POLAR 2.0: An Effective Routability-Driven Placer

Tao Lin
Iowa State University
Ames, Iowa 50011
tlin@iastate.edu

Chris Chu
Iowa State University
Ames, Iowa 50011
cnchu@iastate.edu

ABSTRACT

A wirelength-driven placer without considering routability would lead to unroutable results. To mitigate routing congestion, there are two basic approaches: (1) minimizing the routing demand; (2) distributing the routing demand properly. In this paper, we propose a new placer POLAR 2.0 emphasizing both approaches. To minimize the routing demand, POLAR 2.0 attaches very high importance to maintaining a good wirelength-driven placement in the global placement stage. To distribute the routing demand, cells in congested regions are spread out by a novel routability-driven rough legalization in a global manner and by a history based cell inflation technique in a local manner. The experimental results based on ICCAD 2012 contest benchmark suite show that POLAR 2.0 outperforms all published academic routability-driven placers.

1. INTRODUCTION

Placement is one of the most important and ancient problems in Electronic Design Automation (EDA). Its quality has been greatly improved during the last two decades. However, with the gradually increasing scale of design, a high quality while extremely fast placer is still in urgent need. Besides, [1, 2] pointed out that the commonly used wirelength metric might not capture the key aspects of solution quality. Overemphasis of wirelength as in traditional placement formulation inevitably results in bad quality in other metrics such as power, timing and routability, although optimizing wirelength is beneficial to those metrics to some extent.

Among varieties of metrics, routability is now becoming more and more important due to a significant mismatch between the objectives of wirelength and routing congestion. A wirelength-driven placer without considering routability usually leads to irresolvable routing congestion problem. In the recent years, a series of contests (ISPD 2011, DAC 2012, ICCAD 2012) were held to promote the research in routability-driven placement, and some academic routability-driven placers [3–9] such as coPR [4], Ripple 2.0 [7] and NTUplace4h [9] were produced. Besides, SRP [10] and Ropt [11] try to refine routability-driven placement by using the routing information feedbacked by global router.

There are two challenges in routability-driven placement problem. The first challenge is that the routing congestion is expected to be detected accurately in short runtime. Since directly invoking the whole routing process in the global placement stage is very time consuming, many routing congestion estimation methods were proposed. Fast global routers such as FastRoute [12] and BFG-R [13] have been incorporated into some placers [3–5, 7] to achieve relatively accurate estimation. RUDY [14] adopts a L-shaped probability model and half perimeter wirelength (HPWL) to estimate the actual routing demand. Ripple [5] and NTU-Place4h [9] further extend RUDY's method. Ripple utilizes rectilinear minimum spanning tree (RSMT) to replace HWPL, while NTUPlace4h applies Guassian smoothing to smooth the L-shaped approximation model. The second challenge is that the cells within routing congestion region should be spread out to balance the routing supply and routing demand. Many placers [3–7, 15–17] apply cell inflation. CROP [18] adjusts the boundary of each G-Cell to make sure it has enough available area and routing supply. NTU-Place4h formulates routing congestion as an additional constraint into its non-linear programming framework.

In this paper, we propose a new routability-driven placer, POLAR 2.0, which mitigates routing congestion by the following two basic approaches: (1) minimizing the routing demand; (2) spreading the routing demand properly. To minimize the routing demand, the new placer attaches very high importance to maintaining a good wirelength-driven placement in the global placement stage. To distribute the routing demand, cells in congested regions are spread out by a novel routability-driven rough legalization in a global manner and by a history based cell inflation technique in a local manner. Experimental results on ICCAD 2012 Contest benchmark suite show that POLAR 2.0 outperforms all published academic routability-driven placers. Compared with SimPLR [3], CoPR [18], Ripple 2.0 [7] and NTUPlace4h [9], our placer respectively achieves 5.1%, 3.0%, 2.8% and 1.1% improvement on the scaled HPWL.

The key ideas of this paper are highlighted as follows.

- Maintaining a good wirelengh-driven placement is attached very high importance in our routability-driven optimization flow, in order to minimize the routing demand.
- A novel routability-driven rough legalization is applied

DAC '14, June 01 - 05 2014, San Francisco, CA, USA

to distribute the routing demand. In this technique, routing congestion on the horizontal and vertical directions are handled separately, and the routing congestion regions are effectively spread out in a global manner.

- A history based cell inflation is adopted as a complement in a local manner. Different from some previous cell inflation techniques, the inflation amount is accumulated and kept until the global placement is finished.

The remainder of this paper is organized as follows. Section 2 reviews the preliminaries. Section 3 presents the framework of POLAR 2.0. Section 4 illustrates POLAR 2.0's algorithm. Section 5 shows the experimental results. Finally, Section 6 are the conclusions.

2. PRELIMINARIES

The routability-driven placement relies on the traditional wirelength-driven placement engine. A circuit can be represented by a hypergraph $G = (V, E)$, where V is the set of cells and E is the set of nets. The placement tries to determine the physical positions of the cells without violating the placement density constraints. We denote the x-coordinates of cells by a vector $\mathbf{x} = (x_1, x_2, \cdots, x_{|V|})$, and y-coordinates by $\mathbf{y} = (y_1, y_2, \cdots, y_{|V|})$, the objective is to minimize the half-perimeter wirelength (HPWL):

$$\mathrm{HPWL}(\mathbf{x}, \mathbf{y}) = \Sigma_{e \in E}[\max_{i \in e} x_i - \min_{i \in e} x_i + \max_{i \in e} y_i - \min_{i \in e} y_i] \quad (1)$$

POLAR 2.0 is a natural extension of POLAR [19]. POLAR adopts the rough legalization idea [20], but rough legalization is realized in a different manner: for each density hotspot, POLAR enumerates the optional reasonable windows to find the smallest one that can accommodate all the cells within it, and then a tree-based bisection method spreads the cells evenly.

A good wirelength-driven placement is beneficial to router, since the wirelength has direct influence on the routing demand. Usually, better total wirelength means less total routing demand. However, excessively optimizing the wirelength would lead to routing congestion, since the cells which have lots of connections are pulled together resulting into that the local routing demand substantially exceeds the local routing supply.

Routability-driven placement essentially is to distribute the routing demand rationally according to the routing supply. To simplify the routability formulation, the routing resources are given by a 2-D $m \times n$ mesh, since 3-D mesh can be easily transformed to 2-D mesh by accumulating the routing resources of different layers. The grid in the 2-D mesh is usually called G-Cell, which is denoted by a coordinate (x, y), where $0 \leq x < m$ and $0 \leq y < n$. For any pair of adjacent horizontal G-Cells, the connecting routing channel is called H-edge, while for any pair of adjacent vertical G-Cells, the connecting routing channel is called V-edge. Therefore, for each G-Cell, it is at most associated to two H/V-edges respectively. And the number of trunks in each H/V-edge is fixed as the routing supply. As shown in Fig. 1(a), it is a 4×4 2-D mesh, for the G-Cell $(1, 2)$, the two associated H-edges are coloured red, while the two associated V-edges are coloured blue. The horizontal/vertical routing supply(H/V-supply) of G-Cell is defined as the total number of trunks in its associated H/V-edges. The horizontal/vertical routing demand(H/V-demand) of G-Cell is defined as the number of wires that goes through its associated H/V-edges, as shown in Fig. 1(b), the H-demand and V-demand of G-Cell $(2, 2)$ are respectively 2 and 2.

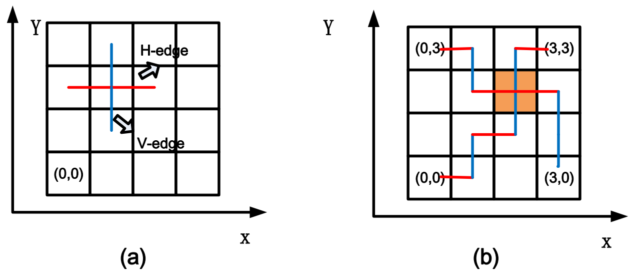


Figure 1: The 2-D routing mesh

To capture a clear picture of design routability, [21] introduced average congestion of G-Cell edges(ACE). Optimizing ACE not only minimizes the total routing overflow but also produces rational distribution of the routing demand. In this paper, the same as previous works [3, 4, 7, 9], the objective of routability-driven placement is to minimize both HPWL and ACE.

3. OVERVIEW

There are two basic approaches to optimize the routability: (1) minimizing the routing demand; (2) spreading the routing demand properly. On one hand, since routing overflow is calculated by routing demand minus routing supply, roughly speaking, minimizing routing demand is beneficial to decrease routing overflow. On the other hand, for each H/V-edge, its routing demand is expected to be not higher than its routing supply. Therefore, the distribution of routing demand should be done properly based on the known distribution of routing supply.

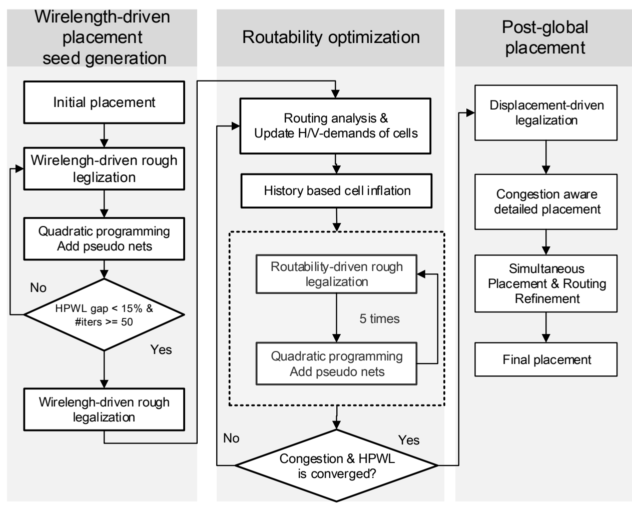


Figure 2: The overview of the new placer.

POLAR 2.0 targets on the above two approaches to optimize the routability. Its overview is presented in Fig. 2.

The whole placement is partitioned into three stages: (1) wirelength-driven seed placement generation; (2) routability-driven cell spreading; (3) post-global placement.

To minimize the routing demand, in POLAR 2.0, maintaining a good wirelength-driven placement is attached very high importance. In the first stage, POLAR [19] is used to generate a good wirelength-driven seed placement: the global placement loop of POLAR is not stopped until the number of iterations is greater than 50 and the gap between the upper bound wirelengh and the lower bound wirelength is less than 15%. Besides, in routability-driven cell spreading stage, for each round of routing analysis, the steps of quadratic programming and routability-driven rough legalization are iterated 5 times to maintain a good wirelength-driven placement.

In the routability-driven cell spreading stage, routing analysis is applied to calculate H/V-demands of G-Cells, and to model the migration of routing demand, H/V-demands of G-Cells are amortized to H/V-demands of movable cells. Based on movable cells' H/V-demands, POLAR 2.0 simultaneously distributes area demand, horizontal routing demand and vertical routing demand by the following two approaches. Firstly, in a global manner, we propose a routability-driven rough legalization which is a natural extension of POLAR's [19] rough legalization idea. During routability-driven rough legalization, both area and routing congestion hotspots are detected. For each hotspot, the smallest window (expansion region) which has enough area and routing resources to satisfy all demands of the enclosed cells is searched by enumeration. Then a tree based bisection spreading technique is applied to distribute those cells within the window. Secondly, to avoid local routing congestion when distributing the cells within the window, a history based cell inflation technique is proposed. The details of this stage are presented in Section 4.

Finally, in the post-global placement stage, we adopt the same method as Ripple 2.0's [7], which has three components: (1) displacement-driven legalization, (2) congestion aware detailed placement and (3) simultaneous placement and routing refinement [10].

4. ROUTABILITY OPTIMIZATION

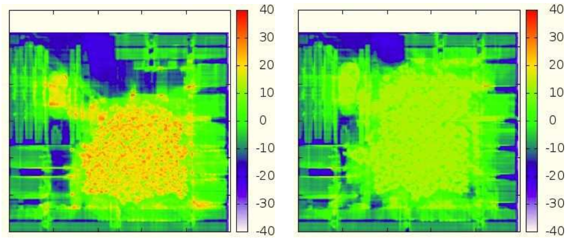
4.1 Routing analysis

4.1.1 Routing supply calculation

3-D routing can easily be transformed to 2-D routing by accumulating the total routing resources of all metal layers. Since different metal layers have different wire pitches, we need to sum up the number of tracks of each metal layer. In addition, there are many fixed routing blockages occupying the routing resources on the metal layers, the supply of H/V-edges need to exclude these blocked routing resources. The work [7] gives the details about the transformation from 3-D routing to 2-D routing.

4.1.2 Routing demand estimation

Different from the routing supply, the routing demand relays on the placement and routing solution. To calculate the exact routing demand, legalized placement and detailed routing are necessary. However, invoking legalization and detailed router during the global placement stage is very time consuming. Therefore, in POLAR 2.0, the routing demand is estimated based on roughly legalized placement and global routing instead: we use roughly legalized placement to calculate the pin locations, and then the congestion aware pattern routing of FastRoute [12] is applied to estimate the routing demand.



(a) (b)

Figure 3: The congestion map of benchmark superblue16 with/without using routability-driven rough legalization. (a) is the one with POLAR's [19] rough legalization instead of POLAR 2.0's routability-driven rough legalization during routability-driven cell spreading stage. Note that the congestion value is scaled from -40 to 40, higher value means more congestion. The congestion map was drew based on the results of FastRoute's pattern routing [12].

When the cells are moved, routing demand would be migrated. During the routability optimization, POLAR 2.0 maintains a good wirelength-driven placement, in which the relative positions of cells are only allowed to modified a little bit. Under this condition, for most of the cells, moving around would not change its H/V-demand much.

To trade off accuracy and runtime, we perform routing demand estimation infrequently. As shown in Fig. 2, the routing demands are updated only once every five times the placement solution is refined. To model the migration of routing demands, we associate the demands to movable cells by introducing two new attributes, the horizontal and vertical routing demand (H/V-demand), for each movable cell. Consider a movable cell i located in G-Cell j. Let the H/V-demand of G-Cell j be denoted by HD_j and VD_j, respectively, and the number of movable cells within G-Cell j be denoted by k_j. Then the H/V-demand of cell i, denoted respectively by hd_i and vd_i, is given by Formulas (2) and (3):

$$hd_i = \frac{HD_j}{k_j} \tag{2}$$

$$vd_i = \frac{VD_j}{k_j} \tag{3}$$

4.2 Routability-driven rough legalization

After routing analysis, each movable cell has three attributes: area demand, H-demand and V-demand. And the routability-driven placement essentially is to distribute these three demands based on the given supplies (available area, horizontal routing supplies, vertical routing supplies). To realize this goal in a global manner, we propose a novel routability-driven rough legalization technique.

The pseudo code of routability-driven rough legalization

Algorithm 1 Routability-driven rough legalization

Require: The H/V-demands of movable cells are known, placement is rough legalized.
Ensure: Good wirelengh is maintained, routability is optimized.

Detect the area/routing congestion hotspots; ▷ The method is similar to POLAR's [19]
for each hotspot s **do**
  $\Gamma = \emptyset$;
  for each window w whose geometrical center is the same as s **do**
    as = the available area of w;
    hs = total H-supplies of the G-Cells contained by w;
    vs = total V-supplies of the G-Cells contained by w;
    ad = total areas of the movable cells located in w;
    hd = total H-demands of the movable cells located in w;
    vd = total V-demands of the movable cells located in w;
    rt = the aspect ratio of w;
    if $as \geq ad$ && $hs \geq \alpha \times hd$ && $vs \geq \alpha \times vd$ && $rt \in \left[\frac{1}{3}, 3\right]$ **then**
      $\Gamma = \Gamma \cup \{w\}$;
    end if
  end for
  Distribute the cells within the minimal window of Γ by tree based bisection [19].
end for

is presented in Algorithm 1. In the original POLAR's [19] rough legalization, for each placement density hotspot, a minimal expansion window is found by enumerating the ones which have enough available area and reasonable aspect ratio, and then the cells are spread out evenly by a tree-based bisection within the chosen window. While, in the routability-driven rough legalization, not only placement density hotspots, but also routing congestion hotspots are detected. Besides, any expansion window w should have enough Horizontal and vertical routing supplies by satisfying the following two additional constraints.

$$\Sigma_{j \in w} HS_j \geq \alpha \times \Sigma_{i \in w} hd_i \quad (4)$$

$$\Sigma_{j \in w} VS_j \geq \alpha \times \Sigma_{i \in w} hd_i \quad (5)$$

Where HS_j and VS_j are respectively the routing supplies of the associated H-edges and V-edges of G-Cell j, α is a parameter due to the inaccuracy of routing estimation. When α is higher than 1, it means that the routing congestion is underestimated; on the contrary, α is less than 1 means that the routing congestion is overestimated. Experimental results verify that α is less than 1, mainly because the time consuming maze routing is not used during routing analysis in POLAR 2.0.

This routability-driven rough legalization technique can effectively distribute the routing demand by mitigating the routing demand to the places which have enough routing supply without being used. Fig. 3 shows the congestion map of benchmark superblue16 with/without this technique. It can be seen that with this technique, the routing congestion is spread out so that the scaled HPWL is significantly improved.

4.3 History based cell inflation

For any design, the distribution of area supply/demand, horizontal routing supply/demand and vertical routing supply/demand are usually not the same. The tree based bisection spreading technique used in routability-driven rough legalization only distributes cells evenly according to area supply/demand. Therefore, for some G-Cells, there are enough available areas to accommodate its enclosed cells, but may no enough horizontal/vertical routing resources to satisfy the routing demands of its enclosed cells. To avoid local routing congestion that routability-driven rough legalization cannot resolve, similear to previous works, the routing demands of some cells are transformed into inflated area by a history based cell inflation.

The pseudo code of history based cell inflation is presented in Algorithm 2. The principle is that only the movable cells located in the most congested G-Cells are inflated by a small ratio and the inflation is accumulated until the routability cannot be improved. Its insight is derived from the history based global routing technique [22] (In global routing, detour is not preferred, but usually inevitable. To route a design, [22] adds a big penalty to detour at the beginning to see whether all the nets can be routed without overflow. If the answer is no, then the detour penalty is decreased slightly and rerouteing is performed. This process is continued until all the nets are finally routed without overflow.) This approach is similar to other cell inflation techniques [3–7, 15–17] functionally. It is simple and works well according to our experimental results.

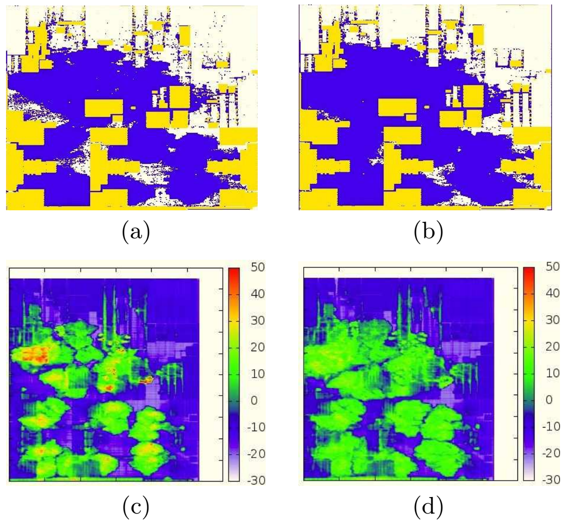


Figure 4: (a) and (b) are respectively the global placement just before/after routability optimization.(c) and (d) are the congestion map of benchmark superblue7 just before/after routability optimization. Note that the congestion value is scaled from -30 to 50, higher value means more congestion. The congestion map was drew based on the results of FastRoute's pattern routing [12].

5. EXPERIMENTAL RESULTS

POLAR 2.0 was implemented in C++ and complied by g++-4.7.2. The benchmarks of ICCAD 2012 contest [23] are ran on a Linux PC with Intel Xeon X5550 2.67GHz CPU and 16GB RAM to verify the efficiency of POLAR 2.0. Routability evaluation is performed by official script in the ICCAD

Algorithm 2 History based cell inflation

Require: Routing analysis is just done intermediately.
Ensure: The movable cells located in the most congested G-Cells are inflated by a small ratio.

$\phi = \emptyset$;
for any G-Cell j **do**
    HD_j = the H-demand of G-Cell j;
    VD_j = the V-demand of G-Cell j;
    HS_j = the H-supply of G-Cell j;
    VS_j = the V-supply of G-Cell j;
    if $HD_j > HS_j$ **then**
        add the ordered pair $(j, HD_j - HS_j)$ into ϕ;
    end if
    if $VD_j > VS_j$ **then**
        add the ordered pair $(j, VD_j - VS_j)$ into ϕ;
    end if
end for
Sort the ϕ based on the overflow (the second item of ordered pair) in descending order;
for each ordered pair t in the top 10% of sorted ϕ **do**
    for each movable cell i in G-Cell $t.first$ **do**
        Inflate the area of cell i by 10%;
    end for
end for

2012 contest [23]. The placement solution is routed by the designate global router-NCTRgr [24]. The scaled wirelength is calculated according to HPWL and ACE [21] penalty.

5.1 Runtime analysis

The runtime of our placer is shown in Table I. It is broken down into three components: wirelengh-driven placement seed generation, routability optimization which includes routing analysis, history based cell inflation and window based cell spreading, and post-global placement. On average, routability optimization seed generation takes 47% of the total runtime, while wirelengh-driven placement seed generation and post-global placement respectively takes 27% and 26% of the the total runtime. And during the routability optimization, routing analysis uses 6% , window based cell spreading uses 26%, and the rest (such as cell inflation and quadratic programming) uses 15% of the total runtime.

Besides, compared with pure wirelength-driven placer POLAR [19], the proposed routability-driven placer is only 2.26× slower. Considering the fact that routablity-driven placement problem is much more complex than pure wirelength-driven placement, POLAR 2.0 is very fast.

5.2 Compared with previous works

The solution quality (including HPWL and ACE [21]) of POLAR 2.0 is shown in Table II. It can be seen that the ACE [21] penalty is decreased to a very low level, which means that POLAR 2.0 can effective immigrate the routing congestion. Fig. 4 shows the global placement and routing congestion map of benchmark superblue7 just before/after the routability optimization stage, it can be seen that the shape of wirelengh-driven placement seed is roughly maintained, while the routing congestion is significant spread out.

As shown in Table III, our placers outperforms other academic routability-driven placers on ICCAD 2012 benchmark suite [23]. Considering the scaled HPWL, on average, our placer respectively achieves 5.1%, 3.0% , 2.8% and 1.1% improvement versus SimPLR [3], coPR [4], Ripple2 [7] and NTUPlace4 [9]. Considering the runtime, on average, we believe POLAR 2.0 is faster than SimPLR, coPR, Ripple2

Table 2: ACE [21] on ICCAD 2012 benchmarks [23].

Benchmark	ACE(%) [21]				RC	HPWL	sHPWL
	0.50	1.00	2.00	5.00	(%)	($\times 10^7$)	($\times 10^7$)
superblue1	102.48	101.24	100.62	100.25	101.15	2.72	2.82
superblue3	102.29	101.15	100.57	100.23	101.06	3.23	3.33
superblue4	102.08	101.04	100.52	100.21	100.96	2.18	2.24
superblue5	101.51	100.76	100.38	100.15	100.70	3.44	3.51
superblue7	101.77	100.88	100.44	100.18	100.82	3.97	4.07
superblue10	102.40	101.20	100.60	100.24	101.11	6.01	6.21
superblue16	102.81	101.41	100.70	100.28	101.30	2.61	2.72
superblue18	103.17	101.58	100.79	100.32	101.47	1.61	1.69

and NTUPlace4. [1]

6. CONCLUSIONS

In this paper, we propose a very simple and fast routability-driven placer, POLAR 2.0, which targets on mitigating routing congestion by the following two basic approaches: (1) minimizing routing demand by maintaining a good wirelength-driven placement; (2) spreading the routing demand properly by a novel routability-driven rough legalization and a history based cell inflation.

Experimental results show that even without applying many techniques that others proposed (such as narrow channel blocking [6, 9], routing path based inflation [4, 7], and reserving space around macros [11], etc), POLAR 2.0 yet outperforms all published academic routability-driven placers. For future work, we will investigate the use of those techniques.

Acknowledgments

This work is partially supported by NSF under grant CCF-1219100.

References

[1] C. J. Alpert, Z. Li, M. D. Moffitt, G.-J. Nam, J. A. Roy, and G. Tellez, "What makes a design difficult to route," ISPD '10, pp. 7–12, 2010.

[2] J. A. Roy, J. F. Lu, and I. L. Markov, "Seeing the forest and the trees: Steiner wirelength optimization in placemen," ISPD '06, pp. 78–85, 2006.

[3] M.-C. Kim, J. Hu, D.-J. Lee, and I. L. Markov, "A SimPLR method for routability-driven placement," ICCAD '11, pp. 67–73, 2011.

[4] J. Hu, M.-C. Kim, and I. L. Markov, "Taming the complexity of coordinated place and route," DAC '13, pp. 150:1–150:7, 2013.

[5] X. He, T. Huang, L. Xiao, H. Tian, G. Cui, and E. F. Y. Young, "Ripple: an effective routability-driven placer by iterative cell movement," ICCAD '11, pp. 74–79, 2011.

[1] For SimPLR [3], Ripple 2.0 [7] and NTUPlace4h [9], their results were referred from the ICCAD 2012 contest [23]; for coPR [4], its runtime was computed based on [4] which claimed that it was 1.01 slower than SimPLR [3]. Besides, the machine used in the ICCAD 12 contest [23] is with Intel Xeon X7560 2.27GHz CPU(much more expensive and powerful than the CPU used in our experimental environment) and 16GB memory. Therefore, the datas in Table III are relatively correct.

Table 1: Runtime breakdown on ICCAD 2012 benchmarks [23]. Runtime is measured in second.

Benchmark	Wirelengh-driven seed generation	Routability optimization			Post-global placement	Total runtime	time ratio POLAR 2.0/POLAR[19]
		Routing analysis	cell spreading	others			
superblue1	411	84	564	273	280	1612	2.49
superblue3	425	75	534	295	431	1760	2.33
superblue4	240	58	490	267	205	1260	2.36
superblue5	328	55	162	141	414	1100	1.69
superblue7	814	61	375	252	361	1863	1.66
superblue10	579	119	316	243	1689	2946	3.09
superblue16	334	41	240	121	292	1028	1.83
superblue18	266	116	420	248	228	1278	2.59
Normalize	0.27	0.06	0.26	0.15	0.26	1.00	2.26

Table 3: Comparison on ICCAD 2012 benchmarks [23]. Runtime is measured in second.

Benchmark	SimplR [3]		coPR [4]		Ripple2 [7]		NTTPlacer4 [9]		POLAR 2.0	
	sHPWL	Time	sHPWL	Time	sHPWL	Time	sHPWL	Time	sHPWL	Time
superblue1	2.79	2319	2.86	2453	2.89	10213	2.79	8759	2.82	1612
superblue3	3.44	2706	3.46	2603	3.60	15114	3.67	7193	3.33	1760
superblue4	2.43	1257	2.37	1816	2.27	8575	2.31	4866	2.24	1260
superblue5	3.60	2154	3.51	2345	3.49	10833	3.59	7322	3.51	1100
superblue7	4.31	3249	4.36	3570	4.29	23017	3.96	15005	4.07	1863
superblue10	6.91	4837	6.51	5098	5.98	26312	6.17	12352	6.21	2946
superblue16	2.86	1797	2.80	1234	2.84	9494	2.78	6024	2.72	1028
superblue18	1.82	1645	1.68	1342	1.84	10989	1.64	4622	1.69	1278
Normalize	1.051	1.54	1.030	1.56	1.0282	8.83	1.0111	5.22	1.000	1.00

[6] J. Cong, G. Luo, K. Tsota, and B. Xiao, "Optimizing routability in large-scale mixed-size placement," ASP-DAC '13, 2013.

[7] X. He, T. Huang, W.-K. Chow, J. Kuang, K.-C. Lam, W. Cai, and E. F. Y. Young, "Ripple 2.0: high quality routability-driven placement via global router integration," DAC '13, pp. 152:1–152:6, 2013.

[8] M.-K. Hsu, S. Chou, T.-H. Lin, and Y.-W. Chang, "Routability-driven analytical placement for mixed-size circuit designs," ICCAD '11, pp. 80–84, 2011.

[9] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, T.-C. Chen, and Y.-W. Chang, "Routability-driven placement for hierarchical mixed-size circuit designs," DAC '13, pp. 151:1–151:6, 2013.

[10] X. He, W.-K. Chow, and E. F. Young, "SRP: simultaneous routing and placement for congestion refinement," ISPD '13, pp. 108–113, 2013.

[11] W.-H. Liu, C.-K. Koh, and Y.-L. Li, "Optimization of placement solutions for routability," DAC '13, pp. 153:1–153:9, 2013.

[12] Y. Xu, Y. Zhang, and C. Chu, "FastRoute 4.0: global router with efficient via minimization," ASP-DAC '09, pp. 576–581, 2009.

[13] J. Hu, J. A. Roy, and I. L. Markov, "Completing high-quality global routes," ISPD '10, pp. 35–41, 2010.

[14] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," DATE '07, pp. 1226–1231, 2007.

[15] W. Hou, H. Yu, X. Hong, Y. Cai, W. Wu, J. Gu, and W. H. Kao, "A new congestion-driven placement algorithm based on cell inflation," ASP-DAC '01, pp. 605–608, 2001.

[16] U. Brenner and A. Rohe, "An effective congestion driven placement framework," ISPD '02, pp. 6–11, 2002.

[17] J. A. Roy, N. Viswanathan, G.-J. Nam, C. J. Alpert, and I. L. Markov, "CRISP: congestion reduction by iterated spreading during placement," ICCAD '09, pp. 357–362, 2009.

[18] Y. Zhang and C. Chu, "CROP: fast and effective congestion refinement of placement," ICCAD '09, pp. 344–350, 2009.

[19] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev, "POLAR: Placement based on novel rough legalization and refinement," ICCAD '13, 2013.

[20] M.-C. Kim, D.-J. Lee, and I. L. Markov, "SimPL: an effective placement algorithm," ICCAD '10, pp. 649–656, 2010.

[21] N. Viswanathan, C. Alpert, C. Sze, Z. Li, and Y. Wei, "GLARE: global and local wiring aware routability evaluation,"

[22] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for fpgas," FPGA '95, pp. 111–117, 1995.

[23] N. Viswanathan, C. Alpert, C. Sze, Z. Li, and Y. Wei, "ICCAD-2012 CAD contest in design hierarchy aware routability-driven placement and benchmark suite," ICCAD '12, pp. 345–348, 2012.

[24] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "Multi-threaded collision-aware global routing with bounded-length maze routing," DAC '10, pp. 200–205, 2010.