Techniques for Scalable and Effective Routability Evaluation

Yaoguang Wei, University of Minnesota
Cliff Sze, IBM Austin Research Laboratory
Natarajan Viswanathan, IBM Systems and Technology Group
Zhuo Li, IBM Austin Research Laboratory
Charles J. Alpert, IBM Austin Research Laboratory
Lakshmi Reddy, IBM Systems and Technology Group
Andrew D. Huber, IBM Systems and Technology Group
Gustavo E. Tellez, IBM Systems and Technology Group
Douglas Keller, IBM Systems and Technology Group
Sachin S. Sapatnekar, University of Minnesota

Routing congestion has become a critical layout challenge in nanoscale circuits since it is a critical factor in determining the routability of a design. An unroutable design is not useful even though it closes on all other design metrics. Fast design closure can only be achieved by accurately evaluating whether a design is routable or not early in the design cycle. Lately, it has become common to use a "light mode" version of a global router to quickly evaluate the routability of a given placement. This approach suffers from three weaknesses: (i) it does not adequately model local routing resources, which can cause incorrect routability predictions that are only detected late, during detailed routing, (ii) the congestion maps obtained by it tend to have isolated hot spots surrounded by noncongested spots, called "noisy hot spots", which further affects the accuracy in routability evaluation, (iii) the metrics used to represent congestion may yield numbers that do not provide sufficient intuition to the designer; moreover, they may often fail to predict the routability accurately. This paper presents solutions to these issues. First, we propose three approaches to model local routing resources. Second, we propose a smoothing technique to reduce the number of noisy hot spots and obtain a more accurate routability evaluation result. Finally, we develop a new metric which represents congestion maps with higher fidelity. We apply the proposed techniques to several industrial circuits and demonstrate that one can better predict and evaluate design routability, and congestion mitigation tools can perform much better to improve the design routability.

Categories and Subject Descriptors: B.7.2 [**Hardware**]: Integrated Circuit—*Design Aids - Routing*

General Terms: Algorithms, Design, Experimentation, Measurement

Additional Key Words and Phrases: Physical design, Routing, Routability evaluation, Local resource modeling, Congestion metric

1. INTRODUCTION

Routability has become an increasingly important and difficult issue in nanometer-scale VLSI designs, and must be addressed across the entire physical synthesis tool stack. This in turn requires fast, yet reasonably accurate techniques to identify routing-challenged regions (hot spots) for routability optimization. This work focuses on the two key components of routability evaluation: (a) the method

This research was conducted when Y. Wei worked as an intern at the IBM Austin Research Laboratory, Austin, TX. Some initial results of this work were published in the proceedings of the Design Automation Conference [Wei et al. 2012].

Author's addresses: Y. Wei and S. S. Sapatnekar, Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN; C. Sze, Z. Li and C. J. Alpert, IBM Austin Research Laboratory, Austin, TX; N. Viswanathan, IBM Systems and Technology Group, Austin, TX; L. Reddy, A. D. Huber and D. Keller, IBM Systems and Technology Group, Hopewell Junction, NY; G. E. Tellez, IBM Systems and Technology Group, Burlington, VT.

DOI: http://dx.doi.org/10.1145/0000000.0000000

used to analyze the congestion of a given placement, and (b) the metric(s) used to score or represent the congestion.

1.1. Congestion analysis techniques

Congestion analysis is related to, but different from, routing. The basic purpose of routing is to find paths to connect all the nets to achieve a correct electrical connection of the circuit. Routing is traditionally divided into two main stages due to its complexity: global routing and then detailed routing. In the global routing stage, the routing region is partitioned into global routing cells (g-cells) and only the g-cell-to-g-cell paths are computed for all the nets (see Section 2 for more details). Next, detailed routing computes the pin-to-pin connection for all the nets, guided by the coarse paths from global routing. Further, detailed routing is constrained by complex design rules, which are usually ignored by global routing, in order to ensure manufacturability. Unlike routing, the goal of congestion analysis is to predict routability, identify routing hot spots, and provide designers or optimization tools fast feedback on congestion to improve routability. In congestion analysis, a congestion map (e.g., Fig. 1(a)), i.e., a region-wise color-coded plot that denotes the congestion in each region of a design, can be generated and used to visualize the congestion level in the design. A thorough discussion about the differences between congestion analysis and routing can be found in [Alpert and Tellez 2010].

Typical approaches for congestion analysis can be categorized as follows:

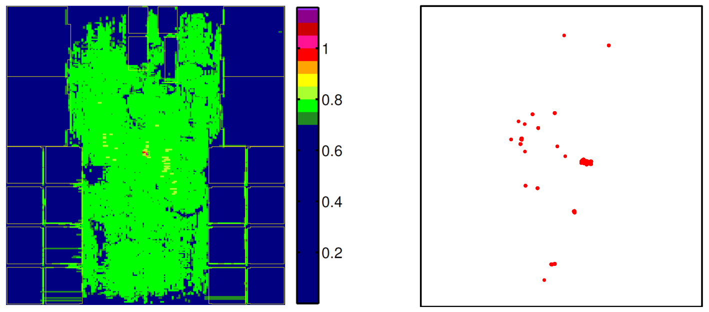
(1) Taking a design through detailed routing to determine whether it is routable or not.
(2) Using a probabilistic congestion estimation procedure without performing any routing [Lou et al. 2002; Westra et al. 2004].
(3) Performing fast global routing and using its solution to perform congestion analysis [Pan and Chu 2007; Roy et al. 2009; Shojaei et al. 2011; Hu et al. 2013].

In principle, an approach based on detailed routing estimates is the most accurate, but is very time-consuming and impractical during the early stages of design closure. Probabilistic methods are highly inaccurate and fail to capture the behavior of global routing, especially in modern designs with numerous IP blockages and a large number of metal layers with varying width and spacing. Lately, the third method has become more attractive and mainstream due to the advent of fast, high-quality global routers [Xu et al. 2009; Chang et al. 2008; Cho et al. 2007; Chen et al. 2009; Wu et al. 2010]. Although global-routing-based congestion analysis provides a happy medium between probabilistic analysis and detailed routing, it suffers from two key drawbacks.

The first drawback in global-routing-based congestion analysis is that global routing solutions cannot effectively predict the problematic regions identified by detailed routing, since they do not effectively model local routing congestion. Here, *local routing congestion*, or simply, *local congestion*, refers to the congestion that does not appear in global routing but shows up in detailed routing. This mismatch appears mainly because the *local routing resources*, or simply, *local resources*, used for *local routing* in detailed routing are not modeled in global routing (see Section 3.1 for detailed discussions).

Roughly speaking, local routing refers to the routing performed in one g-cell. Local routing clearly consumes varying amounts of routing resources depending on the factors such as design rules, the size of the g-cell, and pin density. Fig. 1 demonstrates a concrete instance where ignoring local resources in global routing can lead to significant misprediction of design routability. Without considering local resources, the congestion analyzer only sees very few congestion hot spots in the "combined" congestion map[1] (the spots in yellow, orange, red or pink color in Fig. 1(a)), and cannot predict the locations of opens/shots in detailed routing (the red dots in Fig. 1(b)), where detailed routing cannot complete. Hence, to achieve more accurate routability evaluation, local resources must be modeled

[1]The "combined" congestion map combines all the layers by showing the maximal congestion among all the layers. All the congestion plots in this paper without a qualifier show the combined maps. Moreover, the color map shown in Fig. 1(a) applies to all the subsequent congestion plots skipping a color bar.



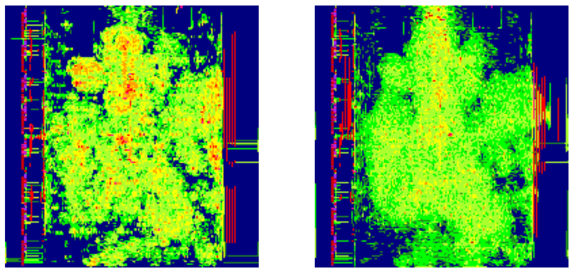
(a) Combined congestion map from a global-routing-based congestion analyzer

(b) The open/short map from a detailed router

Fig. 1. Without considering local resources, global-routing-based congestion analyzers cannot predict the locations of opens/shorts well.

in global routing. Further, any such method should be *flexible* enough to enable it to be adjusted in a straightforward manner from one technology to the next (as design rules are different for each technology). It is also desirable that the local resource model has good scalability on g-cell size so that it can be applied with different g-cell sizes.

The other drawback in global-routing-based congestion analysis is that the routing solutions from the congestion analyzers tend to have hot spots surrounded by noncongested spots, called "noisy" hot spots (further discussed in Section 4), which can usually be mitigated easily by further routing effort, and such noisy hot spots bring inaccuracy to the congestion analysis. The noisy hot spots appear mainly because of the following factor. Due to a large number of invocations in the design flow, congestion analysis tools tend to run very fast by limiting the routing effort, e.g., by limiting the extent to which a net can detour [Alpert and Tellez 2010]. However, during the routing stage, many of these constraints are naturally removed since nets are allowed greater flexibility in detouring. For example, Fig. 2 shows the congestion maps on the design ckt_i from an industrial routing tool in two modes: congestion analysis mode and global routing mode. We observe that in Fig. 2(a), there are quite a few noisy hot spots and they are dissolved by further routing effort in global routing mode (Fig. 2(b)). These hot spots are "noise" that prevents us from obtaining more accurate routability evaluation, and should be addressed properly.



(a) Congestion analysis mode

(b) Global routing mode

Fig. 2. Congestion maps from an industrial routing tool in two modes.

1.2. Metrics to score or represent congestion

Visual inspections of congestion plots, or congestion maps, often serve as a first-order method to compare the routability of different design points. However, optimization tools and designers also require a single metric that can accurately score or represent the design congestion. Commonly-used metrics in academia and industry can be categorized as follows:

Overflow-based metrics include total overflow and maximal overflow that measure the excess of the routing usage over routing capacity on the global routing edges in a global routing graph (defined in Section 2). These metrics do not provide sufficient intuition (e.g., how good/bad is an overflow of $14,253$?), which makes it difficult to quantify how much better one design point is versus another. Further, they may even fail to predict the routability correctly in some cases, as will be demonstrated in Section 7.3.

Net-congestion-based metrics [Alpert and Tellez 2010] include[2]: (a) $ACN(x)$, the average net congestion, defined as the average congestion of the top $x\%$ congested nets, where the congestion of a net is the maximum congestion among all the global routing edges traversed by the net. (b) $WCI(y)$, the worst congestion index, defined as the number of nets with congestion greater than or equal to $y\%$. In practice, $ACN(20)$, $WCI(90)$ and $WCI(100)$ have been used to evaluate routability. The main issue with these metrics is that they fail to differentiate between a net spanning a single congested global routing edge and one that spans multiple congested edges.

In this paper, we propose techniques to enhance the accuracy and effectiveness of routability evaluation. Our key contributions include:

— A study of the inaccuracies in existing global-routing-based congestion analyzers, specifically due to the lack of local routing resource modeling and the existence of noisy hot spots in the congestion maps.
— An analysis of the weaknesses in existing metrics to score or represent design congestion.
— Methods to model and incorporate the effects of local routing resource usage during global routing. Compared with approaches without modeling local routing resources, our methods improve congestion analysis in three aspects: (a) significant improvement in the accuracy of congestion analysis, (b) better prediction of detailed routing issues such as opens and shorts, and (c) accelerated congestion analysis with a larger g-cell size.
— A smoothing technique that reduces the noise in congestion maps and further improves the accuracy of routability evaluation.
— A new congestion metric that provides better intuition and represents the design congestion with high fidelity. This metric has been used in DAC 2012 placement contest [Viswanathan et al. 2012].
— Detailed empirical validation of our proposed techniques on advanced industrial designs.

The rest of this paper is organized as follows. We begin by presenting background and definitions in Section 2. Next, we present our methods for modeling local routing congestion in Section 3, and discuss the smoothing technique proposed to filter out the noise in routability evaluation in Section 4, followed by a description of our new metric for routability evaluation in Section 5. In Section 6, we discuss how our proposed techniques can be integrated together into the traditional routability evaluation process. Empirical validation and concluding remarks are provided in Section 7 and 8, respectively.

2. PRELIMINARIES

Typically, during global routing, the chip is tessellated into rectangular grids (or *g-cells*), and the global routing graph (GRG), $G = (V, E)$, is constructed. A node in V represents a g-cell in the layout, and an edge (called a *g-edge*) in E denotes the boundary between two adjacent g-cells. An example of the GRG is shown in Fig. 3.

We now introduce some notation and terms that will be used in the remainder of this paper. For each edge e in the GRG, we define c_e as edge capacity — the total or maximal capacity of the edge, b_e as blockage usage, and w_e as the routing demand on the edge. In global routing, c_e, b_e, and w_e are generally expressed in the number of routing tracks, where a routing track is the routing resource taken by a single wire passing through an edge in the GRG. We further define total routing usage u_e

[2]We name the metrics differently from [Alpert and Tellez 2010] to facilitate later references.

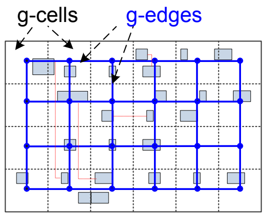


Fig. 3. Global routing graph (GRG).

as sum of b_e and w_e. Then the overflow of an edge e can be defined as $o_e = \max(u_e - c_e, 0)$. The total overflow (TOF) of the layout is given by $\sum_{e \in E} o_e$, and the maximal overflow (MOF) is given by $\max_{e \in E} o_e$. The congestion of edge e, denoted as g_e, is given by $g_e = u_e / c_e$.

3. LOCAL CONGESTION MODELING

In this section, we will analyze the problems associated with existing congestion analysis methods, discuss the sources of local resources, review the previous works related to local resource modeling, and finally propose our methods for local resource modeling. For convenience, in this paper we will use the terms "local resource modeling" and "local congestion modeling" interchangeably.

3.1. Limitations of existing global-routing-based congestion analysis methods

As mentioned in Section 1, global-routing-based congestion analysis is now mainstream. Examples include FastRoute [Pan and Chu 2006] and NTHU-Route 2.0 [Chang et al. 2008], which are used as congestion analyzers within routability-driven placers IPR [Pan and Chu 2007] and CRISP [Roy et al. 2009], respectively. However, the major problem in these academic global routers or congestion analyzers is in their inability to model local resource usage, which can lead to the inaccurate prediction of routing hot spots (opens/shorts) in detailed routing.

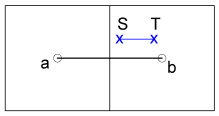
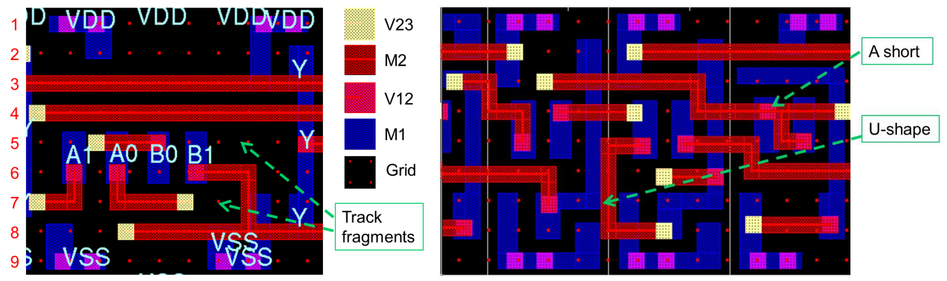


Fig. 4. The local net (S, T) is ignored by traditional global routers.

There are two major consumers of local resources. The first of these are the wires used to connect the *local (sub-) nets*, whose pins are all inside a single g-cell. As illustrated in Fig. 4, the two-pin net (S, T) is a local net in the g-cell centered at b, and the local wire connecting S to T is not modeled in global routing. For convenience, we denote these kinds of resources as **local-net resources**. Traditional global routers generally abstract the routing problem and only focus on g-cell-to-g-cell routing, while the resources used by local nets are ignored.

The second set of consumers, which we refer to as **pin-access resources**, are the resources used for pin access in detailed routing, which are also ignored in traditional global routing. Fig. 5(a) shows a standard cell with five signal pins, shown as blue shapes on metal layer 1 (M1). If we ignore these pins, nine horizontal tracks (marked with number 1–9), on metal layer 2 (M2), are available for global routing in this region. However, from the detailed routing results shown in Fig. 5(a), we can see that horizontal track 5–7 will be mostly blocked in the region for pin access and are no longer available for global routing. Note that only counting the area of wires on layer M2 connecting to pins as local resources used by pins is not enough, since pin access causes many track fragments between the short wires connecting to pins (Fig. 5(a) shows two examples), which are hard to use for routing and should also be amortized to the pin-access resources. In addition, pin-access resources also depend on the pin distribution. The closer pins are packed together, the more difficult it is for a detailed router to access all the pins, since the wires connecting to some pins could block the pin access to other pins and detour may become necessary in such a case. Therefore, more resources could be consumed than the case where the pins are packed more loosely. As an example, Fig. 5(b) shows the pin access for three standard cells with larger pin density[3] than the cell shown in Fig. 5(a). Due to denser pin distribution, pin access becomes more difficult in Fig. 5(b), and therefore, zigzag wires and U-shaped wires have to be used to access the pins, which takes more resources than flat wires or L-shaped wires used for pin access shown in Fig. 5(a). Even worse, a short is caused as shown in Fig. 5(b).



(a) Pin access blocks track 5–7.

(b) Pins with large density consume many routing tracks and a short even occurs.

Fig. 5. Pin access consumes considerable local routing resources. In the legends, "V12" denotes the vias from layer M1 to M2, and "V23" the vias from layer M2 to M3.

In summary, ignoring these local routing resources caused by local nets and pin access will incorrectly make global routing see more tracks than are available, resulting in inaccurate routability evaluation. This motivates the problem of modeling local resources, or local congestion, in global routing. Next, we will first review the previous works for local resource/congestion modeling, and then present our algorithms to address this problem.

3.2. Review of previous works for local congestion modeling

Since the fundamental task of routing is to connect the pins of the same net, pin density is closely related to routing congestion, and high pin density is usually correlated with high routing congestion [Taghavi et al. 2010]. Therefore, pin density has become a key factor to optimize in many placers [Brenner and Rohe 2002; Roy et al. 2009; Hsu et al. 2011]. The general idea involved in these works is to spread the cells so that the pin density is not high in a region. In the routing stage, pin density is also used as a metric to drive routers to achieve more uniform wire distribution to reduce the variations in the chemical mechanical polishing process [Cho et al. 2006; Chen et al. 2007]. Though these works have used pin density as a tool to drive placers or routers in optimization, none of them explicitly studies the relation between pin density and the local congestion, i.e., how the local congestion can be modeled by pin density in global routing stage to achieve a more accurate routability evaluation.

In [Li et al. 2012], an algorithm is proposed to estimate the routing congestion of a circuit considering the local-net resources. It first uses a Steiner tree algorithm to estimate the routes for all the nets (including global nets) in the circuit, and then based on the Steiner solution, computes the maximal track usage within a g-cell using a scan-line algorithm. Though the consideration of local connections in a g-cell improves the accuracy of routability evaluation, it has the following problems. Firstly, using Steiner solution for global nets to estimate the congestion can have large errors, since in real routers, significant detours are used for the nets around the congested regions. Secondly, while

[3]Compared with the standard cell shown in Fig. 5(a), the three standard cells in Fig. 5(b) have only 1.2× larger area but 1.8× more pins.

this method acts more like a congestion estimator (similar to [Lou et al. 2002; Westra et al. 2004]) for a whole circuit, it does not work well in our scenario where the local congestion is to be modeled at the global routing stage. This problem is illustrated in Fig. 6. We show two g-cells and assume that the g-edge (a, b) has a capacity of 4 global routing tracks. Using the algorithm in [Li et al. 2012], the maximal wire density in the g-cell centered at b would be 4 due to four local nets. However, for global routing, if we reduce the capacity of g-edge (a, b) by 4, then this may be too pessimistic, since there may still be some global nets that can be routed through the g-edge, as shown in Fig. 6.

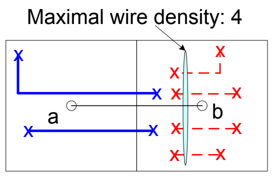


Fig. 6. The maximal wire density model does not work well for the local congestion modeling in global routing.

In [Zhang and Chu 2012], an interleaved global routing and detailed routing framework, GDRouter, is proposed to improve the detailed routing routability. To improve the consistency in routability evaluation between global routing and detailed routing, three techniques are proposed. First, the cost for each g-cell is calculated to consider pin distribution based on a Voronoi diagram method. Second, the capacity of each g-edge is adjusted based on local routing usage estimated by spine routing, which uses a single trunk tree for routing. Third, the capacity of each g-edge is further adjusted by the number of global segments that cannot be assigned to a detailed routing track, which are estimated by performing virtual routing, i.e., fast implementations of FastRoute [Xu et al. 2009] and RegularRoute [Zhang and Chu 2011]. The first technique only uses the pin distribution to adjust the cost of g-cells, but does not adjust the capacity of g-edges to consider the pin-access resources, and thus ignores the fact that pin-access resources would also affect the capacity of g-edges. The second technique could overestimate the local-net resource usage, since the spine routing tree could have more wirelength than the Steiner tree that is used to estimate the local-net usage in advanced industrial detailed routers such as [Gester et al. 2012; 2013]. The third technique involves running of the fast version of global and detailed routers, which in practice could be computationally expensive when applied to a congestion analyzer invoked tens of times in a physical synthesis flow. In summary, these techniques have various limitations when they are used for local resource modeling in congestion analysis.

Some industrial global routers or congestion analyzers also include some methods to model local resources, e.g., some global routers include some form of detailed routing to consider the resources consumed by local net connections and stacked vias [Gester et al. 2012; 2013]. However, these approaches tend to be complex and computationally expensive when such a router acts as a congestion analyzer and is repeatedly invoked during physical synthesis. This is shown in Section 7, where we provide runtime data for such an industrial congestion analyzer. The aforementioned problems motivate our work to develop more effective and efficient methods to model local resources when using a global router for congestion analysis.

Next we will present and discuss three methods of local resource modeling: Method 1 based on Steiner tree wirelength estimation, Method 2 based on pin-density estimation, and Method 3 combining techniques from Method 1 and Method 2 and including further enhancements. Before we proceed further to discuss the details of each method, it will be helpful to briefly introduce how we will evaluate the effects of our methods on the accuracy of routability evaluation. Given a design, we first run an industrial congestion analyzer with complex and accurate local resource modeling to get the reference congestion maps. Next, we run another global-routing-based fast congestion analyzer with one of our methods to model local resources on the same design, to obtain another set of congestion maps. Then we use the correlation between the two set of maps to evaluate the effects of our proposed method. Further details will be presented in Section 7.1 where we discuss the experimental validation for the proposed methods.

3.3. Method 1: Estimation of local resources based on Steiner tree wirelength

In this section, we discuss how to estimate the local resources based on Steiner tree wirelength.

We first discuss the method for local-net resource modeling. We observe that the longer the local wires are, the more likely they are to block global routing tracks. This observation can be formulated by the following equation:

$$t_b = l_r / s_e, \tag{1}$$

where t_b is the number of routing tracks blocked by a local wire, l_r is the length of the local wire, and s_e is the length of a g-edge.

Equation (1) is adopted to calculate blocked tracks on a g-edge and can be easily extended to the more complex cases. Consider the case of a multi-pin local net. To estimate local routing, we first build a Rectlinear Steiner Minimum Tree (RSMT) for the pins[4]; in our experiments, we use Flute [Chu and Wong 2008] for this purpose. We then break each horizontal tree segment into two based on the x-coordinate of the g-cell center and apply Eq. (1) to calculate blocked global routing tracks on the left and right g-edges associated with the g-cell. Similarly each vertical Steiner tree segment can be broken using the y-coordinate of the g-cell center.

An example is shown in Fig. 7, where net (A, B) is a two-pin net while (A, J) and (B, J) are the two segments of a Steiner tree. The global routing tracks blocked by net (A, B) in the horizontal direction can be calculated based on segment (A, J). Since the g-cell center b is between A and J, segment (A, J) blocks global routing tracks on g-edges (a, b) and (b, c). The blocked tracks on g-edge (a, b) can be calculated as $(x_b - x_A)/(x_b - x_a)$, where x_b denotes the x-coordinate of g-cell center b, and other notations are similarly defined. Similarly, the blocked global routing tracks on g-edge (b, c) is $(x_J - x_b)/(x_c - x_b)$. The vertical tracks blocked by net (A, B) can be calculated similarly, based on segment (J, B). As another example, when a segment is completely on the left of (above) or right of (below) the g-cell center, such as the net (C, D) in Fig. 7, the blocked tracks can all be attributed, respectively, to the left (top) or right (bottom) g-edge. The blocked tracks on g-edge (b, c), in this case, can be calculated as $(x_D - x_C)/(x_c - x_b)$. The proposed method can be easily applied to more complex Steiner trees, for example net (A, B, C) in Fig. 8.

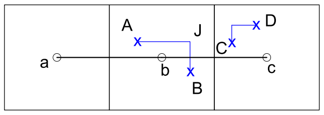


Fig. 7. Local routing resource estimation for two-pin nets.

Next we discuss how we model the pin-access resources. Since most traditional global-routing-based congestion analyzers only produce the g-cell-center-to-g-cell-center connections for nets, we must consider the synergy between global and local routing, i.e., how to connect to real pins. As an approximation, the following method is used. We include the g-cell center as a dummy pin when constructing the Steiner tree to model the local resources[5]. For example, for a net (D, E, F, G) shown in Fig. 8, the Steiner tree connecting b, E, F, G is used to calculate the blocked global routing tracks on the four boundaries of the g-cell with center b. Similarly, the Steiner tree connecting a, D is used to calculate the blocked tracks corresponding to the g-cell with center a.

[4]We use RSMT since it provides a solution with minimum wirelength for a net and most modern routers use RSMT as the initial solution for a net.

[5]Note that g-cell centers are added as dummy pins only for the nets with global wires. For a local net with all the pins inside a g-cell, the g-cell center is not considered for Steiner tree construction since there is no global wire for this net.

To further consider the track fragments blocked by the local wires connecting to pins as shown in Fig. 5(a), we introduce a parameter p ($p > 1$) to scale the estimated local resources using the method discussed above, where p will be tuned empirically for each technology.

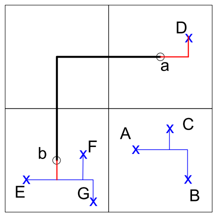


Fig. 8. Local routing resources consumed by two nets.

In summary, to model local resources in global routing, we add a preprocessing step. Specifically, we iterate through each net in the design, identify the local nets inside each g-cell, estimate the local resources using the method presented in this section, and block the global routing tracks from the related g-edges. Local wires inside a g-cell are usually short and for pin accessibility they are typically routed in the second (M2) and third (M3) metal layers during detail routing. Hence, we only block the global routing tracks on g-edges in the M2 and M3 layers during congestion evaluation.

3.4. Method 2: Estimation of local resources based on pin density

In this section, we present the second method for local resource modeling, which is even simpler and faster than Method 1, yet is seen to provide similar effectiveness. This method is based on pin density, and does not involve constructing Steiner trees to estimate the local resources. It is based on the following observations:

— Each pin is associated with a set of local wires connected to it.
— The number of pins in a g-cell is a good indicator of the number of local wires, and is a first-order estimate for routing tracks blocked by local wires within the g-cell.

Based on the above observations, we model the local resources R_l in a g-cell by

$$R_l = kn, \tag{2}$$

where k is a technology-dependent parameter, and n is the number of pins from both local and global nets in the g-cell.

As in Method 1, we use a preprocessing step with Method 2 in a global-routing-based congestion evaluation tool. Specifically, we traverse all the g-cells and nets, and count the number of pins (n), inside each g-cell. Following this, we block kn global routing tracks, due to the local wires in each g-cell, on the four g-edges related to the g-cell. Similar to Method 1, we only block the global routing tracks on g-edges in the M2 and M3 layers.

3.5. Method 3: An enhanced method to model local resources with better scalability

As Method 1 and Method 2 provide simple first-order models for local resources, there is some scope for further improvement. Next, we will first discuss the limitations associated with the two methods, and then present our enhanced method to model local resources.

3.5.1. Limitations in Method 1 and Method 2. As will be shown in Section 7.1.3, both Method 1 and Method 2 work well for small g-cell size, but do not scale well to large g-cell sizes. We will analyze the reasons next.

In Method 1, to consider pin access resources, the local resource estimation counts the connection from the real pins to the corresponding g-cell centers which act as dummy pins. This is based on the assumption that all global wires are connected to g-cell centers. While this seems true in global

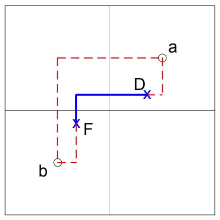


Fig. 9. Artificially connecting pins to g-cell centers overestimates the routing resources used by the two-pin net (D, F).

routing stage, the real situation in detailed routing can be quite different. Fig. 9 shows an example, where a two-pin net (D, F) in detailed routing may only consume the routing resources shown as solid blue segments, while by artificially connecting pins to g-cell centers as in Method 1, the net will consume resources shown as dashed red segments that significantly overestimate the routing resources required. From the example, we can see that adding g-cell centers as dummy pins to consider pin-access resources can bring errors, which will be amplified when g-cell size is increased, since the wires from pin to g-cell center tend to be longer when g-cell size becomes larger.

There are two major reasons why Method 2 does not scale well with g-cell size. Firstly, Equation (2) does not consider the potential relationship between k and g-cell size, and then the k-factor must be tuned for each g-cell size. The dependence of k on the g-cell size is illustrated as follows. Fig. 10 shows four g-cells, a, b, c, d, with the size of 10 tracks, and in each there are 10 pins. Assume 10 pins block 5 tracks (showed as red segments) on each edge, and then k for g-cell size 10 is 0.5. Now assume g-cell size increases to 20 tracks, and look at the g-cell F that contains g-cell a, b, c, d and 40 pins. Since the 40 pins now block 10 tracks on each edge, the k-factor for g-cell size 20 can be calculated as 0.25. This example clearly shows the dependence of k on g-cell size. Secondly, the estimation of local-net resources by pin-density in Method 2 ignores the real wirelength of local nets, and can become inaccurate when g-cell size increases. For example, in Fig. 7, two two-pin nets (A, B) and (C, D) have quite different Steiner wirelengths and likely consume different amount of local resources, while Method 2 assumes they consume the same amount of local resources since they both have 2 pins, which may bring error in local resource estimation. This error may be negligible when g-cell size is small, but could become significant while a large g-cell size is used and the lengths of wires connecting local nets with the same pin count can vary in a large range.

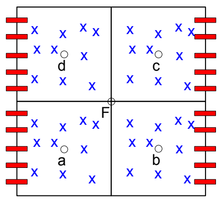


Fig. 10. When g-cell size doubles, blocked tracks also double while the number of pins becomes four times.

In addition to the scalability problem, Method 2 also has other limitations, and can be improved in the following two aspects. Firstly, in Method 2, when adjusting the capacity of four g-edges associated with one g-cell, the amount of blockage added to each g-edge is the same, i.e., kn. This can be further improved by considering the distribution of pins in the g-cell, i.e., if pins are closer to one boundary of the g-cell, more blockages should be added to the g-edge across that boundary. Secondly, Method 2 assumes n pins always consume the same local routing resources, kn, but this can also be improved by considering the pin distribution. If the pins are packed within a region so that their distance becomes smaller than a threshold distance, d_{th}, it could become difficult to access those pins, and more local resources tend to be used for pin access, as illustrated in Fig. 5(b). Therefore, extra weights could be added to scale the local resources used by those pins.

3.5.2. The enhanced method to model local resources: Method 3. Motivated by solving the problems in Method 1 and Method 2, we develop an enhanced method to model local resources, Method 3. In Method 3, we use the following strategy to estimate the local routing resources: we split the estimation of local resources into separate estimation of the local-net resources and the pin-access resources, and use suitable techniques to estimate each category. Specifically, we use a modified Steiner tree method to estimate the local-net resources, and an enhanced pin-density method to estimate the pin-access resources with consideration of pin distribution. Finally, we combine the estimation from the two methods as the final estimation. In this way, we make use of the advantages from each method, and avoid the disadvantages.

Firstly, the modified Steiner tree method is used to estimate the local-net resources. Since, unlike Method 1, we do not consider pin-access resources in this step, we do not add g-cell center as dummy pins when constructing Steiner trees for local nets, and do not scale up the local-net resources. In this way, only the local-net resources are estimated.

Secondly, the enhanced pin-density method is used to estimate the pin-access resources and there are three major improvements. The first improvement is that we change our formulation to make the estimation of the pin-access resources aware of g-cell size. We make the following assumption (note that this assumption will be extended soon to consider the effects of pin distribution):

ASSUMPTION 1. *Each pin will consume the same amount of local routing resource in terms of area, denoted as a_r.*

For the convenience of derivation, we assume the unit for a_r is nm^2. Let $a_u = P^2$ be the unit area, where P is the minimum wire pitch (unit: nm) on layer M2 (layer M3 usually has the same wire pitch as layer M2). Here the pitch of layer M2 and M3 is used because local wires are generally routed on these layers and their area is at least as large as a_u. Assume $a_r = qa_u$, where q is a technology-dependent parameter to be tuned. Given a g-cell c with n pins, the pin-access resources consumed by the n pins will be qP^2n. Further, let S be the g-cell size (unit: nm). The routing resources taken by a global routing track are PS. Then we can calculate the pin-access resources consumed by n pins in the unit of global routing track, denoted as b_p, as follows:

$$b_p = \frac{qP^2n}{PS} = \frac{qP}{S}n. \tag{3}$$

The g-cell size C, in the unit of routing tracks (more commonly used in practice), can be calculated by $C = S/P$. Now combining (3), we have:

$$b_p = \frac{q}{C}n. \tag{4}$$

Note that b_p is inversely proportional to C. Equation (4) verifies the observations from Fig. 10 that the scaling factor associated with pin density n is a function of g-cell size.

The second improvement over Method 2 is that the pin-access resources will be scaled up to be further aware of pin distribution, if the pins are packed together too close so that their distances become smaller than a threshold distance, d_{th}. Note that the distance discussed in this paper refers to Manhattan distance. This enhancement is implemented by extending Assumption 1. Instead of assuming every pin will consume the same amount of local routing resource a_r, now we assume that every pin P_i will consume resources in the amount of $w_i a_r$, where w_i is a weight (≥ 1) associated to each pin P_i which will be computed according to the pin distribution around P_i. Using techniques that are similar to those used to derive (4), we may calculate the pin-access resources blocked by n pins in a g-cell c as

$$b_p = \frac{q}{C}\sum_{j=1}^{n} w_j, \tag{5}$$

where w_j is the weight for pin $P_j (1 \leq j \leq n)$, the j^{th} pin in the g-cell c. Weight w_j is calculated by

$$w_j = 1 + \sum_{k=1}^{n_{cp}} W(d_{k,j}), \tag{6}$$

where n_{cp} is the number of pins with distance to P_j smaller than d_{th}, $d_{k,j}$ denotes the distance between P_k and P_j, and $W(d_{k,j})$ is a function of $d_{k,j}$ which defines how much extra weight should be added to w_j due to the small distance between the pin pair (P_k, P_j). Note that if $n_{cp} = 0$, $w_j = 1$, i.e., no extra weight is added for pin P_j. In this work, we set d_{th} equal to the expected distance of the pins in a design, calculated as: $d_{th} = \sqrt{(1 - r_{mb})A_{ds}/\mathcal{N}}$, where A_{ds} is the design area, $\mathcal{N}$ is the total number of pins in the design, and r_{mb} is the ratio of area of the macro blocks to the design area. Here, the area of macro blocks is subtracted from the calculation, since it is usually impermissible to place pins in such regions. In addition, the weighting function $W(d_{k,j})$ used in this work is set to

$$W(d_{k,j}) = 1.4760 - \arctan(0.5155 + 10d_{k,j}/d_{th}), \tag{7}$$

where the constants in the function are chosen with the following considerations: $W(0) = 1$, $W(d_{th}) = 0$, and W increases slowly when $d_{k,j}$ is close to d_{th}, while increases quickly when $d_{k,j}$ becomes close to 0. The function is illustrated in Fig. 11.

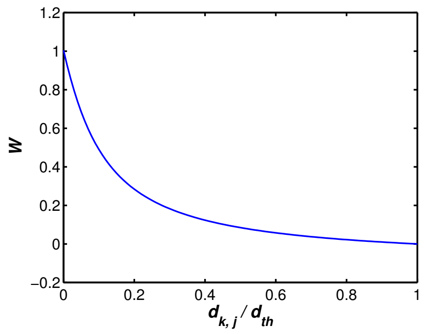


Fig. 11. The function translating the distance of a pin pair to the weight on a pin.

A key step in computing b_p by (5)–(7) is to compute the weight w_i for each pin P_i, which requires us to find all the pins with distance to P_i smaller than d_{th}. For this purpose, we use a gridding method, inspired by the work in [Heckbert 1997]. The pseudocode of our method is listed in Algorithm 1. First, the layout is partitioned to uniform grids with size d_{th}, and all the pins are mapped to each grid. Note that this grid is much finer than the g-cell grid, typically more than $5\times$ smaller, and is only used to compute the weights of all the pins but not used in routing. Next, the weight of each pin is initialized to be 1. Then we process the grid one by one from the bottom left corner to the top right corner. For each grid $G_{x,y}$, we iterate through each pin P_j in it, and find all the pins with distance to P_j smaller than d_{th} by searching the current grid $G_{x,y}$ and the four directly adjacent grids. The five grids searched are illustrated in the shaded region in Fig. 12. Whenever we find a pin P_k satisfying the distance constraint, i.e., $d_{j,k} < d_{th}$, a weight $W(d_{j,k})$ is added to w_j and w_k. The reason why we do not need to search other non-adjacent grids is that all the pins in a grid not adjacent to $G_{x,y}$, such as pin P_l in Fig. 12, must have a distance to any pins in $G_{x,y}$ not less than d_{th}, since the grid size is d_{th}. In addition, when processing a grid $G_{x,y}$, we do not need to search the other four adjacent grids in the left-bottom direction, since the pins in those grids, such as pin P_m in Fig. 12, have been processed in an earlier iteration, and our searching process ensures any pair of pin will be processed only once. Though there are several levels of loops in our algorithm, in most cases, it has linear time complexity in terms of the number of pins $\mathcal{N}$. Assume the largest pin density in a

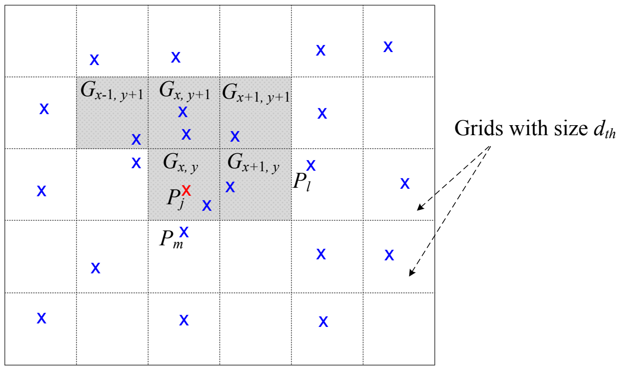


Fig. 12. When searching for pins with distance to P_j smaller than d_{th}, we will only search the five shaded grids.

grid with size d_{th} is α for a given technology. Then for each pin, at most we need to search $5\alpha d_{th}^2$ pins, and therefore, the time complexity of the whole algorithm is $\mathrm{O}(\alpha\mathcal{N})$. In most cases, α could be bounded by a constant for a given technology, and our algorithm runs in linear time.

ALGORITHM 1: Computing the weights for all the pins.

```
Input: All the pins $P_i, 1 \le i \le \mathcal{N}$.
Output: Weight of each pin $w_i$.
Initialize the weight of each pin to 1;
Partition the layout to $n_r \times n_c$ grids with size $d_{th}$, and map all the pins to the grids;
for $y = 0; y < n_r; ++y$ do
    for $x = 0; x < n_c; ++x$ do
        Assign the number of pins in grid $G_{x,y}$ to $n_{x,y}$;
        for $j = 0; j < n_{x,y}; ++j$ do
            for $k = j+1; k < n_{x,y}; ++k$ do
                if pin $P_j$ and $P_k$ are not from the same net, and their distance $d_{j,k}$ is smaller than $d_{th}$ then
                    $w_j += W(d_{j,k})$;
                    $w_k += W(d_{j,k})$;
                end
            end
            foreach pin $P_k$ from grids $G_{x+1,y}$, $G_{x+1,y+1}$, $G_{x,y+1}$ and $G_{x-1,y+1}$ (ignoring the grid with coordinates out of the design boundary) do
                if pin $P_j$ and $P_k$ are not from the same net, and their distance $d_{j,k}$ is smaller than $d_{th}$ then
                    $w_j += W(d_{j,k})$;
                    $w_k += W(d_{j,k})$;
                end
            end
        end
    end
end
```

The third improvement over Method 2 is that the blockages added to each one of the four g-edges associated with a g-cell will be redistributed by considering pin distribution, instead of equal distribution as in Method 2. We will present our method for the horizontal g-edges, and the case for the vertical g-edges is analogous. For a given g-cell b, as illustrated in Fig. 13, denote the two associated horizontal g-edges as e_l and e_r, and the routing tracks blocked by the pins in g-cell b on them as b_l and b_r, respectively. Then we calculate the mean of the x-coordinates of all the pins in the g-cell b, denoted as μ_x. Let x_l and x_r be the x-coordinate of the left and right boundary of g-cell b, respectively, and let $S = x_r - x_l$ be the g-cell size. Then we distribute the pin-access resources b_p (computed by (5) and (6)) consumed by all the pins in g-cell b by the following formulae:

$$b_l = b_p \frac{x_r - \mu_x}{S} \text{ and } b_r = b_p \frac{\mu_x - x_l}{S}. \tag{8}$$

The intuition behind these formulae is that if μ_x is closer to the left boundary of the g-cell, which means more pins are closer to the left horizontal g-edge, we should block more capacity of the left horizontal g-edge, and vice versa.

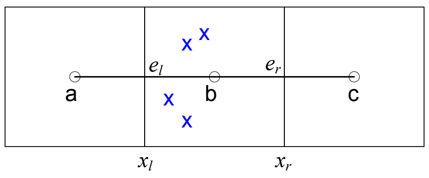


Fig. 13. The pin-access resources consumed by the pins in a g-cell on the horizontal layer will be redistributed to the left and right g-edges associated with the g-cell based on the pin distribution.

Together with all the three improvements, our enhanced pin-access resources algorithm will work in the following steps:

— Compute the weight for each pin using Algorithm 1.
— Iterate through each g-cell c: calculate the pin-access resources b_p for the pins in c by (5), and further redistribute the resources b_p to the four g-edges associated with g-cell c on layer M2 and M3 by (8).

After pin-access resources are computed, the local-net resources will be computed by the improved Steiner method discussed earlier, and finally the blockages from two kinds of local resources will be added to the g-edges on layer M2 and M3. The whole process still works as a preprocessing step, just as in the case of Method 1 or Method 2.

4. FILTERING OUT THE NOISE IN ROUTABILITY EVALUATION

As mentioned in Section 1.1, the noisy hot spots in the congestion maps computed by a congestion analyzer can mainly be attributed to the limited routing effort involved in congestion analysis, and they are likely to be removed in real global routing process by additional routing effort. These noisy hot spots prevent us from obtaining accurate routability evaluation (as compared to real global routing results), and should be addressed properly in congestion analysis. In this section, we will first present quantitative analysis on this problem, and then propose a smoothing technique to deal with it.

To quantitatively study this problem, we introduce a metric called *noise ratio* to measure the ratio of noisy hot spots to the total number of hot spots in a map. If g_{th} is the congestion threshold, then a g-edge e with routing demand $w_e > 0$, and congestion $g_e \geq g_{th}$ will be treated as a hot spot. In this work, g_{th} is set to 80% according to the properties of the industrial routing toolkit we use. Quantitatively, a hot spot is treated as a noisy hot spot when the difference between its congestion and its two parallel adjacent g-edges in the same routing direction is larger than θg_{th}, where θ is a user-defined parameter. With $\theta = 0.25$ (which is the setting used in this work), when routing with a typical g-cell size, e.g., 40 tracks, the difference in the routing usage between a noisy hot spot and its parallel adjacent g-edges will be larger than $40 \cdot \theta g_{th} = 8$ tracks, which is large enough so that it is likely that more routing effort in real global routing could more evenly redistribute the routing usage among these g-edges. Based on these discussions, the noise ratio can be easily calculated given the congestion data for all the layers. Continuing our analysis on the example shown in Fig. 2 (for better readability, we copy the figures to Fig. 14), next we calculate the noise ratios for the routing solutions from congestion analysis mode (Fig. 14(a)) and global routing mode (Fig. 14(b)) of an industrial router (that has built-in local routing resource modeling). For the congestion analysis mode, the noise ratio is 17.08%, while for the global routing mode, the noise ratio is reduced to 9.98%, which means that the insufficient routing effort in congestion analysis results in many more noisy hot spots than those in global routing.

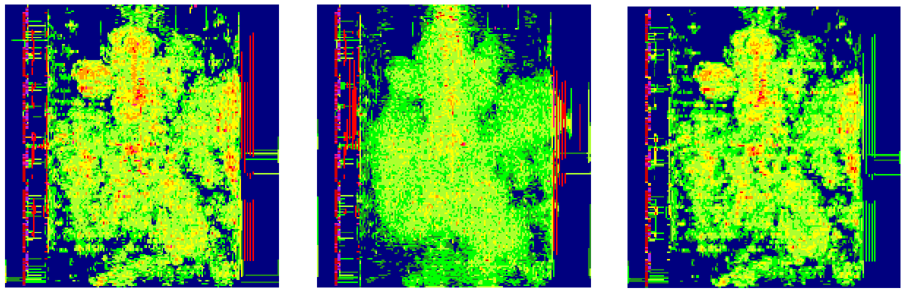

(a) Map from congestion analysis (b) Map from global routing (c) Smoothed map using Gaussian function

Fig. 14. The proposed smooth technique reduces the noisy hot spots in a congestion map.

To reduce the number of the noisy hot spots in the congestion maps and achieve more accurate routability evaluation, a smoothing technique can be applied to the congestion maps obtained by a congestion analyzer. Smoothing is a common technique widely used in many fields, such as signal processing, image processing, and EDA. In the EDA field, smoothing techniques have been widely used in placement algorithms, which use a bell-shaped function [Naylor et al. 2003; Kahng and Wang 2005; Chen et al. 2008b; Chen et al. 2008a; Jiang et al. 2008; Hsu et al. 2011] or the inverse Laplace transform [Chan et al. 2005] to smooth the potential function (that specifies the total area of movable blocks in a placement bin), so that optimization can proceed more effectively. In addition, smoothing has also been used in the works related to design for manufacturability [Tian et al. 2001; Ouma et al. 2002; Wei and Sapatnekar 2010; Chen et al. 2010], where Gaussian function is used to smooth the pattern density (that is the density of metal wires in a given region) across a layout.

However, to the best of our knowledge, smoothing techniques have not been directly applied to congestion maps in the literature. In this work, we propose to use a smoothing technique to filter out the noisy hot spots in the congestion maps obtained by a congestion analyzer. The objective of the smoothing technique is not merely to correct for errors in congestion estimation, but also to mimic the way in which real global routers work. For example, assume there are a large number of pins in the two horizontally-adjacent g-cells, a and b. In congestion analysis, due to limited routing effort, these pins are routed through the g-edge e connecting the two g-cells, which causes a congestion hot spot S_h, while the g-edges in the same routing direction adjacent to e are not congested. In contrast, in real global routing, it is likely these pins in g-cell a and b can be routed with a detour by crossing the g-edges adjacent to e. In other words, the global router could further redistribute the congestion on g-edge e to the adjacent g-edges by spreading the wires with some detour. Smoothing a congestion map will reduce the difference in the congestion between a noisy hot spot and its neighboring g-edges, which could obtain an effect that is similar to realistic routing, where greater detours are allowed. Therefore, it is reasonable to apply smoothing technique on the congestion maps from congestion analysis to mimic the behavior of a global router.

Unlike [Jiang et al. 2008], we apply a one-dimensional Gaussian function to smooth the congestion map for each layer in the direction perpendicular to the preferred routing direction of that layer. The reason to use a one-dimensional function is that in routing, to mitigate the congestion, spreading the overflowing wires to the neighboring g-edges is directional, i.e., vertically/horizontally spreading on the horizontal/vertical layer. The Gaussian function used is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \tag{9}$$

where σ is the standard deviation. Since the Gaussian function will be applied on a discrete congestion map, the Gaussian function will be discretized to the GRG using a method similar to those in [Tian et al. 2001; Wei and Sapatnekar 2010]. We first define the smoothing window size as $(2l+1)$ g-edges, and the smoothing function will be truncated beyond the smoothing window, which means that when we calculate the smoothed congestion value at one g-edge, other g-edges with distance farther than l will not be counted. In the Gaussian function, typical values for σ can be l, $l/2$ or $l/3$. Denote the discretized function as $f(i)$ with i as an integer. Note that $f(i)$ will be 0 for $i > l$ or $i < -l$ due to the truncation. After discretization, we should normalize the function values to make sure they sum to 1. Given a congestion map for a layer, if we denote the congestion of a g-edge with coordinate (x, y) in the map as $g(x, y)$, then the smoothed congestion, $\bar{g}(x, y)$, can be calculated by

$$\bar{g}(x,y) = \begin{cases} \sum_{j=y-l}^{j=y+l} g(x,j) f(j-y) & \text{for horizontal layers,} \\ \sum_{i=x-l}^{i=x+l} g(i,y) f(i-x) & \text{for vertical layers.} \end{cases} \tag{10}$$

In Eq. (10), when the indices of (x, j) [or (i, y)] are out of range (i.e., they lie outside the congestion map), the value of $g(x, j)$ [or $g(i, y)$] is set to $g(x, y)$. Another caveat to use the smoothing technique is that we have to be aware of the case where the routing demand on a g-edge e is small and then its congestion after smoothing will become smaller than b_e/c_e, i.e., the ratio of the blockage usage to the capacity on the g-edge e (that is the minimum possible congestion on e with zero routing demand), which is not realistic. For example, assume we have three adjacent g-edges with the following situations:

— G-edge e_0: $c_{e_0} = 40, b_{e_0} = 20, d_{e_0} = 0, g_{e_0} = 50\%$.
— G-edge e_1: $c_{e_1} = 40, b_{e_1} = 34, d_{e_1} = 2, g_{e_1} = 90\%$.
— G-edge e_2: $c_{e_2} = 40, b_{e_2} = 20, d_{e_2} = 0, g_{e_2} = 50\%$.

Further assume the weights from a smoothing function (with $l = 1$) is: $f(0) = 0.8$, $f(1) = f(-1) = 0.1$. Then the smoothed congestion of g-edge e_1 will become 82%, smaller than $b_{e_1}/c_{e_1} = 85\%$, which becomes unrealistic. To solve this problem, we change our implementation so that whenever such a case occurs on g-edge e, we use b_e/c_e as its congestion after smoothing, and then redistribute the routing demand on g-edge e evenly to its two adjacent g-edges. In addition, in practice, the smoothing technique can be used multiple times until the congestion map reaches the desired extent of smoothness, specifically, until the noise ratio is reduced below a threshold, e.g., 5%.

We now test the smoothing technique on the routing solutions shown in Fig. 14(a), assuming $\sigma = l/2 = 1/2$. Then the smoothing window size would be three g-edges, and the discretized function will have three values: $f(0) = 0.7979$ and $f(1) = f(-1) = 0.1080$. After normalization, $f(0) = 0.7870$ and $f(1) = f(-1) = 0.1065$. Then applying this smoothing function to the congestion maps of all the layers from congestion analysis mode, we obtain the smoothed combined congestion map shown in Fig. 14(c). We can see that as expected, the map in Fig. 14(c) is smoother than the original map in Fig. 14(a). The noise ratio for the smoothed routing solution is calculated as 4.37%, much smaller than 17.08% for the original solution. Moreover, the difference between smoothed map and global routing map becomes smaller than that between congestion analysis map and global routing map. This shows that our proposed smoothing technique can indeed filter out the noisy hot spots, and improve the accuracy of the routability evaluation.

5. METRICS FOR DESIGN CONGESTION

In this section, we will first discuss the limitations of existing routability metrics besides those mentioned in Section 1.2, and then present our new metric.

5.1. Limitations of current metrics

TOF and MOF: Naïve implementations of the TOF and MOF metrics treat the overflow in each layer as identical; however, this is inaccurate as each layer has a different capacity. Normalizing the overflow to the layer capacity can overcome this issue, but other problems remain. The TOF metric does not capture the hot spots in the congestion map, i.e., the severity of congestion in the worst regions of the chip. MOF fares only slightly better, capturing only the maximum overflow value among all the g-edges in the routing graph, and it presents a fairly incomplete picture of the congested regions in the design. These problems could make overflow metrics even fail to predict the routability correctly in some cases, as will be demonstrated in Section 7.3.
$\boldsymbol{ACN(20)}$**,** $\boldsymbol{WCI(100)}$ **and** $\boldsymbol{WCI(90)}$: These metrics fail to differentiate between a net spanning a single congested g-edge and one that spans multiple congested g-edges.

Example 5.1. Consider two nets in the GRG: *net A* traverses g-edges with congestion 0.50, 0.70, 0.80, 0.90 and 1.10, while *net B* traverses g-edges with congestion 0.60, 0.80, 0.95, 1.05 and 1.10. When calculating $ACN(20)$, $WCI(100)$ and $WCI(90)$, both nets will be considered to have the same congestion value of 1.10. However, their routability is different: clearly, *net B* is harder to route compared to *net A*, as it traverses more number of g-edges with high congestion. This fact is not captured by these net-congestion-based metrics.

Additionally, minor design changes can cause large fluctuations in the $WCI(100)$ and $WCI(90)$ metrics.

Example 5.2. Assume a design has a g-edge e, with $c_e = 40$, $b_e = 0$ and $w_e = 39$. Assume, that we reroute a net to pass through this g-edge (say, to improve timing). Then the congestion of e becomes 100%, implying that all 40 nets crossing e now have a congestion of 100%. As a result, $WCI(100)$ will now report 40 additional congested nets, when in reality we only rerouted a single net. A similar example applies to the $WCI(90)$ metric. Such instability renders these metrics unsuitable for guiding routability optimization.

Although, $ACN(20)$ avoids large swings due to minor design changes, it suffers from the limitation of not accurately capturing design congestion (will be demonstrated in Section 7.3.1).

In addition, existing metrics improperly model the congestion along macro boundaries [Alpert and Tellez 2010; Alpert et al. 2010], leading to an artificially high reported congestion. Referring to Fig. 15, a net N routes to a pin on macro block B. Due to the blockage, the congestion of edge e would be rated as being above 90%, but in practice, we find that such nets are easily routable. Including these g-edges with artificially high congestion when calculating the metric introduces unnecessary noise leading to improper estimation of the routability. Note that we only suggest excluding the edges along macro boundaries when calculating the metric *after* global routing to evaluate the routability, but the high congestion of these edges should *not* be ignored during the global routing process.

5.2. New metric (ACE metric) for design congestion

To address the issues with existing metrics, we propose a new metric that is based on the histogram of g-edge congestion. Our metric has two features:

— It downplays the effects of g-edges with artificially high congestion due to the presence of macro blockages.
— It presents congestion as a histogram, instead of a single number.

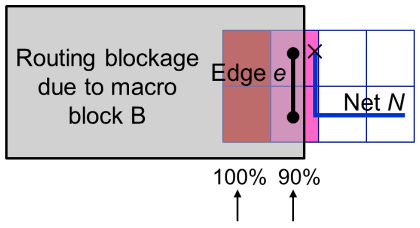


Fig. 15. An example showing a net N traversing g-cells that are 90% blocked due to a macro blockage. This leads to artificially high reported congestion for g-edge e.

To accurately capture the congestion, our metric, denoted as $ACE(x, y)$, computes the average congestion of the top $x\%$ congested g-edges, while excluding g-edges that are along macro boundaries and $\geq y\%$ blocked. The role of the parameter y is to avoid counting the effects of g-edges with artificially high congestion around macro boundaries. The value of y should be appropriately chosen. If y is too small, we may exclude too many g-edges which may have real congestion around the macro boundaries; if y is too large, we may over count the g-edges with high congestion caused by macro blockages. In this work, y is set to 50 to get a tradeoff between the two cases, implying that all g-edges along macro boundaries with $\geq 50\%$ routing blockage are ignored when computing the metric. For convenience, we use $ACE(x)$ to denote $ACE(x, 50)$ in this paper and the new metric is called the **ACE metric**. In computing the ACE metric, we use a simple heuristic to detect the g-edges along macro boundaries. Given a g-edge e, if either of the two adjacent g-edges on the same routing layer is 100% blocked, then e is treated as a g-edge along macro boundaries. In addition, the g-edges with high congestion purely from blockages, i.e., the g-edges with zero routing demand, will also bring noises to the ACE metric, and therefore, the g-edges with zero routing demand will be downplayed and their congestion is treated as zero when computing the ACE metric.

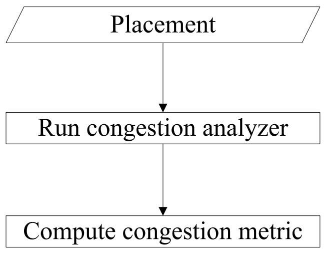
In practice, the new metric is most useful when expressed as a vector, for different values of x, e.g., for $x \in \{0.5, 1, 2, 5, 10, 20\}$. $ACE(x)$, for a small value of x (e.g., 0.5 and 1), provides a highly local view, representing congestion in the regions with the highest contention for wiring resources (hot spots). For larger values of x (e.g., 10 and 20), it provides a broader picture of the design congestion.

6. HOW THE PROPOSED TECHNIQUES WORK TOGETHER

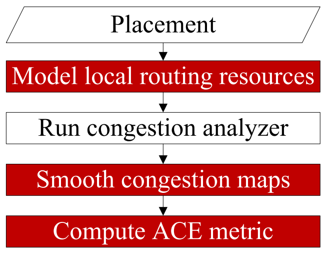
In order to improve the effectiveness of routability evaluation, we have introduced three techniques: the methods to model local routing resources, a smooth technique to filter out noisy hot spots, and an improved metric to represent congestion. In this section, we will discuss how these techniques work together in a typical routability evaluation process.

Traditionally, there are two major steps in routability evaluation as shown in Fig 16(a). First, given a placement, a global-routing-based congestion analyzer is invoked to obtain a routing solution and the corresponding congestion maps. Secondly, a congestion metric is computed to represent the congestion level of the placement, and used to estimate the routability.

The traditional routability evaluation can be improved using our proposed techniques. Firstly, we add a preprocessing step to model the local routing resources before global-routing-based congestion analysis really starts. Any one of the three methods (Method 1 to 3) discussed in Section 3 can be used, but Method 3 is preferred due to its good accuracy and scalability with large g-cell sizes as will be demonstrated in Section 7.1. With Method 3, we first calculate the local routing resources consumed by all the local nets and pins, and then add them as the form of routing blockages to the related g-edges on layer M2 and M3. These routing blockages added in the preprocessing step will propagate to the congestion analyzer by some way such as file exchange. Alternatively, the local routing resource model can be implemented and integrated inside the congestion analyzer as a preprocessing function. Secondly, the congestion analyzer will be run with the blockages from



(a) Traditional routability evaluation process



(b) Improved routability evaluation process with proposed techniques

Fig. 16. Comparison of routability evaluation processes.

the preprocessing step. It is expected that the routing solution will have a higher congestion level compared with that without local resource modeling. Thirdly, our proposed smoothing technique can be applied on the congestion maps from the second step to filter out the noisy hot spots. Finally, the ACE metric will be computed based on the smoothed congestion maps. The improved routability evaluation process is shown in Fig. 16(b). It is worth pointing out that when computing the ACE metric, the "blockages" added by the local routing resource modeling should be treated as routing demand, since they are indeed the routing demand from local nets and pin access.

7. VALIDATION AND ANALYSIS

Our proposed techniques are implemented within a congestion analyzer that performs global routing in the spirit of MaizeRouter [Moffitt 2008]. This section provides a detailed analysis of these techniques on advanced industrial designs. All experiments were run on a 64-bit Linux server with 4 octa-core CPUs (Intel® Xeon® X7560 2.27 GHz).

In this section, we first validate the effectiveness of each proposed technique in Section 7.1–7.3. Specifically, we first analyze the impact of methods about local resource modeling on routability evaluation in Section 7.1, and then present the results of the smoothing technique in Section 7.2, followed by the analysis of the impact of the proposed ACE metric in Section 7.3. Finally, in Section 7.4, we demonstrate how our proposed techniques work together in the improved routability evaluation process described in Section 6.

7.1. Impact of local resource modeling on routability evaluation

For the analyses presented in this section, we use the following congestion analyzers to evaluate the impact of our proposed techniques for local resource modeling:

— **Analyzer A0**: A fast congestion analyzer that is based on MaizeRouter [Moffitt 2008], with the ability to perform global routing on millions of nets in less than 10 minutes[6]. It does not include any local resource modeling.
— **Analyzer A1**: Modification of Analyzer A0, incorporating the Method 1 for local resource modeling.
— **Analyzer A2**: Modification of Analyzer A0, incorporating the Method 2 for local resource modeling.
— **Analyzer A3**: Modification of Analyzer A0, incorporating the Method 3 for local resource modeling.
— **Reference Analyzer:** An industrial congestion analyzer with complex modeling of local routing resources, which is a component of a full-blown industrial routing toolkit[7]. The Reference Analyzer

[6]This is achieved by running on our Linux server with a proper g-cell size on the designs.
[7]This industrial routing toolkit is currently used in industry for real chip tape-out and has three major components: a congestion analyzer, a global router, and a detailed router.

has been proven to be sufficiently accurate in routability evaluation, and will be used to judge the quality of all results. It typically runs at least 10 times slower[8] than Analyzer A0–A3.

Note that Analyzer A1–A3 are adapted from Analyzer A0 by adding a preprocessing step before global routing to model local routing resources using Method 1–3, respectively, and all other parts such as the routing algorithm and configuration are exactly the same.

To quantitatively measure the correlation between Analyzer Ai ($0 \leq i \leq 3$) and the Reference Analyzer, the average error (AVGE) between the congestion maps of Analyzer Ai ($0 \leq i \leq 3$) and those from the Reference Analyzer, is computed for each design. For each g-edge e in the congestion maps, the error is calculated as $\Delta_e = |g_{ye} - g_{xe}|$,where g_{ye} is the congestion of e from Analyzer Ai ($0 \leq i \leq 3$) and g_{xe} is the corresponding congestion computed by the Reference Analyzer. The value of AVGE is the average of Δ_e over all the g-edges considered. Since, in practice, only the g-edges with congestion over a threshold g_{th} are of interest, when calculating AVGE, we only consider the g-edges with congestion larger than g_{th} (set to 80% as stated in Section 4) in either the Reference Analyzer or Analyzers Ai ($0 \leq i \leq 3$): we denote this set of g-edges as S_e. Note that S_e will include all the g-edges with "congestion mismatches" (a notation proposed in [Pan and Chu 2006]), i.e., the g-edges with underestimation (congestion over g_{th} in the Reference Analyzer but not in the other Analyzer) and overestimation (congestion below g_{th} in the Reference Analyzer but not in the other Analyzer) in the congestion maps. In fact, we consider even more g-edges than [Pan and Chu 2006], since the g-edges that are congested ($g_e > g_{th}$) in both Analyzer Ai ($0 \leq i \leq 3$) and the Reference Analyzer will also be counted in the AVGE computation. Moreover, other error statistics, including the number of g-edges considered in the AVGE computation ($N_e = |S_e|$) and standard deviation of the errors on all these g-edges (σ_e), will also be computed and presented for completeness.

In this set of experiments, nine designs from three technology nodes are tested, and their chief characteristics are listed in the first three columns in Table I.

7.1.1. Tuning the parameters in local resource models. As discussed in Section 3, there is a parameter in each of the three proposed local resource models, respectively: p in Method 1, k in Method 2 and q in Method 3. The parameter has to be tuned for each method to be practically useful. Next we will introduce the process to tune the parameters. Without loss of generality, we use q as an example. We have the following two observations to guide our tuning process. Firstly, parameter q for different circuits in the same technology can be quite similar. In Method 3, q is the ratio of the estimated local resources consumed by a pin a_r to the unit area $a_u = P^2$, where P is the minimum wire pitch on layer M2 and M3. For different circuits in a technology, the local routing resources consumed by each pin, a_r, on average, should be similar, since design rules for different circuits in the each technology are the same, such as the minimum area of a shape, the minimum width of wires and the minimum distance between two wires on layer M2 and M3. Since a_r is similar and a_u is the same for different circuits in a technology, parameter q will be similar for different circuits in the same technology. The experimental results in Section 7.1.2 and 7.1.3 will demonstrate that the parameters tuned on one design work well for the other designs in the same technology, verifying this observation. Secondly, q can be different for different technologies, because the design rules are quite different for each technology, and thus a_r and a_u can also be different. Based on these observations, we only need to tune parameter q for a circuit in each technology.

Since automatic parameter configuration is a complex optimization problem that researchers are still actively working on [Ansótegui et al. 2009; Eiben and Smit 2011; Bergstra and Bengio 2012] and it is beyond the scope of this work, a heuristic tuning method is used in this work: a three-pass tuning process. Firstly, a coarse tuning is performed. A series of trial values starting from X_L to X_U with an increment I_1, are tested, and then we select the three adjacent values (X_{i-1}, X_i, X_{i+1}), where X_i produces the smallest error, specifically, AVGE. In the case $X_i = X_L$ or $X_i = X_U$, we set X_{i-1} to X_L or X_{i+1} to X_U, respectively. Secondly, a finer tuning pass is conducted. Another

[8]One goal of this work is to replace the Reference Analyzer in the routing toolkit by creating another congestion analyzer that has good correlation with it but runs much faster than it.

series of values starting from X_{i-1} to X_{i+1} with an increment I_2, are tested, and then we select the three adjacent values (Y_{j-1}, Y_j, Y_{j+1}), where Y_j produces the smallest error (the boundary cases can be treated similar to the first pass). Note that I_2 is usually much smaller than I_1, e.g., $I_2 = 0.1I_1$. Thirdly, a series of values starting from Y_{j-1} to Y_{j+1} with an increment I_3, are tested, and then we select the value that produces the smallest error. The increment I_3 is usually much smaller than I_2, e.g., $I_3 = 0.1I_2$. In our experiments, we set $X_L = 0$, $X_U = 10$, $I_1 = 1$, $I_2 = 0.1$ and $I_3 = 0.01$ by our experience.

In addition, though three local resource models are proposed and will be analyzed in this work, there will be only one winner among the three and thus only one parameter will need tuning in the practical application of our method.

Table I. Major characteristics of the benchmarks and the tuned parameters for each technology in Analyzer A1–A3.

Technology	Circuits	#nets	Tuned parameters		
			A1 (p)	A2 (k)	A3 (q)
32 nm	ckt_fb	320,357	2.20	0.30	7.84
	ckt_dl16	1,191,615			
	ckt_dl12	1,337,659			
45 nm	ckt_i	354,771	1.18	0.15	3.38
	ckt_y	324,102			
	ckt_m	118,911			
65 nm	ckt_12	1,660,223	1.90	0.27	6.90
	ckt_18	533,530			
	ckt_x	464,661			

7.1.2. Improving congestion analysis accuracy. This section presents the impact of our methods about local resources modeling on the overall accuracy of congestion analysis.

With the tuning process discussed in Section 7.1.1, the parameters for Method 1–3 are tuned for the first design from each technology node shown in Table I, and then the tuned parameters are used to test the other two designs in the same technology node. In this set of experiments, a g-cell size equal to 20 tracks is used.

In Table I, the column "tuned parameters" lists the parameters tuned for the three methods of local resource modeling[9] and the error statistics including AVGE, σ_e and N_e are presented in Table II. In Table II, the row "Average" shows the average values over all the designs. The AVGE data of Analyzer A1–A3 are much better than those of Analyzer A0 while the data of σ_e and N_e are quite similar for all the four analyzers, which indicates that the analyzers with the three proposed methods for local resource modeling (Analyzer A1–A3) can achieve more accurate routability evaluation than Analyzer A0 without any local resource modeling. This is also true for the designs which the parameters are not tuned for, demonstrating the effectiveness of the proposed local resource modeling methods. Furthermore, the three methods produce very similar accuracy on average, though Method 2 is a little better than the other two methods in terms of AVGE. Note that the values of N_e for Analyzer A1–A3 are a little larger than that for Analyzer A0, since by modeling local congestion, more g-edges become congested in Analyzer A1–A3. Though some of these g-edges may even overestimate the congestion compared with the Reference Analyzer, the overestimation is limited, as for these analyzers, N_e is close to 1 and σ_e is limited.

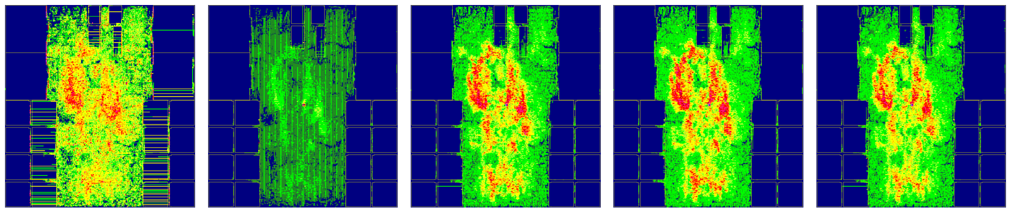
To visually see the impact of local resource modeling on the accuracy of routability evaluation, in Fig. 17, we show the results of running all the five analyzers on design ckt_fb. From Fig. 17(b) we see that Analyzer A0 with no modeling of local routing resources significantly underestimates

[9]The tuned parameters are different from those in [Wei et al. 2012], because our fast congestion analyzer have been improved a lot to be able to make more routing effort in less runtime, compared with the version used in [Wei et al. 2012], and then the local resources added now is more accurate, while those in [Wei et al. 2012] are likely fewer than required, since the not enough routing effort may offset some local resources which should be added but in fact not.

Table II. Comparison of the accuracy in routability evaluation using Analyzer A0–A3, with g-cell size 20 tracks. N_e is shown as normalized by N_e^{RA} that is the number of congested g-edges in the Reference Analyzer.

Circuits	AVGE ($\times 10^{-2}$)				σ_e ($\times 10^{-2}$)				N_e (normalized by N_e^{RA})				N_e^{RA}
	A0	A1	A2	A3	A0	A1	A2	A3	A0	A1	A2	A3	
ckt_fb	15.41	9.21	9.08	9.18	15.94	15.75	15.69	15.85	1.000	1.026	1.028	1.024	80,831
ckt_dl16	13.21	6.90	6.69	6.90	7.40	7.48	7.33	7.48	1.001	1.007	1.008	1.007	491,922
ckt_dl12	14.75	9.66	9.49	9.59	10.46	10.28	10.37	10.33	1.000	1.003	1.004	1.003	407,830
ckt_i	12.93	8.72	8.66	8.78	12.29	12.51	12.55	12.56	1.006	1.060	1.065	1.048	70,037
ckt_y	12.23	8.76	8.87	8.79	9.55	9.40	9.85	9.37	1.009	1.022	1.027	1.021	81,282
ckt_m	11.62	7.82	7.69	7.98	9.72	8.52	8.21	8.96	1.015	1.035	1.036	1.032	21,770
ckt_12	10.44	6.26	6.13	6.28	7.73	7.32	7.30	7.39	1.004	1.043	1.051	1.047	479,775
ckt_18	11.37	8.21	7.96	8.45	9.38	8.85	8.89	8.97	1.009	1.023	1.025	1.021	287,576
ckt_x	8.13	5.84	5.67	5.87	6.11	5.71	5.55	5.73	1.009	1.037	1.042	1.036	252,730
Average	12.23	7.93	7.81	7.98	9.84	9.53	9.53	9.63	1.006	1.028	1.032	1.027	1

the actual congestion. Alternatively, the congestion maps from Analyzer A1–A3 (Fig. 17(c)-17(e)) are much closer to the one obtained from the Reference Analyzer[10], both in terms of the congested regions and their intensity. This result assumes significance in the context of using analyzers within congestion mitigation tools such as CRISP [Roy et al. 2009], where the effectiveness of the tool is highly dependent on accurately identifying the regions of high congestion as well as their relative intensity.



(a) Reference Analyzer (b) Analyzer A0 (c) Analyzer A1 (d) Analyzer A2 (e) Analyzer A3

Fig. 17. Congestion maps for ckt_fb with five analyzers using a g-cell size of 20 tracks.

7.1.3. Scalability of the proposed methods on increasing g-cell size. In last section, we have shown that all the three proposed methods for local resource modeling work similarly well on the nine designs when the g-cell size is set to 20. In this section, we will test their scalability on increasing g-cell size.

We rerun the experiments on all the nine designs using the same tuned parameters but a g-cell size of 80 tracks. In Table III, the error statistics including AVGE, σ_e and N_e are presented, and the row “Average” shows the average values over all the designs. Comparing the error data in Table III with those in Table II, we can see that different analyzers show different scalability. When using the same parameters tuned with the g-cell size of 20 tracks, Analyzer A2, integrated with the simple pin-density-based method (Method 2) for local resource modeling, on average, has the largest error (in terms of AVGE and σ_e) among all the analyzers, and the error is even larger than Analyzer A0 without any local resource modeling on some designs[11]. Analyzer A1, integrated with the Steiner-tree-based method (Method 1) for local resource modeling, has smaller errors (in terms of AVGE

[10]It is expected that the congestion maps obtained from Analyzer A1–A3 do not exactly match those from Reference Analyzer, since they run much faster, and do not work as hard as the Reference Analyzer. However, Analyzer A1–A3 can generally predict the hot spots well.

[11]The reason why in [Wei et al. 2012] Method 2 showed good scalability on design ckt_12 is that the blockages added from local resources at that time were likely fewer than the real amount using the small parameter $k = 0.05$, which is only 19% of the new parameter value 0.27. Thus, when using a large g-cell size 80, the scalability problem was not obviously visible in [Wei et al. 2012].

Table III. Comparison of the accuracy in routability evaluation using different analyzers and pre-tuned parameters from Table I, with g-cell size 80 tracks. N_e is shown as normalized by N_e^{RA} (the same notation as used in Table II).

Circuits	AVGE ($\times 10^{-2}$)				σ_e ($\times 10^{-2}$)				N_e (normalized by N_e^{RA})				N_e^{RA}
	A0	A1	A2	A3	A0	A1	A2	A3	A0	A1	A2	A3	
ckt_fb	8.10	5.82	6.04	3.73	4.72	5.24	5.40	3.04	1.000	1.041	1.045	1.008	4,176
ckt_dl16	13.10	5.95	7.58	6.47	4.26	4.12	4.76	3.45	1.000	1.015	1.018	1.004	28,803
ckt_dl12	10.51	5.81	6.12	5.88	6.34	4.50	4.77	4.15	1.000	1.016	1.025	1.002	20,148
ckt_i	7.97	10.27	13.96	5.43	5.89	8.93	9.66	6.09	1.003	1.146	1.193	1.088	3,713
ckt_y	9.69	10.41	14.40	6.32	5.84	11.05	13.97	5.56	1.005	1.091	1.149	1.028	4,554
ckt_m	9.11	9.11	14.48	4.70	6.04	7.34	11.59	4.13	1.048	1.209	1.411	1.106	1,011
ckt_12	9.23	8.41	11.12	4.49	4.69	5.82	6.47	3.51	1.002	1.133	1.174	1.036	26,116
ckt_18	8.48	5.58	8.27	5.82	7.05	6.32	7.35	5.90	1.006	1.124	1.180	1.021	15,567
ckt_x	5.43	5.85	10.89	3.20	3.69	4.77	6.63	2.67	1.006	1.219	1.347	1.029	13,193
Average	9.07	7.46	10.32	5.12	5.39	6.46	7.85	4.28	1.008	1.110	1.171	1.036	1

and σ_e) than Analyzer A2 on all the designs, but has larger errors than Analyzer A0 for all the three 45 nm designs. Moreover, the values of N_e for Analyzer A1 and A2 are much larger than 1, indicating these analyzers probably have significant overestimation problem, which can be verified in Fig. 18. These results show the bad scalability in Method 1 and Method 2, which is caused by the factors discussed in Section 3.5.1. In contrast, Analyzer A3, integrated with the enhanced method combining the Steiner tree technique and the pin-density technique considering pin distribution (Method 3), achieves the best scalability and the smallest error on average among all the four congestion analyzers A0–A3. This shows the effectiveness of the proposed techniques in Method 3.

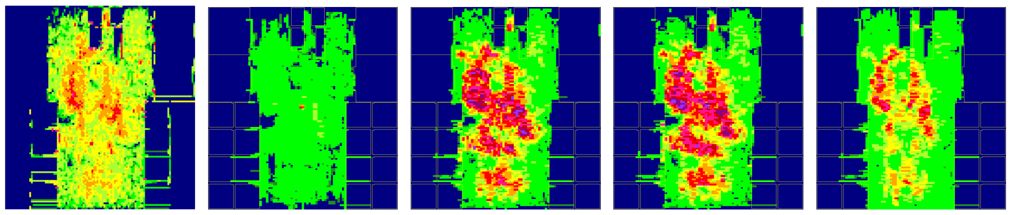

(a) Reference Analyzer (b) Analyzer A0 (c) Analyzer A1 (d) Analyzer A2 (e) Analyzer A3

Fig. 18. Congestion maps for ckt_fb with five analyzers using a g-cell size of 80 tracks.

Fig. 18 shows the congestion maps from all the analyzers on design ckt_fb with the g-cell size set to 80 tracks. Analyzer A0 without local resource modeling still significantly underestimates the actual congestion. Analyzer A1 and A2 both generate over-pessimistic congestion maps as compared with the Reference Analyzer. The congestion map from Analyzer A3 is closest to that from Reference Analyzer among all analyzers (A0–A3), demonstrating the good scalability of the proposed enhanced method (Method 3).

7.1.4. Runtime and acceleration by using a larger g-cell. In this section, we will present the analysis on the runtime of different methods for local congestion modeling and different analyzers, with g-cell sizes of 20 and 80 tracks. We will show that the proposed local resource modeling methods only take a small portion of the runtime of the whole congestion analysis, and with good local resource modeling, congestion analysis can be accelerated by using a larger g-cell size.

In Table IV, the column "LRM (size 20)" lists the CPU time of the preprocessing step for local resource modeling (LRM) in A1–A3 with a g-cell size of 20 tracks, "Total runtime" shows the CPU time for Analyzer A0–A3 ("size 20": g-cell size 20 tracks, and "size 80": 80 tracks), and "RA20" presents the CPU time of Reference Analyzer[12] with a g-cell size of 20 tracks. In the row "ratio", the numbers in the column "LRM (size 20)" list the ratio of LRM CPU time to the total runtime of the corresponding analyzer, and the other numbers show the CPU time normalized by that of Analyzer A3 (g-cell size 20), averaged over all the designs. Let us first look at the runtime of different LRM methods. From column "LRM (size 20)", we can see that LRM with Method 2 (sub-column "A2"), the pin-density method, runs fastest among all the three methods, due to its simplicity. Method 3 (sub-column "A3"), the enhanced method, is the slowest, but still fast enough in practice, taking up only 5.9% of the total routing time on average[13]. Now look at the runtime of different analyzers with g-cell size 20 tracks. Among Analyzer A0–A3, Analyzer A0 runs fastest since it does not have any local resource modeling, and sees the least congestion, translating to the least routing effort. The runtime of Analyzer A1–A3 with LRM is similar and a little longer than that of Analyzer A0, but much shorter than that of the Reference Analyzer[14].

Table IV. Runtime comparison for congestion analyzers with different g-cell sizes.

Circuits	LRM (size 20) (s)			Total runtime (size 20) (s)				RA20 (s)	Total runtime (size 80) (s)			
	A1	A2	A3	A0	A1	A2	A3		A0	A1	A2	A3
ckt_fb	3.5	2.6	6.1	36.7	57.5	63.7	70.1	1499.1	18.7	28.5	28.7	32.0
ckt_dl16	12.8	9.5	25.2	656.4	896.2	953.6	1075.8	129927.5	107.1	171.8	173.3	186.1
ckt_dl12	13.6	9.1	26.5	444.0	725.0	708.9	698.7	29614.2	105.0	157.4	160.0	168.8
ckt_i	3.0	2.1	5.3	39.3	56.4	52.1	58.9	1564.1	19.0	25.3	25.9	29.2
ckt_y	3.0	1.9	5.3	47.6	62.1	63.4	69.7	2790.4	16.9	25.5	25.7	26.9
ckt_m	1.0	0.7	1.6	11.5	15.6	15.7	17.1	503.8	5.7	7.5	7.4	8.5
ckt_12	17.6	11.8	29.1	355.5	513.1	530.7	568.6	41425.5	118.0	159.0	168.3	183.8
ckt_18	5.8	3.5	9.7	227.1	321.3	323.9	361.0	8354.4	46.8	69.4	68.2	76.5
ckt_x	4.9	3.5	8.8	111.8	167.7	178.3	198.7	4984.0	32.7	48.3	52.4	51.7
Ratio	3.8%	2.5%	5.9%	0.62	0.90	0.92	1.00	44.64	0.21	0.30	0.31	0.34

Considering accuracy, scalability and runtime, Method 3 is the winner among all the three methods of local resource modeling, and we use Method 3 for all subsequent analyses in this paper.

Since our method can accurately incorporate the effects of local routing within a g-cell, it provides the freedom to increase the size of the g-cell, thereby accelerating congestion analysis. As shown in Table IV, with Method 3, when increasing g-cell size from 20 to 80, runtime is reduced by 66% on average.

7.1.5. Better prediction of detailed routing issues. Often a design that seems routable after global routing can end up with multiple opens/shorts at the end of detailed routing. Early prediction of such issues without performing the time-consuming step of detailed routing is highly beneficial as it enables designers to take appropriate measures to improve overall turn-around time for design closure. As mentioned in Section 1, a global router that ignores the modeling of local resources may significantly mispredict design routability and cannot predict the opens/shorts locations, as illustrated in Fig. 1. These opens/shorts indicate the problematic locations in detailed routing, which, in our experience, are usually due to high local congestion at these locations. Next, we will demonstrate that our proposed local resource modeling method can enable the congestion analyzer to predict detailed routing opens/shorts with high fidelity. Fig. 19 shows the comparison of the opens/shorts plots (during an intermediate stage of an industrial strength detailed router) and the congestion

[12]The runtime shown in [Wei et al. 2012] for Reference Analyzer is the wall-clock time, which explains the different numbers in the table.
[13]In [Wei et al. 2012], it is mentioned that Method 2 is 3.6 times faster than Method 1. However, the difference in the runtime between the two methods has become smaller in this work, because data structure in the fast congestion analyzer used has been improved since then, which sped up Method 1 more than Method 2 due to the different operations involved in the two methods.
[14]Note that Reference Analyzer is using multi-threads routing, and the numbers listed for Reference Analyzer are total CPU time of all the threads, while the wall time can vary depending on the number of threads used.

maps from Analyzer A0 and Analyzer A3 on design ckt_fb[15]. Comparing these plots, we see that Analyzer A3 (with Method 3 to model local routing resources) clearly indicates congestion hot spots, translating to the problematic regions in detailed routing, which are not captured by Analyzer A0.

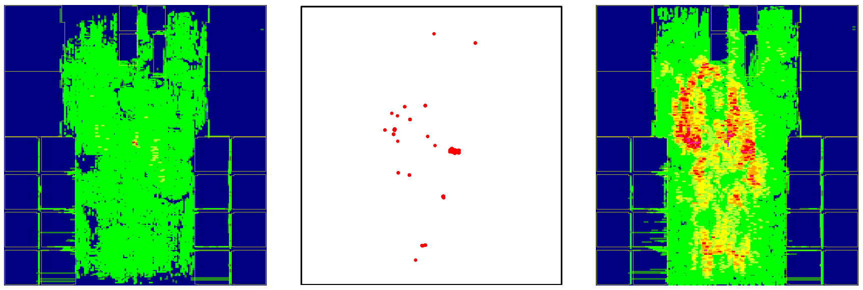

(a) Congestion map from Analyzer A0 (b) Open/short map from detailed routing (c) Congestion map from Analyzer A3

Fig. 19. The open/short map and congestion plots for ckt_fb. Analyzer A3 integrated with Method 3 to model local routing resources could predict the problematic regions in detailed routing with higher fidelity than Analyzer A0.

To quantitatively measure the predictability of the analyzers, we compute the ratio of the number of opens/shorts present in g-cells with congestion greater than 85% to the total number of opens/shorts in the design. This ratio is called *match ratio*, which measures the extent of matching between opens/shorts and the highly congested regions on the design. Here, using 85% as the threshold to consider g-cells in the computation is based on our prior experience that regions with such high global congestion are usually problematic in detailed routing. The match ratios for Analyzer A0 and A3 are computed as 0 and 0.98, respectively. In other words, Analyzer A3 is able to point to the congested regions which capture 98% of all opens/shorts, while A0 does not capture any. This further demonstrates the effectiveness of the proposed local resource model (Method 3) in enabling the congestion analyzer to predict the problematic regions in detailed routing.

7.2. Impact of the smoothing technique on routability evaluation

As for the proposed smoothing technique, we have showed some results in Section 4 on a motivating example, and next we will present more results on several industrial circuits in this section. In our experiments, we use $\sigma = l/2 = 1/2$.

Table V shows the effects of applying the smoothing technique once to the solutions from a congestion analyzer on several designs. In the table, column "C.A." lists the results of an industrial congestion analyzer, column "G.R." the results of an industrial global router and column "Smooth" the results after applying the proposed smoothing technique to "C.A." results. It can be seen that the proposed smoothing technique can reduce the noise ratio from more than 6% down to below 4%, and the ACE metrics computed from the smoothed congestion maps become more accurate than those from the congestion analyzer's maps, when compared with those from the global router's maps, which demonstrates the effectiveness of the smoothing technique.

Fig. 20 shows the congestion maps for the congestion analysis solution, the smoothed solution and the global routing solution on design ckt_x. It can be seen that the congestion map from the congestion analyzer has many more hot spots than that from the global router, and there are obvious noisy hot spots in the map. After smoothing, the map becomes smoother with fewer noisy hot spots than the original map, and the intensity of congestion is reduced by some extent, which causes the calculated ACE metrics to be closer to those for the global routing solution. Note that the congestion

[15]For this experiment, a g-cell size of 40 tracks is used, and Fig. 19(a) and Fig. 19(b) are copied from Fig. 1 for the convenience of comparison.

Table V. Smoothing reduces noisy hot spots and improves the accuracy of routability evaluation by a congestion analyzer.

Circuits	Noise ratio (%)			ACE metrics (0.5, 1, 2, 5) (%)		
	C.A.	Smooth	G.R.	C.A.	Smooth	G.R.
ckt_fb	16.39	2.55	8.64	(92.7, 91.1, 89.3, 86.2)	(91.0, 89.5, 87.8, 85.2)	(88.9, 88.1, 86.9, 84.4)
ckt_y	13.99	3.93	6.24	(99.9, 97.6, 95.8, 92.8)	(98.9, 97.0, 95.1, 92.3)	(95.5, 92.9, 90.2, 86.9)
ckt_x	6.34	1.00	2.86	(97.2, 94.4, 92.2, 88.9)	(96.6, 93.9, 91.6, 88.4)	(93.2, 89.7, 87.3, 84.6)

maps after smoothing do not match well with those from the global router, because routing effort from the congestion analyzer is much less than that from the global router, which cannot be compensated by only using the smoothing technique.

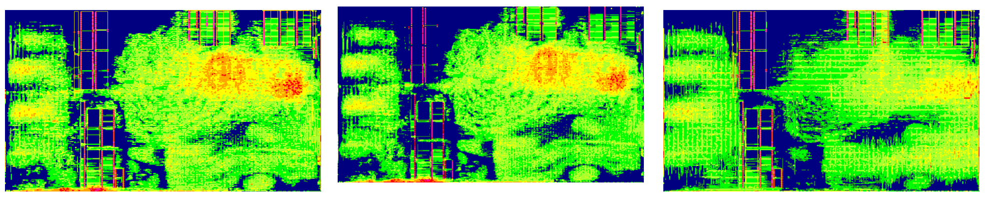

(a) Congestion analysis (b) After smoothing (c) Global routing

Fig. 20. Congestion maps on design ckt_x show that smoothing helps improve the accuracy in routability evaluation compared with the global routing solution.

7.3. Impact of the ACE metric on routability evaluation

7.3.1. Comparison of routability metrics. Visual inspection of a congestion plot is widely used to quickly evaluate the routability of a design point. We now demonstrate that our new metric can represent a congestion plot with higher fidelity compared to prior metrics for routability evaluation. Fig. 21 displays the congestion plots from two global routing solutions on an identical placement for the design ckt_s, which is a 45 nm design with 1,006,029 nets. The corresponding values of the different congestion metrics for the routing solutions are given in Table VI. For the ACE metric, the congestion is expressed as an ordered pair representing (Horizontal, Vertical) layer congestion.

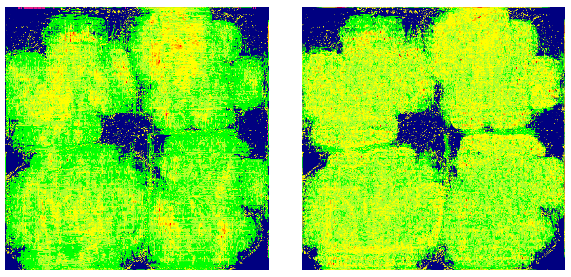

(a) Solution 1 (b) Solution 2

Fig. 21. Congestion plots for two routing solutions on design ckt_s.

From Table VI, the overflow-based metrics[16] indicate that Solution 1 has better routability, while the net-congestion-based metrics indicate that Solution 2 is better, as $ACN(20)$ of Solution 2 is much

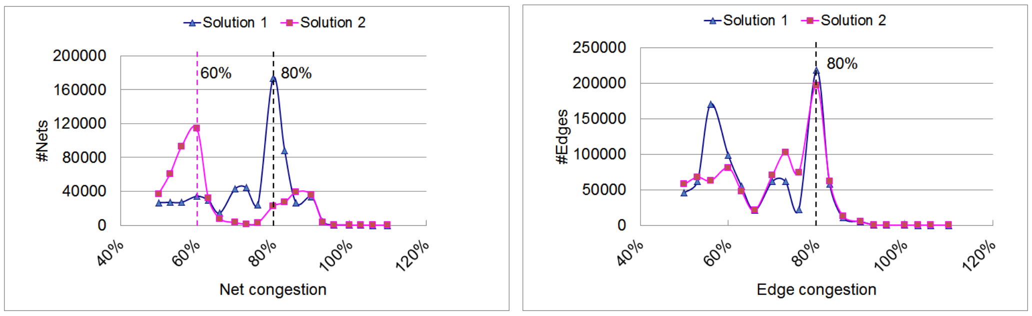
[16]To counteract the drawbacks of overflow metrics discussed earlier, when we calculate overflow, the capacity is scaled down to 80% of the original, and the overflow is in unit of number of minimum-width tracks, e.g., one overflowed track on a layer with 4× width tracks would be counted as four in the overflow number.

better than that of Solution 1, even though $WCI(90)$ and $WCI(100)$ are slightly worse. However, a visual examination of the congestion plots indicates that they are quite similar, demonstrating the deficiencies in the existing metrics. Alternatively, our new metric correctly identifies the congestion of these two routing solutions to be similar.

Table VI. Congestion metrics for two routing solutions on design ckt_s.

Metrics		Solution 1	Solution 2
Overflow based metrics	TOF	194373	217499
	MOF	10	11
Net congestion based metrics	$ACN(20)$	84.41	77.97
	$WCI(90)$	39494	40548
	$WCI(100)$	274	276
New metric (%)	$ACE(0.5)$	(90.47, 90.54)	(90.27, 90.46)
	$ACE(1)$	(89.23, 89.12)	(89.13, 89.10)
	$ACE(5)$	(85.93, 85.45)	(86.14, 85.49)
	$ACE(10)$	(84.16, 83.39)	(84.59, 83.51)
	$ACE(20)$	(82.48, 81.58)	(82.57, 81.53)

The significant difference in the $ACN(20)$ values for the two comparable routing solutions can be explained by Fig. 22(a) which plots the distribution of the worst congestion on the nets. In Fig. 22(a), the dash lines with labels indicate the minimum congestion for the top 20% congested nets: 80% for Solution 1 and 60% for Solution 2. In computing the $ACN(20)$ metrics, a number of nets with low congestion (around 60%) are included for Solution 2, while only the nets with congestion $\geq 80\%$ are considered for Solution 1. This leads to the difference in the two $ACN(20)$ values. Alternatively, Fig. 22(b) plots the distribution of congestion on the g-edges, where the dash line with a label indicates the minimum congestion for the top 20% congested g-edges: 80% for both solutions. Furthermore, we observe that the distributions for the two routing solutions are similar above 80%. This explains why the ACE metrics shown in Table VI for the two solutions are similar, which matches our visual observation that the two solutions have similar congestion.

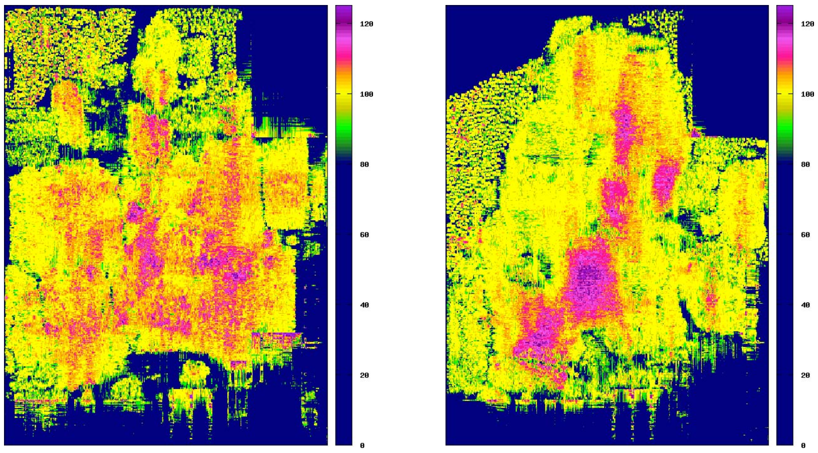


(a) Distribution of net congestion (b) Distribution of g-edge congestion

Fig. 22. The distribution of congestion for two routing solutions of ckt_s (the distribution of congestion $< 50\%$ is skipped since it is not important).

7.3.2. Further comparison between total overflow metric and ACE metric. In this section, we further demonstrate the limitations of the TOF metric in predicting design routability, and the ability of the ACE metric to accurately represent the design congestion.

For our experiments, we evaluate the placements from two of the top-performing teams (Ripple [He et al. 2011] and SimPLR [Kim et al. 2011]) in the ISPD-2011 routability-driven placement contest [Viswanathan et al. 2011].



(a) Ripple with TOF of $542, 786$ (b) SimPLR with TOF of $514, 614$

Fig. 23. The congestion maps and TOF values for Ripple and SimPLR placement solutions on superblue12. The regions colored purple indicate the congestion hot spots.

We recall that TOF was the primary routability evaluation metric used during the ISPD-2011 contest. Let us consider the Ripple and SimPLR placements for the design superblue12. The TOF values as reported by the ISPD-2011 contest congestion analyzer are: Ripple $(542, 786)$ and SimPLR $(514, 614)$, indicating that the SimPLR placement has better routability. Fig. 23 shows the corresponding congestion maps[17]. From Fig. 23, we make two observations. First, on this design, Ripple seems to spread cells more than SimPLR. As a result, the congestion hot spots for Ripple are more distributed than the hot spots for SimPLR. Second, the regions colored purple, with congestion $> 120\%$, are much larger in SimPLR's congestion map as compared to Ripple's. Based on these two observations and our experience on industry designs, we believe that the SimPLR placement is worse in terms of routability as compared to the Ripple placement. However, this is not captured by the TOF metric, highlighting its limitations.

Table VII. The ACE metric (shown as percentage) computed on routing solutions for Ripple's and SimPLR's placement solutions on superblue12.

Solutions	$ACE(0.5)$	$ACE(1)$	$ACE(2)$	$ACE(5)$	$ACE(10)$	$ACE(20)$
Ripple	126.25	123.00	120.62	114.32	109.10	104.55
SimPLR	130.89	126.34	123.17	118.97	113.49	106.74

Next, we compute the ACE metric on the same routing solutions as in Fig. 23, and show the results in Table VII. Note that only the maximum from horizontal and vertical layers is shown for each $ACE(x)$ value. From Table VII, we see that $ACE(0.5)$ and $ACE(1)$ for SimPLR are higher than that for Ripple, implying that the hot spots in the SimPLR congestion map have higher congestion than the ones in the Ripple congestion map. Moreover, all the ACE values for SimPLR are worse than Ripple. Together, they indicate that SimPLR has worse routability compared to Ripple on this design. This correlates well with our observations from the congestion maps.

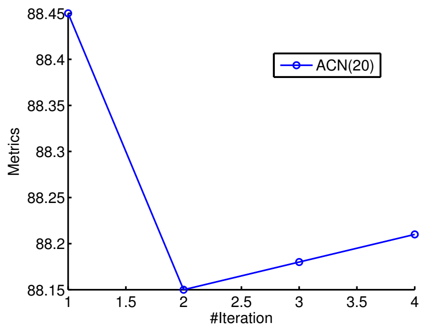
These observations indicate that ACE metric can represent the congestion map with higher fidelity than TOF metric.

[17]Since ISPD-2011 contest used an academic congestion analyzer with a congestion target/threshold different from the industrial analyzers we used in this work, a color map other than that in Fig. 1(a) is used here.

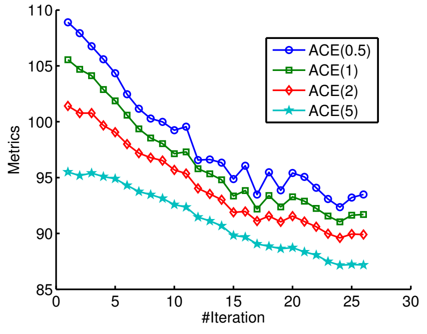
7.3.3. Guiding routability optimization. In this section, we will discuss the performance of different metrics in guiding routability optimization, and demonstrate that a good congestion metric such as the ACE metric is essential to effectively guide routability optimization. We will first explain the experiments in details using circuit ckt_y as an example, and then present the results on the other two circuits.

In our experiments, we employ CRISP [Roy et al. 2009], an incremental placer that iteratively spreads logic cells in congested regions to improve the routability of a design. To track its progress, CRISP relies on a congestion metric to compare the routability of placements over successive iterations. It terminates if the metric does not improve for two consecutive iterations. In other words, it returns the placement solution from iteration i if the congestion metrics for both iterations $i+1$ and $i+2$ are worse than that for iteration i.

We first employ the $ACN(20)$ congestion metric to guide CRISP and check the stopping criterion as outlined above. A plot of the progression of the $ACN(20)$ metric across CRISP iterations on circuit ckt_y is shown in Fig. 24(a). We see that CRISP terminates after just four iterations and returns the placement solution from the second iteration. The reason for the early termination of CRISP is as follows: CRISP spreads cells in the most congested regions of the design. This may reduce the number of nets in regions with 100% congestion ($WCI(100)$), but increase the number of nets in regions with over 90% congestion ($WCI(90)$). As a result, the $ACN(20)$ metric may not change, or in fact degrade (Fig. 24(a), iterations three and four). Such a scenario can cause CRISP to exit prematurely without resolving all the congestion issues.



(a) CRISP terminates after four iterations when using the $ACN(20)$ metric.



(b) CRISP iterates more when using the ACE metric for congestion evaluation.

Fig. 24. The ACE metric provides a more accurate view of the congestion, enabling CRISP to be more effective.

Next, we replace the $ACN(20)$ metric with a set of values obtained from the ACE metric. As mentioned before, the ACE metric expresses congestion as an order pair of (Horizontal, Vertical) layer congestion values. We use the maximum of these two values to check the stopping criterion for CRISP. In this experiment, we use $ACE(0.5)$, $ACE(1)$, $ACE(2)$ and $ACE(5)$ and adopt the following strategy: during each iteration, we determine the four metric values, and consider it as an improvement in congestion if there is a reduction in any one of the metric values. Same as before, CRISP terminates if the congestion evaluated by the ACE metric does not improve for two consecutive iterations. Note that using the ACE metric does not affect the strategy of CRISP to mitigate congestion. The ACE metric is just a metric to score or represent congestion, and it does not change the congestion profile of a placement and the congested bins wherein CRISP needs to operate and alleviate congestion. CRISP uses the same strategy as before to move cells in each iteration: identify the most congested bins, inflate the cells in the those bins, spread the cells out of congested bins and assign them to legal locations. The effect of CRISP using a set of metric values derived from the ACE metric is shown in Fig. 24(b), wherein CRISP runs for up to 26 iterations on ckt_y, and returns the solution in iteration 24.

The intuition behind using a set of edge-congestion-based metric values is as follows: since CRISP spreads cells from the topmost congested g-cells, it should reduce the congestion on the topmost congested g-edges. This is captured by the $ACE(0.5)$ metric value. In doing so it may increase the congestion on other g-edges, but this is still acceptable as it is reducing the peak of the congestion histogram. Over successive CRISP iterations, the $ACE(0.5)$ metric value may saturate, but a reduction in any of the other metric values implies that CRISP is still improving congestion, except that it is now targeting g-edges that are deeper in the congestion histogram.

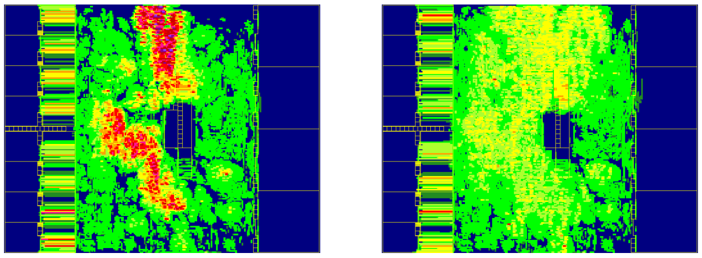

(a) Using the $ACN(20)$ metric (b) Using the ACE metric

Fig. 25. Post-CRISP congestion plots for ckt_y when using different metrics to check the stopping criterion.

In addition, Fig. 25 shows the post-CRISP congestion plots for ckt_y when employing the two congestion metrics. Compared with the solutions obtained by using $ACN(20)$ (Fig. 25(a)), there is a significant improvement in the design routability (Fig. 25(b)) by running CRISP for more iterations, as enabled by the ACE metric.

Table VIII. Comparison of the solutions obtained by CRISP driven by $ACN(20)$ and the ACE metric, respectively. "#iter" lists the iteration number of the CRISP solutions, while $ACN(20)$ and ACE metrics (shown in percentage) indicate the congestion level of the solutions.

Circuits	CRISP driven by $ACN(20)$ (%)			CRISP driven by ACE metric (%)		
	#iter	$ACN(20)$	ACE metrics (0.5, 1, 2, 5)	#iter	$ACN(20)$	ACE metrics (0.5, 1, 2, 5)
ckt_y	2	88.15	(107.93, 104.69, 100.77, 95.18)	24	83.68	(92.35, 91.04, 89.60, 87.15)
ckt_sb2	1	81.36	(86.73, 85.04, 83.22, 81.10)	30	79.56	(82.14, 81.38, 80.64, 80.00)
ckt_pf	2	87.73	(96.08, 94.67, 92.80, 89.08)	14	87.01	(95.41, 93.72, 91.52, 88.04)

Besides ckt_y, we have also found the similar results on circuits from other technologies. Table VIII summarizes the results on ckt_y and another two 32 nm designs including ckt_sb2 and ckt_pf when using $ACN(20)$ and the ACE metric to drive CRISP, respectively. From the table, we can see that the ACE metric can drive CRISP to iterate more and obtain solutions with better routability than the $ACN(20)$ metric.

7.4. Impact of all techniques working together

We have discussed the individual impact of our proposed techniques, and in this section, we will demonstrate how they work together in the routability evaluation process described in Section 6.

Assume there are two placements A and B for ckt_y shown in Fig. 26, and an EDA tool needs to compare their routability and return the one with better routability. For this purpose, the improved routability evaluation process (Fig. 16(b)) will be used. As a comparison, the results using the traditional process (Fig. 16(a)) will also be shown.

First, a congestion analyzer will be invoked on the given two placements to obtain congestion maps. We use Analyzer A0 and A3 (introduced in Section 7.1) for traditional routability evaluation process and the improved process, respectively. Fig. 27 shows the congestion maps obtained by

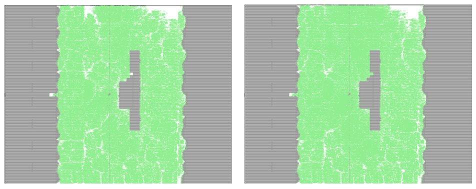

(a) Placement A (b) Placement B

Fig. 26. Two placements for design ckt_y.

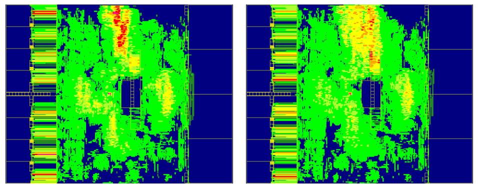

(a) For placement A (b) For placement B

Fig. 27. Congestion maps from Analyzer A0 for ckt_y.

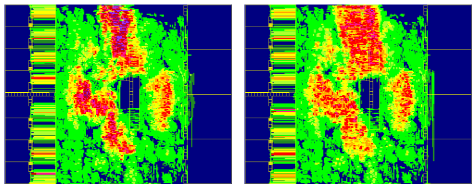

(a) For placement A (b) For placement B

Fig. 28. Congestion maps from Analyzer A3 for ckt_y.

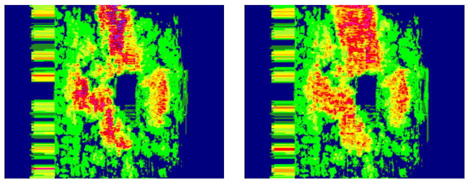

(a) For placement A (b) For placement B

Fig. 29. Smoothed congestion maps based on solutions from Analyzer A3 for ckt_y.

running the Analyzer A0 on placements A and B, while Fig. 28 shows the maps obtained from Analyzer A3. We have two observations:

(1) There are many more hot spots in the congestion maps from Analyzer A3 than from Analyzer A0, which is expected due to the modeling of local routing resources in Analyzer A3. Looking at the congestion map for placement B from Analyzer A0 (Fig. 27(b)), we may think placement B is probably routable, since there are only a few hot spots that are well distributed. However, when looking at the plot from Analyzer A3 in Fig. 28(b), we will think placement B is probably unroutable, since there are several large areas of hot spots in the congestion map.
(2) When comparing the congestion maps for placement A and B from either Analyzer A0 or A3, we find in common that the congestion hot spots in maps for placement B are more distributed and have smaller intensity than for placement A, and it is likely that placement B has better routability than placement A.

As a next step in the improved routability evaluation process, we apply the smoothing technique discussed in Section 4 to the congestion maps from Analyzer A3, and obtain the smoothed congestion maps shown in Fig. 29.

Table IX. Routing metrics for two placements A and B on design ckt_y from traditional and improved routability evaluation processes.

Placement	Traditional process	Improved process (%)						Detailed routing	
	TOF	$ACE(0.5)$	$ACE(1)$	$ACE(2)$	$ACE(5)$	$ACE(10)$	$ACE(20)$	#errors	Time (h)
A	801,520	111.7	108.2	104.1	98.1	93.1	86.6	2,424	49.5
B	804,540	102.7	101.2	99.4	96.2	92.9	87.2	816	29.0

Next, we compute the congestion metrics. We use TOF metric in traditional evaluation process, and the ACE metric in the improved process. The computed metrics are listed in Table IX. The TOF value for placement A is smaller than that for placement B, and then the traditional evaluation process will return placement A as the better solution in terms of routability. Now, looking at the ACE metric (only the maximum between horizontal and vertical layers is reported), we can see that $ACE(0.5)$ and $ACE(1)$ for placement A are higher than that for placement B, implying that the hot spots in the congestion maps for placement A have higher congestion than those in the congestion maps for placement B. Metrics $ACE(2)$ and $ACE(5)$ for placement A are also worse than placement B, while $ACE(10)$ and $ACE(20)$ for both placements are very close, smaller than 1%. Based on these data, the improved routability evaluation process will consider placement B as a better solution than A in terms of routability.

To practically judge which placement has better routability, we run an industrial detailed router on the two placements, and report the results in Table IX. We find that although neither of the two placements is routable, placement B has relatively better routability than placement A, since both the number of routing errors and runtime (wall-clock time reported) on placement B are much better than those on placement A. This verifies our visual observation that placement B has better routability than A. Moreover, this experiment highlights the drawbacks in the traditional evaluation process:

(1) Due to the lack of local routing resource modeling, the traditional routability evaluation tends to be optimistic.
(2) Due to the limitations in TOF metric, the conclusion from the traditional routability evaluation process could be inaccurate.

At the same time, the detailed routing results confirm the advantages of the improved routability evaluation process integrated with our proposed techniques.

Another conclusion can be drawn from this experiment. We have seen that the metrics $ACE(x)$, $x \in \{0.5, 1, 2, 5\}$ for placement A are considerably higher than those for placement B, which correctly indicates that placement B has better routability than A, and is consistent with the results of detailed routing. However, the $ACE(10)$ and $ACE(20)$ metrics for placement B are almost the same or even slightly worse than A, which does not match the fact that placement B has better routability than A. This is because when we use a large value for x, i.e., go too deep in the histogram of g-edge congestion to compute the ACE metrics, the g-edges with small congestion are also counted in, which brings noise to the routability evaluation. Therefore, a conclusion drawn from this experiment is that we should not use an excessively high value for x in $ACE(x)$ when using the ACE metric to evaluate routability, and the values $x = 0.5, 1, 2, 5$ are suggested.

8. CONCLUSION

Fast and accurate routability evaluation techniques are critical to address the increasingly important and difficult issue of routing closure in nanometer-scale physical synthesis. In this work, we have addressed two important aspects of routability evaluation: the accuracy of congestion estimation and a metric for evaluating the routability of a design.

We have shown that ignoring the effects of local congestion can result in large errors during congestion analysis. This observation motivates our models for local routing resources:

(1) Method 1: Based on the Steiner tree wirelength of the local nets;
(2) Method 2: Based on the pin density in each g-cell;
(3) Method 3: An enhanced method combining the good techniques in Method 1 and Method 2, and further considering the scalability on increasing g-cell size and pin distribution.

Experimental results show that the proposed models can improve the accuracy and fidelity of congestion analysis, and better predict detailed routing issues such as opens and shorts. In particular, Method 3 has the best scalability on g-cell size among the three methods, which enables designers to use large g-cells to accelerate the process of congestion analysis, thereby speeding design closure.

Furthermore, we have discussed the effects of noisy hot spots in the congestion maps on routability evaluation, and proposed a smoothing technique, which could be used to obtain more accurate routability evaluation, as demonstrated in our experiments on several industrial circuits.

In addition, we have analyzed the limitations of existing congestion metrics including overflow, etc., and proposed a new metric, the ACE metric, based on g-edge congestion. We have demonstrated that the ACE metric can represent a congestion plot with higher fidelity than other traditional metrics such as overflow. Moreover, we have showed that with the ACE metric, a routability-driven placer can perform better and improve the routability of designs significantly than that with the $ACN(20)$ metric.

Finally, we have presented how our proposed techniques work together in the routability evaluation process and have demonstrated that the improved process with our proposed techniques can predict the routability more effectively and accurately than the traditional routability evaluation process.

ACKNOWLEDGMENTS

The authors would like to thank Dirk Müller and Laleh Behjat for constructive discussions.

REFERENCES

ALPERT, C. AND TELLEZ, G. 2010. The importance of routing congestion analysis. *DAC Knowledge Center Online Article*. http://www.dac.com/back_end+topics.aspx?article=47&topic=2.

ALPERT, C. J., LI, Z., MOFFITT, M. D., NAM, G. J., ROY, J. A., AND TELLEZ, G. 2010. What makes a design difficult to route. In *Proceedings of the ACM International Symposium on Physical Design*. ACM, New York, NY, USA, 7–12.

ANSÓTEGUI, C., SELLMANN, M., AND TIERNEY, K. 2009. A gender-based genetic algorithm for the automatic configuration of algorithms. In *Principles and Practice of Constraint Programming*. Springer, Heidelberg, Germany, 142–157.

BERGSTRA, J. AND BENGIO, Y. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research 13*, 281–305.

BRENNER, U. AND ROHE, A. 2002. An effective congestion driven placement framework. In *Proceedings of the ACM International Symposium on Physical Design*. ACM, New York, NY, USA, 6–11.

CHAN, T., CONG, J., AND SZE, K. 2005. Multilevel generalized force-directed method for circuit placement. In *Proceedings of the ACM International Symposium on Physical Design*. ACM, New York, NY, USA, 185–192.

CHANG, Y.-J., LEE, Y.-T., AND WANG, T.-C. 2008. NTHU-Route 2.0: A fast and stable global router. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Piscataway, NJ, USA, 338–343.

CHEN, H., CHOU, S., AND CHANG, Y. 2010. Density gradient minimization with coupling-constrained dummy fill for CMP control. In *Proceedings of the ACM International Symposium on Physical Design*. ACM, New York, NY, USA, 105–111.

CHEN, H.-Y., CHOU, S.-J., WANG, S.-L., AND CHANG, Y.-W. 2007. Novel wire density driven full-chip routing for CMP variation control. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Piscataway, NJ, USA, 831–838.

CHEN, H.-Y., HSU, C.-H., AND CHANG, Y.-W. 2009. High-performance global routing with fast overflow reduction. In *Proceedings of the Asia-South Pacific Design Automation Conference*. IEEE Press, Piscataway, NJ, USA, 582–587.

CHEN, T., CHAKRABORTY, A., AND PAN, D. Z. 2008a. An integrated nonlinear placement framework with congestion and porosity aware buffer planning. In *Proceedings of the ACM/IEEE Design Automation Conference*. ACM, New York, NY, USA, 702–707.

CHEN, T., JIANG, Z., HSU, T., CHEN, H., AND CHANG, Y. 2008b. NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27,* 7, 1228–1240.

CHO, M., LU, K., YUAN, K., AND PAN, D. Z. 2007. BoxRouter 2.0: Architecture and implementation of a hybrid and robust global router. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Piscataway, NJ, USA, 503–508.

CHO, M., PAN, D., XIANG, H., AND PURI, R. 2006. Wire density driven global routing for CMP variation and timing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. ACM, New York, NY, USA, 487–492.

CHU, C. AND WONG, Y.-C. 2008. Flute: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27,* 1, 70–83.

EIBEN, A. E. AND SMIT, S. K. 2011. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation 1,* 1, 19–31.

GESTER, M., MÜLLER, D., NIEBERG, T., PANTEN, C., SCHULTE, C., AND VYGEN, J. 2012. Algorithms and data structures for fast and good VLSI routing. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. ACM, New York, NY, USA, 459–464.

GESTER, M., MÜLLER, D., NIEBERG, T., PANTEN, C., SCHULTE, C., AND VYGEN, J. 2013. BonnRoute: Algorithms and data structures for fast and good VLSI routing. *ACM Transactions on Design Automation of Electronic Systems 18,* 2, 32:1–32:24.

HE, X., HUANG, T., XIAO, L., TIAN, H., CUI, G., AND YOUNG, E. 2011. Ripple: An effective routability-driven placer by iterative cell movement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Piscataway, NJ, USA, 74–79.

HECKBERT, P. 1997. Fast surface particle repulsion. Tech. Rep. CMU-CS-97-130, Carnegie Mellon University, Pittsburgh, PA, USA.

HSU, M., CHOU, S., LIN, T., AND CHANG, Y. 2011. Routability-driven analytical placement for mixed-size circuit designs. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Piscataway, NJ, USA, 80–84.

HU, J., KIM, M.-C., AND MARKOV, I. 2013. Taming the complexity of coordinated place and route. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. ACM, New York, NY, USA, 1–7.

JIANG, Z., SU, B., AND CHANG, Y. 2008. Routability-driven analytical placement by net overlapping removal for large-scale mixed-size designs. In *Proceedings of the ACM/IEEE Design Automation Conference*. ACM, New York, NY, USA, 167–172.

KAHNG, A. B. AND WANG, Q. 2005. Implementation and extensibility of an analytic placer. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24,* 5, 734–747.

KIM, M.-C., HU, J., LEE, D.-J., AND MARKOV, I. 2011. A SimPLR method for routability-driven placement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Piscataway, NJ, USA, 67–73.

LI, Y., FARSHIDI, A., BEHJAT, L., AND SWARTZ, W. 2012. High performance post-placement length estimation techniques. *International Journal of Information and Computer Science 1,* 6, 144–152.

LOU, J., THAKUR, S., KRISHNAMOORTHY, S., AND SHENG, H. S. 2002. Estimating routing congestion using probabilistic analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 21,* 1, 32–41.

MOFFITT, M. D. 2008. MaizeRouter: Engineering an effective global router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27,* 11, 2017–2026.

NAYLOR, W. C., DONELLY, R., AND SHA, L. 2003. Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer. U.S. Patent 6671859 Bl.

OUMA, D. O., BONING, D. S., CHUNG, J. E., EASTER, W. G., SAXENA, V., MISRA, S., AND CREVASSE, A. 2002. Characterization and modeling of oxide chemical-mechanical polishing using planarization length and pattern density concepts. *IEEE Transactions on Semiconductor Manufacturing 15,* 2, 232–244.

PAN, M. AND CHU, C. 2006. FastRoute: A step to integrate global routing into placement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. ACM, New York, NY, USA, 464–471.

PAN, M. AND CHU, C. 2007. IPR: An integrated placement and routing algorithm. In *Proceedings of the ACM/IEEE Design Automation Conference*. ACM, New York, NY, USA, 59–62.

ROY, J. A., VISWANATHAN, N., NAM, G.-J., ALPERT, C. J., AND MARKOV, I. L. 2009. CRISP: Congestion reduction by iterated spreading during placement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. ACM, New York, NY, USA, 357–362.

SHOJAEI, H., DAVOODI, A., AND LINDEROTH, J. T. 2011. Congestion analysis for global routing via integer programming. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Piscataway, NJ, USA, 256–262.

TAGHAVI, T., ALPERT, C., HUBER, A., LI, Z., NAM, G.-J., AND RAMJI, S. 2010. New placement prediction and mitigation techniques for local routing congestion. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Piscataway, NJ, USA, 621–624.

TIAN, R., WONG, D. F., AND BOONE, R. 2001. Model-based dummy feature placement for oxide chemical-mechanical polishing manufacturability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 20,* 7, 902–910.

VISWANATHAN, N., ALPERT, C., SZE, C., LI, Z., AND WEI, Y. 2012. The DAC 2012 routability-driven placement contest and benchmark suite. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. ACM, New York, NY, USA, 774–782.

VISWANATHAN, N., ALPERT, C. J., SZE, C., LI, Z., NAM, G.-J., AND ROY, J. A. 2011. The ISPD-2011 routability-driven placement contest and benchmark suite. In *Proceedings of the ACM International Symposium on Physical Design*. ACM, New York, NY, USA, 141–146.

WEI, Y. AND SAPATNEKAR, S. S. 2010. Dummy fill optimization for enhanced manufacturability. In *Proceedings of the ACM International Symposium on Physical Design*. ACM, New York, NY, USA, 97–104.

WEI, Y., SZE, C., VISWANATHAN, N., LI, Z., ALPERT, C. J., REDDY, L., HUBER, A. D., TELLEZ, G. E., KELLER, D., AND SAPATNEKAR, S. S. 2012. GLARE: Global and local wiring aware routability evaluation. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. ACM, New York, NY, USA, 768–773.

WESTRA, J., BARTELS, C., AND GROENEVELD, P. 2004. Probabilistic congestion prediction. In *Proceedings of the ACM International Symposium on Physical Design*. ACM, New York, NY, USA, 204–209.

WU, T.-H., DAVOODI, A., AND LINDEROTH, J. T. 2010. A parallel integer programming approach to global routing. In *Proceedings of the ACM/IEEE Design Automation Conference*. ACM, New York, NY, USA, 194–199.

XU, Y., ZHANG, Y., AND CHU, C. 2009. FastRoute 4.0: Global router with efficient via minimization. In *Proceedings of the Asia-South Pacific Design Automation Conference*. IEEE Press, Piscataway, NJ, USA, 576–581.

ZHANG, Y. AND CHU, C. 2011. RegularRoute: An efficient detailed router with regular routing patterns. In *Proceedings of the ACM International Symposium on Physical Design*. ACM, New York, NY, USA, 45–52.

ZHANG, Y. AND CHU, C. 2012. GDRouter: Interleaved global routing and detailed routing for ultimate routability. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. ACM, New York, NY, USA, 597–602.