

Article

CSF: Fixed-Outline Floorplanning Based on the Conjugate Subgradient Algorithm and Assisted by Q-Learning

Xinyan Meng ^{1,†}, Huabin Cheng ^{2,†} , Yu Chen ^{1,*} , Jianguo Hu ^{3,4,*} and Ning Xu ⁵¹ School of Mathematics and Statistics, Wuhan University of Technology, Wuhan 430070, China² Department of Basic Science, Wuchang Shouyi University, Wuhan 430064, China³ Guangdong Provincial Key Laboratory of High-End Integrated Circuit Design and Integration Technology, Sun Yat-Sen University, Zhuhai 519082, China⁴ Shenzhen Research Institute of Sun Yat-Sen University, Shenzhen 528406, China⁵ School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China

* Correspondence: ychen@whut.edu.cn (Y.C.); hujguo@mail.sysu.edu.cn (J.H.)

† These authors contributed equally to this work.

Abstract

Analytical floorplanning algorithms are prone to local convergence and struggle to generate high-quality results; therefore, this paper proposes a nonsmooth analytical placement model and develops a Q-learning-assisted conjugate subgradient algorithm (CSAQ) for efficient floorplanning that addresses these issues. By integrating a population-based strategy and an adaptive step size adjustment driven by Q-learning, the CSAQ strikes a balance between exploration and exploitation to avoid suboptimal solutions in fixed-outline floorplanning scenarios. Experimental results on the MCNC and GSRC benchmarks demonstrate that the proposed CSAQ not only effectively solves global placement planning problems but also significantly outperforms existing constraint graph-based legalization methods, as well as the improved variants, in terms of the efficiency of generating legal floorplans. For hard module-only placement scenarios, it exhibits competitive performance compared to the state-of-the-art algorithms.

Keywords: fixed-outline floorplanning; nonsmooth analytic optimization; conjugate subgradient algorithm; Q-learning; very large-scale integration circuit



Academic Editor: Maciej Ławryńczuk

Received: 13 November 2025

Revised: 5 December 2025

Accepted: 8 December 2025

Published: 12 December 2025

Citation: Meng, X.; Cheng, H.; Chen, Y.; Hu, J.; Xu, N. CSF: Fixed-Outline Floorplanning Based on the Conjugate Subgradient Algorithm and Assisted by Q-Learning. *Electronics* **2025**, *14*, 4893. <https://doi.org/10.3390/electronics14244893>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Floorplanning is a key stage in the physical design flow of very large-scale integration (VLSI) circuits and printed circuit Boards (PCB) [1]. With the rapid development of integrated circuit manufacturing technology, the complexity of VLSI/PCB design has increased dramatically, and modern design usually focuses on fixed-outline floorplanning (FOFP) problems [2]. Compared to layout planning problems without a fixed outline, the FOFP problem is more challenging, due to the additional constraint of placing a large number of modules within a given outline [3]. Additionally, the usage of an intellectual property (IP) core introduces hard modules into the VLSI floorplan, making it much more challenging to optimize the fixed-outline floorplan of mixed-size modules [4].

Floorplanning algorithms generally fall into two categories: floorplanning algorithms based on the combinatorial optimization model (FA-COMs) and floorplanning algorithms based on the analytic optimization model (FA-AOMs). By coding the relative positions of modules by combinatorial data structures, FA-COMs formulate floorplanning problems as combinatorial optimization problems addressed by heuristics and metaheuristics [5–8].

Although the combinatorial codes can be naturally converted into compact floorplans complying with the nonoverlapping constraint, the combinatorial explosion results in poor performance on large-scale cases. A remedy for this is to implement multilevel methods in which clustering or partitioning strategies are introduced to reduce the scale of the problem [3,9,10]. However, multilevel strategies typically optimize the placement in a local way, which, in turn, makes it difficult to converge to the globally optimal placement of VLSI and PCB designs.

FA-AOMs address analytical floorplanning models with continuous optimization algorithms, contributing to their low complexity and fast convergence in large-scale cases [11]. A typical FA-AOM consists of two stages: the global floorplanning stage, generating promising results, and the legalization stage that refines the positions of modules to eliminate constraint violations. However, the cooperative design of global floorplanning and legalization is extremely challenging due to the multimodal nonsmooth characteristics of floorplanning models. Global floorplanning is typically implemented by addressing a smoothed optimization model with a first-order gradient-based analytical algorithm [12,13]; however, the smoothing approximation can introduce some distortion to the characteristics of the mathematical model, and the first-order analytical algorithm usually converges to a local optimal solution. Legalization can be implemented through the refinement of constraint graphs [12,13] or the second-order cone programming strategy [14], which usually leads to a relatively high complexity that hinders their application in large-scale scenarios.

To address the fixed-outline floorplanning problem efficiently, this paper proposes a fixed-outline floorplanning algorithm based on the conjugate subgradient algorithm (CSA) where Q-learning is introduced to adaptively regulate the step size of the CSA. The main contributions of this paper are as follows:

- The nonsmooth wirelength and overlapping metrics are directly optimized by the CSA. Since the CSA searches the solution space using both a deterministic gradient-based search and stochastic exploration, it is much more likely to obtain the optimal floorplan by flexibly regulating its step size.
- The CSA is implemented in a population-based search pattern equipped with a dynamical step size regulated based on Q-learning, which contributes to its fast convergence to the optimal floorplan.
- Two improved variants are proposed for the CG-based legalization algorithm [12,13]. They take less time to obtain legal floorplans with a shorter wirelength.
- We propose a CSA-based fixed-outline floorplanning (CSF) algorithm where both global floorplanning and legalization are implemented by the CSA assisted by Q-learning (CSAQ). It strikes a balance between time complexity and floorplan quality, demonstrating competitive performance among the compared algorithms.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 introduces some preliminaries. Then, the proposed algorithm developed for fixed-outline floorplanning problems is presented in Section 4. The numerical experiment is performed in Section 5 to demonstrate the competitiveness of the proposed algorithm, and, finally, this paper is concluded in Section 6.

2. Related Work

By formulating the floorplanning problem as an analytic optimization model, Zhan et al. [15] constructed an algorithm framework consisting of a global floorplanning stage and a legalization stage which exhibited performance superior to *Parquet-4* based on sequence pairs and simulated annealing [3]. Lin and Hung [14] modeled global floorplanning as a convex optimization problem where modules are transformed into circles, and a push–pull mode was proposed with consideration of wirelength. Then,

second-order cone programming was deployed to legalize the floorplan. By incorporating the electrostatic field model proposed in *eplace* [16], a celebrated algorithm for standard cell placement of VLSI, Li et al. [13] created an analytic floorplanning algorithm for large-scale floorplanning cases. Then, a horizontal constraint graph (HCG) and a vertical constraint graph (VCG) were constructed to eliminate overlap of the floorplanning results. Huang et al. [12] further improved the electrostatics-based analytical method for fixed-outline floorplanning in which module rotation and sizing were introduced to obtain floorplanning results with a shorter wirelength. Yu et al. [17] developed a floorplanning algorithm based on the feasible-seeking approach, achieving better performance than the state-of-the-art floorplanning algorithms.

Since the typical wirelength and overlap metrics of analytical floorplanning models are not smooth, additional smoothing procedures are incorporated to make gradient-based optimization algorithms feasible. Popular smoothing approximation methods for the half-perimeter wirelength (HPWL) include the quadratic model, the logarithm-sum-exponential model, and the L_p -norm model, and the bell-shaped smoothing method can be adopted to smooth the overlap metric [18]. In addition, Ray et al. [19] proposed a nonrecursive approximation to simulate the max function for the nonsmooth HPWL model, and Chan et al. [20] employed a Helmholtz smoothing technique to approximate the density function. However, the additional smoothing process not only introduces the extra time complexity of the FA-AOM, but also leads to its local convergence to a solution significantly different from that of the original nonsmooth model. Zhu et al. [21] proposed addressing the nonsmooth analytical optimization model of cell placement with the CSA, and Sun et al. [22] further developed a CSA-based floorplanning algorithm for scenarios without soft modules.

Recently, reinforcement learning (RL) was introduced to design general floorplanning algorithms. He et al. [23] employed agents guided by the Q-learning algorithm to alleviate the need for sufficient prior human knowledge in the search process. Mirhoseini et al. [24] proposed an RL-based method for chip floorplanning and developed an edge-based graph convolutional neural network architecture capable of learning the transferability of chips. By merging RL with a graph convolutional network (GCN), Xu et al. [25] constructed an end-to-end learning-based floorplanning framework named *GoodFloorplan*. Yang et al. [26] proposed an end-to-end RL framework for learning a floorplanning policy, and promising results were obtained with the assistance of an edge-augmented graph attention network, a position-wise multilayer perceptron, and a gated self-attention mechanism. Moldenhauer et al. [27] developed an open-source RL environment building upon the gymnasium API and compatible with stable-baselines3. DellaRovere et al. [28] proposed a hybrid method that combines RL with beam search. By constructing a search tree to expand the state space of the RL agent and periodically pruning it, this approach improves key floorplanning metrics such as area and wirelength. The generalization ability of well-trained RL algorithms can facilitate their potential applications in diverse floorplanning scenarios, but it is challenging to achieve the fast convergence of RL-based floorplanning algorithms in the case of inadequate training data.

3. Preliminaries

3.1. The Nonsmooth Analytical Optimization Model of Floorplanning

Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and $\mathbf{y} = (y_1, y_2, \dots, y_n)$ be the vectors of the x -coordinate and y -coordinate of all modules, respectively. FOFP aims to place modules within a fixed outline such that all of them are mutually nonoverlapping and can be formulated as follows:

$$\begin{aligned} \min \quad & W(\mathbf{x}, \mathbf{y}), \\ \text{s.t.} \quad & \begin{cases} D(\mathbf{x}, \mathbf{y}) = 0, \\ B(\mathbf{x}, \mathbf{y}) = 0, \\ \mathbf{x}, \mathbf{y} \in \mathbf{R}^n, \end{cases} \end{aligned} \quad (1)$$

where $W(\mathbf{x}, \mathbf{y})$, $D(\mathbf{x}, \mathbf{y})$, and $B(\mathbf{x}, \mathbf{y})$ are the analytical metrics of wirelength, overlap, and boundary violation, respectively.

The Total Wirelength $W(\mathbf{x}, \mathbf{y})$

The total wirelength is here taken as the sum of the half-perimeter wirelength (HWPL):

$$W(\mathbf{x}, \mathbf{y}) = \sum_{e \in E} (\max_{v_i \in e} x_i - \min_{v_i \in e} x_i + \max_{v_i \in e} y_i - \min_{v_i \in e} y_i) \quad (2)$$

The Total Overlapping Area $D(\mathbf{x}, \mathbf{y})$

The sum of the overlapping area is computed by

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i,j} O_{i,j}(\mathbf{x}) \times O_{i,j}(\mathbf{y}) \quad (3)$$

where $O_{i,j}(\mathbf{x})$ and $O_{i,j}(\mathbf{y})$ represent the overlapping lengths of module i and j in the X-axis and Y-axis directions, respectively [21]. Denoting $\Delta_x(i, j) = |x_i - x_j|$, we know that

$$O_{i,j}(\mathbf{x}) = \begin{cases} \min(\hat{w}_i, \hat{w}_j), & \text{if } 0 \leq \Delta_x(i, j) \leq \frac{|\hat{w}_i - \hat{w}_j|}{2} \\ \frac{\hat{w}_i - 2\Delta_x(i, j) + \hat{w}_j}{2}, & \text{if } \frac{|\hat{w}_i - \hat{w}_j|}{2} < \Delta_x(i, j) \leq \frac{\hat{w}_i + \hat{w}_j}{2} \\ 0, & \text{if } \frac{\hat{w}_i + \hat{w}_j}{2} < \Delta_x(i, j) \end{cases} \quad (4)$$

Denoting $\Delta_y(i, j) = |y_i - y_j|$, we have

$$O_{i,j}(\mathbf{y}) = \begin{cases} \min(\hat{h}_i, \hat{h}_j), & \text{if } 0 \leq \Delta_y(i, j) \leq \frac{|\hat{h}_i - \hat{h}_j|}{2} \\ \frac{\hat{h}_i - 2\Delta_y(i, j) + \hat{h}_j}{2}, & \text{if } \frac{|\hat{h}_i - \hat{h}_j|}{2} < \Delta_y(i, j) \leq \frac{\hat{h}_i + \hat{h}_j}{2} \\ 0, & \text{if } \frac{\hat{h}_i + \hat{h}_j}{2} < \Delta_y(i, j) \end{cases} \quad (5)$$

where $\hat{w}_i$ and $\hat{h}_i$ represent the width and height of module i .

The Boundary Violation $B(\mathbf{x}, \mathbf{y})$

For the fixed-outline floorplanning problem, the position of the i th module must meet the following constraints:

$$\begin{cases} 0 \leq x_i - \hat{w}_i/2, & x_i + \hat{w}_i/2 \leq \mathbf{W}^* \\ 0 \leq y_i - \hat{h}_i/2, & y_i + \hat{h}_i/2 \leq \mathbf{H}^* \end{cases}$$

where $\mathbf{W}^*$ and $\mathbf{H}^*$ are the width and height of the fixed outline, respectively. They can be manually assigned or generated by

$$\mathbf{W}^* = \sqrt{(1 + \gamma) \cdot A/R}, \quad \mathbf{H}^* = \sqrt{(1 + \gamma) \cdot A \cdot R}, \quad (6)$$

where A is the area sum of the modules, R is the width–height ratio, and γ is the ratio of whitespace.

For the boundary violation, we individually define the overflow of each module across the four boundaries (left, right, bottom, and top) as

$$\begin{aligned} b_{1,i}(\mathbf{x}) &= \max(0, \hat{w}_i/2 - x_i), & b_{2,i}(\mathbf{x}) &= \max(0, \hat{w}_i/2 + x_i - \mathbf{W}^*), \\ b_{1,i}(\mathbf{y}) &= \max(0, \hat{h}_i/2 - y_i), & b_{2,i}(\mathbf{y}) &= \max(0, \hat{h}_i/2 + y_i - \mathbf{H}^*), \end{aligned}$$

and we define $B(\mathbf{x}, \mathbf{y})$ as

$$B(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n (b_{1,i}(\mathbf{x}) + b_{2,i}(\mathbf{x}) + b_{1,i}(\mathbf{y}) + b_{2,i}(\mathbf{y})). \quad (7)$$

Subsequently, we employ the Lagrangian multiplier method to transform (1) into an unconstrained optimization model [22]

$$\min f_g(\mathbf{x}, \mathbf{y}) = \alpha W(\mathbf{x}, \mathbf{y}) + \lambda D(\mathbf{x}, \mathbf{y}) + \mu B(\mathbf{x}, \mathbf{y}), \quad (8)$$

where α , λ , and μ are weight coefficients to be confirmed.

In the global floorplanning stage, our objective is to optimize (8) with tailored parameters α , λ , and μ . For the legalization stage, the weight α of the wirelength function is set to 0 to obtain a legal solution, resulting in

$$\min f_l(\mathbf{x}, \mathbf{y}) = \lambda D(\mathbf{x}, \mathbf{y}) + \mu B(\mathbf{x}, \mathbf{y}). \quad (9)$$

which serves as the objective function for the legalization stage. Then, the nonsmooth objective functions of (8) and (9) are addressed by the CSA [21,22].

3.2. The Conjugate Subgradient Algorithm

The pseudo-code of the CSA is presented in Algorithm 1 [21]. The iteration process of the CSA starts with an initial solution $\mathbf{u}_0$ and the subgradient $\mathbf{g}_0$, and the conjugate direction is initialized as $\mathbf{d}_0 = \mathbf{0}$. At generation k , the searching direction is determined according to the subgradient $\mathbf{g}_k$ and the conjugate direction $\mathbf{d}_k$, and the searching step size is determined by dividing a scaling factor c by the norm of $\mathbf{d}_k$. When solution $\mathbf{u}$ lies at a nonsmooth point of f , the computation of the subgradient is performed with a random scheme. Thus, the CSA is not necessarily gradient descendant. Consequently, the adaptive regulation of step size plays a critical role and has a significant influence on its convergence performance.

It was proved that the CSA presented in Algorithm 1 can converge to the neighborhood of the local optimum [21]. The CSA exhibits excellent per-iteration efficiency due to its streamlined computational design. The per-iteration cost is low, as the calculation of the L_1 -norm wirelength, the exact density function, and its subgradient are inherently simple and efficient. This approach avoids complex smoothing approximations, thereby reducing the computational overhead per iteration. Concurrently, the conjugate subgradient method requires storage of only the current and previous subgradients, resulting in a minimal memory footprint. To improve global exploration without sacrificing local exploitation, we propose to adaptively regulate the step size of the CSA based on Q-learning.

Algorithm 1 CSA**Input:** Objective function $f(\mathbf{u})$, initial solution $\mathbf{u}_0$ **Output:** Optimal solution $\mathbf{u}^*$

- 1: Set $\mathbf{g}_0 \in \partial f(\mathbf{u}_0)$, $\mathbf{d}_0 = 0$;
- 2: **for** $k = 1$ to k_{max} **do**
- 3: Calculate the subgradient $\mathbf{g}_k \in \partial f(\mathbf{u}_{k-1})$;
- 4: Calculate the Polak-Ribiere parameter $\eta_k = \frac{\mathbf{g}_k^T(\mathbf{g}_k - \mathbf{g}_{k-1})}{\|\mathbf{g}_{k-1}\|_2^2}$;
- 5: Calculate the conjugate direction $\mathbf{d}_k = -\mathbf{g}_k + \eta_k \mathbf{d}_{k-1}$;
- 6: Calculate the step size $\hat{\alpha}_k = \frac{c}{\|\mathbf{d}_k\|_2}$;
- 7: Update the solution $\mathbf{u}_k = \mathbf{u}_{k-1} + \hat{\alpha}_k \mathbf{d}_k$;
- 8: Update the optimal solution $\mathbf{u}^*$;
- 9: **end for**

3.3. Q-Learning

We propose to update the scaling parameter c in Algorithm 1, a typical value-based RL algorithm, using Q-learning [29]. It aims to find for each state–action pair (s, a) a Q-value that represents its long-term expected return. The Q-values record an individual’s learning experience, stored in a Q-table and updated by

$$Q^k(s_i, a_j) = (1 - \alpha_0)Q^{k-1}(s_i, a_j) + \alpha_0(R(s_i, a_j) + \gamma_0 Q_{max}), \quad (10)$$

where $Q^k(s_i, a_j)$ represents the Q-values of the state–action pair (s_i, a_j) at steps k . α_0 and γ_0 are the learning rate and the discount rate, respectively. $R(s_i, a_j)$ is the reward function. In this paper, Q_{max} is confirmed as

$$Q_{max} = m_0 Q^{k-1}(s_i, a_j) + \frac{1 - m_0}{p - 1} \sum_{l \neq j}^p Q_{max}^{k-1}(s_l), \quad (11)$$

where m_0 is a preset coefficient, and $Q_{max}^{k-1}(s_l)$ is the maximum Q-value across all actions of state l at iteration $k - 1$.

3.4. The Constraint Graph

The constraint graph (CG) is a directed acyclic graph (DAG) used to represent the relative position between modules in a layout. Modules are represented by vertexes of a CG, and arcs indicate the constraint relationships between modules. Accordingly, the horizontal constraint graph (HCG) and the vertical constraint graph (VCG) are constructed for a layout to indicate the left–right and down–up positional relationships between modules, respectively. If the lower-left corner coordinates of module A are less than those of module B , the arcs of the HCG and the VCG are constructed by the following proposed method [13]:

1. If module A and B do not overlap in either the horizontal or vertical direction, add arc $A \rightarrow B$ in the HCG and add arc $A \uparrow B$ in the VCG.
2. If module A and B overlap in the vertical direction but not in the horizontal direction, add arc $A \rightarrow B$ in the HCG. Add arc $A \uparrow B$ in the VCG if module A and B overlap in the horizontal direction but not in the vertical direction.
3. For the case where module A and B overlap in both the horizontal and vertical directions, add arc $A \uparrow B$ in the VCG if the overlapping length in the horizontal direction is greater than that in the vertical direction; conversely, add arc $A \rightarrow B$ in the HCG.

Given a set of modules to be placed, a CG pair (HCG, VCG) can be translated into a floorplan where all modules are mutually disjoint. Then, by constructing a CG pair (HCG, VCG) for the result of the global floorplanning and translating it into another floorplan of modules, the overlaps between modules can be eliminated and we obtain a compact floorplan for the modules to be placed.

4. The CSA-Based Fixed-Outline Floorplanning Algorithm

As presented in Section 3.1, global floorplanning and legalization can be implemented by addressing (8) and (9), respectively. Accordingly, we propose a CSA-based fixed-outline floorplanning algorithm (CSF) where the performance of the CSA is enhanced by adaptively regulating the scaling factor c of the step size via Q-learning.

4.1. The Framework of CSF

Figure 1 illustrates the overall flowchart of our algorithm. At the beginning, a solution population Pop of size p is initialized, where the orientations of modules are randomly generated and Latin hypercube sampling (LHS) is employed to generate the initial position vector $\mathbf{Ind}_i$ within the fixed outline. The characteristic of LHS is that it can make the generated samples distribute as evenly as possible in the parameter space, covering a broader solution space and promoting the likelihood of finding the global optimum. Then, the stages of global floorplanning and legalization are successively implemented. If a legal solution $\mathbf{Ind}^*$ is obtained after the legalization stage, this solution is recorded as the final solution and the iteration is terminated. Otherwise, a 90-degree rotation operation is applied to a randomly selected subset of modules within $\mathbf{Ind}_i$. The framework of the CSF is presented as Algorithm 2.

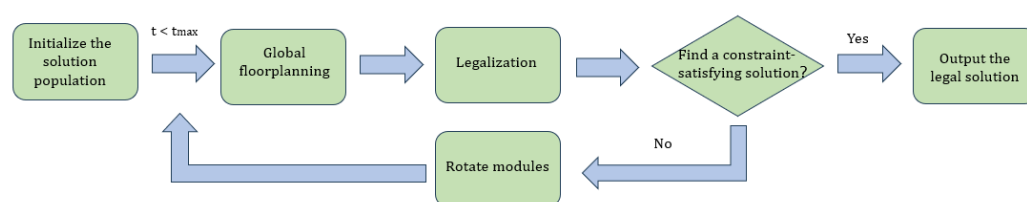


Figure 1. The flowchart of the CSF.

Algorithm 2 The Framework of CSF

Input: Iteration budget t_{max} ;

Output: A floorplan $\mathbf{Ind}^* = (\mathbf{x}^*, \mathbf{y}^*)$;

- 1: Generate a population of floorplan $\mathbf{Pop} = \{\mathbf{Ind}_1, \dots, \mathbf{Ind}_p\}$, where $\mathbf{Ind}_i = (\mathbf{x}^i, \mathbf{y}^i) (i = 1, \dots, p)$ is the initial floorplan sampled within the fixed-outline by LHS;
 - 2: Set $t = 1$;
 - 3: **while** $t < t_{max}$ **do**
 - 4: $\mathbf{Pop} = \text{GFloorplan}(f_g, \mathbf{Pop})$; /*Global floorplanning*/
 - 5: Implement legalization stage to get $\mathbf{Ind}^*$; /*Legalization*/
 - 6: **if** $\mathbf{Ind}^*$ is legal **then**
 - 7: **break**;
 - 8: **end if**
 - 9: Rotate randomly selected modules for each individual $\mathbf{ind}_i (i = 1, \dots, p)$;
 - 10: $t++$;
 - 11: **end while**
-

4.2. The Conjugate Subgradient Algorithm Assisted by Q-Learning

The global floorplanning and legalization address problems (8) and (9) with the CSAQ, which employs a population-based iteration scheme with the principle of the CSA. Thanks to the gradient-like definition and the stochastic characteristics of the subgradient, the CSA incorporates both the fast convergence of gradient-based algorithms and the global convergence of metaheuristics. The parameter c of the CSA (Algorithm 1), which is updated by a Q-learning-assisted strategy, plays a crucial role in achieving a balance between exploration and exploitation [30].

For a solution population **Pop** of size p , a $p \times m$ Q-table is constructed, as shown in Table 1. In our Q-learning framework, the state is defined as the population's individuals [30], and the actions correspond to discrete values of a control parameter c . The selection of an action in each iteration directly dictates the value of c that regulates the CSA's step size. To balance the global exploration and local exploitation, we first define a valid range for c . During the global placement phase, a larger step size is desired to allow modules to move significantly from their initial positions, thereby optimizing the wirelength and ensuring strong global search capability. Based on the module dimensions in our benchmark tests, this requirement informs the upper bound of c . To enhance the local search with promising efficiency, a lower bound for c is also established. These two bounds collectively define the operational range for c throughout the iterative process.

Table 1. The Q-table for the update of parameter c .

State	Action			
	a_1	a_2	...	a_m
$s_1/\mathbf{Ind}_1$	$Q(s_1, a_1)$	$Q(s_1, a_2)$	...	$Q(s_1, a_m)$
$s_2/\mathbf{Ind}_2$	$Q(s_2, a_1)$	$Q(s_2, a_2)$	...	$Q(s_2, a_m)$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$s_p/\mathbf{Ind}_p$	$Q(s_p, a_1)$	$Q(s_p, a_2)$	...	$Q(s_p, a_m)$

Subsequently, we construct the action space by equidistantly selecting m discrete values within this interval. The selection of m necessitates a trade-off; an excessively large m would result in insufficient differentiation among actions, whereas an overly small m would lead to a lack of diversity in the action space. To strike a balance, we set $m = 5$ and $p = 5$, as suggested in [22]. Subsequently, the scaling factor c corresponding to state s_i ($\mathbf{Ind}_i$) is updated via probability distribution

$$p_i^j = \frac{Q^k(s_i, a_j)}{\sum_{l=1}^m Q^k(s_i, a_l)}, \quad j = 1, \dots, m, \quad (12)$$

where $Q^k(s_i, a_j)$ is the Q-value of state–action pair (s_i, a_j) at the k -th iteration. At each iteration, the Q-values are updated by (10), where the reward function is confirmed by

$$R(s_i, a_j) = (f_{pre} - f_{post})/r_c, \quad (13)$$

where f_{pre} and f_{post} are the pre-action value and the post-action value of the objective function f to be minimized and r_c is a preset constant.

With the probability distribution confirmed by (12), the parameter c is updated every k_t generation. Accordingly, we obtain the framework of the CSAQ as presented in Algorithm 3.

Algorithm 3 CSAQ(f, Pop)**Input:** The objective f , the population $\text{Pop} = \{\text{Ind}_i, i = 1, \dots, p\}$;**Output:** The updated solution population Pop ;

```

1: Initialize the parameters  $k_t$  and the Q-table (Table 1) uniformly;
2: for  $k = 1$  to  $k_{max}$  do
3:   for  $i = 1$  to  $p$  do
4:     if  $k == 1$  then
5:       Set  $R(s_i, a_j) = 0$  for all  $j \in \{1, \dots, m\}$ , initialize  $c_j$ ;
6:        $\mathbf{d}_{i,0} = \mathbf{0}$ ,  $\text{Ind}_i^0 = \text{Ind}_i$ ,  $\mathbf{g}_{i,0} \in \partial f(\text{Ind}_i^0)$ ;
7:     end if
8:     if  $k \pmod{k_t} = 0$  then
9:       Update  $Q^k(s_i, a_j)$  according to (10);
10:      Update  $c_i$  according to the distribution (12);
11:    end if
12:    Calculate the subgradient  $\mathbf{g}_{i,k} \in \partial f(\text{Ind}_i^{k-1})$ ;
13:    Calculate the parameter  $\eta_{i,k} = \frac{\mathbf{g}_{i,k}^T(\mathbf{g}_{i,k} - \mathbf{g}_{i,k-1})}{\|\mathbf{g}_{i,k-1}\|_2^2}$ ;
14:    Calculate the conjugate direction  $\mathbf{d}_{i,k} = -\mathbf{g}_{i,k} + \eta_{i,k}\mathbf{d}_{i,k-1}$ ;
15:    Calculate the step size  $\hat{a}_{i,k} = \frac{c_i}{\|\mathbf{d}_{i,k}\|_2}$ ;
16:    Update the solution  $\text{Ind}_i^k = \text{Ind}_i^{k-1} + \hat{a}_{i,k}\mathbf{d}_{i,k}$ ;
17:  end for
18: end for
19: Set  $\text{Pop} = \{\text{Ind}_i^{k_{max}}, i = 1, \dots, p\}$ .

```

4.3. The Global Floorplanning

Global floorplanning is implemented by addressing problem (8) using the CSAQ described by Algorithm 3. Because the weight setting of f_g depends on the case being investigated, we employ a dynamic setting of objective weight λ . As presented in Algorithm 4, the CSAQ is executed iteratively for the parameter setting $\lambda \leftarrow q\lambda$, and q is set as 1.3 in this paper. In this way, it tries to achieve a promising result by striking a balance between wirelength and constraint violation. The iteration process ceases if the total overlapping area of all individuals in the solution population is less than A/v , where A is the sum of the areas of all modules and v is a preset constant.

Algorithm 4 GFloorplan(f_g, Pop)**Input:** The objective f_g , the population $\text{Pop} = \{\text{Ind}_1, \dots, \text{Ind}_p\}$;**Output:** The updated solution population Pop^* ;

```

1: Initialize parameters  $\alpha$ ,  $\lambda$  and  $\mu$  of the objective  $f_g$  (Equation (8));
2: while the termination condition is not met do
3:    $\text{Pop} = \text{CSAQ}(f_g, \text{Pop})$ ;
4:    $\lambda \leftarrow q\lambda$ ;
5: end while
6:  $\text{Pop}^* = \{\text{ind}_1, \dots, \text{ind}_p\}$ .

```

4.4. The Legalization

Li et al. [13] proposed a tailored legalization strategy based on a constraint graph (LA-CG) which generated promising results on large-scale floorplanning cases with soft modules. However, our preliminary experiment indicates that it does not work well on small cases without soft modules. Accordingly, we propose an improved legalization algorithm based on a constraint graph (ILA-CG). Moreover, we also perform legalization by solving problem (9). The CSAQ is demonstrated to be an efficient optimizer, and we obtain an efficient legalization algorithm based on the CSAQ (LA-CSAQ).

4.4.1. The Improved Variants of Legalization Based on a Constraint Graph

The LA-CG proposed by Li et al. [13] starts with the construction of the HCG and the VCG, which are then translated into a floorplan without overlap. If all modules are placed in the constrained region, we obtain a legal floorplan; otherwise, there are some modules placed outside the fixed outline, and constraint relationships in the critical path (the so-called critical path refers to a path composed of blocks that constrain each other in the same direction and are closely arranged [31]) of a CG are selected to be adjusted to eliminate the out-of-bound violation. Since the process to eliminate a horizontal out-of-bound violation is similar to that in the vertical direction, we only present the details of horizontal legalization.

To eliminate the horizontal out-of-bound violation, the LA-CG proposes several definitions for modules and arcs in the CG (please refer to Ref. [13] for the rigorous definitions):

- For module A , the maximum adjustment range of the x-coordinate tolerated by the HCG is defined as the *horizontal slack* S_A^x ; similarly, S_A^y is the *vertical slack*, defined as the maximum adjustment range of the y-coordinate.
- If $S_A^x = S_B^x = 0$, the arc $A \rightarrow B$ (representing a relationship between A and B) in the HCG is termed as a *horizontal critical relationship*, which means the x-coordinates of modules A and B cannot be adjusted without overlapping with other modules. A *vertical critical relationship* $A \uparrow B$ in the VCG can be defined similarly. The arcs in the critical path of an HCG/VCG are a critical relationship.
- For a horizontal critical relationship $A \rightarrow B$, the weight is defined by

$$\text{weight}(A \rightarrow B) = \begin{cases} S_A^y - h_B & \text{if } y_A \leq y_B, \\ S_B^y - h_A & \text{otherwise.} \end{cases} \quad (14)$$

The weight for the vertical critical relationship $A \uparrow B$ can be defined similarly.

- Modules A and B have a *compressible relationship* in the horizontal/vertical direction if A or B may be moved horizontally/vertically such that the floorplan is more compact.

Then, the LA-CG modifies the HCG as follows:

1. If an arc $A \rightarrow B$ is *compressible*, it is removed from the HCG.
2. The *critical relationship* $A \rightarrow B$ with the maximum weight is removed from the HCG and a corresponding relationship is inserted into the VCG at the same time.

By modifying the critical relationship with the maximum weight, the LA-CG aims to reduce the vertical placement range to the greatest extent, employing a greedy strategy that sometimes fail to achieve the optimal modification in terms of wirelength minimization. The modification of the VCG is performed similarly.

Although modification of the critical relationship with the maximum weight can reduce the vertical placement range to the greatest extent, it is a greedy strategy that sometimes does not achieve the best modification in terms of wirelength minimization. To remedy the deficiency of the LA-CG, we propose an improved legalization strategy in the x-direction (named ILG_x), presented in Algorithm 5, and ILG_y , for legalization in the y-direction, is implemented similarly.

According to the weight values quantified by (14), the proposed ILG_x obtains k candidate relationships by selecting critical relationships with the top k weight values. Then, k candidate relationships are sorted in descending order by the potential wirelength increment, and we obtain a probability distribution $\{Pw_i, i = 1, \dots, k\}$, where $Pw_i > Pw_{i+1}$ for all $i \in \{1, \dots, k-1\}$. The distribution plays a significant role in the performance of the CG-based legalization, and we perform a numerical study in Section 5.2.

Algorithm 5 $ILG_x(W^*, HCG, VCG)$ **Input:** Width W^* of the fixed outline, the CG pair (HCG, VCG) ;**Output:** (HCG, VCG) after horizontal legalization.

```

1: Calculate the width  $W_d$  of the current floorplan and locate all the critical relationships in the
    $HCG$ ;
2: while  $W_d > W^*$  do
3:   if there exists a compressible  $A \rightarrow B$  then
4:     Remove  $A \rightarrow B$  from the  $HCG$ ;
5:   else
6:     Calculate the weights of all critical relationships by (14);
7:     Select critical relationships  $\{A_i \rightarrow B_i, i = 1, \dots, k\}$  with  $k$  greatest weights;
8:     Determine the selection probability  $Pw_i$  for  $A_i \rightarrow B_i$  ( $i = 1, \dots, k$ );
9:     Set  $A \rightarrow B$  as a relationship selected from  $\{A_i \rightarrow B_i, i = 1, \dots, k\}$  according to the
       distribution  $\{Pw_i, i = 1, \dots, k\}$ ;
10:    Remove  $A \rightarrow B$  from the  $HCG$ ;
11:   end if
12:   if  $y_A \leq y_B$  then
13:     Insert  $A \uparrow B$  to the  $VCG$ ;
14:   else
15:     Insert  $B \uparrow A$  to the  $VCG$ ;
16:   end if
17:   Calculate the width  $W_d$  and locate all critical relationships in the  $HCG$ ;
18: end while

```

By alternatively executing ILG_x and ILG_y , we try to obtain the improved legalization algorithm based on the constraint graph (ILA-CG) presented in Algorithm 6, trying to reduce the wirelength while legalization is implemented. The iteration ceases when a legal floorplan is obtained or the maximum number of iterations N_{max} is reached.

Algorithm 6 $ILA - CG(W, H, Pop)$ **Input:** The population $Pop = (Ind_1, \dots, Ind_p)$, width W^* and height H^* of the fixed outline;**Output:** The post-legalization solution/floorplan $Ind^* = (x^*, y^*)$;

```

1: for  $i = 1, \dots, p$  do
2:   for  $k = 1$  to  $N_{max}$  do
3:     Generate  $HCG_i$  and  $VCG_i$  according to  $Ind_i$ ;
4:     Update  $Ind_i$  according to the  $HCG_i$  and the  $VCG_i$ ;
5:      $(HCG'_i, VCG'_i) = ILG_x(W^*, HCG_i, VCG_i)$ ; /*Horizontal legalization*/
6:      $(HCG''_i, VCG''_i) = ILG_y(H^*, HCG'_i, VCG'_i)$ ; /*Vertical legalization*/
7:     Update  $Ind_i$  according to the  $HCG''_i$  and the  $VCG''_i$ ;
8:     Calculate the width  $W_d$  and height  $H_d$  of the current floorplan;
9:     if  $W_d \leq W^*$  and  $H_d \leq H^*$  then
10:       $Ind^* = Ind_i$ ;
11:      return;
12:     end if
13:   end for
14: end for

```

4.4.2. The Legalization Algorithm Based on CSAQ

Since the ILA-CG does not explicitly optimize the wirelength, it sometimes generates a legalized floorplan with a poor wirelength metric. Accordingly, Sun et al. [22] proposed to legalize a floorplan by directly optimizing (9) via the CSA. However, the individual-based CSA with constant c can be further improved. Accordingly, we propose a population-based CSA to legalize floorplans. The CSAQ used in the legalization stage is similar to that deployed in the global floorplanning stage, while the iteration process is terminated when either the maximum number of iterations is reached or a legal individual (floorplan) is obtained. As presented in Algorithm 7, the optimization of problem (9) sometimes produces

an floorplan that is not compact, which incurs unnecessary wirelength costs. Accordingly, the CG pair (HCG, VCG) is constructed to update the legalized floorplan to ensure the modules are compactly arranged (Figure 2).



(a) The floorplan generated by the CSAQ.

(b) The floorplan compressed by the CG.

Figure 2. The floorplan compression implemented by the CG.

Algorithm 7 LA – CSAQ(f_l , Pop)

Input: The population $\text{Pop} = (\text{Ind}_1, \dots, \text{Ind}_p)$;

Output: The post-legalization solution/floorplan $\text{Ind}^* = (x^*, y^*)$;

- 1: Initialize parameters λ and μ of the objective f_l , the parameter k_t , initialize the Q-table (Table 1) uniformly;
 - 2: **for** $k = 1$ to $k_{\max}$ **do**
 - 3: **for** $j = 1$ to p **do**
 - 4: **if** $k == 1$ **then**
 - 5: Set $R(s_j, a_i) = 0$ for all $i \in \{1, \dots, m\}$, initialize c_j ;
 - 6: $\mathbf{d}_{j,0} = \mathbf{0}$, $\text{Ind}_j^0 = \text{Ind}_j$, $\mathbf{g}_{j,0} \in \partial f_l(\text{Ind}_j^0)$;
 - 7: **end if**
 - 8: **if** $(k \bmod k_t) = 0$ **then**
 - 9: Update $Q^k(s_j, a_i)$ according to (10);
 - 10: Update c_j according to (12);
 - 11: **end if**
 - 12: Calculate the subgradient $\mathbf{g}_{j,k} \in \partial f_l(\text{Ind}_j^{k-1})$;
 - 13: Calculate the parameter $\eta_{j,k} = \frac{\mathbf{g}_{j,k}^T (\mathbf{g}_{j,k} - \mathbf{g}_{j,k-1})}{\|\mathbf{g}_{j,k-1}\|_2^2}$;
 - 14: Calculate the conjugate direction $\mathbf{d}_{j,k} = -\mathbf{g}_{j,k} + \eta_{j,k} \mathbf{d}_{j,k-1}$;
 - 15: Calculate the step size $\hat{a}_{j,k} = \frac{c_j}{\|\mathbf{d}_{j,k}\|_2}$;
 - 16: Update the solution $\text{Ind}_j^k = \text{Ind}_j^{k-1} + \hat{a}_{j,k} \mathbf{d}_{j,k}$;
 - 17: **if** $f_l(\text{Ind}_j^k) = 0$ **then**
 - 18: $\text{Ind}^* = \text{Ind}_j^k$;
 - 19: **return**;
 - 20: **end if**
 - 21: **end for**
 - 22: **end for**
 - 23: Update Ind^* by the constraint graphs to get tight floorplans.
-

5. Experimental Results

To verify the competitiveness of the CSF, we perform numerical experiments on the GSRC and MCNC benchmarks, where the I/O pads are fixed at the given coordinates for all of the test circuits and all modules are set as hard modules. The characteristics of the benchmarks are presented in Table 2. The compared algorithms are implemented in

C++ and run on a 64-bit Windows laptop with 2.4 GHz Intel Core i7-13700H and 16 GB of memory.

Table 2. Characteristics of the benchmarks.

Benchmark	Instance	# Modules	# IO Pins	# Pins	# Nets	Die Size (μm^2)	
						s1	s2
MCNC	ami33	33	42	480	123	1153.22×1153.22	2058×1463
	ami49	49	22	931	408	6384.53×6384.53	7672×7840
GSRC	n100	100	334	1873	885	454.341×454.341	800×800
	n200	200	564	3599	1585	449.5×449.5	800×800
	n300	300	569	4358	1893	560.487×560.487	800×800

As presented in Table 2, the fixed-outline floorplanning is performed within two different settings of die size. For setting s1, the die sizes are generated by (6) with $R = 1$ and $\gamma = 15\%$. The size setting s2 is cited from Ref. [17] to validate the wirelength minimization capability of the CSF. Via numerical experiments, we first show how the incorporated Q-learning enhances the performance of the CSF and then compare it with several selected floorplanning algorithms. The comparison is performed for the cases without soft modules to highlight how the CSF efficiently addresses the great challenge of minimizing the wirelength within a fixed outline for the scenarios containing only hard modules.

In the proposed CSAQ, the Q-learning strategy is introduced to automatically regulate the scale factor c of the CSA. Influenced by module sizes and the primary objectives of different stages, we specifically design the following distinct action groups for updating the c across various benchmarks:

- For the global floorplanning stage, the following apply:
 - On GSRC benchmarks, the action groups are [8, 12, 15, 20, 25];
 - For *ami33*, the action groups are [80, 100, 120, 140, 160];
 - For *ami49*, the action groups are [130, 180, 220, 270, 330].
- For the legalization stage, the following apply:
 - On GSRC benchmarks, the action groups are [0.1, 0.8, 5, 10, 20];
 - For *ami33*, the action groups are [1, 8, 30, 60, 90];
 - For *ami49*, the action groups are [10, 50, 100, 150, 200].

5.1. The Positive Effects of Q-Learning

The positive effect of Q-learning is validated by comparing the performance of different global floorplanning (GP) and legalization (LA) strategies, where the α_0, γ_0, m_0 of (10) and (11) are 0.4, 0.8, and 0.6 respectively. The weights α, λ, μ of (8) and (9), as well as the parameters k_t, k_{max}, c_0 of the CSA and CSAQ, are presented in Table 3. The fixed-outline die sizes are set as shown in s1 of Table 2.

This study involves multiparameter collaborative optimization and employs a systematic parameter calibration strategy. First, reasonable ranges for parameters are preliminarily delineated based on their functional roles, then a rigorous sensitivity analysis is performed through a single-factor experimental design. While ensuring the stability and generalization performance of parameter combinations, the optimal configuration is finally determined. Given the extensive nature of the complete sensitivity analysis, we present a representative example—taking the average wirelength and computational time from 30 independent runs of the CSF-qq algorithm on the n100 dataset (Figure 3)—to demonstrate the impact of varying core parameters (e.g., $k_{max}, \alpha, \lambda, \mu$) on performance. The calibration of the remaining parameters can be conducted following this methodology.

Table 3. Parameter settings of the CSA and CSAQ in different stages.

Stage	Optimizer	Benchmark	k_t	k_{max}	α	λ	μ	c_0	v
GP	CSA	MCNC	-	50	1	20	10	1000	10
		GSRC	-	50	1	20	100	100	100
	CSAQ	MCNC	10	50	1	20	10	330	10
		GSRC	40	200	1	20	100	25	100
LA	CSA	MCNC	-	2000	-	1	10	500	-
		GSRC	-	2000	-	1	10	50	-
	CSAQ	MCNC	100	2000	-	1	10	100	-
		GSRC	100	5000	-	1	10	10	-

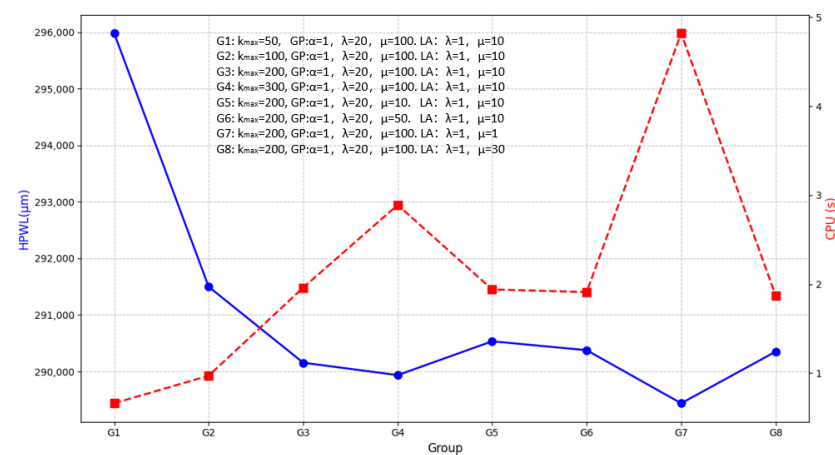
**Figure 3.** Performance comparison among different parameter groups.

Table 4 presents the average wirelength (HPWL) and CPU time obtained from 30 independent runs of the CSF-qq algorithm under different parameter settings: learning rate α_0 , discount factor γ_0 , and population size p . The parameters $\alpha_0(0.4)$, $\gamma_0(0.8)$, $p(5)$ are used for our primary experiments, while all other settings serve as control cases. The values in parentheses indicate the specific value for each parameter.

Table 4. Robustness analysis of parameters α_0 , γ_0 , p .

Parameter	$\alpha_0(0.4), \gamma_0(0.8), p(5)$		$\alpha_0(0.2), \gamma_0(0.8), p(5)$		$\alpha_0(0.6), \gamma_0(0.8), p(5)$		$\alpha_0(0.4), \gamma_0(0.6), p(5)$		$\alpha_0(0.4), \gamma_0(0.9), p(5)$		$\alpha_0(0.4), \gamma_0(0.8), p(3)$	
Circuit	HPWL (μm)	CPU (s)	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU
ami33	84,144	1.03	84,244	1.27	83,922	1.03	85,771	0.99	84,115	1.14	86,110	1.28
ami49	907,992	0.58	922,527	0.57	934,403	0.52	921,425	0.62	906,652	0.59	932,168	0.67
n100	290,156	1.96	290,194	1.83	290,824	1.99	290,354	1.94	290,304	2.02	291,159	1.48
n200	519,790	5.35	519,947	5.06	519,916	5.10	519,829	5.15	520,160	5.28	520,282	3.38
n300	589,641	12.10	590,389	12.70	589,770	12.22	590,494	11.96	590,625	11.88	591,095	7.22

The results demonstrate that, when α_0 and γ_0 are varied, the fluctuation in the HPWL is mostly within a 1% range. The variation in CPU time is also within a reasonable margin of stochastic fluctuation, indicating that our algorithm is highly robust to changes in these two parameters. In contrast, varying the population size p to 3 results in more significant changes in both the HPWL and CPU time. This is because a smaller population size reduces population diversity, which naturally leads to faster execution at the cost of inferior HPWL optimization.

5.1.1. The Performance of CSAQ-Based Global Floorplanning

To validate the promising effect of the CSAQ at the global floorplanning stage, we compare two CSF variants: (1) the CSF-cc implementing both global floorplanning and legalization via the CSA, (2) the CSF-qc implementing global floorplanning and legalization via the CSAQ and CSA. Due to the random characteristics of the CSA, the performances are compared using the averaged results of 30 independent runs. The average runtime of a single run of global floorplanning (t_g), the average runtime of a single run of legalization (t_l), the average runtime to complete the entire floorplanning process (t_w), the success rate (SR), the average wirelength (HPWL), and the improvement ratio (IR) of the HPWL are included in Table 5. Moreover, the one-sided Wilcoxon rank-sum test is performed based on the wirelength data of 30 independent runs. The p -values and the test results (R) with a significance level of 0.05 are also shown in Table 5, where “+” indicates that CSF-qc significantly outperforms CSF-cc and “~” means that the difference is not significant. The null hypothesis H_0 for the test posits that the distribution of CSF-qc is stochastically greater than or equal to that of CSF-cc. A p -value less than the significance level of 0.05 leads to the rejection of H_0 , indicating that the distribution of CSF-qc is statistically significantly smaller than that of CSF-cc.

Table 5. Performance comparison of CSA- and CSAQ-based global floorplanning.

Circuit	Method	Runtime			SR (%)	HPWL (μm)	IR (%)	p -Value	R
		t_g (s)	t_l (s)	t_w (s)					
ami33	CSF-cc	0.11	0.05	2.02	43.3	132,336	−1.34	0.21	~
	CSF-qc	0.32	0.06	1.40	73.3	134,142			
ami49	CSF-cc	0.03	0.06	2.10	86.7	904,333	−1.30	0.19	~
	CSF-qc	0.11	0.05	0.45	100	916,218			
n100	CSF-cc	0.35	0.11	0.68	100	293,202	0.14	0.20	~
	CSF-qc	1.23	0.11	1.42	100	292,780			
n200	CSF-cc	1.29	0.31	1.67	100	524,905	0.32	0.01	+
	CSF-qc	4.25	0.30	4.70	100	523,229			
n300	CSF-cc	2.07	0.48	2.66	100	595,735	0.33	0.10	~
	CSF-qc	11.11	0.46	11.68	100	593,784			

Bold texts highlight the better results of t_w , SR and HPWL.

Compared to CSF-cc, CSF-qc demonstrates an enhanced success rate at the cost of a marginally longer t_g . However, the runtime of legalization t_l is shorter, which indicates that the better floorplanning results obtained by the CSAQ can even shorten the runtime of the legalization implemented by the CSA. Consequently, the CSF-qc can compete with CSF-cc in terms of averaged wirelength for most of the test benchmarks. However, the CSAQ-assisted GP alone cannot improve the floorplanning results to a large extent, and the hypothesis test shows that CSF-qc only outperforms CSF-cc significantly on the *n200* case.

5.1.2. The Superiority of CSAQ-Based Legalization to CSA-Based Legalization

The CSF-qc is further compared with the CSF-qq, where both global floorplanning and legalization are implemented by the CSAQ. The results are presented in Table 6, and the same hypothesis testing procedure is applied.

By employing the Q-learning strategy, the CSAQ leads to significantly better legalization results at the expense of slightly increased runtime. The statistical test indicates that CSF-qq outperforms CSF-qc on four benchmark circuits. Although the test does not

demonstrate significant superiority of CSF-qq in the *ami49* case, it still beats CSF-qc in terms of the average wirelength.

Table 6. Performance comparison between CSA- and CSAQ-based legalization.

Circuit	Method	Runtime			SR (%)	HPWL (μm)	IR (%)	<i>p</i> -Value	R
		t_g (s)	t_l (s)	t_w (s)					
ami33	CSF-qc	0.32	0.06	1.40	73.3	134,142	37.27	9.29×10^{-10}	+
	CSF-qq	0.31	0.13	1.03	93.3	84,144			
ami49	CSF-qc	0.11	0.05	0.45	100	916,218	0.90	0.22	~
	CSF-qq	0.11	0.13	0.58	100	907,992			
n100	CSF-qc	1.23	0.11	1.42	100	292,780	0.90	4.61×10^{-5}	+
	CSF-qq	1.27	0.31	1.96	100	290,156			
n200	CSF-qc	4.25	0.30	4.70	100	523,229	0.66	2.05×10^{-6}	+
	CSF-qq	4.18	0.70	5.35	100	519,790			
n300	CSF-qc	11.11	0.46	11.68	100	593,784	0.70	0.01	+
	CSF-qq	11.31	0.74	12.05	100	589,641			

Bold texts highlight the better results of t_w , SR and HPWL. “+” indicates that CSF-qq significantly outperforms CSF-qc and “~” means that the difference is not significant.

5.2. Comparison Between CG- and CSAQ-Based Legalization Algorithms

To further confirm the superiority of the CSAQ-based legalization algorithm, we compare the LA-CSAQ (Algorithm 7) with the LA-CG [13] by starting the legalization with initial floorplans generated by the CSAQ-based global floorplanning algorithm *GFloorplan*(f_g, Pop) (Algorithm 4). Meanwhile, two variants of the ILA-CG (Algorithm 6) are investigated to highlight the superiority of the LA-CSAQ. Both variants of the ILA-CG set $k = 3$ with two different probability distributions:

- ILA-CGm: $Pw_1 = 0.9, Pw_2 = 0.05, Pw_3 = 0.05$;
- ILA-CGs: $Pw_1 = 1, Pw_2 = 0, Pw_3 = 0$.

The statistical results of 30 independent runs are included in Table 7, where MinW and HWSd represent the wirelength of minimum value and the standard deviation, respectively.

Table 7. Performance comparison of the LA-CG and LA-CSAQ algorithms.

Circuit	Method	HPWL (μm)	MinW (μm)	CPU (s)	SR (%)	HWSd (μm)
ami33	LA-CG [13]	126,824	111,099	1.97	100	6718.66
	ILA-CGm	110,642	88,483.3	2.13	100	11,454.52
	ILA-CGs	107,601	76,477.9	2.93	100	12,191.70
	LA-CSAQ	84,144	74,550.4	1.03	93.3	4070.33
ami49	LA-CG [13]	1,583,553	1,286,850	0.59	100	152,821.63
	ILA-CGm	1,369,867	911,640	1.2	100	205,699.06
	ILA-CGs	1,406,771	1,063,960	1.41	100	169,781.00
	LA-CSAQ	907,992	823,193	0.58	100	52,178.06
n100	LA-CG [13]	386,978	363,292	28.88	93.3	10,737.92
	ILA-CGm	367,064	328,286	13.05	100	13,898.46
	ILA-CGs	377,695	324,515	16.08	100	20,007.87
	LA-CSAQ	290,156	286,120	1.96	100	2307.93

Table 7. Cont.

Circuit	Method	HPWL (μm)	MinW (μm)	CPU (s)	SR (%)	HWSD (μm)
n200	LA-CG [13]	722,184	698,852	344.81	10	20,308.29
	ILA-CGm	692,972	645,998	237.34	70	22,120.87
	ILA-CGs	679,861	649,888	211.50	50	23,953.68
	LA-CSAQ	519,790	515,707	5.35	100	2229.27
n300	LA-CG [13]	-	-	-	0	-
	ILA-CGm	-	-	-	0	-
	ILA-CGs	-	-	-	0	-
	LA-CSAQ	589,641	580,012	12.05	100	4712.07

Bold texts highlight the best results for compared algorithms.

The results show that the improved CG-based legalization algorithms (ILA-CGm and ILA-CGs) generally outperform the LA-CG in terms of both the average wirelength and the minimum wirelength. Since the ILA-CGs always selects the candidate correction relationship contributing to the smallest wirelength, it optimizes the wirelength better than the ILA-CGm. Compared with the CG-based legalization methods, including the LA-CG, ILA-CGm, and ILA-CGs, the LA-CSAQ can obtain legal floorplans with a shorter wirelength, demonstrating its superiority in optimization of wirelength.

From the perspective of the average time and the success rate, the LA-CSAQ obtains legal floorplans with the shortest runtime and the highest success rate, and the significantly smaller HWSD values demonstrate its superior robustness regarding the wirelength of the floorplan. Although both the ILA-CGm and ILA-CGs outperform the LA-CG on circuits *ami33*, *ami49*, *n100*, and *n200*, none of the CG-based legalization algorithms can eliminate the constraint violations for the *n300* circuit, which further confirms the superior performance of the LA-CSAQ in the task of floorplan legalization.

5.3. Comparison Between the CSF and the State-of-the-Art Floorplanning Algorithms

The competitiveness of the proposed CSAQ-based floorplanning algorithm is further validated by comparing CSF-qq (where both global floorplanning and legalization are addressed by the CSAQ) with some selected floorplanning algorithms, Parquet-4.5 [3], Fast-FA [8], FFA-CD [22], and Per-RMAP [17]. Among them, Parquet-4.5 and Fast-FA are based on the B*-tree coding structure of the floorplan, and the optimization of wirelength is addressed by the simulated annealing algorithm; FFA-CD utilizes the distribution evolution algorithm based on the probability model to optimize the orientation of modules and employs the CSA to optimize the module coordinates; and Per-RMAP models the floorplanning problem as a feasibility-seeking problem and then deals with the floorplanning problem by introducing a perturbed resettable strategy into the method of alternating projection.

For the fixed-outline floorplanning scenario, the introduction of a soft module is beneficial to achieve legal floorplans with a short wirelength. However, this paper is dedicated to a scenario involving fixed-outline floorplanning without soft modules. Note that the experimental data for Fast-FA and Per-RMAP are taken from Ref. [17], in which the success rates of the benchmark circuits are not presented.

5.3.1. Floorplanning Within Compact Layout Regions

We first validate the performance of CSF-qq on floorplanning scenarios with a tightly arranged layout region by comparing it with Parquet-4.5 and FFA-CD. The layout region is set as the setting *s1* presented in Table 2. The generated floorplans of CSF-qq on the GSRC

benchmarks are illustrated in Figure 4, and the statistical results of 30 independent runs are presented in Table 8, where IR represents the wirelength improvement ratio of CSF-qq compared to other methods. Here, the “-” indicates that the data are not provided in the relevant papers.

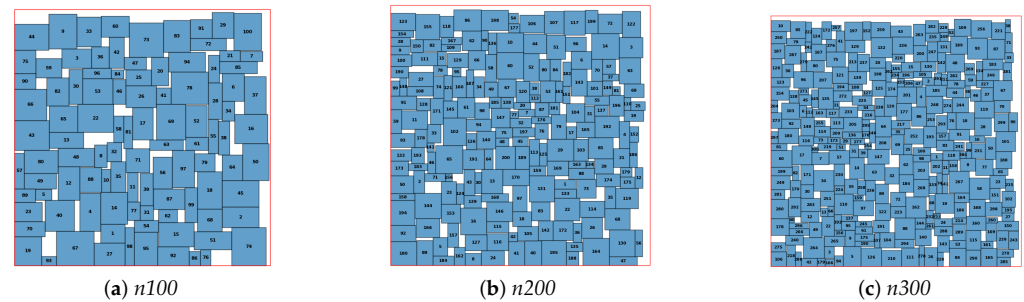


Figure 4. The floorplan compression implemented by the CG.

Table 8. Performance comparison for floorplanning restricted in layout regions with 15% whitespace.

Circuit	Method	HPWL (μm)	CPU (s)	SR (%)	IR (%)
ami33	Parquet-4.5	78,650	0.16	60	−6.53
	FFA-CD	-	-	-	-
	CSF-qq	84,144	1.03	93.3	-
ami49	Parquet-4.5	962,654	0.37	60	5.68
	FFA-CD	-	-	-	-
	CSF-qq	907,992	0.58	100	-
n100	Parquet-4.5	334,719	1.73	100	13.31
	FFA-CD	293,578	0.70	100	1.17
	CSF-qq	290,156	1.96	100	-
n200	Parquet-4.5	620,097	7.78	100	16.18
	FFA-CD	521,140	1.95	100	0.26
	CSF-qq	519,790	5.35	100	-
n300	Parquet-4.5	768,747	16.86	100	23.30
	FFA-CD	588,118	3.44	100	−0.26
	CSF-qq	589,641	12.10	100	-

Bold texts highlight the best results for compared algorithms.

The numerical results show that CSF-qq is generally competitive with the compared algorithms. Compared to Parquet-4.5, the wirelength on the GSRC benchmarks is improved by 13.31~23.39%, and the running time is reduced to a large extent. In comparison with FFA-CD, the CSF-qq generates floorplans on *n100* and *n200* with a shorter wirelength, and the wirelength of problem *n300* is −0.14% worse than that of FFA-CD. Since a population-based CSA framework assisted by Q-learning is employed by CSF-qq, its running time on GSRC benchmarks is slightly longer than that of FFA-CD.

Meanwhile, the global convergence of CSF-qq is guaranteed by the optimizer CSAQ. It can address the MCNC benchmarks better, achieving 100% success rate (SR) on *ami49*. For *ami33*, a higher success rate is achieved at the cost of a certain degree of wirelength optimization performance. In comparison, the SR of Parquet-4.5 is 60%, and there are no reference data available for FFA-CD in Ref. [22].

5.3.2. Wirelength Minimization Within Spacious Layout Regions

To validate the performance of CSF-qq on the minimization of wirelength, we compare it with two representative algorithms, Fast-SA and Per-RMAP, and the fixed outlines of layout regions are generated according to the setting *s2* presented in Table 2 [17]. As presented in Table 9, CSF-qq outperforms both Fast-SA and Per-RMAP on the GSRC benchmarks in terms of wirelength; however, it obtains a slightly longer wirelength when addressing the MCNC benchmarks. With regard to the running time, both CSF-qq and Per-RMAP beat Fast-SA markedly. Note that the results for Per-RMAP are cited from Ref. [17], where it is implemented on a server equipped with two-way Intel Xeon Gold 6248R@3.0-GHz CPUs and 768 GB DDR4-2666 MHz memory. CSF-qq is competitive with Per-RMAP in terms of running time.

Table 9. Performance comparison on [17] fixed-outline floorplanning.

Circuit	Method	HPWL (μm)	CPU (s)	SR (%)	IR (%)
ami33	Fast-SA	-	-	-	-
	Per-RMAP	63,079	0.14	-	−1.19
	CSF-qq	63,841	0.15	100	-
ami49	Fast-SA	-	-	-	-
	Per-RMAP	689,296	1.70	-	−5.02
	CSF-qq	725,744	0.27	100	-
n100	Fast-SA	287,646	10.72	-	3.79
	Per-RMAP	282,596	1.01	-	2.07
	CSF-qq	276,749	1.74	100	-
n200	Fast-SA	516,057	69.86	-	2.56
	Per-RMAP	518,722	2.93	-	3.06
	CSF-qq	502,838	3.75	100	-
n300	Fast-SA	603,811	133.63	-	4.81
	Per-RMAP	626,061	4.11	-	8.19
	CSF-qq	574,772	5.99	100	-

Bold texts highlight the best results for compared algorithms.

6. Conclusions

This paper introduces a Q-learning-accelerated conjugate subgradient algorithm (CSF-qq) for fixed-outline floorplanning for directly optimizing the primal nonsmooth model where Q-learning is employed to tune the step size parameter for improved convergence. The incorporation of population-based CSA search with adaptive regulation of step size demonstrates significant improvements in global exploration and exploitation, contributing to both fast convergence and refined optimized wirelength. Experimental results show that CSF-qq achieves competitive wirelength optimization compared to existing methods in hard module-only scenarios.

Nevertheless, the algorithm's performance is less satisfactory on the small-scale MCNC benchmark than on the GSRC, and its population-based strategy incurs a slight increase in runtime. This performance gap can be attributed to the inherent limitations of the analytical subgradient method when it is applied to the irregular problem structures presented by the MCNC benchmarks, which could be remedied by simultaneous optimization of the module orientations. Future work will focus on enhancing the algorithm through

orientation optimization, reducing its time complexity, and scaling it to larger problems and mixed-module floorplanning.

Author Contributions: Conceptualization, X.M. and Y.C.; methodology, H.C.; software, X.M.; validation, X.M. and H.C.; formal analysis, Y.C.; investigation, J.H.; resources, N.X. and J.H.; writing—original draft preparation, X.M.; writing—review and editing, H.C.; supervision, Y.C.; project administration, N.X. and J.H.; funding acquisition, J.H. and Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported in part by Shenzhen Science and Technology Program (no. JCYJ20220818102002005), in part by the Science and Technology Planning Project of Guangdong Province (no. 2024B1212020007), and in part by the Fundamental Research Funds for the Central University (no. 104972024KFYjc0055).

Data Availability Statement: The datasets analyzed during the current study are available from the corresponding author for academic research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Srinivasan, B.; Venkatesan, R.; Aljafari, B.; Kotecha, K.; Indragandhi, V.; Vairavasundaram, S. A Novel Multicriteria Optimization Technique for VLSI Floorplanning Based on Hybridized Firefly and Ant Colony Systems. *IEEE Access* **2023**, *11*, 14677–14692. [\[CrossRef\]](#)
2. Kahng, A.B. Classical Floorplanning Harmful? In Proceedings of the 2020 International Conference on Computer-Aided Design, Virtual, 2–5 November 2020; pp. 207–213.
3. Adya, S.N.; Markov, I.L. Fixed-Outline Floorplanning: Enabling Hierarchical Design. *IEEE Trans. VLSI Syst.* **2003**, *11*, 1120–1135. [\[CrossRef\]](#)
4. Weng, Y.; Chen, Z.; Chen, J.; Zhu, W. A Modified Multi-Objective Simulated Annealing Algorithm for Fixed-Outline Floorplanning. In Proceedings of the 2018 IEEE International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), Shenyang, China, 16–18 November 2018; pp. 35–39. [\[CrossRef\]](#)
5. Zou, D.; Wang, G.; Pan, G.; Qi, H. A Modified Simulated Annealing Algorithm and an Excessive Area Model for Floorplanning Using Fixed-outline Constraints. *Front. Inf. Technol. Electron. Eng.* **2016**, *17*, 1228–1244. [\[CrossRef\]](#)
6. Ye, Y.; Yin, X.; Chen, Z.; Hong, Z.; Fan, X.; Dong, C. A Novel Method on Discrete Particle Swarm Optimization for Fixed-Outline Floorplanning. In Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIS), Dalian, China, 20–22 March 2020; pp. 591–595. [\[CrossRef\]](#)
7. Chen, J.; Zhu, W. A Hybrid Genetic Algorithm for VLSI Floorplanning. In Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, Xiamen, China, 29–31 October 2010; pp. 128–132. [\[CrossRef\]](#)
8. Chen, T.-C.; Chang, Y.-W. Modern Floorplanning Based on B*-Tree and Fast Simulated Annealing. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2006**, *25*, 637–650. [\[CrossRef\]](#)
9. Ji, P.; He, K.; Wang, Z.; Jin, Y.; Wu, J. A Quasi-Newton-based Floorplanner for Fixed-Outline Floorplanning. *Comput. Oper. Res.* **2021**, *129*, 105225. [\[CrossRef\]](#)
10. Yan, J.Z.; Chu, C. DeFer: Deferred Decision Making Enabled Fixed-Outline Floorplanning Algorithm. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2010**, *29*, 367–381. [\[CrossRef\]](#)
11. Huang, Z.; Li, X.; Zhu, W. Optimization Models and Algorithms for Placement of Very Large Scale Integrated Circuits. *Oper. Res. Trans.* **2021**, *25*, 15–36. [\[CrossRef\]](#)
12. Huang, F.; Liu, D.; Li, X.; Yu, B.; Zhu, W. Handling Orientation and Aspect Ratio of Modules in Electrostatics-Based Large Scale Fixed-Outline Floorplanning. In Proceedings of the 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD), San Francisco, CA, USA, 28 October 2023–2 November 2023; pp. 1–9. [\[CrossRef\]](#)
13. Li, X.; Peng, K.; Huang, F.; Zhu, W. PeF: Poisson's Equation-Based Large-Scale Fixed-Outline Floorplanning. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *42*, 2002–2015. [\[CrossRef\]](#)
14. Lin, J.-M.; Hung, Z.-X. UFO: Unified Convex Optimization Algorithms for Fixed-Outline Floorplanning Considering Pre-Placed Modules. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2011**, *30*, 1034–1044. [\[CrossRef\]](#)
15. Zhan, Y.; Feng, Y.; Sapatnekar, S.S. A Fixed-Die Floorplanning Algorithm Using an Analytical Approach. In Proceedings of the 2006 Asia and South Pacific Conference on Design Automation, Yokohama, Japan, 24–27 January 2006; p. 6. [\[CrossRef\]](#)
16. Lu, J.; Chen, P.; Chang, C.-C.; Sha, L.; Huang, D.J.-H.; Teng, C.-C.; Cheng, C.-K. ePlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method. *ACM Trans. Des. Autom. Electron. Syst.* **2015**, *20*, 17. [\[CrossRef\]](#)

17. Yu, S.; Censor, Y.; Luo, G. Floorplanning with I/O Assignment via Feasibility-Seeking and Superiorization Methods. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2025**, *44*, 317–330. [[CrossRef](#)] [[PubMed](#)]
18. Chang, Y.-W.; Jiang, Z.-W.; Chen, T.-C. Essential Issues in Analytical Placement Algorithms. *IPSJ Trans. Syst. LSI Des. Methodol.* **2009**, *2*, 145–166. [[CrossRef](#)]
19. Ray, B.N.B.; Balachandran, S. An Efficient Wirelength Model for Analytical Placement. In Proceedings of the 2013 Design, Automation & Test in Europe Conference & Exhibition, Melbourne, Australia, 18–22 March 2013; pp. 1711–1714. [[CrossRef](#)]
20. Chan, T.F.; Cong, J.; Shinnerl, J.R.; Sze, K.; Xie, M. mPL6: Enhanced Multilevel Mixed-Size Placement. In Proceedings of the 2006 International Symposium on Physical Design, San Jose, CA, USA, 9–12 April 2006; pp. 212–214.
21. Zhu, W.; Chen, J.; Peng, Z.; Fan, G. Nonsmooth Optimization Method for VLSI Global Placement. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 642–655. [[CrossRef](#)]
22. Sun, J.; Cheng, H.; Wu, J.; Zhu, Z.; Chen, Y. Floorplanning of VLSI by Mixed-Variable Optimization. In *Intelligence Computation and Applications*; Springer Nature: Singapore, 2024; pp. 137–151.
23. He, Z.; Ma, Y.; Zhang, L.; Liao, P.; Wong, N.; Yu, B.; Wong, M.D.F. Learn to Floorplan Through Acquisition of Effective Local Search Heuristics. In Proceedings of the 2020 IEEE 38th International Conference on Computer Design (ICCD), Hartford, CT, USA, 18–21 October 2020; pp. 324–331. [[CrossRef](#)]
24. Mirhoseini, A.; Goldie, A.; Yazgan, M.; Jiang, J.W.; Songhori, E.; Wang, S.; Lee, Y.-J.; Johnson, E.; Pathak, O.; Nova, A.; et al. A Graph Placement Methodology for Fast Chip Design. *Nature* **2021**, *594*, 207–212. [[CrossRef](#)] [[PubMed](#)]
25. Xu, Q.; Geng, H.; Chen, S.; Yuan, B.; Zhuo, C.; Kang, Y.; Wen, X. GoodFloorplan: Graph Convolutional Network and Reinforcement Learning-Based Floorplanning. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *41*, 3492–3502. [[CrossRef](#)]
26. Yang, B.; Xu, Q.; Geng, H.; Chen, S.; Kang, Y. Miracle: Multi-Action Reinforcement Learning-Based Chip Floorplanning Reasoner. In Proceedings of the 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE), Valencia, Spain, 25–27 March 2024; pp. 1–6. [[CrossRef](#)]
27. Moldenhauer, T.; Uhlmann, Y.; Scheible, J. An Open-Source Environment for Evaluating Reinforcement Learning Algorithms for Analog IC Floorplanning. In Proceedings of the 2025 21st International Conference on Synthesis, Modeling, Analysis and Simulation Methods, and Applications to Circuits Design (SMACD), Istanbul, Türkiye, 7–10 July 2025; pp. 1–4. [[CrossRef](#)]
28. Rovere, S.J.D.; Basso, D.; Bortolussi, L.; Videnovic-Misic, M.; Habal, H. Enhancing Reinforcement Learning for the Floorplanning of Analog ICs with Beam Search. In Proceedings of the 2025 21st International Conference on Synthesis, Modeling, Analysis and Simulation Methods, and Applications to Circuits Design (SMACD), Istanbul, Türkiye, 7–10 July 2025; pp. 1–4. [[CrossRef](#)]
29. Watkins, C.J.C.H.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [[CrossRef](#)]
30. Li, H.; Gao, K.; Duan, P.-Y.; Li, J.-Q.; Zhang, L. An Improved Artificial Bee Colony Algorithm with Q-Learning for Solving Permutation Flow-Shop Scheduling Problems. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 2684–2693. [[CrossRef](#)]
31. Moffitt, M.D.; Ng, A.N.; Markov, I.L.; Pollack, M.E. Constraint-driven Floorplan Repair. In Proceedings of the 2006 43rd Annual Design Automation Conference, San Francisco, CA, USA, 24–28 July 2006; pp. 1103–1108. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.