ATLAS: A Two-Level Layer-Aware Scheme for Routing with Cell Movement

Xinshi Zang, Fangzhou Wang, Jinwei Liu, Martin D.F. Wong
Department of Computer Science and Engineering, The Chinese University of Hong Kong
{xszang, fzwang, jwliu}@cse.cuhk.edu.hk, mdfwong@cuhk.edu.hk

ABSTRACT

Placement and routing are two crucial steps in the physical design of integrated circuits (ICs). To close the gap between placement and routing, the routing with cell movement problem has attracted great attention recently. In this problem, a certain number of cells can be moved to new positions and the nets can be rerouted to improve the total wire length. In this work, we advance the study on this problem by proposing a two-level layer-aware scheme, named ATLAS. A coarse-level cluster-based cell movement is first performed to optimize via usage and provides a better starting point for the next fine-level single cell movement. To further encourage routing on the upper metal layers, we utilize a set of adjusted layer weights to increase the routing cost on lower layers. Experimental results on the ICCAD 2020 contest benchmarks show that ATLAS achieves much more wire length reduction compared with the state-of-the-art routing with cell movement engine. Furthermore, applied on the ICCAD 2021 contest benchmarks, ATLAS outperforms the first place team of the contest with much better solution quality while being 3× faster.

ACM Reference Format:
Xinshi Zang, Fangzhou Wang, Jinwei Liu, Martin D.F. Wong. 2022. ATLAS: A Two-Level Layer-Aware Scheme for Routing with Cell Movement. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22), October 30-November 3, 2022, San Diego, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3508352.3549470

1 INTRODUCTION

In physical design, placement and routing (P&R) are two crucial problems that significantly impact the circuit performance, power, and area (PPA). Due to their high complexity, both placement and routing are further split into two sub-problems - global and detailed placement, and global and detailed routing. Many efficient algorithms have been proposed to deal with these sub-problems separately [3, 8–10].

However, the final solution quality of P&R will inevitably be degraded with many sub-problems defined and solved independently, due to the misalignment between objectives. For example, several approximations need to be made in placement to model wire length and routability, e.g., by half perimeter wire length (HPWL) and pin density respectively. The actual routed wire length and routability will be degraded since these approximations are not accurate enough.

To address the inconsistency between the objectives of placement and routing, several different algorithms have been proposed, which can be divided into two categories. In the first category [2, 11, 13, 14], global routing is integrated into placement by invoking global routers in the placement stage to get a congestion map, which can help improve the wire length estimation accuracy during placement. The second category contains several recent works [4, 7, 15, 16], which incorporate placement into global routing by allowing cell movement in the routing stage. Starfish [15], Huang et al. [7] and Zou et al. [16] propose different heuristic algorithms to incrementally optimize wire length by conducting iterative cell movement and net rerouting. ILP-GRC [4] formulates this problem as an integer linear programming problem to simultaneously perform cell movement and net rerouting. It is worth noting that these two categories are complementary to each other and can be combined to further bridge the gap between placement and routing. In this work, we focus on advancing the study on the routing with cell movement problem by taking full advantage of higher routing layers.

Most global routers today will perform 3D routing with multiple horizontal and vertical routing layers. Compared with routing on lower layers, routing on upper metal layers has two major advantages, that is, reduced power consumption and better routability. According to [6], the R/C characteristic on each layer is different, resulting in different power consumption for routing on different layers. In general, the power consumption of routing on higher metal layers is smaller than that on lower layers. Besides, routing on higher layers can help relieve the routing congestion on lower layers, thus improving the routability. To encourage routing on upper layers, a minimum routing layer (MRL) constraint is adopted to restrict the lowest routing layer for some nets [5, 6, 15]. Furthermore, Hu et al. [6] propose a layer-based power factor to model routing costs for different layers.

In this work, we propose a two-level layer-aware scheme for routing with cell movement, named ATLAS. At the beginning, ATLAS will perform a coarse-level cell movement where some cells connected with nets having MRL constraints will be first merged together to form cell clusters, and these clustered cells will be moved as a whole in this stage and the affected nets will be rerouted. By doing so, those good movement directions in favor of multiple cells can be explored. In the next fine-level cell movement, those clustered cells will be unclustered and treated as single cells. By moving each cell individually, the placement and routing solution

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 14209320).

ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA

ACM ISBN 978-1-4503-9217-4/22/10…$15.00
https://doi.org/10.1145/3508352.3549470

can be further improved. Besides, we utilize a set of adjusted layer weights to encourage routing on upper metal layers, leading to more usage of the lower-cost routing resources there. Finally, wire length-driven pattern routing and maze routing are combined in ATLAS to further refine the routing solution.

The main contributions of this work are summarized as follows.

- We propose a two-level layer-aware scheme to solve the routing with cell movement problem. With coarse-level and fine-level cell movement, the placement and routing solution can be improved significantly.
- A set of adjusted layer weights is proposed and incorporated into the two-level framework to encourage routing on higher metal layers.
- ATLAS is capable of handling all kinds of constraints specified in both the ICCAD 2020 and 2021 contest and can significantly improve the total weighted wire length compared with the state-of-the-art methods [1] on each contest benchmark by 1.7% and 0.8% [2] respectively. Furthermore, ATLAS is 3 times faster than the first place team [12] on the ICCAD 2021 contest benchmarks.

The rest of the paper is organized as follows. Section 2 introduces the problem formulation and the proposed framework is discussed in Section 3. Section 4 shows the experimental results and analysis. The conclusion is drawn in Section 5.

2 PRELIMINARIES

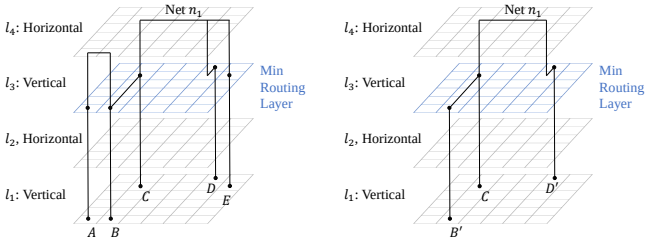
In this section, we will present a unified formulation of the routing with cell movement problem, which covers both the ICCAD 2020 [5] and 2021 [6] contest. Important terms and their meanings are summarized in Tab. 1.

Table 1: Terminology.

Term	Description
$L \times Y \times X$	The routing grid with L layers, Y rows and X columns.
G_{l_i,y_i,x_i}	The grid cell on layer l_i, row y_i, and column x_i
$Sup(l_i, y_i, x_i)$, $Dem(l_i, y_i, x_i)$, $Cap(l_i, y_i, x_i)$	The resource supply, demand and capacity of G_{l_i,y_i,x_i}
lw_i, lw'_i	The weight and adjusted weight of layer i
C, c_i	C denotes the cell set and c_i is the i-th cell
M	The maximum number of moved cells
Pos_{c_i}, Reg_{c_i}	The position and movable region of c_i, $Pos_{c_i} \in \mathbb{R}^2, Reg_{c_i} \subset \mathbb{R}^2$
$N, n_i, \lvert n_i \rvert$	N denotes the net set, n_i is the i-th net and $\lvert n_i \rvert$ is the weighted wire length of n_i
nw_i, mrl_i	The weight and minimum routing layer of net i
Seg_i	The routing segments of n_i
$Seg_{i,j}$	The routing segments of n_i on layer j
$\lvert Seg_i \rvert, \lvert Seg_{i,j} \rvert$	The length of the respective routing segments

2.1 Resource Supply and Demand of P&R

Different from traditional routing grids, in this work, the resource supply and demand defined on each grid cell apply to both cell placement and net routing. A cell placed at $\langle y_i, x_i \rangle$ will consume some of $Sup(l', y_i, x_i)$ on layer l' depending on the size and location of its blockages. The contest [5] is different from [6] that there are some extra demand rules for cells placed in the same position or adjacent positions. Each net n_j passing through G_{l_i,y_i,x_i} will occupy one unit of $Sup(l_i, y_i, x_i)$. The variable $Dem(l_i, y_i, x_i)$ is all resource demands consumed by cell placement and net routing on G_{l_i,y_i,x_i} and the variable $Cap(l_i, y_i, x_i)$ is calculated by $Sup(l_i, y_i, x_i) - Dem(l_i, y_i, x_i)$. An example is shown in Fig. 1(a), where the total resource demand of routing n_1 on each layer is 5, 5, 9 and 7 respectively (from l_1 to l_4).



(a) Original placement and routing. **(b) After cell clustering.**

Figure 1: One net with MRL on a $4 \times 6 \times 6$ routing grid.

2.2 Weighted Wire Length (WWL)

In consideration of timing, a net weight nw_i will be set for each net n_i to represent the timing criticality. A layer weight lw_j will also be assigned to each layer l_j to represent the cost of routing on l_j. The total weighted wire length (TWWL) is then defined as

$$TWWL = \sum_{i=1}^{|N|} |n_i| = \sum_{i=1}^{|N|} \sum_{j=1}^{L} nw_i * lw_j * |Seg_{i,j}| \quad (1)$$

Here, $|Seg_{i,j}|$ is equal to the resource demand of routing n_i on l_j. It is worth noting that the TWWL can also be applied to those routing problems where there is no specification for layer weights and net weights. The weights in those problems can be regarded as 1. For instance, in Fig. 1(a), the $TWWL$ will just be 26 with net weights and layer weights all being 1.

2.3 Problem Formulation

The routing with cell movement problem can be formulated as follows:

Given a 3D routing grid and an initial placement and routing solution, move cells and reroute nets to minimize TWWL while satisfying:

(1) There should be no routing segments except vias under mrl_i for each net n_i (Routing Constraint).
(2) The position of a cell must be in its movable region, i.e., $Pos_{c_i} \in Reg_{c_i}$. The number of moved cells cannot exceed M (Movement Constraint).
(3) No overflow will be allowed, i.e., $Cap(l_i, y_i, x_i) \geq 0$ for each G_{l_i,y_i,x_i} (Overflow Constraint).

[1] Starfish [15] and the first place team [12] are the state-of-the-art on the ICCAD 2020 and 2021 contest benchmarks respectively.
[2] 1.7% and 0.8% can be regarded as significant improvements since the gaps between the top 2 teams in the contest 2020 [5] and 2021 [6] are just 0.7% and 0.5% respectively.

3 METHODOLOGY

3.1 Overview

In this section, we will introduce the overall framework of ATLAS which is shown in Fig. 2. Given an initial placement and routing solution, ATLAS will first perform a congestion-aware incremental 3D global routing to reduce the TWWL and to free resources for later cell movements (Sec. 3.2). After this preprocessing, a coarse-level cell movement will be conducted by first clustering cells to share the routing vias caused by the MRL constraint and then iteratively moving cell clusters and rerouting nets, which will be introduced in Sec. 3.3 and Sec. 3.4 respectively. After the cell cluster movement, those clustered cells will be unclustered and an iterative single cell movement and net rerouting process will be performed. In the coarse-level and fine-level cell movement, a layer weight adjustment technique is applied to encourage routing on upper metal layers, which will be discussed in detail in Sec. 3.5. Finally, ATLAS will utilize a wire length-driven incremental 3D global routing to reduce the TWWL as much as possible.

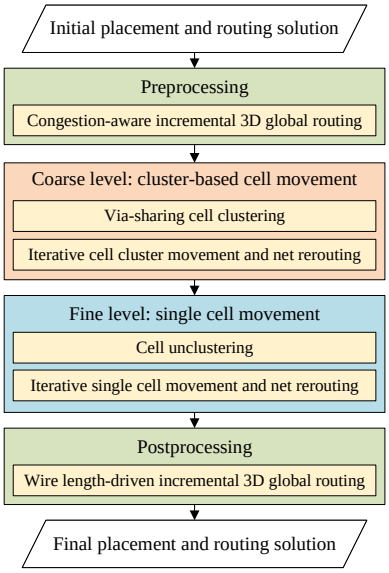


Figure 2: Proposed framework.

3.2 Incremental 3D Global Routing

To satisfy the overflow constraint, 3D global routing of ATLAS is conducted incrementally. Based on the existing routing solution, nets to be rerouted will be ripped up and rerouted one by one according to a net ordering. In ATLAS, the ordering of the nets depends on the net weight, half-perimeter wire length (HPWL), and the pin number. A net with a larger net weight, shorter HPWL, and smaller pin number will be routed earlier. Both pattern routing and maze routing are adopted to route a net.

3.2.1 Cost Function. In ATLAS, Eq. (2) is used to represent the routing cost of passing through G_{l_i,y_i,x_i}. If there is still unused resource in the grid cell, the cost is computed according to the layer weight and congestion cost. Otherwise, the cost is assigned with an infinite value to avoid overflow. λ_1 is a parameter to balance wire length and congestion. For congestion-aware routing, λ_1 is set to be 5 in our implementation since this can help achieving the best wire length without resource overflow in the experiments. In wire length-driven routing, λ_1 is set to be 0 to ignore the congestion impact.

$$Cost(l_i, y_i, x_i) = \begin{cases} lw_i + \frac{\lambda_1}{1+\exp(Cap(l_i,y_i,x_i))}, & Cap(l_i, y_i, x_i) > 0 \\ inf, & \text{otherwise} \end{cases} \tag{2}$$

3.2.2 Pattern Routing. Traditional approaches that perform 2D pattern routing and layer assignment separately to generate 3D routing topologies cannot optimize well towards different layer weights. To better handle the layer weight factor, the 3D pattern routing in [10] is adopted here to directly generate routing topologies to minimize the cost in Eq. (2). In this 3D pattern routing, FLUTE [1] will first be invoked to generate a rectilinear Steiner minimum tree (RSMT) for a multi-pin net. With the reverse order of a depth-first search (DFS) traversal on the RSMT, the nodes will be visited sequentially to compute their routing cost. A state of the DP algorithm is defined as $msc(i, l)$ or the minimum cost of routing the sub-tree rooted at node i and finishing the routing on layer l. There is no on-layer edge sharing and only the sharing of via edges is considered in caculating each $msc(i, l)$. The minimum cost of routing the whole tree will be $min_{0<=l<=L} msc(r, l)$, where node r is the root. Consequently, the state space for the DP algorithm will be $O(NL)$, where N is the number of nodes in the RSMT and L is the number of layers. The computational complexity of the algorithm will be $O(NL^D)$ where D is the maximum degree of the nodes. Since D is almost always smaller than or equal to 4 and L is usually small, the algorithm can run very efficiently.

3.2.3 Maze Routing. Due to the limited routing quality of pattern routing, maze routing can be applied to further improve the incremental routing result. Similar to [10, 15], a two-step routing strategy is adopted. In the first step, a coarsened routing grid graph is constructed by compressing a block of 5×5 grid cells into one coarsened cell. An RSMT for each net is generated by FLUTE on the coarsened grid graph and a route guide is then created with those grid cells within the coarsened cells covered by the RSMT. In the second step, maze routing is applied to connect all pins with the minimum cost within the route guide. This two-step routing strategy can reduce search space for maze routing while not sacrificing much routing quality. This two-step maze routing is conducted after pattern routing for all nets.

3.3 Via-Sharing Cell Clustering

In the coarse-level cell movement, ATLAS will firstly perform cell clustering to address the vias issue resulting from the MRL constraint. Fig. 1(a) shows an example of such a via issue. In the problem formulation, pins of the same net and residing in the same grid cell will share one via to move to the MRL. By clustering cell A and B, D and E into new hyper-cell B' and D' as in Fig. 1(b), A and B can share one via and so do D and E. Therefore, the number of vias needed can be reduced by almost half. By this via-oriented cell clustering, ATLAS can free plenty of resources for later cell movement and net routing.

Algorithm 1: Via-sharing cell clustering

```
Input: Single cell set C
Output: Cell cluster set C'
1  foreach c_i ∈ C do
2      Insert a new cell cluster containing c_i into C'
3  CluteringGainQ ← PriorityQueue()
4  foreach connected cluster pair, ⟨c'_i, c'_j⟩ with c'_i, c'_j ∈ C' do
5      gain ← reduced #MRL-vias by clustering ⟨c'_i, c'_j⟩
6      if gain > minGain && dist(c'_i, c'_j) < maxDist then
7          Insert {⟨c'_i, c'_j⟩, gain} into CluteringGainQ
8  while CluteringGainQ ≠ {} do
9      {⟨c'_i, c'_j⟩, gain} ← CluteringGainQ.pop()
10     if no overflow when merging ⟨c'_i, c'_j⟩ && c'_i, c'_j ∈ C' then
11         Move c'_i to c'_j (or c'_j to c'_i) and reroute affected nets
12         Form a new cell cluster c'_k with cells in c'_i and c'_j
13         Remove c'_i and c'_j and insert c'_k in C'
14         foreach cluster c'_p ∈ C' connected with c'_k do
15             gain ← reduced #MRL-vias by clustering ⟨c'_k, c'_p⟩
16             if gain > minGain && dist(c'_k, c'_p) < maxDist then
17                 Insert {⟨c'_k, c'_p⟩, gain} into CluteringGainQ
```

The detailed cell clustering algorithm is shown in Algo. 1. Firstly, every single cell forms an initial cell cluster (line 2). The clustering gain of a pair of cell clusters is defined as the reduction in via number due to the MRL constraint after clustering. To improve the clustering efficiency, two thresholds, *minGain* and *maxDist*, are used to filter out cluster pairs which only bring small gains or are too far away from each other (line 6 to 7). From line 8 to 17, a pair of cell clusters are selected greedily to maximize the clustering gain and these two cell clusters will be moved into the same grid cell and affected nets will be rerouted if there is no overflow. The newly generated cell cluster will be processed recursively in this clustering procedure. To efficiently maintain the priority queue for clustering gain, the removal of those outdated elements from the queue is delayed to the validity checking step in line 10.

3.4 Iterative Cell Movement and Net Rerouting

The iterative cell movement and net rerouting play a crucial role in both coarse-level and fine-level cell movement. Since cell clusters can be regarded as special hyper-cells, we will introduce the following algorithm from the perspective of a single cell.

3.4.1 Target Region. The basic target movement region of a cell is a median box formed by the median x and y coordinates of its connected cells. To increase the search space, we will expand the basic target region with an extra margin. The final target region is the intersection of the expanded target region and the specified movable region. Each cell can be moved to its target region if the move can bring at least a certain amount of reduction on the WWL.

3.4.2 Net Rerouting. To speed up the net rerouting after moving one cell, the A^*-based partial rerouting technique used in [7, 15] is adopted. For each net to be rerouted after a cell movement, some of its routing segments may become redundant and will be removed.

Algorithm 2: Iterative cell movement and net rerouting

```
Input: Cell set C; net set N; unique net weight set NW
Output: New placement and routing solution
1  foreach cell c_i ∈ C, p' ∈ targetRegion(c_i) do
2      estMoveGain(c_i, p') ← 0
3      foreach n_j ∈ nets connected with c_i do
4          sn_j ← remaining sub-net of n_j after removing c_i
5          l_1, ⟨y_1, x_1⟩ ← layer of n_j pin on c_i, p'
6          d ← min_{l_2,y_2,x_2}∈sn_j LUT(Δ_x, Δ_y, l_1, l_2, mrl_j)
7          estMoveGain(c_i, p') += |sn_j| + d − |n_j|
8  Sort targetRegion(c_i), ∀c_i ∈ C ; // Large estMoveGain first
9  Sort NW ;                          // Large weight first
10 for nw ∈ NW do
11     Candidate cell set C* ← {}
12     foreach net n_i ∈ N do
13         if nw_i ≥ nw then
14             Insert the cells connected with n_i into C*
15     minMoveGain ← An initial gain threshold
16     converge ← false
17     while not converge do
18         totalMoveGain ← 0
19         foreach cell c_i ∈ C* do
20             foreach p' ∈ targetRegion(c_i) do
21                 if estMoveGain(c_i, p') < minMoveGain then
22                     Break
23                 moveGain ← move c_i to p' and reroute nets
24                 if moveGain > minMoveGain then
25                     totalMoveGain += moveGain
26                     Break
27                 else
28                     Move c_i back to its original position
29         minMoveGain ← max(10^−6, minMoveGain − 1)
30         Update cell estMoveGain and reroute affected nets
31         if totalMoveGain == 0 then
32             converge ← true
```

The remaining segments will form a sub-net connecting the other cells except the moved one. A multi-source single-target A^* search algorithm will then be applied to find a path connecting the sub-net to the new position of the moved cell with the minimum cost.

3.4.3 Movement Gain Estimation. The gain of moving one cell is defined as the WWL reduction after cell movement and net rerouting. To evaluate quickly the gain of moving one cell to each position of its target region, we first pre-compute a look-up table (LUT) to calculate the minimum routing distance between any two points in the routing grid considering the layer weights and the MRL. Given two points, (l_1, y_1, x_1) and (l_2, y_2, x_2), and the mrl, $LUT(\Delta_y, \Delta_x, l_1, l_2, mrl)$ will return the minimum WWL of connecting these two points. The new WWL after moving one cell can then be estimated by the WWL of the remaining sub-net plus the minimum routing distance between the sub-net and the new position. It is worth noting that this LUT-based gain estimation provides an upper bound for the gain of each cell movement without considering

routing resource constraints. Although the actual routing gain can sometime be smaller than the estimation one, no cell movement with potential gain will be omitted with this gain estimation.

3.4.4 Main Flow. The whole algorithm of iterative cell movement and net routing is shown in Algo. 2. The estimated movement gain for each cell is initialized from line 1 to 7. In line 8, the points in the target region of each cell are sorted in descending order of their estimated movement gain. Since the nets with higher weights will have more impacts on TWWL, cells connected by nets with larger weights will be first put into the candidate set of cells to move (line 12 to 14). Furthermore, due to resource limitations, cells with larger estimated movement gain should be given higher priority to move. Therefore, a minimum movement gain threshold is set and decreased gradually in each iteration (line 21, 24, and 29). An A^* search-based partial routing is applied in line 23 to speed up runtime. At the end of each iteration, the maze routing is invoked to reroute the whole nets which are affected to improve the TWWL (line 30).

3.4.5 Cell Putting back. Although the maximum number of moved cells is constrained in the problem formulation, we temporarily ignore this constraint and keep moving cells and rerouting nets until there is no movement gain anymore. After that, we will put cells that only bring fewer gains back to their original positions to satisfy the movement constraint. The cell putting back process is conducted systematically in a similar way as for the iterative cell movement and net rerouting in Sec. 3.4. The only difference is that the target position for each moved cell is now their original position. The net rerouting technique in Sec. 3.4.2 is also adopted to reroute those affected nets after a cell is put back. This technique can avoid getting trapped in local optima due to some bad moves that use up the movement quota. Note that this technique is applied in both coarse-level cell cluster movement and fine-level single cell movement.

3.5 Adjustment on Layer Weights

In the ICCAD 2021 contest [6], the metal layers are divided into several groups vertically and each group is assigned a layer weight (In alignment with this setting, metal layers in traditional global routing without this layer grouping can be regarded as being all in the same group with layer weight one). Although upper layers may have enough routing resources and smaller layer weights, they can be hardly accessed when the lower routing layers are congested. Encouraging the use of upper layers at the early optimization stage may help improving the performance of our greedy algorithms and lead to a better optimum, where lower routing layers are less congested and low-cost higher routing layers are more sufficiently used. To that end, we adjust the weights of the metal layers in the same group by increasing the weights of the lower layers based on Eq. (3). Here, layer j is the top layer in the group of layer i and λ_2 is an adjustment step. For example, if the initial layer weights are $[1.0, 1.0, 1.0, 0.8, 0.8, 0.8]$ (from l_1 to l_6), the adjusted layer weights will be $[1.04, 1.02, 1.0, 0.84, 0.82, 0.8]$ with λ_2 being 0.02.

$$lw_i' = lw_i + (j - i) * \lambda_2 \tag{3}$$

In ATLAS, these adjusted layer weights will be applied in Eq. (2) throughout the coarse-level cell movement to encourage routing on upper metal layers. In the fine-level cell movement and postprocessing stages, the actual layer weights are used back to align with the final objective while following the routing guidance obtained in the coarse-level cell movement stage.

4 EXPERIMENTS

We used C++ to implement ATLAS and perform all the experiments on a 64-bit Linux workstation with Intel Xeon 2.9 GHz CPUs and 256 GB memory. Our algorithm is tested on both ICCAD 2020 (CAD20) and ICCAD 2021 (CAD21) routing with cell movement benchmarks [5, 6]. Consistent with the two contests, eight threads are enabled for our program.

Table 2: Benchmark statistic.

Case ID	#Cells	#Nets	$L \times Y \times X$
CAD20-case3	2,735	2,644	$7 \times 27 \times 33$
CAD20-case3B	2,604	2,563	$7 \times 29 \times 29$
CAD20-case4	204,206	179,996	$12 \times 277 \times 277$
CAD20-case4B	207,347	183,137	$12 \times 277 \times 277$
CAD20-case5	96,682	92,546	$16 \times 104 \times 103$
CAD20-case5B	96,689	92,559	$16 \times 104 \times 103$
CAD20-case6	352,269	332,080	$16 \times 237 \times 236$
CAD20-case6B	352,234	332,045	$16 \times 237 \times 236$
CAD21-case3	2,735	2,644	$7 \times 27 \times 33$
CAD21-case3B	2,738	2,644	$7 \times 27 \times 33$
CAD21-case4	96,689	92,559	$16 \times 104 \times 103$
CAD21-case4B	96,690	92,566	$16 \times 104 \times 103$
CAD21-case5	204,206	179,996	$12 \times 277 \times 277$
CAD21-case5B	204,206	179,996	$12 \times 277 \times 277$
CAD21-case6	352,234	332,063	$16 \times 237 \times 236$
CAD21-case6B	352,302	332,118	$16 \times 237 \times 236$

4.1 Results on CAD20 and CAD21 Benchmarks

The statistics of CAD20 and CAD21 benchmarks are summarized in Tab. 2. In general, CAD20 and CAD21 benchmarks are similar in terms of the number of cells, the number of nets, and the routing grids but with different initial placement and routing solutions. Besides, the optimization objectives and constraints in the two benchmark sets are also slightly different. In CAD20 benchmarks, net weights, layer weights, and movable region constraints are not specified, but a number of extra demand rules are imposed on cell placement. Necessary adaptations are made in order to run ATLAS on CAD20 benchmarks, e.g., the net weights and layer weights are all set to 1.

For CAD20 benchmarks, four baselines are chosen from the latest works, including Starfish [15], Huang et al. [7] and Zou et al. [16], and the winner in the ICCAD 2020 contest. Note that we cannot compare with ILP-GRC [4] since it does not handle those constraints defined in both CAD20 and CAD21 benchmarks. The results are listed in Tab. 3. Among the four baselines, Starfish outperforms Huang et al. and Zou et al. with a slight advantage in terms of scores. As a contrast, ATLAS can reduce 1.7% wire length compared to Starfish which is significantly more than the improvement of Starfish over the other baselines in Tab. 3. The runtime of ATLAS is longer than that of Starfish and Zou et al.

Table 3: Results on CAD20. The top 1 team is from the ICCAD *2020* contest. The results of the four baselines are obtained from papers [7, 15, 16]. The score is calculated by final TWWL − initial TWWL and weights of nets and layers are all 1. RT denotes runtime.

Benchmarks		Starfish [15]		Huang et al. [7]		Zou et al. [16]		1st Place Team		ATLAS	
Case ID	Initial TWWL	Score	RT (s)	Score	RT (s)	Score	RT (s)	Score	RT (s)	Score	RT (s)
case3	32,600	11,610	2	11,574	24	11,591	2	11,425	34	**11,745**	13
case3B	29,748	11,327	1	11,309	18	11,176	2	11,237	31	**11,451**	11
case4	4,680,681	2,064,790	260	2,064,165	1952	2,064,519	433	2,046,811	2,221	**2,083,357**	1,098
case4B	4,886,698	2,200,820	284	2,196,644	2050	2,198,445	429	2,182,574	2,299	**2,218,962**	1,195
case5	1,763,627	699,626	111	695,344	999	694,338	113	695,219	903	**707,887**	452
case5B	1,721,530	668,351	103	664,440	989	664,605	156	664,347	886	**678,901**	428
case6	7,188,481	2,737,028	684	2,737,732	2918	2,735,891	1,357	2,721,274	3,171	**2,791,708**	2,565
case6B	7,340,802	2,785,061	932	2,787,052	3265	2,773,149	1,566	2,748,097	3,406	**2,859,700**	3,241
Sum	27,644,167	11,178,613	2,377	11,168,260	12215	11,153,714	4,058	11,080,984	12,951	**11,363,711**	9,002
Ratio	-	1.000	1.000	0.999	5.139	0.998	1.707	0.991	5.448	**1.017**	3.787

Table 4: Results on CAD21. The top 3 teams are from the ICCAD *2021* contest. Their results are provided by the contest organizer [6].

Benchmarks		1st Place Team		2nd Place Team		3rd Place Team		ATLAS	
Case ID	Initial TWWL	Score	RT (s)	Score	RT (s)	Score	RT (s)	Score	RT (s)
case3	29,707	**9,901**	2,060	9,573	6	9,569	6	9,787	11
case3B	30,074	**10,124**	2,141	9,813	6	9,812	6	10,018	11
case4	1,590,850	610,719	3,526	606,053	474	597,955	726	**611,618**	447
case4B	1,590,850	584,771	3,527	579,469	479	571,747	760	**585,863**	436
case5	3,671,190	1,252,540	3,546	1,245,000	1,546	1,234,230	1,515	**1,256,390**	1,582
case5B	3,665,610	1,252,670	3,543	1,243,930	1,485	1,234,840	1,531	**1,256,050**	1,612
case6	5,602,310	1,207,500	3,564	1,204,010	3,153	1,162,360	2,271	**1,228,050**	2,166
case6B	5,554,350	1,186,130	3,564	1,184,140	2,923	1,143,930	2,287	**1,205,880**	2,035
Sum	21,735,080	6,114,354	25,471	6,081,987	10,072	5,964,443	9,102	**6,163,655**	8,300
Ratio	-	1.000	1.000	0.995	0.395	0.975	0.357	**1.008**	0.326

and this is due to much more time needed to explore cell cluster movement and routing on upper layers in ATLAS. Compared with Huang et al. whose performance is nearly equal to Starfish's, the runtime of ATLAS is even smaller with one possible reason that the LUT-based movement gain estimation in ATLAS should be much faster than the search-based estimation used in Huang et al.

For CAD21 benchmarks, we compare our results with the top three winners in the ICCAD 2021 contest. Those existing works, including Starfish, Huang et al. and Zou et al., do not handle those constraints in CAD21 benchmarks and cannot be compared with. As shown in Tab. 4, our algorithm improves over the first place in the contest by around 0.8 percent with respect to the TWWL reduction and spends only around 32.6% time used by them.

4.2 Trend in Layer Utilization

We conduct further experiments to reveal one reason why ATLAS can reduce more wire length on CAD20 benchmarks. Fig. 3 shows the difference in the total routing resource demand on each layer between ATLAS and Starfish. A bar below zero means ATLAS creates less demand on that layer. The figure shows that ATLAS is able to shift part of the demand on lower layers to upper layers and thus can reduce the routing demand on the lower metal layers which are usually more congested.

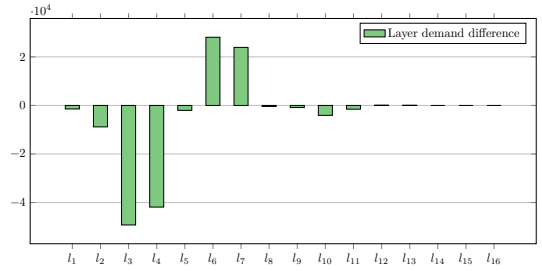


Figure 3: Difference of layer demand on case6 of CAD20 (ATLAS − Starfish).

4.3 Experiment without Cell Movement Limit

Besides, to fully test the potential of our algorithm, we run ATLAS without cell movement limit and let ATLAS optimize the placement and routing as much as possible. Note that originally the maximum number of the moved cells cannot exceed 30% the number of cells in each benchmark. We compare our results with 30% movable cells and 100% movable cells to the same results of the first place in ICCAD 2021 contest as disclosed in [6]. The comparison of weighted wire length reduction is illustrated in Fig. 4. The results show that relaxing the cell movement limit, ATLAS can still further improve the quality of placement and routing and the wire length reduction of ATLAS still has a slight advantage over that of the first place.

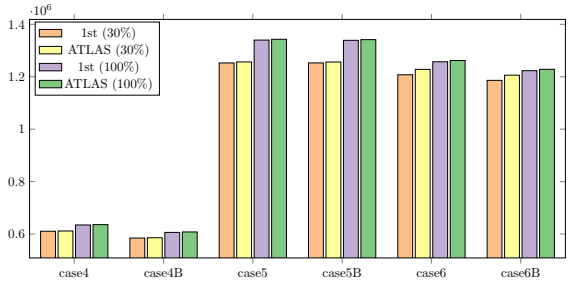
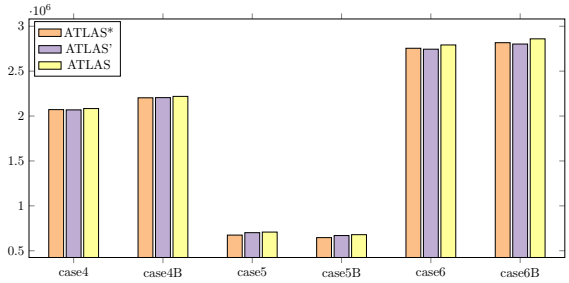


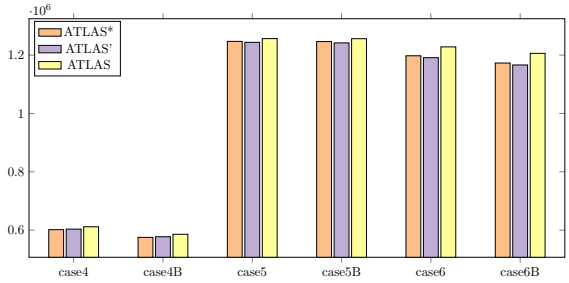
Figure 4: Weighted wire length reduction with different percentages of movable cells.

4.4 Ablation Study on Two Level Scheme

Finally, an ablation study is conducted to investigate the individual contribution of coarse level movement and fine level movement in the proposed two-level scheme. Two different versions of ATLAS (i.e., ATLAS* and ATLAS') are created by only keeping coarse level movement and fine level movement respectively in the original framework shown in Fig. 2. The comparison of weighted wire length reduction on six largest benchmarks in CAD20 and CAD21 is depicted in Fig. 5. Compared with ATLAS*, ATLAS' performs slightly better on smaller benchmarks, like case5 in CAD20 and case4 in CAD21, but its performance drops on larger benchmarks, like case6 in CAD20 and CAD21. Taking the advantages of both coarse level and fine level movement, ATLAS can always further improve the wire length reduction in all benchmarks.



(a) CAD20.



(b) CAD21.

Figure 5: Weighted wire length reduction for only coarse level movement (ATLAS*), only fine level movement (ATLAS') and both (ATLAS).

5 CONCLUSION

To bridge the gap between placement and routing, we propose a novel two-level layer-aware scheme for routing with cell movement. These two coarse-level and fine-level cell movements are combined to significantly reduce the number of vias and the total weighted wire length. Furthermore, a layer weight adjustment technique is proposed to encourage routing on higher layers. Experimental results show the superiority of our algorithm compared with the state-of-the-art methods on both CAD20 and CAD21 benchmarks.

6 ACKNOWLEDGEMENT

The authors truly appreciate Prof. Evangeline F.Y. Young for her kind guidance and valuable suggestions.

REFERENCES

[1] Chris Chu. 2004. FLUTE: Fast lookup table based wirelength estimation technique. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* IEEE, 696–701.

[2] Ke-Ren Dai, Chien-Hung Lu, and Yih-Lang Li. 2009. GRPlacer: Improving routability and wire-length of global routing with circuit replacement. In *Proceedings of the 2009 International Conference on Computer-Aided Design.* 351–356.

[3] Yixiao Ding, Chris Chu, and Wai-Kei Mak. 2017. Pin accessibility-driven detailed placement refinement. In *Proceedings of the 2017 ACM on International Symposium on Physical Design.* 133–140.

[4] Tiago Augusto Fontana, Erfan Aghaeekiasaraee, Renan Netto, Sheiny Fabre Almeida, Upma Gandh, Aysa Fakheri Tabrizi, David Westwick, Laleh Behjat, and José Luís Güntzel. 2021. ILP-Based Global Routing Optimization with Cell Movements. In *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI).*

[5] Kai-Shun Hu, Ming-Jen Yang, Tao-Chun Yu, and Guan-Chuen Chen. 2020. ICCAD-2020 CAD contest in routing with cell movement. In *Proceedings of the 39th International Conference on Computer-Aided Design.* 1–4.

[6] Kai-Shun Hu, Tao-Chun Yu, Ming-Jen Yang, and Chin-Fang Cindy Shen. 2021. 2021 ICCAD CAD Contest Problem B: Routing with Cell Movement Advanced. In *Proceedings of the 40th International Conference on Computer-Aided Design.*

[7] Zhipeng Huang, Haishan Huang, Runming Shi, Xu Li, Xuan Zhang, Weijie Chen, Jiaxiang Wang, and Ziran Zhu. 2021. Detailed Placement and Global Routing Co-Optimization with Complex Constraints. *Electronics* 11, 1 (2021), 51.

[8] Haocheng Li, Gengjie Chen, Bentian Jiang, Jingsong Chen, and Evangeline FY Young. 2019. Dr. cu 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD).* IEEE, 1–7.

[9] Yibo Lin, Zixuan Jiang, Jiaqi Gu, Wuxi Li, Shounak Dhar, Haoxing Ren, Brucek Khailany, and David Z Pan. 2020. Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 4 (2020), 748–761.

[10] Jinwei Liu, Chak-Wa Pui, Fangzhou Wang, and Evangeline FY Young. 2020. CUGR: Detailed-routability-driven 3D global routing with probabilistic resource model. In *2020 57th ACM/IEEE Design Automation Conference (DAC).* IEEE, 1–6.

[11] Wen-Hao Liu, Cheng-Kok Koh, and Yih-Lang Li. 2013. Optimization of placement solutions for routability. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC).* IEEE, 1–9.

[12] Canhui Luo, Jinghu Liang, Zhenxuan Xie, Zhouxing Su, and Zhipeng Lü. 2021. The 1st Place Team Video Show. *ICCAD-2021 CAD Contest, http://iccad-contest.org/2021/ProblemB-cada0136.mp4* (2021).

[13] Min Pan and Chris Chu. 2006. FastRoute: A step to integrate global routing into placement. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design.* 464–471.

[14] Min Pan and Chris Chu. 2007. IPR: An integrated placement and routing algorithm. In *Proceedings of the 44th Annual Design Automation Conference.* 59–62.

[15] Fangzhou Wang, Lixin Liu, Jingsong Chen, Jinwei Liu, Xinshi Zang, and Martin DF Wong. 2021. Starfish: An Efficient P&R Co-Optimization Engine with A*-based Partial Rerouting. In *2021 IEEE/ACM International Conference on Computer-Aided Design (ICCAD).* IEEE.

[16] Peng Zou, Zhifeng Lin, Chenyue Ma, Jun Yu, and Jianli Chen. 2021. Late Breaking Results: Incremental 3D Global Routing Considering Cell Movement. In *2021 58th ACM/IEEE Design Automation Conference (DAC).* IEEE.