Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A modified simulated annealing algorithm and an excessive area model for floorplanning using fixed-outline constraints*

De-xuan ZOU[†‡1], Gai-ge WANG[2], Gai PAN[1], Hong-wei QI[1]

([1]*School of Electrical Engineering and Automation, Jiangsu Normal University, Xuzhou 221116, China*)

([2]*School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, China*)

[†]E-mail: zoudexuan@163.com

Received Nov. 7, 2015; Revision accepted Feb. 16, 2016; Crosschecked Oct. 17, 2016

Abstract: Outline-free floorplanning focuses on area and wirelength reductions, which are usually meaningless, since they can hardly satisfy modern design requirements. We concentrate on a more difficult and useful issue, fixed-outline floorplanning. This issue imposes fixed-outline constraints on the outline-free floorplanning, making the physical design more interesting and challenging. The contributions of this paper are primarily twofold. First, a modified simulated annealing (MSA) algorithm is proposed. In the beginning of the evolutionary process, a new attenuation formula is used to decrease the temperature slowly, to enhance MSA's global searching capacity. After a period of time, the traditional attenuation formula is employed to decrease the temperature rapidly, to maintain MSA's local searching capacity. Second, an excessive area model is designed to guide MSA to find feasible solutions readily. This can save much time for refining feasible solutions. Additionally, B*-tree representation is known as a very useful method for characterizing floorplanning. Therefore, it is employed to perform a perturbing operation for MSA. Finally, six groups of benchmark instances with different dead spaces and aspect ratios—circuits n10, n30, n50, n100, n200, and n300—are chosen to demonstrate the efficiency of our proposed method on fixed-outline floorplanning. Compared to several existing methods, the proposed method is more efficient in obtaining desirable objective function values associated with the chip area, wirelength, and fixed-outline constraints.

Key words: Fixed-outline floorplanning, Modified simulated annealing algorithm, Global search, Excessive area model, B*-tree representation

http://dx.doi.org/10.1631/FITEE.1500386 **CLC number:** TN4

1 Introduction

In the initial design procedure for implementing a physical design, a number of circuit modules need to be placed in suitable positions on a chip. This well-known procedure is floorplanning (Heller *et al.*, 1982; Maling *et al.*, 1982; Otten 1982), which plays an important role in the improved performance of very large scale integrated (VLSI) circuits. As information science develops and the customers' practical needs increase, the sizes of VLSI circuits become larger and larger. Accordingly, the design requirements of floorplanning become more rigorous and complex, and require the floorplanners to reduce design complexity. Most importantly, the floorplanners have to use available geometric information on modules to develop faster and more accurate methods to improve the floorplanning. In a word, the rationality and complexity of a representation will significantly affect the quality of the final floorplan solutions.

During the past decades, several classical methods have been proposed to address floorplanning problems. Wong and Liu (1986) presented a normalized Polish expression to address floorplanning problems. To be more specific, there exist three types of operations in the normalized Polish

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61403174 and 61503165), the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No. 14KJB520011), and the Jiangsu Provincial Science Foundation for Youths (No. BK20150239)

ORCID: De-xuan ZOU, http://orcid.org/0000-0002-6500-5393

expression: (1) swap two adjacent operands; (2) complement some chain of nonzero length; (3) swap two adjacent operand and operators. For the normalized Polish expression corresponding to a given floorplan, it can be perturbed by randomly selecting one from the above three types of operations. Therefore, this representation is capable of carrying out a neighborhood search with flexibility. Murata *et al*. (1996) proposed a solution space where each packing is represented by a pair of module name sequences, called a sequence-pair. A procedure called Gridding is used for describing a packing on the chip, and the packing is encoded to a sequence-pair. Furthermore, two types of lines—the positive and the negative step-lines—are drawn according to the relative positions of their modules, respectively. Accordingly, an arbitrary sequence-pair can be converted to a packing. By applying a sequence-pair, the authors were able to introduce a finite solution space containing an optimal solution. Guo *et al*. (1999) developed an O-tree representation for non-slicing floorplans. They adopted the following procedure to implement perturbation: (1) choose a module in the original O-tree; (2) delete this module from the O-tree; (3) move this module to the position with the best value for the cost function among all possible inserting positions; (4) carry out steps (1)–(3) on the orthogonal O-tree. In short, traversing all possible inserting positions is a direct and efficient strategy. It can produce a number of promising configurations for the perturbation step of the O-tree representation. Chang *et al*. (2000) designed a B*-tree representation for non-slicing floorplans. B*-tree is based on ordered binary trees and admissible placement (Guo *et al*., 1999), and the packing order of all modules is based on depth-first search (DFS). There are three flexible perturbing operations for hard modules, stated as follows: (1) rotate a module; (2) move a module to another place; (3) swap two modules. By adopting these three operations, a set of preferable experimental results can be obtained in a short runtime. Lin and Chang (2001) introduced a transitive closure graph (TCG) based representation for non-slicing floorplans. TCG uses horizontal and vertical transitive closure graphs to represent the horizontal and vertical relations for each pair of modules, respectively. Moreover, TCG gradually carries out operations, and records useful information, including boundary modules as well as the shapes and the relative positions of the modules. In other words, the geometric relationship among modules can be fully used by the operations of the TCG representation. This helps promote the convergence to a desired solution. The above representations have existed for nearly two decades. They are still in widespread use today, and will have a profound impact on subsequent research of floorplanning.

At first, outline-free floorplanning was studied because it is easy to handle. However, as technology advances, researchers have found that this type of problem is useless according to modern design criteria. As a result, a series of new complicated issues with various purposes were rapidly developed, including 3D floorplanning (Cong *et al*., 2004; Khan *et al*., 2014), interconnect-driven floorplanning (Wong and Young, 2003; Yan, 2006; Chen *et al*., 2008), bus-driven floorplanning (Xiang *et al*., 2004; Ma and Young, 2008; Wu and Ho, 2011), and fixed-outline floorplanning (Adya and Markov, 2003; Yan and Chu, 2010; Lin and Hung, 2012; Hoo *et al*. 2014). Among these issues, fixed-outline floorplanning may be the most fundamental design consideration. It limits all the objective modules within a given fixed-outline. In this paper, we emphasize fixed-outline floorplanning, and try to devise a fast and accurate method to cope with this interesting issue.

2 Problem description for fixed-outline floorplanning

Given a floorplan, all of its modules are restricted within a fixed-outline, and no overlap is allowed. Based on this prerequisite, the goal of fixed-outline floorplanning is to minimize the chip area and wirelength simultaneously. Accordingly, the problem formulation of fixed-outline floorplanning is given by

$$\begin{aligned} &\min A,\\ &\min L, \end{aligned} \tag{1}$$

s.t.

$$\begin{cases} H \le H_0, W \le W_0,\\ x_i+w_i \le x_j \text{ or } y_i+h_i \le y_j \text{ or } x_i-w_j \ge x_j \text{ or } y_i-h_j \ge y_j,\\ 1 \le i \le M, 1 \le j \le M, i \ne j, \end{cases}$$

where A and L denote the chip area and wirelength, respectively, H and W denote the height and width of the floorplan, respectively, H_0 and W_0 denote the height and width of the fixed-outline, respectively, $x_i(x_j)$ and $y_i(y_j)$ are the x- and y-coordinate of the lower-left corner of module $i(j)$, respectively, $h_i(h_j)$ and $w_i(w_j)$ are the height and width of module $i(j)$, respectively, and M represents the total number of modules. We can easily obtain the additional relationship among the above problem variables as follows:

$$A = WH, \tag{2}$$

$$W = \max_{1\le i\le M}\{x_i + w_i\}, \tag{3}$$

$$H = \max_{1\le i\le M}\{y_i + h_i\}, \tag{4}$$

$$H_0 = \sqrt{(1+\Gamma)A_{\text{sum}}R}, \tag{5}$$

$$W_0 = \sqrt{(1+\Gamma)A_{\text{sum}}/R}, \tag{6}$$

where A_{sum} represents the total area of all circuit modules, Γ represents the maximum allowed percentage of dead space, and R=H/W represents the aspect ratio. Note that wirelength L cannot be calculated precisely since actual wiring is not implemented in the floorplanning step. Therefore, we have to employ an approximation to compute the wirelength. For a net connecting a number of modules, a commonly used approximation (Kahng, 2000; Adya and Markov, 2003) is to compute its half-perimeter wirelength (HPWL). To be more specific, HPWL is the half-perimeter length of the smallest bounding box which surrounds all pins. If the coordinates of the pins are unavailable, we can measure HPWL according to the center coordinates of the module. Let L_i ($1\le i\le N$) be the wirelength of the ith net. It can be measured using HPWL. Thus, the total wirelength of all nets L can be obtained by summing all values of L_i ($1\le i\le N$).

2.1 Four functions based on width and height violations

Cost function is a criterion that determines which is the best among the candidate floorplan solutions. Its rationality and accuracy directly determine the quality of the obtained floorplan solutions. For outline-free floorplanning, a linear combination of area and wirelength is used to construct the cost function (Chen and Chang, 2006), given by

$$\text{Cost} = \alpha\cdot\frac{A}{A_{\text{norm}}} + (1-\alpha)\cdot\frac{L}{L_{\text{norm}}}, \tag{7}$$

where A_{norm} and L_{norm} stand for the average area and wirelength, respectively, and α is a user-defined parameter, depending on the users' practical needs. Given a floorplan, it is perturbed n_{p} times to generate other n floorplans, and A_{norm} (L_{norm}) can be obtained by averaging all area (wirelength) values of the n_{p} floorplans. In this study, n_{p} is set to $10\sqrt{n}$, where n is the total number of modules.

For fixed-outline floorplanning, a popular way of constructing its cost function is to synthesize the cost function of outline-free floorplanning and fixed-outline constraints. The second constraints of Eq. (1) can be guaranteed by using sequence pair representation. To deal with the first constraints in Eq. (1), Adya and Markov (2003) addressed two functions according to the excessive height and width. We adopt B*-tree representation to meet the first constraints. In addition, a penalty term is incorporated into the cost function to give consideration to both constraints (height and width limits) and objectives (area and wirelength minimization). In short, a universal penalty function framework is adopted, given by

$$\text{Cost} = \alpha\cdot\frac{A}{A_{\text{norm}}} + (1-\alpha)\cdot\frac{L}{L_{\text{norm}}} + \theta\cdot f_{\text{P}}, \tag{8}$$

where f_{P} represents the penalty term, which is actually a function associated with constraint violations, and θ is the penalty coefficient used to impose a penalty on f_{P}. Two versions of the functions (Adya and Markov, 2003) are used to facilitate the satisfaction of the fixed-outline constraints:

$$\text{Cost} = \max\{H - H_0, 0\} + \max\{W - W_0, 0\}, \tag{9}$$

$$\text{Cost} = \max\{H - H_0, W - W_0\}. \tag{10}$$

In Eq. (9), the excessive height and width have the same weight that equals 1. However, if the aspect ratio is larger than 1, the modules will have a worse flexibility in the x direction than in the y direction, and vice versa. In other words, in x and y directions, the flexibilities of the modules will be quite different if the aspect ratio is far from 1. It is concluded that minimizing the sum of the excessive width and height

is better than minimizing the greater one of the two. Here, both functions are considered as the alternatives of f_P. Similarly, two other functions (Liu *et al.*, 2005; Chen and Yoshimura, 2008) are chosen as candidates for f_P, expressed as

$$\text{Cost}=\max\{H-H_0,0\}/H_0+\max\{W-W_0,0\}/W_0, \quad (11)$$

$$\begin{aligned}\text{Cost}=&E_W+E_H\cdot\lambda+C_1\cdot\max\{E_W,E_H\lambda\}\\&+C_2\cdot\max\{W,H\lambda\}.\end{aligned} \quad (12)$$

In Eq. (11), the excessive height and width are normalized by multiplying $1/H_0$ and $1/W_0$, respectively. Therefore, this function creates a tradeoff between the excessive height and width. In Eq. (12), λ is defined as the width/height, and thus $\lambda=1/R$. Additionally, $E_W=\max\{W-W_0, 0\}$ and $E_H=\max\{H-H_0, 0\}$ are the excessive width and height, respectively. Coefficients C_1 and C_2 were set to 1 and 1/16, respectively.

All the above four functions are based on width and height violations. Most of them work well for fixed-outline floorplanning problems with small circuit sizes. However, a floorplan solution generally converges very slowly in the middle and late evolutionary processes for fixed-outline floorplanning problems, especially those with large circuit sizes. Not even one feasible floorplan solution can be obtained for large size problems. In short, the four models all penalize the largest constraint violations, but ignore the smaller ones. Also, it is difficult to say exactly which is the most suitable among these functions, and they may also conflict with each other in some cases.

2.2 A function based on area violation

In addition to the aforementioned four functions based on length violation (width and/or height), a function based on area violation is proposed, into which both the width and height outlines are integrated, in case that any fixed-outline violation occurs. There are four types of expressions, given as

$$f_P=\begin{cases}0, & H\le H_0, W\le W_0,\\ (W-W_0)H_0, & H\le H_0, W\ge W_0,\\ W_0(H-H_0), & H\ge H_0, W\le W_0,\\ WH-W_0H_0, & \text{otherwise.}\end{cases} \quad (13)$$

The area-based function is constructed according to the geometric characteristics of a floorplan from the perspective of a two-dimensional space. Moreover, there are four conditions corresponding to the above four types of expressions (Fig. 1).

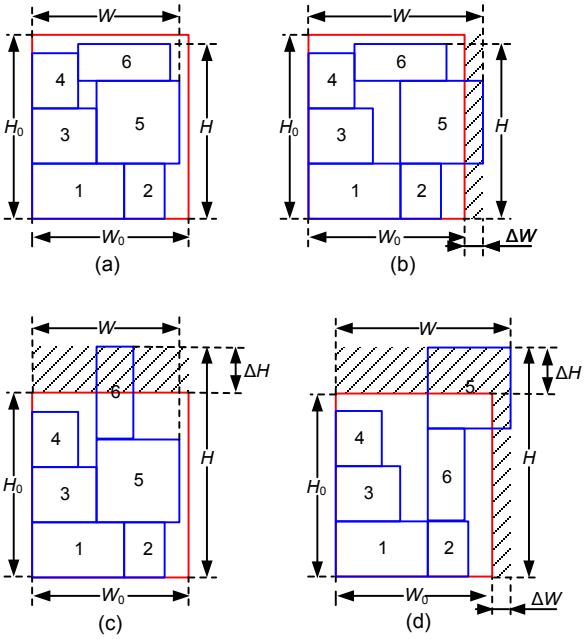

Fig. 1 Four types of floorplans: (a) no outline violation; (b) width violation; (c) height violation; (d) width and height violation

However, in some situations, Eq. (13) may mislead the normal search of the method used, and prevent the improvement of the final floorplan solution. Take Fig. 2 as an example. According to Eq. (5), Figs. 2a and 2b have the same area violation, marked with shadow, indicating that Figs. 2a and 2b have the same floorplan solution quality. However, a close observation reveals that Fig. 2a is better than Fig. 2b, since Fig. 2a contains only one violated module. Obviously, the area-based function is incomplete to some extent. Therefore, there is an urgent need to exploit more reasonable and accurate functions to actively guide the current floorplan towards a desirable one.

2.3 A function based on the excessive area

To speed up the normal search of the method used, we extract finer information from the current floorplan and propose a function based on the

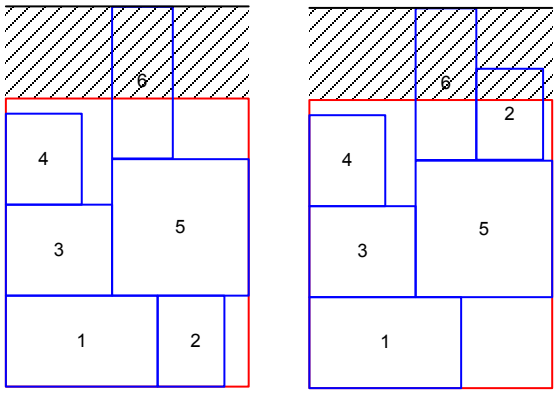

Fig. 2 Area violations in two different situations: (a) violated module 6; (b) violated modules 6 and 2

excessive area. This new function is composed of two parts, given as

$$f_{\mathrm{P}} = A_{\mathrm{O}} + \sum_{i \in I} A_i, \tag{14}$$

where A_{O} is defined as the outer excessive area, produced by the maximum width and height violations, and it has the same expression as Eq. (13). In addition to A_{O}, we introduce a kind of inner excessive area $\sum_{i \in I} A_i$, which is produced by all the modules violating outline constraints, where I denotes the violated modules set and A_i denotes the excessive area of the ith violated module. Furthermore, Fig. 3 explains the necessity of introducing A_i.

In Fig. 3, the largest solid rectangular frame (in red) is the fixed outline of the floorplan, the largest dotted rectangular frame (in black) is the boundary of all the modules, and the shadows represent the violated parts of the modules. Floorplan 1 shown in Fig. 3a contains 10 modules, which are all within the fixed outline. Thus, it is a feasible floorplan solution. On the contrary, floorplans 1–4 in Figs. 3b–3e are not desirable solutions, since some modules exceed the fixed outline. For floorplan 2, modules 6 and 7 partially exceed the outline, and module 10 is completely outside the outline. Thus, these three modules are involved in the calculation of $\sum_{i \in I} A_i$. For floorplan 4, module 6 completely exceeds the fixed outline, and the positions of the other modules are the same as those of floorplan 2. Moreover, floorplans 2 and 4 have the same outer excessive area A_{O}. Therefore, according to Eq. (14), floorplan 2 is better than floorplan 4. Similarly, floorplan 4 is better than floorplan 5. For floorplan 3, although its inner excessive area $\sum_{i \in I} A_i$ is equal to that of floorplan 4, its outer excessive area A_{O} is larger. Thus, floorplan 4 is better than floorplan 3. Finally, A_i is determined by

$$A_i = \begin{cases} 0, & x_2 \le W_0,\ y_2 \le H_0, \\ w_i h_i, & x_1 \ge W_0,\ y_1 \ge H_0, \\ w_i (y_2 - H_0), & x_2 \le W_0,\ y_2 > H_0, y_1 < H_0, \\ (x_2 - W_0) h_i, & y_2 \le H_0,\ x_2 > W_0, x_1 < W_0, \\ w_i h_i - \Delta w \Delta h, & \text{otherwise}, \end{cases} \tag{15}$$

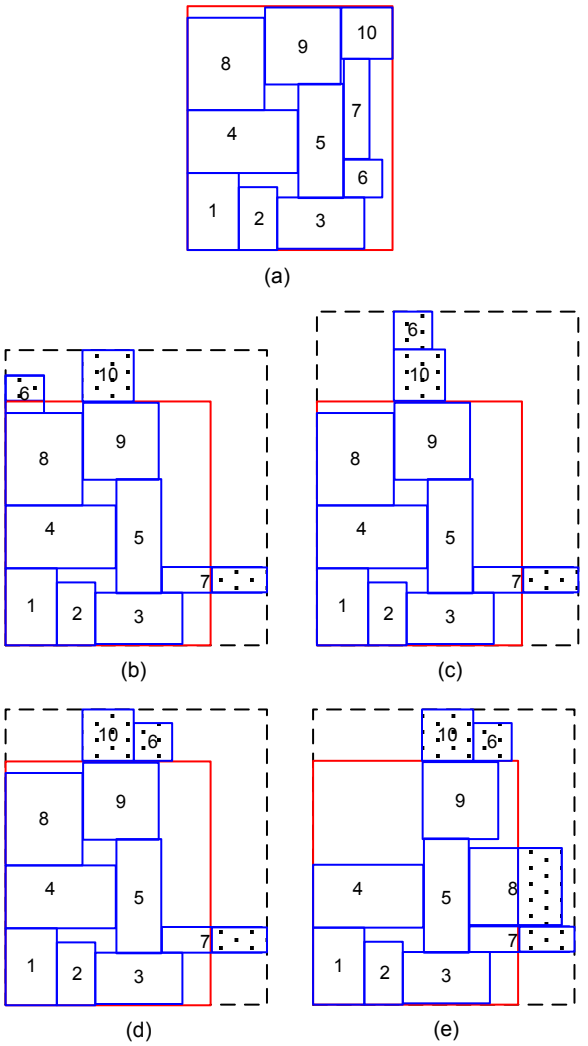

Fig. 3 Excessive areas produced by the modules violating outline constraints: (a) floorplan 1; (b) floorplan 2; (c) floorplan 3; (d) floorplan 4; (e) floorplan 5; (f) floorplan 6

References to color refer to the online version of this figure

where $\Delta w=W_0-x_1$, and $\Delta h=H_0-y_1$. Unlike traditional functions based on length violation, our proposed cost function targets on area violation, and is composed of two parts:

1. The first part is the outer excessive area A_O arising from the maximum violations of width and height. This function efficiently limits the violated area of the floorplan boundary, and provides a potential global searching direction for modules.

2. The second part is the inner excessive area $\sum_{i\in I} A_i$ caused by individual modules violating the outline constraints. Given an infeasible floorplan solution, if its boundary has not been changed after perturbation, we will employ $\sum_{i\in I} A_i$ to further check the total area of the violated modules within the boundary. This part is an important auxiliary function, which enables modules to move towards potential positions.

3 Two simulated annealing algorithms

The simulated annealing (SA) algorithm (Kirpatrick *et al*., 1983) was derived from the physical phenomena in the solidification process of certain fluids. Although very simple in operational principle, SA is very powerful and competitive for a wide variety of optimization problems, such as multiconstraint team orienteering (Lin and Yu, 2015), dynamic double row layout (Wang *et al*., 2015), multi-compartment vehicle routing (Goodson, 2015), log transport system scheduling (Haridass *et al*., 2014), and redundancy allocation (Chambari *et al*., 2013). In addition, it is found more commonly in physical design, especially in floorplanning (Murata *et al*., 1996; Chang *et al*., 2000; Lin and Chang, 2001; Adya and Markov, 2003). In fact, SA has been playing a very important role in the field of floorplanning since it was first introduced by Kirpatrick *et al*. (1983).

3.1 Simulated annealing algorithm

SA is basically a stochastic algorithm, which is quite different from a greedy algorithm, since it allows uphill moves with a probability. More specifically, this probability is determined by two factors: the first is the variation of the cost function value ΔC per generation, and the second is temperature T. Accordingly, a candidate solution will be accepted if the following Boltzman criterion is satisfied:

$$r < \min\{p_a, 1\}, \tag{16}$$

where r is a random number uniformly distributed between 0 and 1, $p_a=\exp(-\Delta C/T)$ denotes the probability of accepting worse solution, $\Delta C=C_{current}-C_{old}$, where $C_{current}$ and C_{old} stand for the current and the previous cost function values, respectively. In addition, temperature T is a positive real number, calculated by $T_{current}=\omega\cdot T_{old}$, where $T_{current}$ and T_{old} stand for the current and the previous temperatures, respectively. ω ($0<\omega<1$) is the attenuation coefficient, which controls the cooling speed of T. In summary, a simple flowchart of SA is provided in Algorithm 1.

Algorithm 1 Simulated annealing

```
1  Initialize a solution S, and S_best=S; set temperature T
   (T>0) and ω (ω<1);
2  while the 'frozen' state is not reached
3    for k=1 to k_max
4      Perturb S to obtain a new one S_new, and calculate
       ΔC=cost(S_new)−cost(S);
5      if ΔC<0
6        S=S_new;
7        if cost (S_new)<cost (S_best)
8          S_best=S_new;
9        end if
10     else
11       if r<exp(−ΔC/T)
12         S=S_new;    // up-hill move
13       end if
14     end if
15   end for
16   T=ωT;           // Cooling temperature
17 end while
18 return S_best
```

In Algorithm 1, S_{best} denotes the best solution, and k_{max} denotes the maximum number of perturbations at the same temperature, which is beneficial for refining the candidate solution. The 'frozen' state represents a termination condition, and it can be the predefined minimum temperature or the allowable maximum runtime. Unfortunately, for $\Delta C>0$, under the action of ω after a number of iterations, T decreases sharply, and p_a achieves almost 0. Obviously, SA gradually loses the chance of accepting inferior solutions in the beginning of the annealing process.

It acts more like a greedy algorithm at the remaining annealing process. Consequently, SA can hardly deal with premature convergence in large-scale floorplanning problems due to the shortage of sufficient uphill moves during most of the annealing time. Therefore, it is necessary to take some effective measures to further promote the convergence of SA.

3.2 Modified simulated annealing algorithm

To avoid the need for a great deal of ineffective searching, we propose a new version of SA, called the modified simulated annealing (MSA) algorithm. To have a better understanding of MSA, its procedure is explained explicitly as follows:

Step 1: initializing algorithm parameters

These parameters include the initial temperature T_0, the attenuation coefficient ω, the number of perturbations at the same temperature k_{max}, the allowable maximum period of stagnation P_s, the number of stagnations N_s=0, the intervals of saving a historical best solution I_h, and the number of updated historical best solutions N_h=0. The last four newly introduced parameters will be elaborated in the subsequent steps.

Step 2: initializing solutions

There are two kinds of solutions that need to be initialized in this step. The first is denoted by S, and it is generated by initializing a B*-tree with all modules as a complete binary tree (Chen and Chang, 2006). The second is the historical best solution S_{best}, and it is generated in the same way as S in this step.

Step 3: perturbation

B*-tree representation is widely applicable for various floorplanning problems. It is employed to perturb S to obtain a new solution S_{new}. In brief, three types of perturbations are used in this step, and they are rotating, moving, and swapping, respectively (Fig. 4).

Step 4: updating solutions

If S_{new} is better than S, directly replace S with S_{new}. If S_{new} is better than S_{best}, replace S_{best} with S_{new}. Otherwise, reserve S_{new} with a probability of p_a= exp(−ΔC/T). Note that steps 3 and 4 are performed k_{max} times for each temperature T.

Step 5: adjusting temperature

The new adjusting strategy partially builds on the traditional one with an important study on how temperature reduction affects the evolutionary process of MSA. At the beginning of the evolutionary process, a new attenuation formula of temperature is used, given by $T = T_0 / \sqrt{N_g}$, where N_g represents the current generation number. Accordingly, the temperature T drops slowly, and the probability of accepting inferior solutions remains large ($0.3 \le p_a \le 0.9$) for a long period of time during the evolutionary process. This early attenuation step is necessary for improving MSA's climbing capacity. However, it should not be used through the whole evolutionary process, since a large number of climbing moves will seriously affect the convergence speed of MSA, especially in the late evolutionary process. Therefore, the early attenuation step is stopped in a timely manner (generally within 1/10 of the running time), followed by using a traditional attenuation formula in the remaining

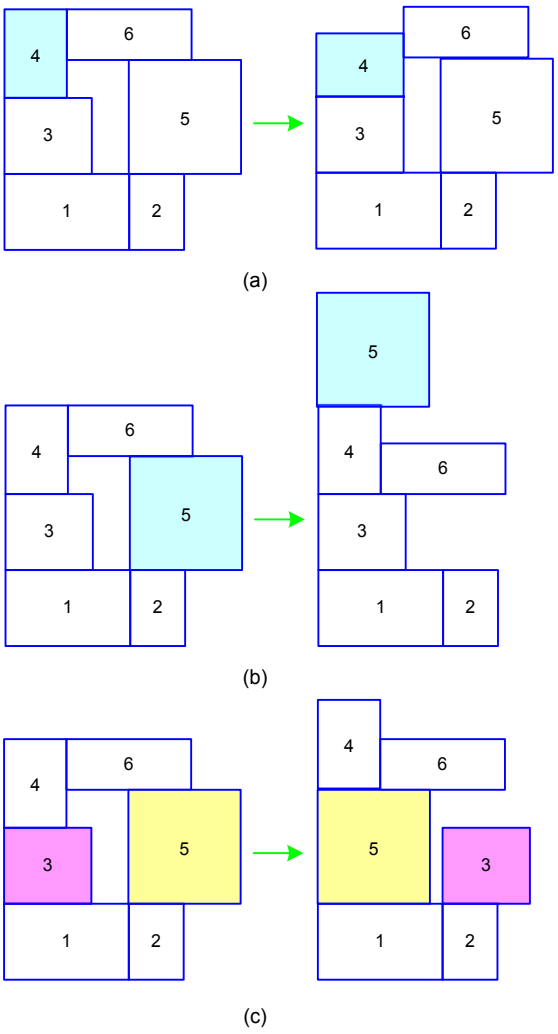

Fig. 4 Three perturbations of B*-tree representation: (a) rotate a module; (b) move a module to another place; (c) swap two modules

time. In short, the formula of adjusting temperature is given by

$$T=\begin{cases}T_0/\sqrt{N_g}, & t_{\text{current}}<t_{\max}/10,\\ \omega T, & \text{otherwise},\end{cases} \tag{17}$$

where t_{current} stands for the current running time, and $t_{\max}$ stands for the allowable maximum running time. This adjusting strategy enables MSA to implement a global search in the early optimizing stage, and a local search in the late optimizing stage. With respect to both global and local searches, this strategy is therefore able to offer a better compromise solution than using a single attenuation formula.

Step 6: monitoring the 'frozen' state

The 'frozen' state is an exact terminating condition. When the allowable maximum runtime is reached, stop the MSA procedure, and output the best solution. Otherwise, repeat steps 3–5.

In the light of the above detailed representation, an algorithm of MSA is shown below.

Algorithm 2 Modified simulated annealing

```
Initialize a solution S, and S_best=S; set temperature T
(T>0) and ω(ω<1);
while the 'frozen' state is not reached
  for k=1 to k_max
    Perturb S to obtain a new one S_new, and calculate
    ΔC=cost(S_new)−cost(S);
    if ΔC<0
      S=S_new;
      if cost(S_new)<cost(S_best)
        S_best=S_new;
      end if
    else
      if r<exp(−ΔC/T)
        S=S_new;      // up-hill move
      end if
    end if
  end for
  if t_current<t_max/10
    T = T_0 / sqrt(N_g); // Decrease temperature at
                         // a quite rapid rate
  end if
end while
return S_best
```

In the field of physical design, SA is generally recognized as the most suitable approach for solving floorplanning problems. Furthermore, it can be well combined with some efficient representations, such as B*-tree and sequence pair. In addition to SA, there are some other heuristics approaches (Kennedy and Eberhart, 1995; Storn and Price, 1997) which may possibly be alternatives in solving floorplanning, such as particle swarm optimization (PSO) and differential evolution (DE). However, these approaches are rarely applied for floorplanning in practice, indicating that their performance will not be better than SA for floorplanning. Concretely, SA is different from other heuristic approaches in the following two aspects:

First, the perturbation step for SA is based on B*-tree representation, and it can always satisfy the nonoverlapping constraints (as in Eq. (1)) in each generation. Velocity updating and position updating are the two main perturbation steps of PSO, and mutation and crossover are the two main perturbation steps of DE. Unfortunately, these steps cannot meet the nonoverlapping requirements because of their particular characteristics. In other words, both PSO and DE require extra effort to eliminate the nonoverlapping among all modules through the evolutionary process, which is really very time-consuming, and will significantly affect the computational efficiency of PSO and DE.

Second, SA accepts worse solutions, which means it has a strong capacity of climbing. Sometimes a worse solution is necessary, since it can serve as a bridge between poor and potential solution spaces. This feature is very important for SA, since it can help SA avoid premature convergence. On the contrary, some other heuristic approaches (such as PSO and DE) will miss a lot of chances of searching for potential solutions, since they always omit worse solutions. These approaches have large convergence rates in the early evolutionary process. However, this trend can easily get them into local optima during the middle and late evolutionary processes. Once caught in a local optimum, it is hard for them to quickly get rid of this predicament.

4 Experimental results and discussions

We conducted three groups of comprehensive experiments to investigate the overall properties of the combination of the excessive area model and

MSA algorithm. First, SA is used to optimize six types of functions to verify the efficiency of the excessive area model for handling fixed-outline constraints. Second, the excessive area model is used to guide SA and MSA, respectively, to cope with the constrained single-objective optimization—area (wirelength) minimization constrained by fixed-outline requirements. Finally, the excessive area model is used to help SA and MSA, respectively, to handle constrained multi-objective optimization—area and wire-length minimization, constrained by fixed-outline requirements. These three groups of experiments adopted the same GSRC floorplan benchmark instances (Kahng and Markov, 2007) to test and compare the performances of different approaches, and these instances were circuits n10, n30, n50, n100, n200, and n300. To be more specific, they contained 10, 30, 50, 100, 200, and 300 modules, respectively, and 118, 349, 485, 885, 1585, and 1893 nets, respectively. In addition, C++ language was used to implement the above experiments in the environment of an Intel Core i5-2410M CPU at 2.30 GHz. Both SA and MSA were based on B*-tree floorplan representation and under the same initial temperature. For each floorplanning instance, the dead space was set to Γ=10% and the aspect ratio was set to R=1.0, 2.0, and 3.0, respectively. For fair comparison, the I/O pads for all circuits were fixed at the given coordinates in the benchmark.

4.1 Testification of the efficiency of six methods for handling fixed-outline constraints

We focus only on fixed-outline constraints with $H \le H_0$ and $W \le W_0$. That is, this kind of problem is not considered solved until all modules have been put inside a given fixed-outline. To meet the requirements of fixed-outline constraints, six functions were considered, and each function was used solely to drive the evolution of SA to find a desirable feasible solution. Furthermore, these six functions are from Eqs. (9)–(14). The first four functions are based on the excessive length, denoted by EL1, EL2, EL3, and EL4, respectively. The last two functions are based on the excessive area, denoted by EA1 and EA2, respectively. For a fair comparison, other than the number of generations, the running time of all approaches was kept the same for each problem. The allowable maximum runtime of each method for n10, n30, n50, n100, n200, and n300 is set to 10, 10, 10, 30, 40, and 60 s, respectively. For SA, the attenuation coefficient was set to ω=0.85. The initial temperature was set to $T_0 = 10\overline{\Delta C_{\text{uphill}}}$, where $\overline{\Delta C_{\text{uphill}}}$ denotes the average uphill cost during $\left\lfloor 5\sqrt{n} \right\rfloor$ perturbations, and n is the number of modules. The number of perturbations was set to $k_{\max}$=7 at the same temperature. Additionally, an initial solution was generated by initializing a B*-tree with all modules as a complete binary tree. Given this solution method, it is regarded as a success if it meets all the fixed-outline constraints. Otherwise, it is regarded as a failure. Each problem was implemented 20 times. The efficiencies of the six methods for the floorplanning problems, considering only fixed-outline constraints, were tested and compared.

Table 1 lists the success rates and average running times of six methods for the floorplanning problems, considering only fixed-outline constraints. SA/MSA-EL*m*/EA*m* represents the SA/MSA algorithm based on objective function EL*m*/EA*m*. SR and AR denote the success rate and average running time, respectively. The numbers in bold indicate the best results. Note that the average running time is basically the average time to achieve the feasible solutions for any of the six methods. According to the marked best results, it is clear that in driving SA to find feasible solutions, the EA2-based function is more efficient than the other five functions. In detail, SA-EA2 is capable of obtaining feasible solutions with SR=100% for all instances except n10. Furthermore, SA-EA2 uses smaller ARs to obtain feasible solutions in most cases, suggesting its strong convergence and stability. On the contrary, it is hard for the other five methods to obtain desirable solutions for circuits n10, n200, and n300. Among these five methods, SA-EL3 and SA-EA1 are comparable for most problems, and they perform slightly better than the other three methods. With respect to n10, all six methods failed to achieve a success rate of 100%, indicating that n10 is to some extent more intractable than the other five problems. In fact, the average size of the modules in any of the other five circuits is much smaller than the dead space of the corresponding floorplan, which makes it easier to move a module from its current position to a better one. Hence, most modules for these circuits have higher flexibilities than those for n10. The average size of the modules in n10

Table 1 Success rates and average running times of six methods for the floorplanning problems considering only fixed-outline constraints

Circuit	R	SR (%)						AR (s)					
		SA-EL1	SA-EL2	SA-EL3	SA-EL4	SA-EA1	SA-EA2	SA-EL1	SA-EL2	SA-EL3	SA-EL4	SA-EA1	SA-EA2
n10	1.0	40	**70**	45	25	55	40	6.07	3.22	5.56	7.53	4.60	4.59
	2.0	80	70	**85**	65	60	55	2.03	3.07	1.55	3.54	4.04	4.54
	3.0	**65**	50	40	25	45	30	3.62	5.09	6.04	7.52	5.96	7.21
n30	1.0	100	100	100	100	100	100	0.24	0.23	0.26	**0.22**	0.26	0.23
	2.0	100	100	100	100	100	100	0.38	0.21	0.31	0.27	0.25	**0.15**
	3.0	100	100	100	100	100	100	0.30	0.29	0.32	**0.25**	0.26	0.26
n50	1.0	100	100	100	100	100	100	0.64	0.56	0.78	0.51	0.46	**0.35**
	2.0	100	100	100	100	100	100	0.91	0.78	0.69	0.61	0.70	**0.44**
	3.0	100	100	100	100	100	100	0.88	0.74	0.92	0.79	0.81	**0.37**
n100	1.0	100	100	100	100	100	100	2.51	6.01	2.59	3.00	1.89	**1.38**
	2.0	100	100	100	100	100	100	5.80	6.75	4.89	4.28	2.05	**1.91**
	3.0	100	100	100	100	100	100	8.99	7.26	7.82	6.21	7.11	**2.23**
n200	1.0	95	30	100	95	100	100	12.83	37.76	9.87	18.27	9.62	**4.22**
	2.0	30	10	95	60	90	**100**	32.12	39.19	13.42	31.53	13.08	**5.87**
	3.0	0	0	35	5	25	**100**	40.00	40.00	31.29	39.79	31.95	**6.76**
n300	1.0	100	0	90	75	100	100	33.35	60.00	33.55	38.84	29.98	**11.11**
	2.0	25	0	85	30	75	**100**	53.44	60.00	28.58	50.24	31.28	**17.58**
	3.0	5	0	55	5	40	**100**	57.61	60.00	35.76	57.92	45.72	**13.99**

SA/MSA-EL*m*/EA*m* represents the SA/MSA algorithm based on objective function EL*m*/EA*m*. The numbers in bold indicate the best results. SA: simulated algorithm; MSA: modified simulated algorithm; EL: excessive length; EA: excessive area; SR: success rate; AR: average running time; *R*: aspect ratio calculated by height/width

closely approaches the dead space of the corresponding floorplan. Consequently, most modules of n10 lack flexibility. Even worse, once trapped in a local optimum, SA can hardly get rid of this optimum.

A more intuitive test was employed to visualize the evolutionary process of SA based on the six objective functions. To be more specific, the objective function values against the number of generations were recorded to obtain the convergence curves of each method over 20 independent runs. Fig. 5 portrays the convergence curves of six methods for the largest scale problem, n300, with R=3.0.

As shown in Fig. 5, SA-EA2 is more efficient than the other five approaches, since all its objective function values achieve 0 in 20 runs. Note that most convergence curves of SA-EA2 descend at a fast pace, and reach the lowest level in the early evolutionary process, suggesting that SA-EA2 possesses not only a fast convergence speed, but also a strong capacity to overcome local optima. Meanwhile, SA-EL1, SA-EL3, and SA-EL4 succeed to reach the lowest level only for several times. SA-EL2 fails to reach the lowest level in 20 runs, and thus this approach has exhibited a poor performance in finding feasible solutions.

4.2 Area (wirelength) minimization constrained by fixed-outline requirements

After the excessive area model was validated, it was also employed to solve constrained area (wirelength) minimization. We used the combinations of three functions and two SA algorithms to deal with the area (wirelength) minimization of the floorplanning constrained by fixed-outline requirements. More specifically, the three functions were obtained from the first three best ones of the aforementioned six functions—EL3, EA1, and EA2. The two SA algorithms are the original SA and our proposed MSA. The goal of using these combinations is to find the most suitable floorplanner for constrained area (wirelength) minimization. We used a penalty function method to synthesize the objective (area or wirelength minimization) and constraints (fixed-outline requirements), and it can be expressed as $\text{Cost}=f_S+\theta\cdot f_P$, where f_S represents a single-objective

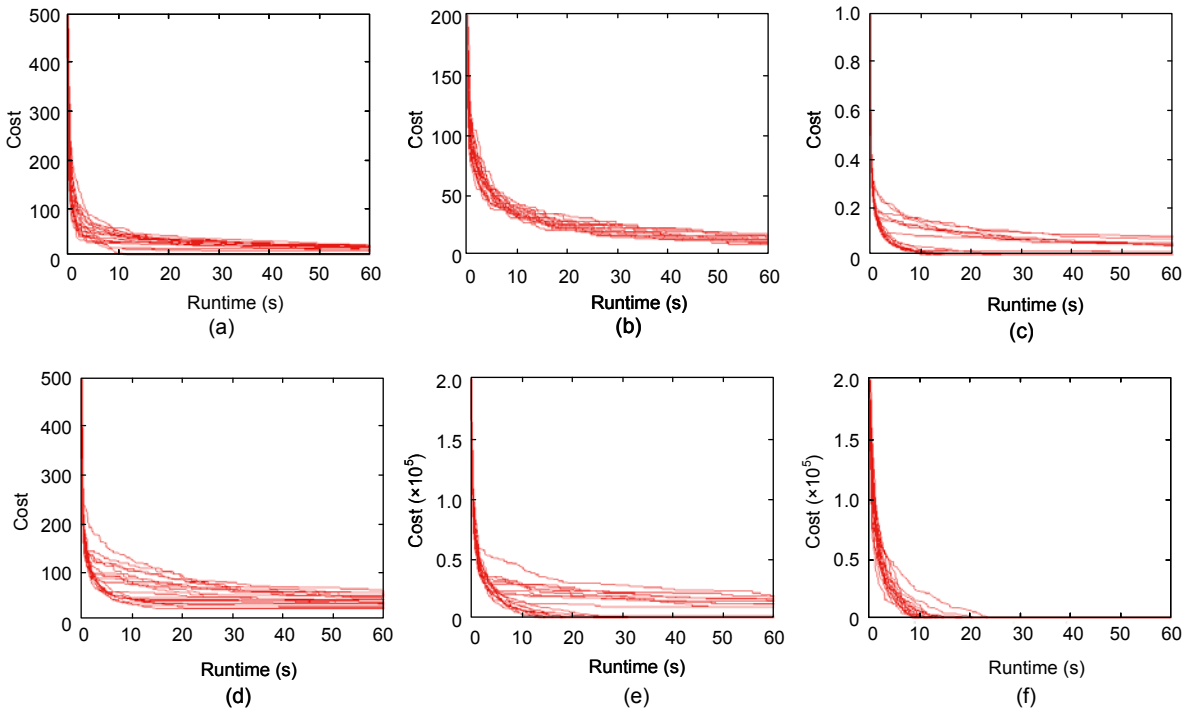

Fig. 5 Convergence curves of six methods for n300 with R=3.0: (a) SA-EL1; (b) SA-EL2; (c) SA-EL3; (d) SA-EL4; (e) SA-EA1; (f) SA-EA2

SA/MSA-EL*m*/EA*m* represents the SA/MSA algorithm based on objective function EL*m*/EA*m*

function, whose dependent variable can be either the area A or wirelength L, and the penalty coefficient θ was set to θ=10^{10}; the penalty term f_P can be replaced by any of the three selected functions. Regarding constrained area minimization, the allowable maximum runtime of each method for n10, n30, n50, n100, n200, and n300 was set to 10, 10, 10, 30, 40, and 60 s, respectively. However, unlike constrained area minimization, constrained wirelength minimization is more time-consuming. Therefore, the allowable maximum runtime of each method for n10, n30, n50, n100, n200, and n300 was set to 30, 30, 30, 60, 120, and 200 s, respectively. For both SA and MSA algorithms, the attenuation coefficients were set to ω=0.85, the initial temperatures were set to $T_0=10\overline{\Delta C_{\text{uphill}}}$, and the numbers of perturbations were set to $k_{\max}$=7 at the same temperature. Each case was tested 20 times. The average area and wirelength were obtained by performing the six combinations, reported in Tables 2 and 3, respectively.

In Table 2, AA represents the average area. The numbers in bold indicate the best results. Regarding n10 with R=1.0, 2.0, and 3.0, three approaches based on MSA are able to obtain higher success rates than the other three approaches. Moreover, MSA-EA2 obtained 100% success rates for n10 with all three aspect ratios. In the meantime, the AAs of MSA-EL3 were smaller than those of the other five approaches for n10 with R=1.0 and 2.0, and the AA of MSA-EA2 was smaller than those of the other five approaches for n10 with R=3.0. However, for the problem that n10 with different aspect ratios, the other three approaches based on SA failed to obtain SR=100%. Obviously, compared with SA, MSA can escape from the local optima more easily. For any of the other five circuits including n30, n50, n100, n200, and n300, two approaches guided by the EA2 function can find feasible solutions with a 100% success rate in 20 runs. In contrast, the success rates of the other four approaches became worse as the circuit size increased. Thus, the EA2 function was more efficient and effective than the other two functions in driving two SAs to search for desirable floorplan solutions. Additionally, MSA-EA2 yielded the nine smallest AAs, which account for half of the best results of all 18 problems. SA-EA1 yielded the four smallest AAs, but failed to obtain a 100% success rate for 12 instances. Although SA-EA2 yields only one smallest AA,

Table 2 Average area obtained by six combinations on solving constrained area minimization

Circuit	R	AA						SR (%)					
		SA-EL3	SA-EA1	SA-EA2	MSA-EL3	MSA-EA1	MSA-EA2	SA-EL3	SA-EA1	SA-EA2	MSA-EL3	MSA-EA1	MSA-EA2
n10	1.0	239 182	241 503	243 579	**231 430**	232 496	232 065	25	35	25	100	100	100
	2.0	238 718	240 871	241 409	**233 402**	235 068	234 460	30	55	60	100	100	100
	3.0	242 712	244 117	249 239	237 797	239 295	**238 084**	10	35	35	95	95	**100**
n30	1.0	218 562	**217 723**	218 075	218 034	218 425	217 778	100	100	100	95	95	100
	2.0	217 480	**217 263**	218 669	217 259	216 859	217 793	90	100	100	90	90	100
	3.0	218 152	218 292	217 357	217 778	**216 544**	217 244	85	100	100	95	95	100
n50	1.0	205 666	205 659	205 947	206 489	205 871	**205 472**	100	100	100	95	95	100
	2.0	205 729	205 652	206 396	206 038	205 836	**205 493**	100	100	100	85	85	100
	3.0	205 708	**205 379**	206 599	205 273	205 469	205 583	75	100	100	85	85	100
n100	1.0	187 073	**186 761**	187 124	188 139	187 789	187 430	100	100	100	100	100	100
	2.0	186 547	186 883	186 728	187 539	186 829	**186 518**	80	100	100	80	80	100
	3.0	188 015	186 198	186 800	188 167	186 818	**185 990**	35	100	100	20	20	100
n200	1.0	**186 367**	187 802	186 756	190 614	190 994	187 261	100	95	100	15	15	100
	2.0	186 158	188 710	**185 834**	191 126	194 161	185 962	85	75	100	0	0	100
	3.0	189 110	195 446	185 247	189 978	197 006	**185 200**	0	15	100	0	0	100
n300	1.0	295 718	294 671	295 333	306 002	303 404	**294 264**	100	100	100	0	0	100
	2.0	295 694	292 039	293 261	310 136	308 033	**292 323**	30	95	100	0	0	100
	3.0	294 678	292 080	291 755	302 632	315 086	**291 557**	45	95	100	0	0	100

SA/MSA-EL*m*/EA*m* represents the SA/MSA algorithm based on objective function EL*m*/EA*m*. The numbers in bold indicate the best results. SA: simulated algorithm; MSA: modified simulated algorithm; EL: excessive length; EA: excessive area; AA: average area; SR: success rate; *R*: aspect ratio calculated by height/width

Table 3 Average wirelength obtained by six combinations on solving constrained wirelength minimization

Circuit	R	AWL						SR (%)					
		SA-EL3	SA-EA1	SA-EA2	MSA-EL3	MSA-EA1	MSA-EA2	SA-EL3	SA-EA1	SA-EA2	MSA-EL3	MSA-EA1	MSA-EA2
n10	1.0	42 924	44 247	45 050	39 726	39 629	**39 227**	15	30	30	100	100	100
	2.0	43 831	45 237	46 413	**42 414**	42 847	42 550	25	60	45	100	100	100
	3.0	46 769	49 698	49 504	**45 375**	45 481	46 323	10	55	35	100	100	100
n30	1.0	127 487	120 623	**119 436**	116 579	120 119	121 023	0	100	100	55	100	100
	2.0	119 892	123 951	125 204	123 974	125 098	**124 655**	20	100	100	80	100	100
	3.0	127 175	133 555	133 135	134 333	134 339	**132 013**	10	100	100	15	100	100
n50	1.0	153 210	155 227	**153 848**	151 525	155 368	154 204	40	100	100	45	100	100
	2.0	152 563	158 987	164 483	152 527	157 518	**156 947**	15	100	100	25	100	100
	3.0	160 209	169 358	170 322	157 884	168 844	**168 222**	15	100	100	40	100	100
n100	1.0	237 378	241 977	243 782	230 941	**239 211**	241 323	20	100	100	25	100	100
	2.0	232 568	248 516	257 461	229 381	247 809	**247 150**	10	100	100	0	100	100
	3.0	241 300	269 278	266 581	242 690	**260 735**	266 090	0	100	100	0	100	100
n200	1.0	442 768	463 923	458 269	430 563	529 162	**450 291**	0	95	100	0	45	100
	2.0	455 256	474 813	**465 320**	431 886	591 249	467 029	15	90	100	0	0	100
	3.0	498 297	545 192	**505 975**	453 130	635 068	506 574	20	65	100	0	0	100
n300	1.0	618 886	680 069	641 884	595 545	765 977	**639 793**	0	100	100	0	45	100
	2.0	662 567	687 059	**673 267**	615 440	872 540	684 506	0	90	100	0	10	100
	3.0	695 732	821 299	737 587	638 237	956 978	**731 336**	0	60	100	0	0	100

SA/MSA-EL*m*/EA*m* represents the SA/MSA algorithm based on objective function EL*m*/EA*m*. The numbers in bold indicate the best results. SA: simulated algorithm; MSA: modified simulated algorithm; EL: excessive length; EA: excessive area; AWL: average wirelength; SR: success rate; *R*: aspect ratio calculated by height/width

it succeeded in obtaining a 100% success rate for 15 instances, suggesting a higher stability of SA-EA2 than that of SA-EA1. Both SA-EL3 and MSA-EL3 underperform for most instances, since they yield only few desirable results according to two criteria—success rate and the number of the smallest AA obtained. In light of the above comprehensive comparison, MSA-EA2 significantly outperforms the other five approaches in solving area minimization constrained by fixed-outline requirements.

Table 3 shows the average wirelengths (AWLs) and SRs obtained using six combinations. It should be emphasized that the combination which encounters any failure of 20 runs was excluded from the comparison among six approaches for any problem, since it contains the results of infeasible solutions which are considered meaningless. According to the number of marked results, MSA-EA2 performed better than the other five approaches. It yielded the nine smallest AWLs, which account for half of the 18 best results for all problems. Meanwhile, the total number of thebest results obtained using SA-EL3, SA-EA1, SA-A2, MSA-EL3, MSA-EA1 is nine, which is exactly the same as that of MSA-EA2. According to the criterion AWL, SA-EA2 is the second best approach with five best results for 18 problems. With the same smallest AWL, MSA-EL3 and MSA-EA1 are comparable to each other. Unfortunately, neither SA-EL3 nor SA-EA1 can obtain even one best result for 18 problems, indicating the poor convergence of both approaches. In addition, the superiority of MSA-EA2 over the other five approaches can be further observed in the experimental results associated with its success rate. As shown in Table 3, MSA-EA2 can obtain feasible solutions with a 100% success rate for any instance, suggesting its strong stability during the evolutionary process. The second best approach is SA-EA2, since it is usually successful, except in the case of handling n10 with various aspect ratios. SA-EL3 failed to yield a 100% success rate for any instance, and what is worse, all its SRs were smaller than 50%, indicating the inefficiency of SA-EL3 in finding feasible solutions. In view of the two criteria AWL and SR, MSA-EA2 performed the best among all the six approaches, and it exhibited good convergence and stability on solving constrained wirelength minimization.

4.3 Area and wirelength minimization constrained by the fixed-outline requirements

The combinations of three functions (EL1, EA1, and EA2) and two SA algorithms were used in an attempt to simultaneously minimize the area and wirelength under fixed-outline constraints. Moreover, the penalty function method stated in Eq. (8) was employed to synthesize the multiple objectives and constraints. The coefficient of the normalized area was set to α=1/2, the penalty coefficient was set to θ=10^{10}, and the penalty term f_P can be chosen from the three candidate functions. Both A_{norm} and L_{norm} were produced by averaging the area and wirelength of all floorplans, respectively, in $\lfloor 5\sqrt{n} \rfloor$ perturbations. The allowable maximum runtime of each method for n10, n30, n50, n100, n200, and n300 is set to 30, 30, 30, 60, 120, and 200 s, respectively. For both SA and MSA, ω=0.85, $T_0 = 10\overline{\Delta C_{\text{uphill}}}$, and $k_{\max}$=7. Each instance was conducted 20 times. Table 4 shows the average cost function values obtained by six approaches.

As shown in Table 4, MSA-EA2 performed best among the six approaches in terms of the two criteria average, the objective function value (AOFV) and SR. For example, among the six approaches, MSA-EA2 achieved eight smallest objective function values for all 18 problems. Moreover, MSA-EA2 can obtain feasible solutions with a 100% success rate for any of the 18 problems. In addition, the other five approaches obtained the 10 totally smallest AOFVs for the 18 problems, and each of them achieved two best results. SA-EA2 can succeed in obtaining feasible solutions with a 100% success rate for all problems, except n10 with R=1.0, 2.0, and 3.0. On the contrary, the success rates of SA-EL3, SA-EA1, MSA-EL3, and MSA-EA1 tended to have lower results as the circuit scale and aspect ratio increased. On the whole, our excessive area model (EA2) is more efficient than the other two models (EL3 and EA1) in assisting SA and MSA in the exploration of the feasible solution space. Furthermore, combined with

EA2, MSA and SA are capable of quickly approaching the feasible solution space, which can gain enough time to further refine high-quality feasible solutions. Basically, we attribute the high efficiency of the EA2 model to the reasonable evaluation mechanism for constraint violations. In contrast, the poor efficiency of the EL3 and EA1 models is due to the deficiency of the proper utilization of the geometrical features regarding floorplans.

To testify the robustness of MSA-EA2, the initial solution was continuously perturbed N_{perturb} times before MSA-EA2 started. Moreover, we will investigate the effects of N_{perturb} on the results obtained using MSA-EA2. The dead space and aspect ratio were set to Γ=10% and R=3.0, respectively. Each instance was conducted 20 times. Table 5 shows the results obtained by MSA-EA2 for various initial solutions.

AMR represents the allowable maximal runtime for each case. It increased as the circuit size increased. According to Table 5, the SR of MSA-EA2 was 100% for each problem. Furthermore, the differences

Table 4 Average cost function values obtained using six approaches

Circuit	R	AOFV						SR (%)					
		SA-EL3	SA-EA1	SA-EA2	MSA-EL3	MSA-EA1	MSA-EA2	SA-EL3	SA-EA1	SA-EA2	MSA-EL3	MSA-EA1	MSA-EA2
n10	1.0	1.131484	1.133801	1.158394	**1.050490**	1.100118	1.077886	45	50	35	100	100	100
	2.0	1.158049	1.150787	1.165778	1.110397	1.114380	**1.101369**	55	65	60	100	100	100
	3.0	1.227295	1.224293	1.231851	1.159313	1.205179	**1.153789**	45	25	30	100	55	100
n30	1.0	1.052716	1.058293	1.047826	1.049503	1.046095	**1.045172**	100	100	100	100	100	100
	2.0	**1.073819**	1.076343	1.084704	1.076567	1.077146	1.080114	100	100	100	100	100	100
	3.0	1.123534	1.118898	1.125955	**1.114918**	1.123375	1.115012	100	100	100	100	100	100
n50	1.0	**0.897405**	0.904454	0.899078	0.900381	0.894795	0.900166	100	100	100	100	100	100
	2.0	0.915673	0.910948	0.923015	0.915470	**0.909410**	0.913261	100	100	100	100	100	100
	3.0	0.954025	0.956104	0.954699	0.941319	0.947424	**0.940996**	100	100	100	100	100	100
n100	1.0	0.691027	0.687007	0.689286	0.684166	**0.682224**	0.685029	100	100	100	100	100	100
	2.0	0.699437	0.702461	0.703290	0.703154	0.700334	**0.697567**	100	100	100	100	100	100
	3.0	0.734666	0.729840	0.725235	0.720365	0.729039	**0.720136**	95	100	100	100	100	100
n200	1.0	0.598852	**0.588953**	0.591166	0.657804	0.631980	0.590825	90	100	100	20	60	100
	2.0	0.628175	0.608225	**0.603381**	0.693873	0.694236	0.610766	70	90	100	0	0	100
	3.0	0.725340	0.627114	0.620991	0.728373	0.727570	**0.619072**	5	95	100	0	0	100
n300	1.0	0.497865	0.500331	**0.495205**	0.574662	0.550421	0.502643	100	100	100	0	35	100
	2.0	0.539793	**0.507663**	0.510876	0.596936	0.596934	0.511838	65	100	100	0	0	100
	3.0	0.620987	0.534674	0.533615	0.625650	0.614755	**0.529251**	0	95	100	0	10	100

SA/MSA-ELm/EAm represents the SA/MSA algorithm based on objective function ELm/EAm. The numbers in bold indicate the best results. SA: simulated algorithm; MSA: modified simulated algorithm; EL: excessive length; EA: excessive area; AOFV: average objective function value; SR: success rate; R: aspect ratio calculated by height/width

Table 5 Results obtained using MSA-EA2 for various initial solutions (Γ=10%, R=3.0)

Circuit	AMR(s)	AOFV					SR (%)				
		N_{perturb}= 50	N_{perturb}= 100	N_{perturb}= 200	N_{perturb}= 500	N_{perturb}= 1000	N_{perturb}= 50	N_{perturb}= 100	N_{perturb}= 200	N_{perturb}= 500	N_{perturb}= 1000
n10	30	1.168389	1.171100	1.168716	1.171715	1.147228	100	100	100	100	100
n30	30	1.125225	1.124228	1.128384	1.118624	1.119547	100	100	100	100	100
n50	30	0.947668	0.945060	0.955124	0.940019	0.954123	100	100	100	100	100
n100	60	0.722553	0.723566	0.721898	0.722232	0.736069	100	100	100	100	100
n200	120	0.629260	0.629209	0.630130	0.625429	0.626835	100	100	100	100	100
n300	200	0.528583	0.533549	0.537897	0.540180	0.529990	100	100	100	100	100

MSA-EA2 represents the MSA algorithm based on objective function EA3. MSA: modified simulated algorithm; EA: excessive area; AMR: allowable maximal runtime; AOFV: average objective function value; SR: success rate; N_{perturb}: the times that the initial solution perturbed before MSA-EA2 starts

among different AOFVs were slight for each problem. Due to the suitable convergence of MSA and desirable accuracy of EA2, MSA-EA2 was still able to cause a solution to rapidly move towards potential solution space, even though the initial solution has been continuously perturbed for many times. Therefore, the proposed approach has demonstrated strong robustness on solving fixed-outline floorplanning.

MSA-EA2 results were used in high-quality floor-plan solutions for all problems. Also, the floorplans of all problems obtained using MSA-EA2 can fit into the given outline. For simplicity, three feasible floorplans of the largest scale circuit n300 are shown in Fig. 6.

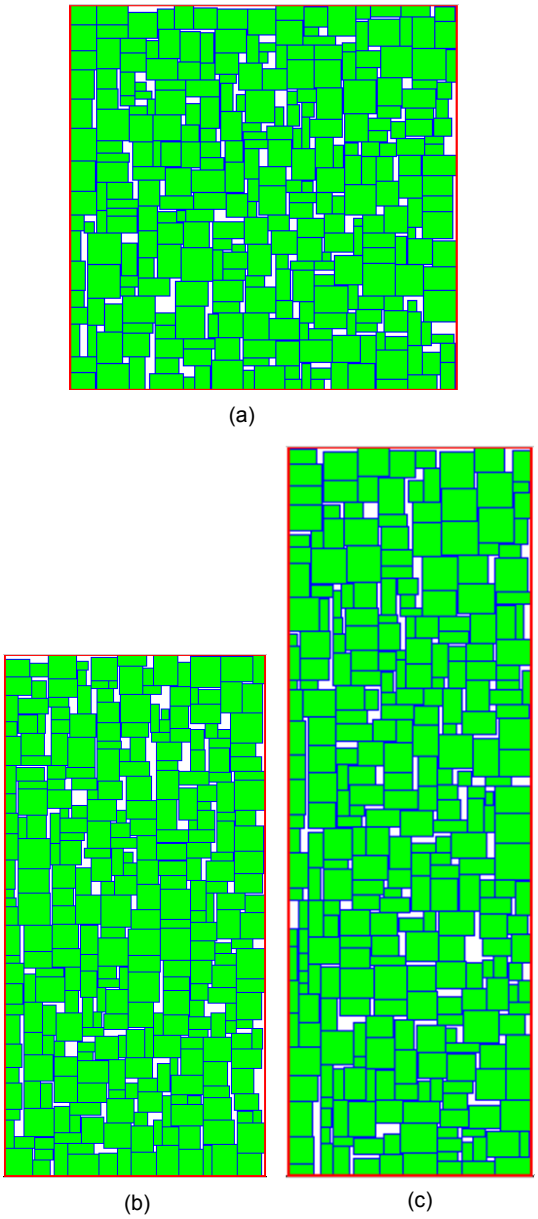

Fig. 6 Three feasible floorplanns for n300 with various aspect ratios: (a) *R*=1.0; (b) *R*=2.0; (c) *R*=3.0

5 Conclusions

We present a useful method to deal with modern floorplanning with fixed-outline constraints. The method is based on the MSA algorithm and the excessive area model. More specifically, MSA updates the temperature by using two different attenuation formulas. The first formula is adopted to implement a slow reduction for the temperature in the early evolutionary stage, and the second one is used to implement a sharp reduction for the temperature in the subsequent evolutionary stage. Compared to SA, MSA has a stronger capacity of exploiting the solution space, and thus it is able to better alleviate the occurrence of premature convergence. On the other hand, the excessive area model is quite different from previous models, which concentrate on width and height violations. Also, this model is composed of two parts, the outer and inner excessive areas. It is more accurate than previous models in driving MSA to find feasible solutions. Additionally, B*-tree representation is used to perturb a candidate solution in each iteration. This is beneficial for improving the convergence of MSA. By incorporating the excessive area model and B*-tree representation into MSA, it can be guaranteed that a candidate solution will gradually suit fixed-outline during the annealing progress of MSA. Experimental results show that the proposed method is very efficient in finding feasible solutions with different dead spaces and aspect ratios. Overall, the proposed method is superior to the other five methods for most test instances, and is thus a promising choice for floorplanning with fixed-outline constraints.

References

Adya, S.N., Markov, I.L., 2003. Fixed-outline floorplanning: enabling hierarchical design. *IEEE Trans. VLSI Syst.*, **11**(6):1120-1135.

http://dx.doi.org/10.1109/TVLSI.2003.817546

Chambari, A., Najafi, A.A., Rahmati, S.H.A., *et al.*, 2013. An efficient simulated annealing algorithm for the redundancy allocation problem with a choice of redundancy strategies. *Reliab. Eng. Syst. Safety*, **119**:158-164. http://dx.doi.org/10.1016/j.ress.2013.05.016

Chang, Y.C., Chang, Y.W., Wu, G.M., *et al.*, 2000. B*-trees: a new representation for non-slicing floorplans. Proc. Design Automation Conf., p.458-463. http://dx.doi.org/10.1109/DAC.2000.855354

Chen, S., Yoshimura, T., 2008. Fixed-outline floorplanning: block-position enumeration and a new method for calculating area costs. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.*, **27**(5):858-871. http://dx.doi.org/10.1109/TCAD.2008.917968

Chen, T.C., Chang, Y.W., 2006. Modern floorplanning based on B*-tree and fast simulated annealing. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.*, **25**(4):637-650. http://dx.doi.org/10.1109/TCAD.2006.870076

Chen, T.C., Chang, Y.W., Lin, S.C., 2008. A new multilevel framework for large-scale interconnect-driven floorplanning. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.*, **27**(2):286-294. http://dx.doi.org/10.1109/TCAD.2007.907065

Cong, J., Wei, J., Zhang, Y., 2004. A thermal-driven floorplanning algorithm for 3D ICs. IEEE/ACM Int. Conf. on Computer Aided Design, p.306-313. http://dx.doi.org/10.1109/ICCAD.2004.1382591

Goodson, J.C., 2015. A priori policy evaluation and cyclic-order-based simulated annealing for the multi-compartment vehicle routing problem with stochastic demands. *Eur. J. Oper. Res.*, **241**(2):361-369. http://dx.doi.org/10.1016/j.ejor.2014.09.031

Guo, P.N., Cheng, C.K., Yoshimura, T., 1999. An O-tree representation of non-slicing floorplans and its applications. Proc. Design Automation Conf., p.268-273. http://dx.doi.org/10.1109/DAC.1999.781324

Haridass, K., Valenzuela, J., Yucekaya, A.D., *et al.*, 2014. Scheduling a log transport system using simulated annealing. *Inform. Sci.*, **264**:302-316. http://dx.doi.org/10.1016/j.ins.2013.12.005

Heller, W.R., Maling, K., Sorkin, G., 1982. The planar package for system designers. Proc. Design Automation Conf., p.253-260. http://dx.doi.org/10.1109/DAC.1982.1585509

Hoo, C.S., Kanesan, J., Ramiah, H., 2014. Enumeration technique in very large-scale integration fixed-outline floorplanning. *IET Circ. Dev. Syst.*, **8**(1):47-57. http://dx.doi.org/10.1049/iet-cds.2013.0003

Kahng, A.B., 2000. Classical floorplanning harmful? Proc. Int. Symp. on Physical Design, p.207-213. http://dx.doi.org/10.1145/332357.332401

Kahng, A.B., Markov, I., 2007. GSRC Floorplan Benchmark. Available from http://vlsicad.eecs.umich.edu/BK/GSRCbench/HARD/.

Kennedy, J., Eberhart, R.C., 1995. Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks, p.1942-1948. http://dx.doi.org/10.1109/ICNN.1995.488968

Khan, A.K., Vatsa, R., Roy, S., *et al.*, 2014. A new efficient topological structure for floorplanning in 3D VLSI physical design. IEEE Int. Advance Computing Conf., p.696-701. http://dx.doi.org/10.1109/IAdCC.2014.6779409

Kirpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. *Science*, **220**(4598):671-680. http://dx.doi.org/10.1126/science.220.4598.671

Lin, J.M., Chang, Y.W., 2001. TCG: a transitive closure graph-based representation for non-slicing floorplans. Proc. Design Automation Conf., p.764-769. http://dx.doi.org/10.1145/378239.379062

Lin, J.M., Hung, Z.X., 2012. SKB-tree: a fixed-outline driven representation for modern floorplanning problems. *IEEE Trans. VLSI Syst.*, **20**(3):473-484. http://dx.doi.org/10.1109/TVLSI.2011.2104983

Lin, S.W., Yu, V.F., 2015. A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows. *Appl. Soft Comput.*, **37**:632-642. http://dx.doi.org/10.1016/j.asoc.2015.08.058

Liu, R., Dong, S., Hong, X., *et al.*, 2005. Fixed-outline floorplanning with constraints through instance augmentation. IEEE Int. Symp. on Circuits and Systems, p.1883-1886. http://dx.doi.org/10.1109/ISCAS.2005.1464979

Ma, T.L., Young, E.F.Y., 2008. TCG-based multi-bend bus driven floorplanning. Design Automation Conf., p.192-197. http://dx.doi.org/10.1109/ASPDAC.2008.4483938

Maling, K., Heller, W.R., Mueller, S.H., 1982. On finding most optimal rectangular package plans. Proc. Design Automation Conf., p.663-670. http://dx.doi.org/10.1109/DAC.1982.1585567

Murata, H., Fujiyoshi, K., Nakatake, S., *et al.*, 1996. VLSI module placement based on rectangle-packing by the sequence pair. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.*, **15**(12):1518-1524. http://dx.doi.org/10.1109/43.552084

Otten, R.H.J.M., 1982. Automatic floorplan design. Proc. Design Automation Conf., p.261-267. http://dx.doi.org/10.1109/DAC.1982.1585510

Storn, R., Price, K., 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.*, **11**(4):341-359. http://dx.doi.org/10.1023/A:1008202821328

Wang, S.L., Zuo, X.Q., Liu, X.Q., *et al.*, 2015. Solving dynamic double row layout problem via combining simulated annealing and mathematical programming. *Appl. Soft Comput.*, **37**:303-310.

http://dx.doi.org/10.1016/j.asoc.2015.08.023

Wong, D.F., Liu, C.L., 1986. A new algorithm for floorplan design. Proc. Design Automation Conf., p.101-107. http://dx.doi.org/10.1109/DAC.1986.1586075

Wong, K.W.C., Young, E.F.Y., 2003. Fast buffer planning and congestion optimization in interconnect-driven floorplanning. Proc. Design Automation Conf., p.411-416. http://dx.doi.org/10.1109/ASPDAC.2003.1195050

Wu, P.H., Ho, T.Y., 2011. Thermal-aware bus-driven floorplanning. Int. Symp. on Low Power Electronics and Design, p.205-210. http://dx.doi.org/10.1109/ISLPED.2011.5993637

Xiang, H., Tang, X.P., Wong, M.D.F., 2004. Bus-driven floorplanning. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.*, **23**(11):1522-1530. http://dx.doi.org/10.1109/TCAD.2004.836728

Yan, J.T., 2006. Simultaneous wiring and buffer block planning with optimal wire-sizing for interconnect-driven floorplanning. IEEE Proc. on Computers and Digital Techniques, p.335-347. http://dx.doi.org/10.1049/ip-cdt:20060077

Yan, J.Z., Chu, C., 2010. DeFer: deferred decision making enabled fixed-outline floorplanning algorithm. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.*, **29**(3):367-381. http://dx.doi.org/10.1109/TCAD.2010.2041850