# Net Resource Allocation: A Desirable Initial Routing Step

Zhisheng Zeng
SKLP, Institute of Computing Technology, CAS
Peng Cheng Laboratory
University of Chinese Academy of Sciences

Jikang Liu
Shenzhen University

Zhipeng Huang
Peng Cheng Laboratory

Ye Cai
Shenzhen University

Biwei Xie
Institute of Computing Technology, CAS
Peng Cheng Laboratory

Yungang Bao
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences

Xingquan Li*
Peng Cheng Laboratory

## ABSTRACT

In modern IC design, routing significantly impacts chip performance, power, area, and design iteration count. Critical challenges in routing include generating a rectilinear Steiner minimum tree (RSMT) for each net and handling routing resources among nets. Due to limited resources and net order, congestion is inevitable in VLSI circuit routing. Most competitive routers address congestion after routing without prior net guidance, leading to difficulty in managing resources among nets. We suggest introducing a net resource allocation step to tackle routing and congestion as a potentially desirable initial routing stage. Firstly, we introduce the net region probability density (NRPD) concept to achieve suitable net resource allocation. Using a prior NRPD, we model the resource allocation problem as linear programming (LP). We solve the LP problem and obtain a posterior NRPD for each net on each grid. Based on the posterior NRPD and congestion map, we introduce a cost scheme to guide net routing. This cost scheme supports a weighted RSMT construction technique for better topological solutions. We propose an iterative method for global routing and track assignment, improving detailed routing quality and optimizing design rule violations. Experimental results show the effectiveness of net resource allocation and demonstrate the superior performance of our router over OpenROAD's router across multiple metrics.

## KEYWORDS

Net resource allocation, global routing, detailed routing, linear programming, rectilinear Steiner tree

## 1 INTRODUCTION

With the continuous reduction of transistor size and increase of the number of gates, designing chips becomes more and more difficult [1]. For a modern chip design, the number of nets is generally greater than the number of gates. Chip design automation tools play a crucial role, especially for routing. The solution space of routing depends not only on the number of nets but also on the availability of routing tracks. For the routing problem, obtaining an exact solution is extremely difficult, even for an approximate solution. To conveniently solve the complex routing problem, designers deal with the routing problem in three steps: global routing, track assignment and detailed routing.

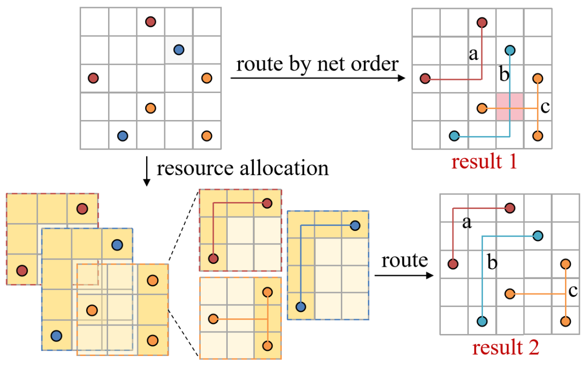


**Figure 1: Net order.**

For a given placement result, global routing partitions the routing region into a grid map and generates routing results for all nets in grid units. Several effective approaches have been proposed to address congestion among nets in global routing. These works can be summarized into two categories. 1) Some works([2], [3], [4], [5]) first route nets one by one, and then they use various techniques to relieve congestion among nets. 2) Some works ([6], [7], [8]) obtain candidate routing solutions through the form of integer linear programming or multi-commodity flow. Track assignment is used to bridge the gap between global routing and detailed routing. [9] proposed a track assignment algorithm based on Weighted Bipartite Matching with integrated heuristics to optimize the solution. TraPL [10] proposes a net-by-net track assignment algorithm that considers both local nets and the interconnectivity between nets. TRADER [11] presents a practical track-assignment-based detailed router to deal with the most representative design rules in modern designs. Detailed routing optimizes DRC violations based on the results of global routing through techniques such as ripup and rerouting. RegularRoute [12] encourages regular routing patterns and considers design rules through the maximum independent set. [13] presents a highly effective detailed routing algorithm for modern circuit designs, including real industrial constraints. TritonRoute-WXL [14] performs parallel routing after design partitioning and uses multiple iterations to converge the number of DRC violations. Dr.CU [15] proposes a detailed router that judiciously handles hard-to-access pins and new design rules.

Routing poses two crucial challenges: generating a desirable rectilinear Steiner minimum tree for each net and handling congestion among the nets during routing. For RSMT generation, since the number of pins in a net is generally not too many, there are some effective techniques. However, with the increasing number of nets, properly dealing with routing resources among nets is very hard under limited resources. Most competitive routers route nets one by one. The quality of the solution relies heavily on the order of nets, and congestion is hard to avoid.

*Corresponding author: Xingquan Li (lixq01@pcl.ac.cn)

*DAC '24, June 23–27, 2024, San Francisco, CA, USA*
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3658463

To reduce the congestion, they typically perform repair operations after routing. If we can provide prior guidance for each net before routing, then we can achieve an ideal routing resource coordination among nets. As shown in Fig. 1, if we route nets in the order of $a$, $b$, $c$, then we may obtain a routing result (result 1) in which congestion occurs between $b$ and $c$. Moreover, if we conduct a resource allocation operation before routing, we can achieve a desirable routing result (result 2).

To address the problem of handling routing congestion, we propose resource allocation as the initial routing step. Resource allocation assigns a net region probability density (NRPD) for each net. After that, we will combine the resource allocation result with the congestion map to guide the routing. Finally, through track assignment and detailed routing, the final routing result is obtained. Our main contributions to this work are listed as follows:

- To tackle high-density and large-scale routing and congestion challenges, we suggest introducing a net resource allocation step as a potentially desirable initial routing stage.
- By introducing a concept of NRPD, we formulate the resource allocation problem as linear programming (LP). Moreover, we solve the LP problem and obtain a posterior NRPD for each net on each grid.
- Taking into account the resource allocation map and congestion map, we introduce a cost scheme to guide net routing. With this cost scheme, we design a weighted RSMT construction technique that can find a desirable routing topology under a weighted layout.
- We propose a track assignment algorithm that effectively supports the non-preferred direction and achieves iterating optimization with global routing. Additionally, we have developed a pathfinding algorithm for detailed routing, which is specifically designed to optimize the number of corners.
- Compared to the state-of-the-art OpenROAD's [16] router, the experimental results indicate that our router achieves a 12.6% reduction in vias, effectively resolving DRC violations in open-source benchmarks. In the case of 28nm process benchmarks, DRC violations are reduced by 348.3%, the number of vias is reduced by 5.9%, our router performs well on multiple metrics.

## 2 RESOURCE ALLOCATION

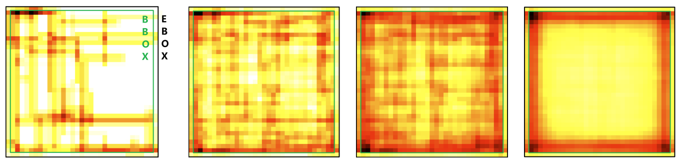
Most routers depend heavily on the net order, which can result in resource competition between nets and lead to congestion. We aim to allocate resources to each net before routing, guiding the routing based on the results of resource allocation. Firstly, we calculate the resources and estimate the net demands. Then, the concept of NRPD is introduced, which represents the probability density of the net region. Finally, the resource allocation is modeled as LP and solved.

### 2.1 Resource Calculation and Demand Estimation

In global routing, nets are assigned to available routing tracks, which are evenly distributed on different layers. Each GCell contains several routing tracks, which are used as the initial resources. According to design rules, there are some routing blockages that block routing resources on one or more layers and make those regions unavailable for routing. Therefore, we need to consider the impact of these obstacles on GCell resources as follows:

$$r_i = Num(track_i) \cdot [1 - \frac{Area(obs_i)}{Area(g_i)}], \quad (1)$$

where $r_i$ is the routing track resource (resource for short) of the $i$-th GCell $g_i \in G$, and $G$ is the set of GCells. And $Num(track_i)$ is the number of routing tracks in the $g_i$. $Area(g_i)$, $Area(obs_i)$ are the area of $g_i$ and the obstacle area in $g_i$, respectively.



(a) (b) (c) (d)

**Figure 2: Net region statistical density. (a) Ten nets. (b) 50 nets. (c) 200 nets. (d) 3000 nets.**

The demand of a net $n_j \in N$ ($N$ is the set of nets) is determined by the total routing length of the rectilinear Steiner tree after routing this net. We use the length of the rectilinear Steiner minimum tree (RSMT) as the demand for each net, which will serve as the lower bound for the final wirelength of this net. A possible estimated demand $d_j$ for $n_j$ is calculated as:

$$d_j = WL(RSMT_j). \quad (2)$$

It can be seen that $WL(RSMT_j)$ is the lower bound of demand of net $n_j$. To obtain a more practical (closer to final routing demand), we will make a statistic for all net wire lengths after routing.

### 2.2 Net Region Probability Density

A basic motivation is that if we know which locations on the layout are prioritized for a certain net routing, then we can achieve an ideal routing resource coordination. For this purpose, we define a concept of net region probability density to guide net routing on the layout.

Definition 1 (**NRPD**). *The net region probability density (NRPD) is a probability density function:*

$$pd_j : g_i \xrightarrow{pd_j} [0, 1], \ \forall g_i \in G, \ and \sum_{g_i \in G} pd_j(g_i) = 1,$$

*where $pd_j$ is the probability density function of net $n_j \in N$, $pd_j(g_i)$ is the probability density value of net $n_j$ in GCell $g_i$, $G$ is the domain set of GCells in a specified 2-D routing region.*

In the first step, we should ensure the prior probability density function for each net. Through the analysis of the previous works, the routing results of a net $n_j$ have a high probability of being contained within its bounding box (BBox), and a small number of nets are routed outside their BBox. To cover as high of the probability as possible, we set EBox as the extension BBox by adding $K$ rectangular rings around BBox. According to our experimental statistics, we set $K = 1$ in this paper. Therefore, we limit the specified routing region as the EBox $E_j$ of net $n_j$ in this paper.

Next, we show two considered prior probability density functions in this work. First, we consider a simple distribution, i.e., the uniform probability density function inside the EBox, as the prior distribution. Naturally, the routing probability density is 0 for the GCell outside the EBox. The uniform prior probability density $p_u\_pd_j(g_i)$ of net $n_j$ on GCell $g_i$ is expressed as follows:

$$p_u\_pd_j(g_i) = \begin{cases} \frac{1}{|G_j|}, & \forall g_i \in G_j \\ 0, & \text{o.w.}, \end{cases} \quad (3)$$

where $G_j$ is the set of GCells associated with EBox of $n_j$, and $G_j \subseteq G$.

The uniform prior probability density $p_u\_pd_j(g_i)$ is a relatively ideal distribution. To achieve a more practical prior probability distribution, we implement a statistic on the net distribution of the final routing result as shown in Fig. 2. For a net $n_j$, after final routing, we can count which GCells it goes through, i.e., we can figure out a routing map for net $n_j$. Similarly, the routing maps of all nets can be shown. To align routing maps of various sizes, we perform affine transformations to ensure overlapping and uniform size. By the above operation, we show the overlapping routing map with ten nets, 50 nets, 200 nets, and 3000 nets in Fig. 2. We can see that the superimposed routing map is distributed randomly under ten nets, and it tends to a rectangular ring around EBox as the number of nets increases.

To more theoretically formulate the above conclusion, we introduce the statistical frequency $sf(g_i)$ for each GCell $g_i$ and define it as follows:

$$sf(g_i) = \frac{T(g_i)}{\sum_{g_i \in G} T(g_i)}, \text{ and } \sum_{g_i \in G} sf(g_i) = 1, \tag{4}$$

where $T(g_i)$ is the overlapping times of all nets in GCell $g_i$. $sf(g_i)$ reflects a group effect for all nets. From Fig. 2, we have following claim:

**Claim 1.** The routing map of all nets with the same design configuration is identically distributed (i.d.).

Though Fig. 2 counts the routing map with 3000 nets or more, routing map distribution may be different for different designs, tools, methods, constraints, and so on. To improve the generalization, we try to fit a more general function under a given functional form. By comparison and analysis, we choose the Bohachevsky function [17] form as follows. The index $i$ of $g_i$ has $i = L * k + l$, which is a 1D numbering for a 2D sequence, where $L$ is the column number in the overlapping routing map. Let $x$ and $y$ be the continuous variable of $l$ and $k$.

$$sf(x,y) = a_1x^2 + a_2y^2 + a_3\cos(a_4\pi x) + a_5\cos(a_6\pi y) + a_7, \ \forall g_i \in G_j. \tag{5}$$

We utilize the nonlinear least square fitting (NLSF) method to fit a statistical frequency function of overlapping routing map as:

$$\begin{aligned} sf(x,y) =& 0.001(x^2+y^2) - 0.019\cos(1.066x) \\ &- 0.016\cos(1.086y) + 0.0726, \ \forall g_i \in G_j. \end{aligned} \tag{6}$$

The integral of the fitted function over its domain is approximately equal to 1. If we obtain the statistical frequency function $sf(x,y)$ of overlapping routing map for all nets, we can obtain an approximated statistical frequency $sf_j(g_i)$ of net $n_j$ by performing an inverse affine transformation.

And we set the statistical frequency $sf_j(g_i)$ as a new prior probability density $p_s\_pd_j(g_i)$ for each net $n_j \in N$ on GCell $g_i$ as follow:

$$p_s\_pd_j(g_i) = \begin{cases} sf_j(g_i), & \forall g_i \in G_j \\ 0, & \text{o.w.} \end{cases} \tag{7}$$

## 2.3 Linear Programming for Posterior Probability Density

The above prior probability density function for each net can be used to weigh GCell. For each net, we will concurrently assign a suitable resource allocation from a global view. Then, we can obtain a posterior probability density for each net in each GCell. In general, if we compress a detailed routing result with good routablity, we can get a uniform distributed wire density. Resource contention may occur if the bounding boxes of two nets overlap. Therefore, pre-allocating resources in overlapping areas to nets can alleviate resource contention between them during routing.

We show the expected effect of resource allocation in Fig. 3. Different colors represent the demand of the net, and darker colors indicate greater demand. In the initial stage, the distribution of the demand for each net is uniform. However, the total demand in the overlapping area is equal to the sum of the demands of the related nets. Since the resources of the GCell are limited, this may cause congestion due to the influence of the net order. After iterations, the resources in the GCell will be allocated to different nets. Even though the demand distribution is uniform throughout the design, it is not uniform in the net. Therefore,

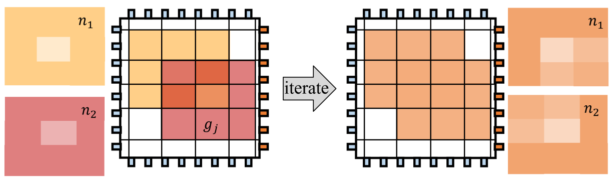


**Figure 3: The expected effect of resource allocation on design and nets.**

we transform the demands of the net into a corresponding weight map, which is used to guide the following routing.

To obtain a suitable posterior probability density for each net, we should reasonably allocate the limited resources of a GCell to its corresponding nets. By analyzing the relationship between GCell resources and net demands, we model it as a constraint minimization problem (10). Before that, we define a weight $w_{i,j}$ of net $n_j$ on GCell $g_i$, the relationship between $w_{i,j}$ and prior probability density $p\_pd_j(g_i)$ is as follow:

$$w_{i,j} = \frac{1 - p\_pd_j(g_i)}{|G_j| - 1}.$$

If $p\_pd_j(g_i) = p_u\_pd_j(g_i)$, i.e., choosing uniform prior probability density, then it has:

$$w_{i,j} = \begin{cases} \dfrac{1 - 1/|G_j|}{|G_j| - 1} = \dfrac{1}{|G_j|}, & \forall g_i \in G_j \\ 0, & \text{o.w.} \end{cases} \tag{8}$$

If $p\_pd_j(g_i) = p_s\_pd_j(g_i)$, i.e., choosing statistical frequency prior probability density, then it has:

$$w_{i,j} = \begin{cases} \dfrac{1 - sf_j(g_i)}{|G_j| - 1}, & \forall g_i \in G_j \\ 0, & \text{o.w.} \end{cases} \tag{9}$$

The objective function of Problem (10) is to minimize the sum of the gap between total allocated resources and the resource of each GCell. The constraint is that the sum of the allocated resources by the net in the covered GCell should equal the demand of the net, which is calculated in the prior probability density function (2). By solving the constraint minimization problem, GCell's resources can be fully utilized to meet all the net demands.

$$\begin{aligned} \min_x \ & \sum_{g_i \in G} \max\{0, \sum_{n_j \in N} w_{i,j}x_{i,j} - r_i\} \\ \text{s.t.} \ & \sum_{g_i \in G} w_{i,j}x_{i,j} = d_j, \ \forall n_j \in N, \end{aligned} \tag{10}$$

where $g_i \in G_j$ denotes the $i$-th GCell, $n_j \in N$ denotes the $j$-th net. $x_{i,j}$ is the posterior probability density of net $n_j$ in GCell $g_i$. Due to the non-smoothness of the $\max\{0,*\}$ function, we construct auxiliary variables $x_i^{aux}$ to achieve relaxation of the maximum function.

$$\begin{aligned} \min_x \ & \sum_{g_i \in G} x_i^{aux} \\ \text{s.t.} \ & \sum_{g_i \in G} w_{i,j}x_{i,j} = d_j, \ \forall n_j \in N \\ & x_i^{aux} \geq \sum_{n_j \in N} w_{i,j}x_{i,j} - r_i, \ \forall g_i \in G \\ & x_i^{aux} \geq 0, \ \forall g_i \in G. \end{aligned} \tag{11}$$

Let $N_i \subseteq N$ be the set of all nets associated with the GCell $g_i$. Let $G' \subseteq G$ be the set of GCells whose corresponding $N_i \neq \emptyset$. Let $m = |G|$, $n = |N|$, weight matrix $W = [w_{i,j}]_{m \times n}$ and variable matrix $X = [x_{i,j}]_{m \times n}$. Let $w_i = (w_{i,1}, w_{i,2}, ..., w_{i,n})^T$ and $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,n})^T$.

Let $x = ({x_1}^T, {x_2}^T, ..., {x_m}^T, x_1^{aux}, ..., x_m^{aux})^T$, where $x_i^{aux}$ is the introduced auxiliary variable. Furthermore, we transform resources allocation problem (11) as the linear programming problem (12).

$$\begin{aligned} \min_x \; & cx \\ \text{s.t. } & Ax = b \\ & A'x \geq b' \\ & x \geq 0. \end{aligned} \tag{12}$$

In Problem (12), the vector $c = (\,\mathbf{0}\;\; \mathbf{1}_m\,)$ with size $mn + m$. The matrix $A = (\,W\;\; \mathbf{O}\,)$ has dimensions $n \times (mn + m)$, where $W$ represents the relationship between the variable $x$ and the nets. The vector $b$ consists of the demands of all nets and is given by $b = (d_1, d_2, ..., d_n)^T$. The matrix $A' = (\,W'\;\; \mathbf{I}_m\,)$ is of size $m \times (mn + m)$, where $W'$ represents the inequality constraint relationship. The vector $b' = (-r_1, -r_2, ..., -r_m)^T$ has a size of $m$.

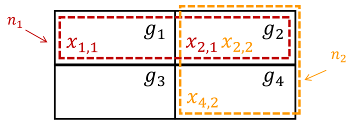


**Figure 4: An example of LP.**

We take an example to illustrate Problem (12). As shown in Fig. 4, there are four GCells ($g_1, g_2, g_3, g_4$) and two nets ($n_1$ in red and $n_2$ in yellow). $g_1, g_2 \in G_1$, and $g_2, g_4 \in G_2$. $n_1$ has two variables, $x_{1,1}$ and $x_{2,1}$, while net $n_2$ has two variables, $x_{2,2}$ and $x_{4,2}$. In practical calculations, we remove the invalid variable $x_{i,j}$ when its weight is equal to 0. Let $\hat{x}, \hat{c}, \hat{A}, \hat{b}, \hat{A}', \hat{b}'$ represent $x, c, A, b, A', b'$ after removing the invalid variables. $\hat{x} = [x_{1,1}, x_{2,1}, x_{2,2}, x_{4,2}, x_1^{aux}, x_2^{aux}, x_4^{aux}]^T$, and $\hat{c}, \hat{A}, \hat{b}, \hat{A}', \hat{b}'$ are as follows.

$$\hat{c} = \begin{bmatrix} \mathbf{0} & \mathbf{1}_3 \end{bmatrix}_{1\times 7} \text{ and } \hat{A} = \begin{bmatrix} W & \mathbf{O} \end{bmatrix}_{2\times 7}$$

$$W = \begin{bmatrix} w_{1,1} & w_{2,1} & 0 & 0 \\ 0 & 0 & w_{2,2} & w_{4,2} \end{bmatrix} \text{ and } \hat{b} = \begin{bmatrix} d_1 & d_2 \end{bmatrix}^T$$

$$\hat{A}' = \begin{bmatrix} W' & \mathbf{I}_3 \end{bmatrix}_{3\times 7} \text{ and } \hat{b}' = \begin{bmatrix} -r_1 & -r_2 & -r_4 \end{bmatrix}^T$$

$$W' = \begin{bmatrix} -w_{1,1} & 0 & 0 & 0 \\ 0 & -w_{2,1} & -w_{2,2} & 0 \\ 0 & 0 & 0 & -w_{4,2} \end{bmatrix}$$

For Problem (12), it is obviously a linear programming problem. Furthermore, its objective and constraint forms are significantly sparse, so its optimal solution can be quickly obtained through the simplex method.

## 3 ROUTING

The proposed router's flow is shown in Fig. 5. Initially, we calculate routing resources and estimate net demand. Then, we figure out the prior probability density and calculate the weight for each net on each GCell. With the weighting scheme, we model the resource allocation problem as an LP problem. By solving the LP problem, we obtain the posterior net probability density (NRPD) for each net. Combining the NRPD and congestion map generates a new weight map. For each net to be routed, the new weight map takes into account not only the impact of the previous routed net on it but also its impact on the subsequent nets. Under this weight map, the optimal Steiner tree topology solution is obtained using a lookup table called ASR, which can obtain a better solution under a weighted layout compared with FLUTE [18]. At global routing (GR), the Steiner tree topology will be decomposed to multiple

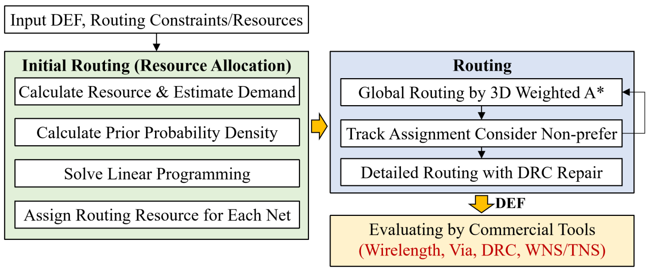

Input DEF, Routing Constraints/Resources
Initial Routing (Resource Allocation)
Calculate Resource & Estimate Demand
Calculate Prior Probability Density
Solve Linear Programming
Assign Routing Resource for Each Net
Routing
Global Routing by 3D Weighted A*
Track Assignment Consider Non-prefer
Detailed Routing with DRC Repair
DEF
Evaluating by Commercial Tools
(Wirelength, Via, DRC, WNS/TNS)

**Figure 5: Overall flow.**

two-pin nets for 3D weighted A* routing. We extend the existing track assignment (TA) technique to consider non-prefer tracks. In addition, to achieve global routing considering detailed routability, we iteratively refine the GR and TA. After performing detailed routing with DRC repair, the final routing results are obtained. The advanced commercial tool is used to evaluate the results based on wirelength, via number, DRC number, and timing metrics.

### 3.1 Global Routing

After allocating resources, the resulting posterior probability density for each net becomes the resource map. The resource map is then merged with the congestion map to generate a weighted map for the nets. Furthermore, the coordinates of the provided pins are compressed into a standardized format compatible with ASR. The weighted map and pin coordinates are utilized in ASR to establish a minimum-weight topology tree. Then, this tree is decomposed into several two-pin nets and weighted A* routing is performed. The global routing flow is shown in Fig. 6.

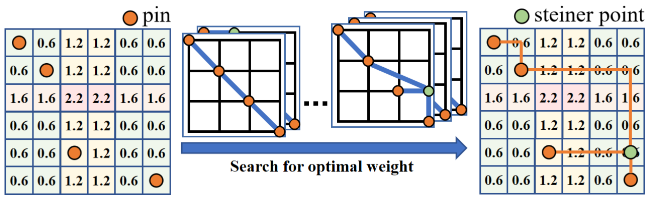


**Figure 6: Global routing flow.**

To control the resources of each GCell more accurately, the number of resources $r$ in each horizontal/vertical direction on each layer is determined by the number of routing tracks as Equation (1) and $r = round(r_i)$ denotes rounding, The resource used in GCell is $d_g$. The $d_g$ obtained from the demand of vias $d_{via}$ and routing wires $d_{wire}$ as follows:

$$d_g = d_{wire} + \lambda \cdot d_{via}. \tag{13}$$

Therefore, the congestion weight function $W_c$ can be calculated as:

$$W_c = d_g \cdot (1 + exp(1 - \eta \cdot \frac{d_g}{r}))^{-1}. \tag{14}$$

In this paper, the weight of GCell not only needs to consider routing congestion but also depends on the result of resource allocation. To fully consider these impacts, we propose an effective routing cost function $W_g$ as follows:

$$W_g = (c \cdot W_c + r \cdot W_r) \cdot (1 + \exp(1 - \xi \cdot \frac{c \cdot W_c + r \cdot W_r}{c + r}))^{-1}, \tag{15}$$

where $W_r$ is the resource weight function, $c$ and $r$ are the congestion and resource factor, respectively.

Previous global routers used FLUTE [18] to quickly obtain the RSMT. FLUTE establishes a Steiner tree lookup table for nets with less than 9 points, which is still accurate for nets with less than 100 points. However, FLUTE's single Steiner tree topology for each point combination

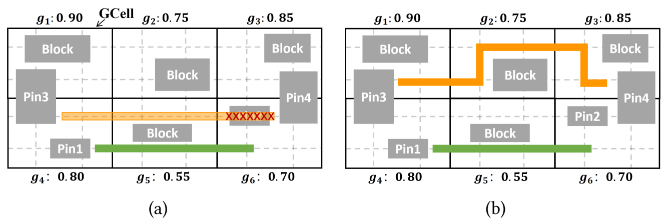


**Figure 7: Result of track assignment. (a) Inaccurate resource estimation. (b) The result after iteration.**

is not sufficient when dealing with varying weights. To address this, ASR, an improved lookup table, is constructed with additional Steiner tree topologies. ASR can generate more comprehensive Steiner tree topologies for each point combination, resulting in better results. ASR can find the smallest weighted Steiner tree under an optimal wirelength. To generate the candidate Steiner tree lookup table, start by generating point combinations with no more than 9 points. To minimize duplicate point combinations and reduce the generation time, ASR utilizes rotation and folding techniques. The length of the H-V tree route for the point combination is also used as the upper limit for the generated Steiner tree length to control the number of candidate trees. After obtaining the topology of the Steiner tree, it is decomposed into multiple two-pin nets and then subjected to 3D weighted A* routing to obtain the result of global routing.

## 3.2 Track Assignment

As shown in Fig. 7 (a), there is a total of 6 GCells, and each GCell has its corresponding routing cost. For instance, the routing cost of $g_1$ is 0.9. There are two global routing paths from Pin3 to Pin4, referred to as $path1 = (g_1, g_2, g_3)$ and $path2 = (g_4, g_5, g_6)$. The overall routing cost of $path1$ is 2.5, while that of $path2$ is 2.05. Considering that these two paths have the same wirelength and via count, in terms of a lower routing cost priority, global routing will select $path2$ as the routing result. However, during the TA stage, $path2$ will overlap with Pin2. This indicates there is a gap in resource control between global routing and track assignment. Therefore, we propose an iterative approach to short this gap. After global routing, track assignment evaluates the results and identifies all paths that violate the constraints as overflows, guiding the rip-up and reroute in global routing. Repeat this iterative process until the overflows converge. The results of the iteration for TA are shown in Fig. 7 (b). To maximize the utilization of routing resources, we propose an assignment algorithm that utilizes a pathfinding algorithm to enhance routability. Initially, a graph is constructed by considering the H/V direction tracks within the panel. Next, track assignment tasks are generated using the global routing results. Then, the weights of the assignment tasks' endpoints are computed with the intention of optimizing the connection points proximity to the pins. Lastly, an assignment result is obtained through the utilization of a pathfinding algorithm.

## 3.3 Detailed Routing

During the detailed routing phase, the design is divided into multiple boxes, and a sliding window approach is utilized to perform rip-up and rerouting for each box. In this rip-up and rerouting process, the routing cost is increased in violation regions to achieve both routing diffusion and convergence of violations. The reduction in DRC violations becomes progressively gradual, even after multiple iterations. To address the remaining DRC violations, the iteration is prematurely terminated, and post-processing methods are employed. Detailed routing refers to the process of using a routing algorithm to connect all pins of each net while satisfying design rule constraints. In order to better utilize resources, we propose a corner-aware A* algorithm.

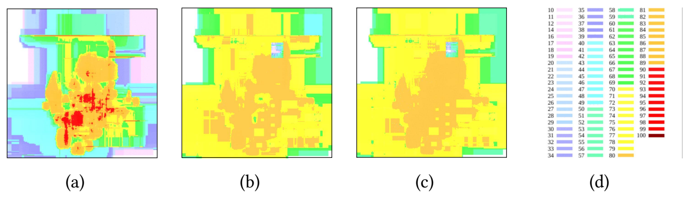


**Figure 8: Resource allocation. (a) Initial state. (b) 10 iterations results. (c) 30 iterations results. (d) level table.**

## 4 EXPERIMENTAL RESULTS

In this section, we first demonstrate the outstanding performance of the proposed resource allocation technique in guiding global routing. Subsequently, we apply this approach to enhance the quality of detailed routing solutions and optimize DRCs. We evaluate the performance of our router against the state-of-the-art OpenROAD's [16] router by calling an advanced commercial tool. Our router is implemented in C++ programming language, and all our experiments are performed on the same platform with a 64-bit Linux workstation using Intel Xeon 2.50GHz CPU and 128 GB memory. For fair comparisons, we use 128 threads for both routers. In our proposed algorithm, the parameters $\lambda$ in Eqution (13) are set to 0.8 and 1, respectively. Moreover, $\eta$ in Eqution (14) and $\xi$ in Eqution (15) were both set to 2.

OpenROAD integrates fast route and TritonRoute [14], which is currently the best open-source or academic router for fixing DRC violations. Additionally, OpenROAD is capable of reading LEF/DEF files of multiple technology process nodes. Due to the inability to showcase our advantages well in the ISPD 2018 benchmarks on 65nm and 45nm process nodes, we conduct a sufficient comparison on the 28nm process node. And ISPD 2018 benchmarks without RTL cannot generate DEF files for the 28nm technology process node. We used the "ysyx" design set[1] with RTL to generate DEF files on the 28nm process using a commercial tool.

## 4.1 Resource Allocation Performance

We demonstrate the effectiveness of our routing resource allocation through an iterative process as shown in Fig. 8. In the initial state, the regions with dense nets have higher routing demand, which is colored red, while the surrounding regions have lower demand, which is colored blue or green. After ten iterations, the routing demands in the red regions spread around, and the demand level in the surrounding area improved. After 30 iterations, the overall demand distribution tends to be uniform and flat. Actually, in Fig. 8, there are multiple macros in the lower-right corner, which block 70% of routing resources. Correspondingly, the demand level of this region after resource allocation is 70, which indicates that resource allocation is also effective for obstacles. Consequently, our resource allocation method can effectively balance routing demands and resources, which guides net routing and effectively releases the contention of routing.

## 4.2 Comparisons After Detailed Routing

Since the improvement during global routing may not actually reflect the optimization of the detailed routing solution and DRCs. Therefore, we proceed with the subsequent steps and evaluate the results using commercial tools. In Table 2, we compare our router with OpenROAD [16] on the open-source ISPD 2018 benchmarks [19], which utilization ranges from 0.57 to 1. From Table 2, we observe our router

[1] ysyx is a RISC-V processor chip talent plan.

**Table 1: Experimental results on design set "ysyx".**

| Benchmark | Statistics | | | Wirelength(um) | | Via count | | WNS | | TNS | | DRC count | | CPU (h) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #Cells | #Nets | Util | [16] | Ours | [16] | Ours | [16] | Ours | [16] | Ours | [16] | Ours | [16] | Ours |
| ysyx_1 | 89737 | 86602 | 0.61 | 996104 | **982381** | 789030 | **753429** | **-1.909** | -2.042 | **-241.722** | -260.482 | 58120 | **11268** | 2.18 | **1.81** |
| ysyx_2 | 95581 | 92630 | 0.65 | 896928 | **889114** | 781131 | **759249** | **-0.429** | -0.579 | -3.933 | **-0.579** | 63388 | **18685** | **3.34** | 3.73 |
| ysyx_3 | 96381 | 93104 | 0.60 | 971123 | **960662** | 795390 | **762392** | **-1.01** | -1.025 | -42.363 | **-34.859** | 180707 | **135022** | 2.3 | **1.68** |
| ysyx_4 | 141612 | 138428 | 0.70 | 1585169 | **1568690** | 1268986 | **1212195** | -2.602 | **-2.296** | -329.497 | **-291.058** | 93578 | **22554** | 13.64 | **7.68** |
| ysyx_5 | 145244 | 140266 | 0.50 | 1855755 | **1831080** | 1297931 | **1213173** | **-0.017** | -0.193 | **-0.017** | -6.864 | 86538 | **17694** | 1.95 | **1.67** |
| ysyx_6 | 145160 | 141757 | 0.70 | 1544978 | **1531242** | 1239512 | **1190702** | **-0.611** | -0.872 | -283.792 | **-207.228** | 92354 | **21943** | 5.49 | **2.19** |
| ysyx_7 | 278050 | 273234 | 0.50 | 3112095 | **3087993** | 2312893 | **2182281** | -0.603 | **-0.377** | -66.852 | **-61.792** | 174790 | **53292** | 11.83 | **7.01** |
| ysyx_8 | 341779 | 338158 | 0.50 | 3978912 | **3950379** | 2907658 | **2736294** | -0.460 | **-0.424** | -142.077 | **-121.499** | 215480 | **59849** | **7.8** | 8.7 |
| ysyx_9 | 371692 | 368331 | 0.50 | 4202756 | **4172103** | 3094471 | **2913300** | -0.78 | **-0.738** | -495.563 | **-391.805** | 233526 | **70490** | 18.36 | **8.08** |
| ysyx_10 | 441727 | 439416 | 0.50 | 5396117 | **5358167** | 3767737 | **3527228** | -0.965 | **-0.816** | -195.533 | **-125.89** | 271325 | **82135** | 44.26 | **11.82** |
| ysyx_11 | 451568 | 449090 | 0.50 | 5347497 | **5305053** | 3798216 | **3556911** | **-0.747** | -1.168 | **-302.927** | -335.876 | 274426 | **86135** | 13.16 | **7.77** |
| ysyx_12 | 472095 | 471895 | 0.50 | 5767591 | **5727769** | 4087891 | **3818688** | -0.554 | **-0.445** | -214.484 | **-146.247** | 298730 | **93271** | **13.95** | 15.8 |
| ysyx_13 | 564643 | 530881 | 0.50 | 6058000 | **6014827** | 4373642 | **4120168** | -1.167 | **-0.966** | -1052.6 | **-361.866** | 329574 | **105101** | 17.09 | **10.46** |
| ysyx_14 | 558390 | 507709 | 0.48 | 6072748 | **6030898** | 4248156 | **3975738** | -0.571 | **-0.377** | -1360.3 | **-696.163** | 304291 | **88182** | 11.61 | **10.39** |
| ysyx_15 | 471606 | 458065 | 0.40 | 5580351 | **5536864** | 3827478 | **3584521** | -1.591 | **-1.218** | -1911.8 | **-1201.3** | 280554 | **88931** | 14.85 | **8.4** |
| ysyx_16 | 503697 | 466947 | 0.40 | 5815550 | **5765702** | 3877494 | **3611242** | -0.772 | **-0.671** | -278.067 | **-173.038** | 279229 | **81067** | 8.62 | **7.06** |
| ysyx_17 | 514710 | 487311 | 0.50 | 5631684 | **5593650** | 4000381 | **3761932** | -1.442 | **-1.185** | -1481.9 | **-558.699** | 298443 | **92903** | 12.16 | **7.33** |
| ysyx_18 | 502794 | 462863 | 0.50 | 5246412 | **5211605** | 3732174 | **3500831** | -1.365 | **-1.315** | -623.62 | **-561.056** | 272655 | **83444** | 11.83 | **10.15** |
| Avg. Ratio | - | - | - | 1.009 | **1.000** | 1.059 | **1.000** | 1.046 | **1.000** | 1.706 | **1.000** | 3.483 | **1.000** | 1.592 | **1.000** |

**Table 2: Experimental results on ISPD 2018 benchmarks.**

| Benchmark | Statistics | | | Wirelength(um) | | Via count | |
|---|---|---|---|---|---|---|---|
| | #Cells | #Nets | Util | [16] | Ours | [16] | Ours |
| ispd18_test1 | 8879 | 3153 | 0.85 | **86222** | 86525 | 36339 | **34696** |
| ispd18_test2 | 35913 | 36834 | 0.57 | **1148524** | 1152370 | 369451 | **343583** |
| ispd18_test3 | 35977 | 36700 | 0.65 | **1218622** | 1220947 | 369036 | **344138** |
| ispd18_test4 | 72094 | 72410 | 0.89 | **2438199** | 2443710 | 747883 | **703306** |
| ispd18_test5 | 71954 | 72394 | 0.92 | **2523261** | 2527960 | 918154 | **786478** |
| ispd18_test6 | 107919 | 107701 | 0.99 | **3307386** | 3313511 | 1384648 | **1183752** |
| ispd18_test7 | 179881 | 179863 | 0.9 | **5777213** | 5785905 | 2264330 | **1956610** |
| ispd18_test8 | 192003 | 179863 | 0.9 | **5808885** | 5809937 | 2343392 | **1966186** |
| ispd18_test9 | 192911 | 178858 | 0.91 | **4963915** | 4969154 | 2324116 | **1953100** |
| ispd18_test10 | 290386 | 182000 | 1 | 6392759 | **6348327** | 2539770 | **2257917** |
| Avg. Ratio | - | - | - | **0.999** | 1.000 | 1.126 | **1.000** |

can reduce 12.6% via count, while DRC violations are roughly resolved in both tools (below 20).

To further demonstrate the applicability of our proposed method, we conduct experimental comparisons based on large-scale cases under the 28nm process node. In Table 1, the wirelength, the number of vias, worst negative slack (WNS), total negative slack (TNS), and the number of DRCs are reported by some advanced commercial tools. The CPU (h) represents the runtime for hours of the tested router. As shown in the table, our router can achieve 0.9%, 5.9%, and 348.3% improvement in wirelength, vias, and DRC violations while the runtime is only utilized approximately 62% of the runtime. Surprisingly, even though our method is not optimized specifically for timing, the optimizations for other metrics also resulted in 4.6% WNS and 70.6% TNS improvement.

## 5 CONCLUSION

In this work, we propose a resource allocation-based router to reduce congestion during routing. First, we introduce the concept of net region probability density (NRPD) to evaluate probability in routing regions of nets. We model resource allocation as linear programming and solve it. Then, we utilize a weighted scheme that combines the resource map and congestion map to guide weighted routing. To accommodate complex weight environments, we construct a lookup table called ASR to generate desirable Steiner trees. Finally, we improve routability by utilizing techniques including global routing and track assignment iterations. We optimize the routing results by employing non-prefer track assignment and corner-aware A* in detailed routing. Our router demonstrates effectiveness in solving large-scale circuits and exhibits advantages in multiple metrics compared to previous routers.

## ACKNOWLEDGMENT

This work is supported by the Major Key Project of PCL (No. PCL2023A03).

## REFERENCES

[1] Mark S. Lundstrom and Muhammad A. Alam. Moore's law: The journey ahead. *Science*, 378(6621):722–723, 2022.

[2] Yen-Jung Chang, Yu-Ting Lee, and Ting-Chi Wang. NTHU-Route 2.0: A fast and stable global router. In *Proc. ICCAD*, pages 338–343, 2008.

[3] Yue Xu, Yanheng Zhang, and Chris Chu. FastRoute 4.0: Global router with efficient via minimization. In *Proc. ASP-DAC*, pages 576–581, 2009.

[4] Jinwei Liu, Chak-Wa Pui, Fangzhou Wang, et al. CUGR: Detailed-Routability-Driven 3D Global Routing with Probabilistic Resource Model. In *Proc. DAC*, pages 1–6, 2020.

[5] Jiayuan He, Udit Agarwal, Yihang Yang, et al. SPRoute 2.0: A detailed-routability-driven deterministic parallel global router with soft capacity. In *Proc. ASP-DAC*, pages 586–591, 2022.

[6] C. Albrecht. Global routing by new approximation algorithms for multicommodity flow. *IEEE TCAD*, 20(5):622–632, 2001.

[7] Tai-Hsuan Wu, Azadeh Davoodi, and Jeffrey T Linderoth. GRIP: Scalable 3D global routing using Integer Programming. In *Proc. DAC*, pages 320–325, 2009.

[8] Minsik Cho, Katrina Lu, Kun Yuan, et al. BoxRouter 2.0: architecture and implementation of a hybrid and robust global router. In *Proc. ICCAD*, pages 503–508, 2007.

[9] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou. Track assignment: a desirable intermediate step between global routing and detailed routing. In *Proc. ICCAD*, pages 59–66, 2002.

[10] Daohang Shi and Azadeh Davoodi. TraPL: Track planning of local congestion for global routing. In *Proc. DAC*, pages 1–6, 2017.

[11] Zhen Zhuang, Genggeng Liu, Tsung-Yi Ho, et al. TRADER: A Practical Track-Assignment-Based Detailed Router. In *Proc. DATE*, pages 766–771, 2022.

[12] Yanheng Zhang and Chris Chu. Regularroute: An efficient detailed router applying regular routing patterns. *IEEE TVLSI*, 21(9):1655–1668, 2013.

[13] Fan-Keng Sun, Hao Chen, Ching-Yu Chen, Chen-Hao Hsu, and Yao-Wen Chang. A multithreaded initial detailed routing algorithm considering global routing guides. In *Proc. ICCAD*, pages 1–7, 2018.

[14] Andrew B. Kahng, Lutong Wang, and Bangqi Xu. TritonRoute-WXL: The Open-Source Router With Integrated DRC Engine. *IEEE TCAD*, 41(4):1076–1089, 2022.

[15] Gengjie Chen, Chak-Wa Pui, Haocheng Li, et al. Dr. CU: Detailed Routing by Sparse Grid Graph and Minimum-Area-Captured Path Search. *IEEE TCAD*, 39(9):1902–1915, 2020.

[16] OpenROAD. https://github.com/The-OpenROAD-Project/OpenROAD.

[17] Momin Jamil and Xin-She Yang. A Literature Survey of Benchmark Functions For Global Optimization Problems. *arXiv:1308.4008*, 4(2):150–194, 2013.

[18] Chris Chu and Yiu-Chung Wong. FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design. *IEEE TCAD*, 27(1):70–83, 2008.

[19] Stefanus Mantik, Gracieli Posser, Wing-Kai Chow, et al. ISPD 2018 Initial Detailed Routing Contest and Benchmarks. In *Proc. ISPD*, page 140–143, 2018.