# CR&P: An Efficient Co-operation between Routing and Placement

Erfan Aghaeekiasaraee*, Aysa Fakheri Tabrizi*, Tiago Augusto Fontana†, Renan Netto†,
Sheiny Fabre Almeida†, Upma Gandhi*, José Luís Güntzel†, David Westwick* and Laleh Behjat*
* Dept. of Electrical and Software Engineering, University of Calgary, Calgary, Canada
† Dept. of Computer Science and Statistics (INE/PPGCC), Federal University of Santa Catarina (UFSC), Florianópolis, Brazil
erfan.aghaeekiasarae@ucalgary.ca

***Abstract*—Placement and Routing (P&R) are two main steps of the physical design flow implementation. Traditionally, because of their complexity, these two steps are performed separately. But the implementation of the physical design in advanced technology nodes shows that the performance of these two steps is tied to each other. Therefore, creating efficient co-operation between the routing and placement steps has become a hot topic in Electronic Design Automation (EDA). In this work, to achieve an efficient collaboration between the routing and placement engines, an iterative replacement and rerouting framework facilitated with an Integer Linear Programming (ILP)-based legalizer is proposed and tested on the ACM/IEEE International Symposium on Physical Design (ISPD) 2018 contest's benchmarks. Numerical results show that the proposed framework can improve detailed routing vias and wirelength by 2.06% and 0.14% on average in a reasonable runtime without adding new Design Rule Violations (DRVs). The proposed framework can be considered as an add-on to the physical design flow between global routing and detailed routing.**

***Index Terms*—Global Routing, Detailed Routing, Placement, Physical Design**

## I. Introduction

The placement and routing steps are the key parts of the physical design implementation of Integrated Circuits (IC). The quality of each step greatly affects the circuit performance, area, and power consumption [1]. A common approach is to divide these two major steps to further sub-problems and solve them separately. This is known as a divide-and-conquer technique, which results in a manageable and converged solution in a reasonable run-time [2].

However, there are a couple of weak points in this approach that can adversely affects the solution. First, there can be a conflict between sub-problem's objective that optimizing one sub-problem can cause a misalignment or divergence in another one. Second, to cope with the first weak point, physical design tools try to estimate the impact that earlier sub-problems will have on the latter ones. Based on these estimations, it can set margins on the earlier sub-problems to help the convergence of the next sub-problems. In some cases, these margins are not well correlated with the next steps. For example, cell density or pin density in the placement is considered a constraint to preserve the circuit routability. But there is a possibility that these margins are not very well correlated with the routing problem [3]. A way to reduce this miscorrelation is to integrate optimization engines called co-operation [4].

In this work, an efficient collaboration between routing and placement steps to reduce this miscorrelation is proposed. The main contributions of this work are summarized as follows:

- An efficient Co-operation between Routing and Placement engines called CR&P is proposed. This framework can be used to fill the gap between placement and routing steps to empower the routing solution.
- A cost function based on an initial global routing solution is employed to target the critical areas. This can find cells with high congested edges and move them.
- An ILP-based legalizer is used to generate new legal locations for the candidate cells, and therefore the solutions can be fed directly to a detailed router.
- A 3D global router to estimate the path cost of each movement is developed. Then, an Integer Linear Programming (ILP)-based model is presented to find the best solution. The result of ILP will guarantee the legalized solution in each iteration.

In order to evaluate the effectiveness of the proposed framework, the routing benchmarks from ISPD-2018 [5] are used which are compatible with detailed routing. The obtained results showed that our framework is able to improve the number of vias by 2.06% and wirelength by 0.14% on average in detailed routing. The rest of the paper is organized as follows: In section II background and related works are surveyed. The problem formulation is described in section III. In section IV, CR&P flow is proposed. Experimental results are discussed in section V. Finally, section VI concludes this paper with future work.

## II. Background and Related Work

The placement step is divided into global placement, legalization, and detailed placement. The main objective of a global placer engine is to provide a placement solution while optimizing the wirelength, pin density, congestion, and satisfying technology constraints imposed by technology nodes [6]–[11]. Following that, the legalization and detailed placement steps' goal is to remove possible overlaps between cells while minimizing the displacement and improving solution quality respectively. To achieve this improvement many detailed placers use greedy heuristics, graph traversal, and combinatorial optimization algorithms including techniques such as global swap, local reordering, independent set matching, row &multi

row-based refinement and chain move [12]–[15]. However, most detailed placers try to minimize half perimeter wire length (HPWL), which is not a well-correlated factor with the detailed routing [16].

The routing step, on the other hand, is divided into two sub-steps: global and detailed routing. Global routing defines the routing regions of each net without going into the details of manufacturing and design rules of the circuit which are handled during the detailed routing stage [2]. After completing the placement, the cell positions are usually fixed, and routing methods are applied to connect the cells. This means that small inefficiencies in the placement solutions can be amplified during the routing and impact the quality. This can be avoided by integrating routing and placement [4].

A development of routing with cell movement was the subject of two recent ICCAD contests [3], [17]. To the best of our knowledge, [18] is the only work that directly addressed this problem so far. They proposed a cluster-based technique, where for each cell its median with related route is calculated. Then by using an ILP model, they find the best placement and routing arrangement. However, this work suffers from a couple of issues. First, although a significant improvement in estimated vias and wirelength in a global routing solution is reported, it is not clear what is the consequence of that in the detailed routing. Second, their proposed framework runtime is exponential and suffering from scalability issues. Third, moving cells only to the median could increase congestion and this adversely affects the detailed routing solution.

A development of framework that can integrate routing with cell movements has a couple of challenges. The first challenge is that for any new candidate position a legalized placement solution for the entire circuit must be guaranteed. Secondly, in modern benchmarks, such as those included in the ISPD-2018 contest [5], there is almost no empty space between cells. Therefore, many techniques such as chain [13], ripple movement [8], sliding and shifting inside an arbitrary window [6], which require empty spaces are impractical. Hence, finding appropriate candidate positions for cell movement in a reasonable runtime is critical. Therefore, an efficient method that can handle the above mentioned challenges is required.

## III. Problem Formulation

This problem is considered as a global routing problem that is collaborating with a detailed placement. In the global routing formulation, the area of the circuit is partitioned into regions called GCells (Grid Cells), where the 3D routing space can be modeled as a 3D graph of GCells (G). Each edge of graph G is notated by e, and it has two attributes: demand ($D_e$) and capacity ($C_e$). Two files are given as input: LEF (technology file) and DEF (design information) [19]. In the proposed framework, it is assumed that an initial placement solution is given and we can reroute and move cells. In Equation 1, it states that the objective of the problem is to minimize the total cost of routed nets. The description of all symbols is presented in Table I.

TABLE I: Symbols used in routing and placement formulation.

| Symbol | Definition | Symbol | Definition |
|---|---|---|---|
| $n$ | net n in netlist N | $X_{y_c^p}$ | x coordinate of cell c in position p |
| $x_n^r$ | $x_n^r$=1, then route r is selected for net n | $Y_{y_c^p}$ | y coordinate of cell c in position p |
| $R_n$ | Set of routes for net n | $H, H_x$ | Circuit's Height; Height of component x |
| $cost_n^r$ | Cost route r for net n | $W, W_x$ | Circuit's width; width of component x |
| $c$ | A cell in a list of cells C | $G$ | GCell Graph |
| $P_c$ | Set of placement (location) for cell c | $e$ | An edge in graph G |
| $y_c^p$ | $y_c^p$=1, then placement p is selected for cell c | $D_e$ | Demand edge e |
| $cost_c^p$ | Cost placement p for cell c | $C_e$ | Capacity edge e |

$$\min \quad \sum_{n=1}^{N} \sum_{r \in R_n} cost_n^r \times x_n^r \tag{1}$$

On the other hand, the constraints can be divided into two parts: (A) Routing Constraints (Equations 2-4), and (B) Placement Constraints (Equations 5-8):

$$\sum_{r=1}^{R} x_n^r = 1,\ \forall n \in N \tag{2}$$

$$\sum_{p \in P_c} y_c^p = 1,\ \forall c \in C \tag{3}$$

$$x_n^r, y_c^p \in \{0, 1\} \tag{4}$$

$$\frac{W_c}{2} \leq X_{y_c^p} \leq W - \frac{W_c}{2},\ \frac{H_c}{2} \leq Y_{y_c^p} \leq H - \frac{H_c}{2} \tag{5}$$

$$|X_{y_c} - X_{y_{c'}}| \geqslant \frac{W_{y_c} + W_{y_{c'}}}{2} \tag{6}$$

$$X_{y_c^p} = \alpha \times W_{site}, \alpha \in \mathbb{N} \tag{7}$$

$$Y_{y_c^p} = \alpha \times H_{row}, \alpha \in \mathbb{N} \tag{8}$$

In Equation 2, it states that for every net one route must be selected otherwise the solution will have an open net which invalidates the solution. Equation 3 ensures that each cell must have a position. Note: It is a strict constraint that selecting a cell position, a related route also must be selected in Equation 1 (Due to lack of space we didn't put this constraint in the equations). In Equation 4, a $x_n^r$ and $y_c^p$ either can be 1 or 0. In Equation 5, it states that each position of cell must be inside the circuit. After that, Equation 6 states that cells must not have any overlap to be considered as a legalized solution. In Equation 7, it ensures that each cell is aligned to sites horizontally. The sites are designed to ensure that in the detailed routing pins are aligned to the tracks [20]. Finally, Equation 8 states that each cell is aligned to power and ground rails in the top and bottom side of each row. Note: Each row has a specific orientation that all cells placed in the row must follow [20].

With the aforementioned constraints, we can ensure that the circuit is fully routed (no open nets), and is completely legalized. This means that the input is valid to apply any detailed router.

## IV. Proposed Solution

The flow is composed of three main steps: global routing, Co-operation between Routing and Placement (CR&P), and detailed routing (Figure 1). This work uses CUGR [21] as a global router and TritonRoute [16] as a detailed router. As you can see in Figure 1, the five steps in CR&P are the main phases proposed in this work. First, according to the initial global routing provided by CUGR, the cost of each cell is calculated in "*Labeling Critical Cells*". Then, in "*Generate Candidate Positions*" step, different candidates are generated for critical cells. After that, the cost of each candidate is estimated in "*Estimation Cost*" step. Then, an ILP model is constructed according to cell candidates and related routing cost estimation. The main goal of this step is to find another possible placement for critical cells to reduce the total cost of routing. Subsequently, after finding a new position for cells the database is updated such as the location of cells and related nets in "*Update Database*" step. This flow can be iterated k times and this can be continued to satisfy expected requirements. Finally, the global routing solution in the format of a guide file and a new position of cells in the format of DEF file is passed to the detailed router to investigate the result of improvement on the detailed routing solution.

In this section, the cost function that is used in this work is described, first. Next, the five steps in CR&P are discussed in more details.

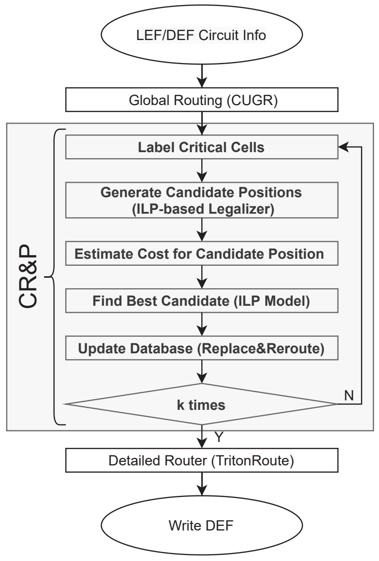
LEF/DEF Circuit Info
Global Routing (CUGR)
CR&P
Label Critical Cells
Generate Candidate Positions (ILP-based Legalizer)
Estimate Cost for Candidate Position
Find Best Candidate (ILP Model)
Update Database (Replace&Reroute)
k times
N
Y
Detailed Router (TritonRoute)
Write DEF

Fig. 1: The methodology flow incorporates three main steps. (1) Global Routing; (2) Co-operation between Routing & Placement (CR&P); (3) Detailed Routing. The framework inputs are LEF and DEF files, and the output is a DEF file.

## A. Cost Function

Two main concepts of Capacity ($C_e$) and Demand ($D_e$) of an edge, inspired from [21], are used to describe connection characteristics in 3D global routing graph G.

*Definition 1 (Capacity):* a capacity of an arbitrary edge e is the total number of tracks available in the edge.

*Definition 2 (Demand):* the demand of an edge can be formulated as follows:

$$D_e = U_w(e) + U_f(e) + \beta \times \delta_e \tag{9}$$

where it is the summation of wire usage ($U_w(e)$) passing through an edge, the usage of fixed components ($U_f(e)$) (like blockages and fixed nets) intersected with the edge, in addition to a rough estimation of Vias ($\beta \times \delta_e$). $\beta$ is a constant number, and the value of $\beta$ is 1.5 in this work, and $\delta_e$ can be modeled as $\sqrt{\frac{V_{e.src}+V_{e.dst}}{2}}$, which is inspired from [21]. It is notable that $V_{e.src}$ and $V_{e.dst}$ represent the number of vias in the source and destination of an edge $e$.

*Definition 3 (Edge Cost):* The cost of an edge can be defined as follows:

$$cost_e = Unit_e \times Dist(e) \times (1 + \text{penalty}(e)) \tag{10}$$

where $Unit_e$ defines the unit's weight of an edge. An edge can be either a wire or via edge. The Manhattan distance between the center of gcells is notated by ($Dist(e)$), and a penalty scheme ($penalty(e) = \frac{1}{1+\exp^{(-S\times(D_e-C_e))}}$). To describe the penalty of an edge, a logistic function of demand ($D_e$) and capacity ($C_e$) with a slope factor ($S$) is used [22]. Using logistic function for penalty means that the range of penalty can be a number between 0 and 1. Increasing the value of $S$ will cause faster overflow in an edge and reducing that will give more flexibility in the cost of edges.

## B. Co-operation between Routing and Placement (CR&P)

The main objective of this step is to find possible improvements in the position of the cells and the overall quality of routing. This step comprises labeling the critical cells, generating new candidate positions for each critical cell and estimating cost for each candidate position, and finally finding a possible solution to improve routing quality respect to wirelength and vias (Figure 1).

*1) Label Critical Cells:* algorithm 1 shows how this work finds a set of critical cells to move in each iteration. First, a set of cells from database (db) is copied in the cells' set of C (line 1). Next, this set is sorted according to the cost of initial global routing for the nets of each cell (line 3). Therefore, cells with a higher cost of nets will be ordered on top of the cell's set C. Following that in the loop, line 4, each cell is investigated according to three main factors: (1) No connected cell to cell c should be found in the critical cell set, line 6; (2) If cell c was already a critical cell in the earlier iterations of the flow, therefore $hist_c$ is 1, line 9. (3) If cell c already moved in the earlier iterations of the flow, therefore $hist_m$ is 1, line 10. Factors (2) and (3) both are modeled to reduce the probability of selecting cells, affected in the earlier iterations of CR&P. To model this probability, a simplified version of simulated annealing [23] is applied to the selection of each cell, line 11. In other words, if a cell is only selected as a critical cell in earlier iterations (not moved) the possibility of selecting the cell again in the current iteration is 36% (exp(-1)). If this cell is moved too, this possibility reduces to 13% (exp(-2)). This approach will help the framework to address more cells to move and not be stuck with critical cells in congested areas. Next, in line 15, the value of $\gamma$ (it is 0.6 in this work) tells the framework what percentage of cells, the flow wants to address to move. At the end, the selected critical cells are returned in algorithm 1.

**Algorithm 1:** Label Critical Cells

```
   Data: A set of cells C.
   Result: A set of critical cells $critical_{set}$.
1  $C$ = copy(db.cells);                          // A set of cells
2  $critical_{set}$ = {};                        // A set of critical cells
3  sort(C) ;                                     // Sort the set of Cells
4  foreach $c$ in $C$ do
5      $connected\_cells$ = c.getConnectedCells();
6      if $critical_{set}.find(connected\_cells)$ then
7          $continue$;
8      end
9      $hist_c$ = db.$critical_{set}^{hist}$.find(c);
10     $hist_m$ = db.$moved_{set}$.find(c);
11     simulated_annealing = exp(-1*($hist_c$+$hist_m$)) / T ;
12     if $simulated\_annealing > random(0, 1)$ then
13         $critical_{set}$.insert(c);
14     end
15     if $len(critical_{set}) > \gamma * len(C)$ then
16         $break$ ;
17     end
18 end
19 return $critical_{set}$
```

*2) Generate Candidate Positions (ILP-based Legalizer):* Generating candidate positions step is presented in algorithm 2. There are three loops that the flow is using. First loop, between line 1-6, is used to find placement candidates for each critical cell, and it runs in parallel. Then, next loop, line 7-10, runs sequentially to add conflict candidate placements. At the end, the calculating cost for each candidate loop, line 11-13, is executed in parallel.

In line 2, each cell gets its initial placement position as one of the placement candidates. This means that in the worst-case scenario the critical cells will keep their initial position. Then, an ILP-based legalizer is proposed to find a legalized solution for all cells after a movement in a local area. In line 3, the proposed ILP-based legalizer receives three inputs: a cell c, a number of sites ($N_{site}$) and a number of rows ($N_{rows}$) around the cell c. The legalizer will return a pair set of placement candidates for the critical cell c ($pl\_cds$), and a list of conflict cells with their new legalized positions (conflict_cells). This will later sequentially be pushed to the conflict cells in line 7-10. This means that for any position for critical cells there can be multiple legalized solution. The ILP-based legalizer can be modeled as follows:

$$\min \sum_{c \in C} \sum_{j=1}^{N_{row}} \sum_{i=1}^{N_{site}} cost_c^{(i,j)} \times y_c^{(i,j)} \qquad (11)$$

where

$$\begin{aligned} cost_c^{(i,j)} &= W_{site} \times |X_{y_c^{(i,j)}} - X_{y_c^{med}}| \\ &+ \\ &H_{row} \times |Y_{y_c^{(i,j)}} - Y_{y_c^{med}}| \end{aligned}$$

where $cost_c^{(i,j)}$ is the cost of cell c, in site i and row j. The cost function in legalizer encourages a cell to move toward its median position locally. The complexity of legalizer is $O(|cells| \times |sites|^{|rows|})$. It depends on the number of cells, sites, and rows in the local area that is targeted. In this work,

**Algorithm 2:** Generate Candidate Position and Cost Estimation for Critical Cells Algorithm

```
   Data: A list of critical cells to move $critical_{set}$
   Result: Find new candidate positions for critical cells and related cost
   // run parallel
1  foreach $c$ in $critical_{set}$ do
2      $c.placement\_candidates$.add(getCurrentPos($c$));
3      [$pl\_cds$, $conflict\_cells$] = legalizer.run($c$,$N_{site}$,$N_{row}$);
4      $c.placement\_candidates$.add($pl\_cds$);
5      $c.conflict\_cells$ = $conflict\_cells$;
6  end
   // run sequential
7  foreach $c$ in $critical_{set}$ do
8      [$c'$, $pl\_cds$] = $c.conflict\_cells$;
9      $c'.placement\_candidates$.add($pl\_cds$);
10 end
   // run parallel
11 foreach $c$ in $C$ do
12     c.calcCostPlacementCandidates(){};          // algorithm 3
13 end
```

$|cells|$ is 3, $|sites|$ is 20, and $|rows|$ is 5 in an execution of ILP-based legalizer. These number are achieved experimentally and it is a trade-off between runtime and a number of candidates for each cell.

*3) Candidate Position Cost Estimation:* After generating possible placement candidates for each critical cell, it is required to investigate the quality of new placement candidates. This quality is investigated by a cost function (Equation 10). As it is shown in algorithm 3, line 1, for each cell the connected nets are queried. Then, all nets with the new position of a cell (candidate placement) are calculated with a fast 3D pattern route between line line 5-6. The 3D pattern route used in this framework is inspired from [21]. The path cost of 3D route are used to calculate the cost of new position of cell, line 7. Note that to keep the routing solution validated for each net, in each iteration, only one cell is allowed to be moved and the other connected cells are fixed.

**Algorithm 3:** Cost Estimation for Candidate Placements (calcCostPlacementCandidates())

```
   Result: Estimating a cost value for placement candidates of each
           critical cell
1  $N$ = getNetsConnectedToCell($c$);
2  foreach $n$ in $N$ do
3      $C_n$ = getCellsConnectedToNet($n$);
4      foreach $pl\_cd$ in $c.placement\_candidates$ do
5          $flute$ = getFlute($C_n$,$pl\_cd$);
6          $n\_routed3D$ = getPatternRoute3D($flute$);
7          $pl\_cd.cost$ += $n\_routed3D$.getCost();
8      end
9  end
```

*4) Find Best Candidate by Solving Integer Linear Programming (ILP) Model:* This section discusses how a candidate between possible placement candidates of a cell will be selected. As explained in earlier sections, for each candidate placement there is a cost value, therefore the problem can be modeled as an integer linear optimization problem as follows:

$$\min \sum_{c \in C} \sum_{p \in P_c} cost_c^p \times y_c^p \qquad (12)$$

TABLE II: ISPD-2018 Contest Benchmarks [5] Statistics

| Circuit | # nets | # cells | Tech. Node | Circuit | # nets | # cells | Tech. Node |
|---|---|---|---|---|---|---|---|
| ispd18_test1 | 3K | 8K | 45nm | ispd18_test6 | 107K | 107K | 32nm |
| ispd18_test2 | 36K | 35K | 45nm | ispd18_test7 | 179K | 179K | 32nm |
| ispd18_test3 | 36K | 35K | 45nm | ispd18_test8 | 179K | 192K | 32nm |
| ispd18_test4 | 72K | 72K | 32nm | ispd18_test9 | 178K | 192K | 32nm |
| ispd18_test5 | 72K | 71K | 32nm | ispd18_test10 | 182K | 290K | 32nm |

where the objective is to select a candidate placement to minimize the cost value of Equation 12 ($cost_c^p$ is calculated by algorithm 3). Solving this model means that a new placement will be generated for the circuit.

*5) Update Database:* After finding the new position for critical cells, they are moved to their new position and the database is updated. Updating database after movement means that the nets connected to moved cells need to be rerouted by the global router (CUGR). Also, the congestion and blockage maps in the database are required to be updated. After updating database, the result of a new global routing solution and the cells' new positions can be passed to the detailed router or it can be iterated to achieve further improvement.

## V. Experimental Results

### A. Experimental Setup and Evaluation Metrics

The proposed framework is developed in C++ with Boost C++ libraries 1.68 [24] and CPLEX [25] as an ILP Solver. Then, all experiments are performed by using ISPD-2018 benchmarks [5]. The information of benchmarks are summarized in Table II. All experiments are executed on a Linux desktop (Ubuntu20.04) with an Intel® Core® i7-8700 CPU running at 3.20 GHz and 16GB RAM (DDR3 at 1600MHz) with 8 threads.

To evaluate the proposed framework, the evaluation metrics and the official evaluator of ISPD-2018 contest [5] are used. The evaluation metrics are: (1) routing metrics such as wirelength and vias' count in the detailed routing, (2) Design Rule Violations (DRVs) such as short, min area, spacing violations. For further information please refer to [5]. It is notable that the main reasons that ISPD-2018 are chosen versus ICCAD-2020&2021 benchmarks is that the ISPD-2018 benchmarks are more realistic benchmarks, and they are compatible with the detailed routing.

### B. Detailed Routing Results

To investigate the quality of the proposed framework, the results of routing solution with a baseline and state-of-the-are [18] are compared for each test benchmark in Table III. The flow of CUGR [21] and TritonRoute [16] is considered as our baseline. This includes only global routing and detailed routing without any cell's movement. Then, the improvement percentage of the state-of-the-art [18] and the proposed framework is compared with the baseline. We have received the binary of the state-of-the-are (*"ILP-Based Global Routing Optimization with Cell Movements"*) from [18] to collect all results in our machine.

According to Table III, the proposed framework is able to improve total vias' count and wirelength after first and tenth iterations compared to baseline and the state-of-the-art. In only two case benchmarks (ISPD2018_test2 and test3), [18] has the higher improvement than our method, where the circuits is less congested. This means that our proposed method is performing more efficient in congested circuits than less congested ones. In average, CR&P is able to reduce the number of vias and wirelength by 2.06% and 0.14%. All these improvements are achieved while we are not generating any further DRVs compared to the baseline. According to ISPD-2018 contest evaluation metrics [5], the weight of a unit of a wire is 0.5 while the weight of a unit of a via is 2. This means that via insertion weight is 4 times expensive than wire insertion. Therefore, vias significantly play dominant role in our cost equation model (section IV.A). In other words, improvement of a path costs depends more on via reduction than wirelength reduction. That's why we have achieved much bigger improvement in vias compared to wirelength.

This level of improvement in the proposed method compared to the state-of-the-art can be elaborated in two main reasons. The first reason is the way each work modeled its cost function. In [18], the cost function is only modeled by the length and a number of detours in each route. While, in our model, the cost function has the penalty of congestion and encourages cells to be moved to less congested areas. The second reason is that, in [18], all cells are tried to be moved to their median with no priority while in our model the critical cells are selected based on their path cost. These two reasons are the main reasons that the proposed method could achieve a better results in detailed routing compared to the state-of-the-art.

In terms of runtime, Figure 2 shows that CR&P in first iteration adds a small margin to the runtime which is much less than the state-of-the-art [18]. Even after ten iterations this runtime increase by a constant value and it is not increased exponentially. This means that the runtime is managable over the iterations. Also, according to Figure 3 where the percentage of different steps of the proposed framework is shown, CR&P in most of the benchmarks has less runtime than global router. It can be seen, in CR&P, the estimation of candidates costs has the highest overhead compared to other steps. This is because CR&P is using 3D global router to investigate the cost of each movement.

## VI. Conclusions

In this paper, a Co-operation between Routing and Placement (CR&P) is proposed. To prove the efficiency of the proposed model, we performed all experiments on the available benchmarks of ISPD-2018 Contest, which is compatible with detailed routing and investigated the results by the official evaluator of ISPD-2018 contest. These results show that the proposed model can improve the via insertion and wirelength of the detailed routing by 2.06% and 0.14% on average through 10 iterations. Also, CR&P by using iterative approach gets a reasonable runtime to achieve this improvement. The proposed framework can assist other routing algorithms to work collaboratively with the placement algorithms, making both placement and routing more agile and efficient.

TABLE III: Comparison of wirelength, Design Rule Violations (DRVs), and total via counts evaluated by an official evaluator of ISPD-2018 contest [5] between: CUGR+TritonRoute (column Baseline (BL)); state-of-the-art [18]; CUGR+CR&P using ILP-based legalizer+TritonRoute with 1 iteration (Ours,k=1) and 10 iterations (Ours,k=10).

| Benchmarks | Total Wirelength | | | | Design Rule Violations | | | | Total Via Count | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BL x $10^9$ | Improvement (%) | | | BL (#) | Number of Violations (#) | | | BL x $10^5$ | Improvement (%) | | |
| | | [18] | Ours, K=1 | Ours, K=10 | | [18] | Ours, K=1 | Ours, K=10 | | [18] | Ours, K=1 | Ours, K=10 |
| ispd18_test1 | **0.17** | -6.03 | -0.02 | -0.35 | 0.0 | 0.0 | 0.0 | 0.0 | 0.4 | 0.73 | 0.07 | **0.79** |
| ispd18_test2 | **3.12** | -0.40 | -0.10 | -0.10 | 0.0 | 0.0 | 0.0 | 0.0 | 3.8 | **5.19** | 1.37 | 2.86 |
| ispd18_test3 | 3.51 | **0.12** | 0.00 | 0.09 | 0.0 | 0.0 | 0.0 | 0.0 | 3.8 | **3.85** | 1.35 | 3.02 |
| ispd18_test4 | **5.24** | -0.05 | -0.01 | 0.00 | 0.0 | 0.0 | 0.0 | 0.0 | **7.2** | -0.90 | -0.05 | 0.00 |
| ispd18_test5 | 5.49 | 0.02 | 0.04 | **0.13** | 0.0 | 0.0 | 0.0 | 0.0 | 8.4 | -1.82 | 0.93 | **2.70** |
| ispd18_test6 | 7.10 | -0.22 | 0.08 | **0.48** | 0.0 | 0.0 | 0.0 | 0.0 | 12.8 | -0.24 | 0.22 | **0.91** |
| ispd18_test7 | 12.91 | -0.04 | 0.08 | **0.31** | 2.0 | 2.00 | **0.0** | **0.0** | 20.9 | -0.01 | 1.16 | **2.64** |
| ispd18_test8 | 13.02 | -0.01 | 0.20 | **0.58** | 0.0 | 2.00 | 0.0 | 0.0 | 21.4 | -0.05 | 1.24 | **3.34** |
| ispd18_test9 | 10.80 | -0.05 | 0.07 | **0.22** | 0.0 | 0.00 | 0.0 | 0.0 | 21.5 | -0.05 | 1.62 | **4.26** |
| ispd18_test10 | 13.54 | **Failed** | 0.02 | **0.02** | 0.0 | **Failed** | 0.0 | 0.0 | 23.0 | **Failed** | 0.03 | **0.03** |
| Avg | 7.49 | -0.74 | 0.04 | **0.14** | - | - | - | **-** | 12.31 | 0.74 | 0.80 | **2.06** |

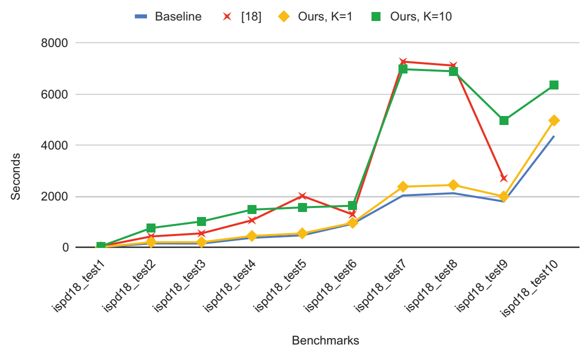

Fig. 2: Runtime comparison between baseline, state-of-the-art [18], the proposed framework with k=1 and 10. It is important to mention that [18] is failed in ispd18_test10.

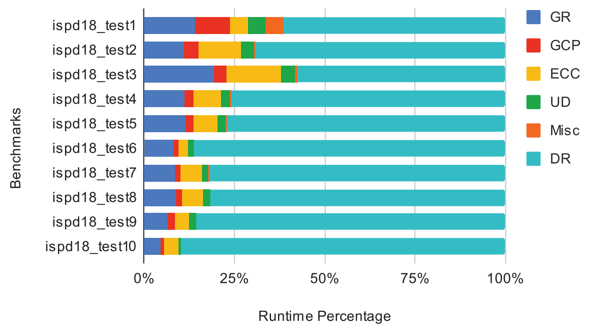

Fig. 3: The percentage of runtime breakdown of CUGR+CR&P+DetailedRoute. In this figure: CUGR (GR); Generate Candidate Position (GCP); Estimating Candidates Cost (ECC); Update Database(UD); Other steps of CR&P(Misc); DetailedRoute(DR).

## References

[1] C. Alpert and et al, *"Handbook of Algorithms for Physical Design Automation"*, 1st ed. Auerbach Publications, Nov. 2008.

[2] A. Kahng and et al, *"VLSI Physical Design: From Graph Partitioning to Timing Closure"*, 2011th ed. Springer Netherlands, 2011.

[3] K. Hu and et al, "ICCAD-2020 cad contest in routing with cell movement," in *ICCAD2020*, 2020, pp. 1–4.

[4] A. Kahng, "Advancing placement," in *ISPD2021*, March 2020, pp. 15–22.

[5] S. Mantik and et al, "Ispd 2018 initial detailed routing contest and benchmarks," in *ISPD2018*, 2018, pp. 140–143.

[6] N. K. Darav and et al, "Eh? placer: A high-performance modern technology-driven placer," *ACM TODAES*, vol. 21, no. 3, pp. 1–27, 2016.

[7] T. Lin and et al, "Polar: A high performance mixed-size wirelengh-driven placer with density constraints," *TCAD*, vol. 34, no. 3, pp. 447–459, 2015.

[8] X. He and et al, "Ripple 2.0: High quality routability-driven placement via global router integration," in *50th DAC*, 2013, pp. 1–6.

[9] C.-C. Huang and et al, "Ntuplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints," *TCAD*, vol. 37, no. 3, pp. 669–681, 2018.

[10] Y. Lin and et al, "Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement," in *56th DAC*, 2019, pp. 1–6.

[11] F. Gessler and et al, "A shared-memory parallel implementation of the replace global cell placer," in *VLSID*, 2020, pp. 78–83.

[12] Y. Lin and et al, "Abcdplace: Accelerated batch-based concurrent detailed placement on multithreaded cpus and gpus," *TCAD*, vol. 39, no. 12, pp. 5083–5096, 2020.

[13] L. Yibo and et al, "Mrdp: Multiple-row detailed placement of heterogeneous-sized cells for advanced nodes," *TCAD*, vol. 37, no. 6, pp. 1237–1250, 2018.

[14] J. Chen and et al, "Mixed-cell-height detailed placement considering complex minimum-implant-area constraints," *TCAD*, pp. 1–1, 2020.

[15] N. Viswanathan and et al, "Fastplace: efficient analytical placement using cell shifting, iterative local refinement,and a hybrid net model," *TCAD*, vol. 24, no. 5, pp. 722–733, 2005.

[16] A. Kahng, L. Wang, and B. Xu, "Tritonroute: The open-source detailed router," *TCAD*, vol. 40, no. 3, pp. 547–559, 2021.

[17] K. Hu and et al, "ICCAD-2021 cad contest in routing with cell movement," in *ICCAD2021*, 2021, pp. 1–4.

[18] T. A. Fontana and et al, "Ilp-based global routing optimization with cell movements," in *ISVLSI*, 2021, pp. 25–30.

[19] "Lef/def language reference," 2009, http://www.ispd.cc/contests/18/lefdefref.pdf.

[20] J. Chen and et al, "Toward optimal legalization for mixed-cell-height circuit designs," in *DAC*, 2017, pp. 1–6.

[21] J. Liu and et al, "CUGR: detailed-routability-driven 3d global routing with probabilistic resource model," in *DAC*, 2020, pp. 1–6.

[22] Y. J. Chang and et al, "Nthu-route 2.0: A fast and stable global router," in *ICCAD*, 2008, pp. 338–343.

[23] C. Sechen and et al, "The timberwolf placement and routing package," *IEEE Journal of Solid-State Circuits*, vol. 20, no. 2, pp. 510–522, 1985.

[24] "The boost c++ libraries." [Online]. Available: https://www.boost.org/users/history/version_1_68_0.html

[25] "IBM ILOG CPLEX." [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio