Multi-layer Global Routing Considering Via and Wire Capacities*

Chin-Hsiung Hsu†, Huang-Yu Chen†, and Yao-Wen Chang†‡
†Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
‡Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

***Abstract*— Global routing for modern large-scale circuit designs has attracted much attention in the recent literature. Most of the state-of-the-art academic global routers just work on a simplified routing congestion model that ignores the essential via capacity for routing through multiple metal layers. Such a simplified model would easily cause fatal routability problems in subsequent detailed routing. To remedy this deficiency, we present in this paper a more effective congestion metric that considers both the in-tile nets and the residual via capacity for global routing. With this congestion metric, we develop a new global router that features two novel routing algorithms for congestion optimization, namely *least-flexibility-first routing* and *multi-source multi-sink escaping-point routing*. The least-flexibility-first routing processes the nets with the least flexibility first, facilitating a quick prediction of congestion hot spots for the subsequent nets. Enjoying lower time complexity than traditional maze and A*-search routing, in particular, the linear-time escaping-point routing guarantees to find the optimal solution and achieves the theoretical lower-bound time complexity. Experimental results show that our global router can achieve very high-quality routing solutions with more reasonable via usage, which can benefit and correctly guide subsequent detailed routing.**

I. Introduction

As IC technology continues to advance, the number of transistors per die will still grow dramatically in the near future, which incurs substantial manufacturing challenges, especially for the modern very-large-scale routing. As indicated in [1], [8], routing is a key step to bring the design-for-manufacturability/yield (DFM/DFY) upstream into the design process, since it determines most of the layout geometries in the physical design flow.

A modern chip may contain billions of transistors and millions of nets. To handle the increasing complexity, a high-quality global router is desired. As reported in [14], a placer integrated with an efficient global router as an interconnect model has a great potential to enhance the optimization consistency of placement and routing. In addition, suffered from the manufacturing closure issues, global routing is the key step to address various nanometer electrical effects, such as crosstalk [17] and CMP (chemical-mechanical polishing) [3]. In 2007 and 2008, the ACM International Symposium on Physical Design (ISPD) held global routing contests to boost the research [9]. The 2007 contest released two sets of benchmarks for 2D and 3D global routing and defined a performance cost metric which ranks the routing results based on the prioritized order: (1) the total overflow, (2) the maximum overflow, and (3) the total wirelength. The 2008 contest further considered the trade-off between wirelength and runtime. The contests signify the importance of global routing and substantially drive the evolution of global routers.

Although the recent global routers demonstrate their superiority over the ISPD'07 benchmarks, we shall point out that the global routing contest just handled a simplified (possibly, over simplified) global routing problem, without considering some crucial design rules such as vias for routing through multiple metal layers. For example, the released 3D benchmarks of the contest only set vertical and horizontal capacities and lack the via capacity between layers, which allows the participants to directly map their 2D routing solutions to 3D ones by layer assignment [4], [16]. These simplifications may not be realistic for practical applications, mainly for attracting more contest participants and also due to the limited contest time.

Specifically, all these routers ignore the *residual via capacity* in a routed region, which might complicate subsequent detailed routing. Chen and Cengiz [2] measured routing congestion with via consideration, but their model is too complex to handle large-scale circuit designs. For example, their experiments handled only 3-track global tiles while the track number of a global tile in the ISPD07 benchmarks can be up to 55. As shown in Fig. 1 (a), a tile may contain both stacked and unstacked vias during detailed routing, and all the stacked vias occupy extra resources. The *in-tile wires* are referred to as the wires routed inside the tile, while the *cross-tile wires* run through the tile. In-tile wires affect not only wire capacity [5] but also via capacity because the in-tile wires might block other detailed routing paths to pass through this tile. As the configuration shown in Fig. 1 (b), a misled global router may guide other nets into a tile already full of in-tile wires since the via capacity is not considered, which would greatly degrade the routability of subsequent detailed routing. In addition, the via capacity would decrease as the number of cross-tile wires increases; Fig. 1 (c) shows that the residual via capacity highly depends on the cross-tile wires, which changes *dynamically* during routing. Consequently, the resulting global routing solutions (even without any overflow) may not lead to the completion of detailed routing, which might further complicate the routing problem and worsen the design convergence.

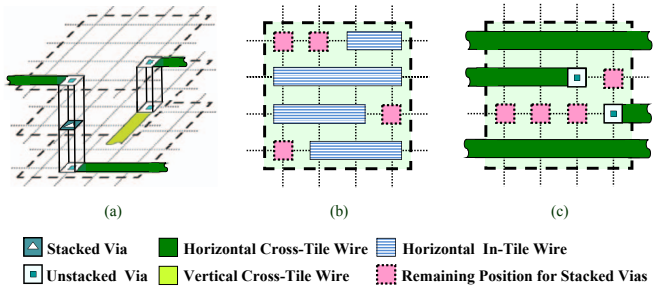

Fig. 1. The routing abnormality incurred by a pathological global routing congestion metric. (a) During detailed routing, a global tile may contain stacked and unstacked vias. (b) A tile with four routing tracks is full of in-tile wires. Without considering the via capacity, a misled global router may route more stacked vias through this congested tile. (c) The residual via capacity decreases as the cross-tile wires increase, which is also ignored in the congestion metric for the ISPD global routing contests.

In order to reduce the gap between global routing and detailed routing, we shall consider via usages and in-tile nets for our global routing congestion model. Besides, we observe that the popular dynamic-programming (DP) based monotonic routing that has been

*This work was supported in part by Etron, SpringSoft, Synopsys Inc., TSMC, and National Science Council of Taiwan under Grant No's. NSC 96-2752-E-002-008-PAE, NSC 96-2628-E-002-248-MY3, NSC 96-2628-E-002-249-MY3, and NSC 96-2221-E-002-245.

successfully applied to the 2D cases [7], [15] cannot be directly extended to the 3D ones. The reason is that the 2D routing graphs are directed acyclic when the routing paths are constrained to be monotonic, and thus finding a shortest path on a directed acyclic graph can be done in linear time. However, the 3D routing graphs are not directed acyclic when the routing paths are constrained to be *aerial-monotonic*. (A routing path is said to be aerial-monotonic if it does not contain any detour from the aerial (top) view.) To remedy these deficiencies, we propose an efficient algorithm to find the optimal aerial-monotonic routing paths, and extend the algorithm to handle the detoured routing in only *linear time*. The major contributions of this paper are summarized as follows:

- We develop a more effective congestion metric that considers in-tile wires and residual via capacity for global routing. This metric can prevent our router from producing over congested global tiles that would complicate subsequent detailed routing.
- Two types of routing paths, the *aerial-monotonic* and *escaping-point* routing paths, are introduced. The aerial-monotonic routing extends the efficient 2D monotonic search to a 3D routing graph, whereas the escaping-point routing allows reasonable detours to reduce the congestion and overflow. The escaping-point routing can explore more paths than the A*-search routing to reduce the overflow and has lower time complexity than the maze routing while preserving the solution optimality.
- With our congestion metric, we develop a new global router that features two novel routing algorithms for congestion optimization, namely *least-flexibility-first routing* and *multi-source multi-sink escaping-point routing*. In a routing box of V nodes, in particular, the $O(V)$-time escaping-point routing algorithm guarantees to find the optimal solution and achieves the theoretical lower-bound time complexity. Compared with the $O(V \lg V)$-time complexity of the multi-source multi-sink maze routing [15], our router can effectively find better routing paths in faster runtime.
- Compared with the state-of-the-art global routers, the experimental results show that our global router can achieve better routing solutions with more reasonable via distribution that can benefit and correctly guide subsequent detailed routing. Further, ours is much more efficient than these global routers.

The rest of this paper is organized as follows. Section II describes the routing model and the problem formulation. Section III presents the methodologies and the algorithm flow of our global router. Experimental results are reported in Section IV, and conclusions are given in Section V.

II. Problem Formulation

For global routing, the routing region is partitioned into *tiles* (or called *global cells*) and a 2D or 3D routing graph composed of nodes (called *global tile nodes*) and edges (called *global edges*) models the routing region, where the global tile node represents a tile, and the global edge models the relationship between adjacent tiles. Each global edge is associated with a *capacity* to model the limited routing resource such as the number of available detailed routing tracks on the tile boundary or the maximum allowable via count between adjacent layers. The main objective of global routing is to minimize the total overflow, which is calculated by the total amount of routing demand that exceeds the capacity for each edge, and/or the maximum edge overflow.

To address the via-capacity issue, we associate a *dynamic via capacity* with each global node. Note that a via capacity is not associated with an edge connecting different layer tiles because resources on metal layers are much more critical than that on via layers. We estimate the dynamic via capacity by the area of the corresponding tile because wires and vias are geometrical rectangles in the layout. For stacked via consideration, we not only integrate layer assignment into global routing, but also constrain the in-tile stacked via count. Consequently, we define the *via capacity* of a tile t as

$$v_t = \left\lfloor \frac{\max\{0,\ (a_t - a_o - a_i) - a_w\}}{(v_w + v_s)^2} \right\rfloor, \tag{1}$$

where a_t, a_o, a_i, v_w, and v_s denote the area of the tile t, the total area of the obstacles in t, the area of in-tile nets in t, the via width, and via spacing, respectively. Here, a_w represents the probabilistic area of wires passing through the tile boundary, and the probabilistic length of a passing wire is averagely the half of the tile width (based on uniform distribution). So a_w is defined as

$$a_w = \sum_{w \in B} (w_w + w_s) t_w / 2, \tag{2}$$

where $w \in B$ refers to a wire crossing through the boundary B of the tile t. By this model, the via resource of a global routing tile t depends on the usage of wires but does not affect the wire capacity of t.

A more practical 3D global routing problem that considers the aforementioned two issues by estimating the area of the layout and routing on a 3D routing graph is defined as follows:

- **The 3D Routing Problem:** Given a netlist, widths and spacings of vias and wires, and wire capacities, find 3D routing paths connecting pins of each net such that the total number of wire and via overflow, and the total wirelength are minimized.

Note that we minimize the total number of wire and via overflow first to make fair comparisons with the recent works. Alternatively, we may minimize the maximum wire and via overflow in a tile first, which is arguably a more suitable objective for global routing.

III. Routing Methodology

There are two kinds of commonly used routing approaches for 3D global routing: (1) direct 3D global routing, and (2) 2D global routing followed by layer assignment. Direct 3D global routing is more flexible, but it is often very time-consuming. Further, its solution space is much bigger, and thus finding a legal routing solution becomes harder. In contrast, the approach of 2D global routing followed by layer assignment is more efficient, but it tends to introduce more via overflow. Our routing approach combines the advantages of the two methods and avoids their disadvantages. We first find a 2D minimal-overflow routing solution and then a 3D minimal-overflow one, with layer assignment serving as an intermediate step to bridge the 2D and 3D routing solutions.

Fig. 2 shows the flow chart of our 3D global routing. Initially, each net is decomposed into 2-pin connections by the minimum spanning tree (MST) to maximize the flexibility of the topology, as mentioned in [16]. Then, we apply *least-flexibility-first routing* to route the nets with lower flexibility to guide the subsequent monotonic routing in a look-ahead manner. If the current 2D routing does not incur any 2D (wire) overflow, layer assignment is performed to produce an initial 3D routing solution. If the initial 3D routing solution also does not cause any 3D (wire and via) overflow, the final global-routing solution is obtained.

If the routing solution is not overflow-free, an iterative negotiation-based rip-up and rerouting scheme is performed to find a minimal-overflow solution (whose total overflow is zero or cannot be further reduced) for 2D and 3D routing. Once the 2D overflow cannot be reduced any more, layer assignment is performed. Different from the previous work, the layer assignment is not the final step when via overflow exists. For 3D global routing, 3D routing edges inherit historical costs from 2D routing edges; we minimize via overflow without increasing wire overflow. We detail the distinguished features of the least-flexibility-first routing and negotiation-based rip-up and rerouting in the following.

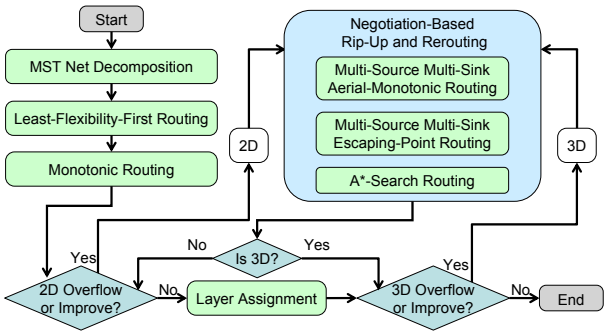

Fig. 2. Flow of our global router.

A. Least-Flexibility-First Routing

In the least-flexibility-first routing, we first route the nets located in a congested region with higher pin density (larger than a threshold) or with shorter wirelength (smaller than a threshold) since they enjoy less routing flexibility than longer ones. This also enables the subsequent routing to quickly predict the congestion hot spots for overflow reduction. Figs. 3 (a) and (b) show the congestion maps for the 2D newblue1 circuit after the least-flexibility-first routing and the final routing, respectively. From the figure, we can see the great correspondence between them in the congestion maps.

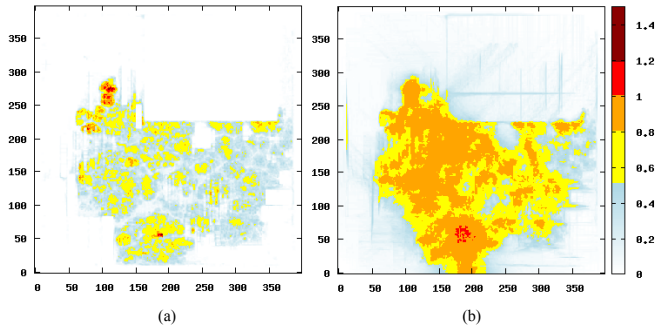

Fig. 3. The congestion map for the 2D newblue1 circuit after the (a) least-flexibility-first routing and (b) the final routing. Both red and dark-red colors represent overflow (and its degree) in global tiles.

B. Negotiation-Based Rip-Up and Rerouting

For rip-up and rerouting, our global router applies the negotiation-based cost of using an edge, similar to that in PathFinder [11]. We observe that many 2D routing techniques, such as monotonic routing cannot directly be applied to the 3D routing because the pins of nets may be located in the same layer, and thus the monotonic routing cannot be propagated upward followed by downward between layers. FGR [16] tried to solve the 3D routing (without constraining via count) by 3D maze routing alone; its results show that it is time-consuming, and its solution quality degrades in some test cases because of the high $O(V \lg V)$-time complexity and the huge solution space, where V is the number of nodes in a routing box. To remedy this deficiency, we introduce two new types of 3D routing paths, namely *the 3D aerial-monotonic routing path* and *the escaping-point routing path*, to make the solution space manageable.

The first type is the 3D aerial-monotonic routing path. A monotonic routing path is referred to as a 2D path connecting the source and the target without any detour [15], as shown in Fig. 4 (a). Monotonic routing is indeed an important technique for 2D global routing, but it is not suitable for 3D routing and may become useless for a straightforward 3D extension. For 3D routing, monotonic routing paths are quite rare; typically, pins are on the lowest layer, while wires are on higher layers, and thus finding a path needs to switch among upper and lower layers by vias. Such routing paths, detours with vias, are no longer monotonic. We thus introduce the 3D aerial-monotonic routing path which does not contain any detour except detouring with vias, as shown in Fig. 4 (b). We define the 3D aerial-monotonic routing path as a path that does not contain any detour from an aerial (top) view, but it may contain detours from a lateral (side) view. Consequently, a 3D aerial-monotonic routing path contains not only wire information, but also via information, and thus it can facilitate 3D routing.

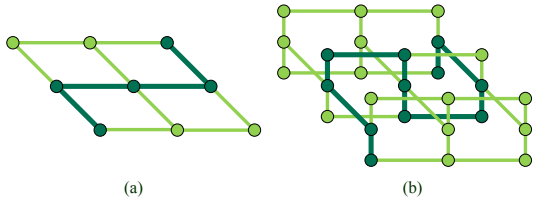

Fig. 4. Illustration of the aerial-monotonic routing. (a) The 2D monotonic routing path. (b) The 3D aerial-monotonic routing path, which looks like the 2D monotonic one from an aerial view.

The second type is the escaping-point routing path. When a net is in a very congested region, it is desired to find a path with U-turns or detours to escape from the net bounding box for congestion optimization. Figs. 5 (a) and (c) show two typical routing paths from s to t that detour from the net bounding box. Here, we define the escaping-point routing path as the path that can be split into at most two monotonic routing paths by one escaping point. For example, the escaping-point routing paths $s \rightarrow t$ of Figs. 5 (a) and (c) can be decomposed into two monotonic routing paths $s \rightarrow p$ and $p \rightarrow t$ via the escaping point p as shown in Figs. 5 (b) and (d), respectively. Note that the escaping-point routing paths also include the aerial-monotonic routing paths. With escaping-point routing, a router can explore more paths inside the searching box for congestion optimization and overflow reduction.

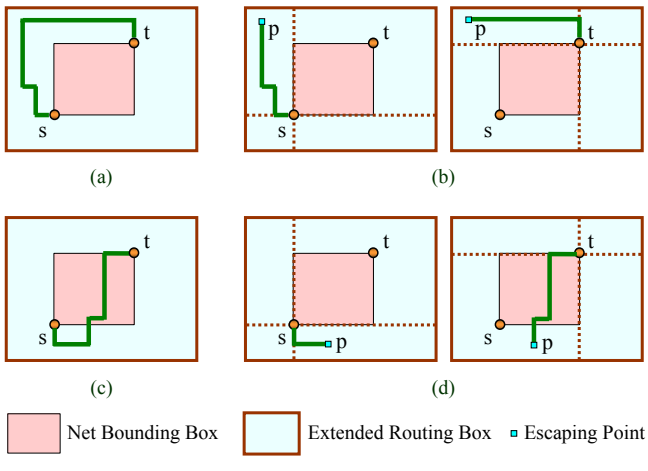

Fig. 5. Illustration of the escaping-point routing path. The escaping-point routing paths $s \rightarrow t$ shown in (a) and (c) can be decomposed into two monotonic routing paths $s \rightarrow p$ and $p \rightarrow t$ via the escaping point p as shown in (b) and (d), respectively.

We propose efficient 3D routing algorithms to find the corresponding optimal routing paths for the aerial-monotonic routing and the escaping-point routing.

1) Multi-Source Multi-Sink Aerial-Monotonic Routing (MSAMR): We first develop an aerial-monotonic routing (AMR) algorithm to find an optimal 3D aerial-monotonic routing path and then extend it to a multi-source multi-sink 3D aerial-monotonic routing (MSAMR) algorithm without sacrificing the optimality. Note that the 3D routing graph is not directed acyclic when the routing paths are constrained to be 3D aerial-monotonic. The z-direction global edge connecting two global tiles on two layers can be upward and downward, and thus cycles may be introduced in the 3D routing graph. As a result, existing linear-time algorithms that find the shortest path on a directed acyclic graph is not applicable to this case, and it is thus desirable to develop an efficient algorithm for this problem.

The AMR algorithm iteratively performs the two-phase operations of the *pile update* and *pile propagation*, where a *pile* represents a set of global tile nodes in the z-direction of a global routing graph. From an aerial view, the pile propagation behaves like the propagation of 2D monotonic routing, propagating the cost of the nodes in a pile ρ to the nodes of ρ's adjacent piles. The main difference from 2D monotonic routing is that the AMR algorithm requires each pile ρ to conduct the pile update which updates the cost of the nodes in ρ *upward* followed by *downward* directions before applying the pile propagation on ρ. With the pile update, the AMR algorithm can guarantee to find an optimal 3D aerial-monotonic routing path. Fig. 6 illustrates the AMR algorithm on a three-layer 3D routing graph. Fig. 6 (a) shows the pile propagation from the piles ρ_1 and ρ_2 to the pile ρ, and Fig. 6 (b) depicts the upward and downward pile updates of ρ before it continues to propagate to adjacent piles ρ_3 and ρ_4, as shown in Fig. 6 (c). Note that the dynamic via capacity can be addressed during the upward and downward pile updates. Note also that we can use the *coloring method* to avoid unnecessary downward updating and thus speed up the runtime.

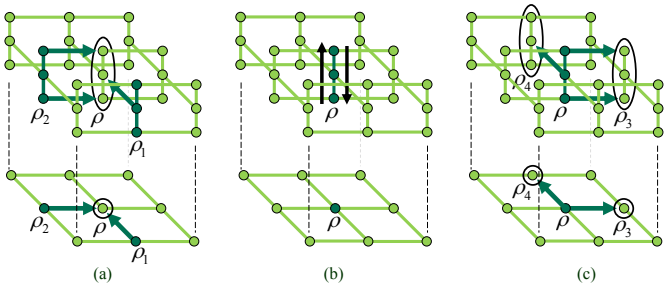

Fig. 6. The pile update and pile propagation of the AMR algorithm. (a) The pile ρ consisting of three nodes is updated by piles ρ_1 and ρ_2 via the pile propagation. (b) The pile update is performed on the pile ρ upward followed by downward to update the cost of nodes in ρ. (c) After pile update, ρ continues to update the next two adjacent piles ρ_3 and ρ_4.

Theorem 1: For a net bounding box containing V nodes, the AMR algorithm finds an optimal aerial-monotonic routing path in $O(V)$ time, achieving the lower-bound complexity.

With the increasing number of congested regions, the multi-source and multi-sink routing can effectively find alternative paths to connect two subtrees, as illustrated in Fig. 7 (a). Note that the multi-source multi-sink maze routing [15] requires $O(V \lg V)$-time and becomes very slow when V is large, because it spends significant time in maintaining a large priority queue. In contrast, extending from our AMR, the MSAMR algorithm can find an optimal 3D multi-source multi-sink aerial-monotonic routing path in only $O(V)$ time, which achieves *the theoretical lower-bound complexity* (since such a routing path may contain $O(V)$ nodes).

The MSAMR algorithm is detailed below. We find two subtrees, T_s and T_t, which are connected by the source s and the target t, as shown in Fig. 7 (a). We classify the topological relationship of the two subtrees into four categories, the xy non-overlap, the x overlap, the y overlap, and the xy overlap, as shown in Figs. 7 (b)–(e). According to the topological relationship of the two subtrees, we can find minimum-area routing box that covers all possible aerial-monotonic routing paths with a specific direction. For instance, in Fig. 7 (b), the relationship is the xy non-overlap, and we find a routing box covering possible paths from s_1 to t_1. For other categories, like the x overlap, we need to find one more routing box to cover possible paths from s_2 to t_2. Thus, we perform AMR on each routing box to find an optimal aerial-monotonic routing path with a specific direction. Note that all nodes in the subtree T_s should be treated as source nodes. AMR selects a minimum-cost node from the nodes on the target tree T_t and in the routing box, and then back traces to any node in the source subtree T_s. Finally, we report the minimum-cost routing path, an optimal aerial-monotonic routing path.

Theorem 2: For a bounding box of a multi-pin net, with V nodes, the MSAMR algorithm finds an optimal multi-source multi-sink aerial-monotonic routing path in $O(V)$ time, achieving the lower-bound complexity.

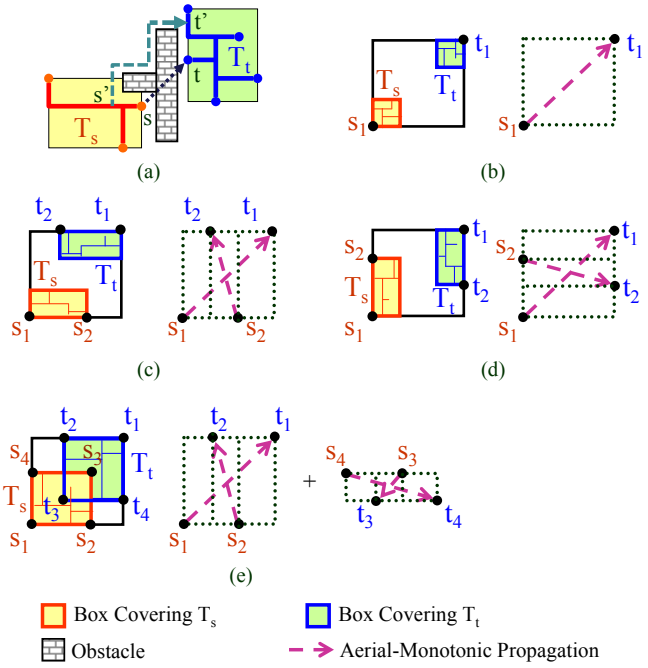

Fig. 7. Illustration of the MSAMR algorithm. (a) A better multi-source multi-sink aerial-monotonic routing path $s' \rightarrow t'$ replaces the original path $s \rightarrow t$. (b)–(e) The MSAMR algorithm uses aerial-monotonic propagation based on the topological relationship of the source and the target subtrees.

2) Multi-Source Multi-Sink Escaping-Point Routing: We first propose a two-phase aerial-monotonic routing (TAMR) algorithm to find an optimal escaping-point routing path in $O(V)$ time, and then extend it to the two-phase multi-source multi-sink aerial-monotonic routing (TMSAMR) algorithm for multi-source multi-sink escaping-point routing. Note that our algorithm achieves the theoretical lower bound of $O(V)$ time since such a routing path may contain $O(V)$

nodes. As shown in Fig. 8, the TAMR algorithm can get an optimal escaping-point routing path because an optimal path is composed of two aerial-monotonic routing paths from both the source and the target. Note that the TAMR algorithm is different from the traditional bidirectional search. The traditional bidirectional search stops when the two search paths meet in the middle, and then it finds a path just like the path found by the unidirectional search. In contrast, TAMR can find a routing path allowing one aerial detour while the unidirectional search cannot.

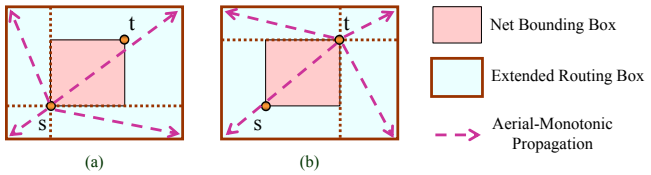

Fig. 8. Two-phase routing for escaping-point routing. (a) The aerial-monotonic propagation from s to the four corners of the routing box. (b) The aerial-monotonic propagation from t to the four corners of the routing box.

The detail of the TAMR algorithm, as shown in Fig. 9, is described below. In line 1, we decide the size of box according to the user-specified number ext. In lines 2–8, four 3D aerial-monotonic propagations are performed from the source to the four corner points of the extended routing box, and then bookkeep the information about the distances and the propagation directions. The same process is performed for the target in lines 9–15. In line 16, we combine the distances to the source and to the target of all nodes in the box, and then find a node with the minimum distance. In lines 17–19, we back trace to the source and the target from the node with the minimum distance, and then combine the two paths, as shown in Fig. 9. Lines 2–8, lines 9–15, and line 16 run in $O(V)$ time, and thus the algorithm runs in $O(V)$ time. Note that this algorithm achieves the lower-bound complexity since such a routing path may contain $O(V)$ nodes.

Theorem 3: The TAMR algorithm finds an optimal escaping-point routing path in $O(V)$ time, achieving the lower-bound complexity.

Note that like the MSAMR algorithm, we can extend the TAMR algorithm to the TMSAMR one by performing the two-phase MSAMR algorithm from both the source and the target subtrees according to their topological relation.

Theorem 4: The TMSAMR algorithm finds an optimal multi-source multi-sink escaping-point routing path in $O(V)$ time, achieving the lower-bound complexity.

IV. Experimental Results

Our global router was implemented in the C++ programming language on a 2.0 GHz Intel Xeon Linux workstation with 16 GB memory. We compared our global router with the latest results of BoxRouter 2.0 [4], NTHU-Route [7], Archer [13], FGR [16], FGR 1.1's best-seen [6], and MaizeRouter [12], based on the ISPD'07 benchmarks.

A. Global Routing Results without Considering Via Capacity

Table I shows the comparison of 3D global routing results on the ISPD'07 global routing benchmarks. For this experiment, we turned off the via-capacity and in-tile-net considerations for our global router to make fair comparisons with other routers. (Note that as mentioned earlier, *we do not consider such a simplification to be realistic for real-world applications.*) In this table, "OF" and "WL" give the number of overflow and the total wirelength. Our global router completed six out of eight circuits for the 3D benchmarks.

Algorithm: TAMR(G, s, t, ext, L)
Input : G - a 3D routing graph;
s, t - 3D coordinates of source and target points;
ext - the extending value of the bounding box;
L - number of layers.
Output : P_{opt} - an optimal escaping-point routing routing path.
begin
1 $B \leftarrow$ the extended searching box by ext;
2 **for** $g \in B$
3 $d(g) \leftarrow \infty$; $\pi(g) \leftarrow null$;
4 $g_s \leftarrow G(s)$; $d(g_s) \leftarrow 0$; $\pi(g_s) \leftarrow null$;
5 **for** each direction $\delta \in$ {top-right, top-left, bottom-right, bottom-left}
6 $p_a \leftarrow s$ and decide p_b by B in δ direction;
7 Aerial-Monotonic Routing (G, p_a, p_b, L);
8 $G_s \leftarrow G$;
9 **for** $g \in B$
10 $d(g) \leftarrow \infty$; $\pi(g) \leftarrow null$;
11 $g_t \leftarrow G(t)$; $d(g_t) \leftarrow 0$; $\pi(g_t) \leftarrow null$;
12 **for** each direction $\delta \in$ {top-right, top-left, bottom-right, bottom-left}
13 $p_a \leftarrow t$ and decide p_b in δ direction
14 Aerial-Monotonic Routing (G, p_a, p_b, L);
15 $G_t \leftarrow G$;
16 find $v_{st} \in B$ such that $d(G_s(v_{st})) + d(G_t(v_{st}))$ is minimum;
17 $P_s \leftarrow$ back track from v_{st} on G_s;
18 $P_t \leftarrow$ back track from v_{st} on G_t;
19 $P_{opt} \leftarrow$ combination of P_s and P_t;
end

Fig. 9. The two-phase aerial-monotonic routing (TAMR) algorithm.

Note that it is known that newblue3 is unroutable. For the newblue3 circuit, our router obtained the best routing solution among all these routers, with only 31246 overflow; our router achieved 4.0x–36.0x runtime speedups (with similar total wirelength), which is the fastest runtime reported in the literature. The experimental results justify the high routability and efficiency of our router. According to the metric of the ISPD'07 global routing contest [9], the total wirelength is compared only when the number of total overflow and the maximum overflow are the same. In the ISPD'08 contest [10], the runtime is also considered when the number of total overflow and the maximum overflow are the same. However, the wirelength metrics are different between the ISPD'07 and ISPD'08 contests (due to the via-length modeling). Since the results for the ISPD'08 global routing contest are not available as time of this submission, we only test on the ISPD'07 benchmarks and adopt the wirelength metric of the ISPD'07 contest to make fair comparisons with other routers.

Note that the platform of Archer (Intel Xeon 3.6GHz) is faster than ours (Intel Xeon 2.0GHz), and the runtime information of BoxRouter 2.0 is not available. (However, it is known that BoxRouter 2.0 is slower than these routers.) FGR 1.1 has two modes, the default mode, which can be run on our platform, and the one with its best-seen results [6], which was run on an AMD Opteron machine with a 2.4 GHz CPU. Compared with FGR 1.1, our router achieves better routability and results in smaller overflow. Although the total wirelength of our router is about 3% worse than FGR 1.1's default mode and FGR 1.1's best-seen results, our runtime is at least 15.7x faster than that of FGR 1.1. Note that the best-seen results of FGR 1.1 were obtained with the running time bound of 2880 min for each circuit. (Note also that the ISPD'08 scoring metric penalizes 4% wirelength for each 2x slower runtime, up to 10% penalty for wirelength.)

B. Via-Aware Global Routing Results

Table II shows the comparisons of via-aware 3D global routing results on the ISPD'07 benchmarks. (Note that we excluded the newblue3 circuit in this experiment because it is known that newblue3 is trivially not routable due to its high-pin-density problem.) In this table, "VOF" reports the number of via overflow, which is the excess of the stacked vias. Unlike the simplified metric of the ISPD'07 routing contest [9], we constrained the in-tile via count by via

TABLE I

COMPARISON OF THE 3D GLOBAL ROUTING RESULTS ON THE ISPD'07 BENCHMARKS.

Circuit	FGR 1.1 [16]			FGR 1.1 best [6]§		BoxRouter 2.0 [4]§		Archer [13]#			MaizeRouter [12]			NTHU-Route [7]			Ours		
	OF	WL (e^5)	CPU (min)	OF	WL (e^5)	OF	WL (e^5)	OF	WL (e^5)	CPU (min)	OF	WL (e^5)	CPU (min)	OF	WL (e^5)	CPU (min)	OF	WL (e^5)	CPU (min)
adaptec1	0	87.79	431.9	0	88.02	0	92.04	0	113.80	87.0	0	99.70	143.0	0	90.56	90.3	0	90.53	16.1
adaptec2	0	89.63	41.5	0	89.96	0	94.28	0	112.56	23.0	0	99.53	90.6	0	92.17	13.8	0	91.60	4.7
adaptec3	0	198.77	237.7	0	200.14	0	207.41	0	244.08	51.0	0	210.19	157.8	0	205.04	59.1	0	203.12	21.6
adaptec4	0	182.87	33.0	0	178.90	0	186.42	0	221.57	12.0	0	190.81	26.2	0	188.43	8.1	0	188.28	2.5
adaptec5	0	258.64	931.8	0	260.53	0	270.41	0	334.09	248.0	0	303.34	4865.7	0	265.03	250.3	0	264.91	29.3
newblue1	310	94.01	1441.3	234	90.68	394	92.94	682	116.08	50.0	1372	100.38	892.3	352	90.91	41.7	62	91.78	1046.8
newblue2	0	132.08	10.2	0	129.30	0	134.64	0	166.50	7.0	0	139.22	14.3	0	136.01	3.3	0	132.88	3.4
newblue3	44854	172.79	1547.1	38386	163.41	38958	172.44	33394	198.77	163.0	32234	181.68	663.4	31800	168.40	318.7	31246	239.29	374.0
Comp.*	-	1.00	15.79	-	1.00	-	1.04	-	1.26	4.67	-	1.10	36.06	-	1.03	4.01	-	1.03	1.00

*: Wirelength and runtime comparisons are based on the overflow-free cases.

§: The runtime of FGR 1.1's best-seen result is 48 hours [6].

§ : No runtime is reported for BoxRouter 2.0 [4], but it is known that BoxRouter 2.0 is slower than FGR 1.1.

#: Archer's runtime is reported in [13] based on a faster platform (Intel Xeon 3.6 GHz).

TABLE II

COMPARISON OF THE 3D GLOBAL ROUTING RESULTS ON THE ISPD'07 BENCHMARKS WITH DYNAMIC VIA CAPACITY CONSTRAINTS.

Circuit	FGR 1.1 best [6]			MaizeRouter [12]			BoxRouter 2.0 [4]			NTHU-Route [7]			Ours		
	OF	VOF	WL (e^5)	OF	VOF	WL (e^5)	OF	VOF	WL (e^5)	OF	VOF	WL (e^5)	OF	VOF	WL (e^5)
adaptec1	2048	2048	88.02	1933	1933	99.70	1963	1963	92.04	1700	1700	90.56	1081	1081	96.61
adaptec2	193282	193282	89.96	158409	158409	99.53	218379	218379	94.28	179171	179171	92.17	161463	161463	96.47
adaptec3	32719	32719	200.14	15850	15850	210.19	37711	37711	207.41	28614	28614	205.04	10390	10390	218.76
adaptec4	7072	7072	178.90	2469	2469	190.81	7091	7091	186.42	6611	6611	188.43	1616	1616	206.14
adaptec5	487200	487200	260.53	438331	438331	303.34	534040	534040	270.41	435053	435053	265.03	427068	427068	270.81
newblue1	66007	65769	90.68	49169	47797	100.38	64597	64203	92.94	52978	52626	90.91	51472	51410	95.71
newblue2	8284	8284	129.30	5252	5252	139.22	11423	11423	134.64	7567	7567	136.01	4343	4343	136.15
Comp.	2.14	2.13	1.00	1.29	1.28	1.10	2.33	2.33	1.04	1.90	1.90	1.03	1.00	1.00	1.08

capacity and set the number of via overflow as one of the major metrics. We believe that this is a more reasonable congestion metric for global routing. The via capacity, as shown in Eq. (1), should be decided by the remaining routing resource of the global tiles. In other words, the via capacity is dynamic and is affected by the number of wires passing through the same global tile.

Table II shows the routing results of considering the dynamic via capacity, decided by Eq. (1). These results show that minimizing the 3D total wirelength without considering the via capacity increases the stacked via overflow since almost all the overflows are contributed by the via overflow. For the experiments on dynamic via capacity, our router achieved the best overall results for the total overflow (including the via and wire overflow) and the total via overflow. The experimental results show the high quality and superiority of our router.

V. CONCLUSION

In this paper, we have derived a congestion metric, dynamic via capacity, for global routing; the metric practically considers the via capacity as well as the in-tile nets. With this metric, we have developed a new global router that features two novel effective routing algorithms, least-flexibility-first routing and multi-source multi-sink escaping-point routing, for congestion optimization. In particular, the linear-time escaping-point routing algorithm is optimal in the sense that it achieves the theoretical lower-bound complexity. Experimental results have shown that our global router can achieve better routing solutions with more reasonable via distribution that can benefit and correctly guide subsequent detailed routing.

REFERENCES

[1] P. Bishop, "Viewpoint: Routing Is Key to Implementing DFM within The Design Flow," *EE Times*, Aug. 1, 2007.

[2] T. Chen and A. Cengiz, "Measuring Routing Congestion for Multi-layer Global Routing," *In Proc. of GLSVLSI*, pages 59–62, 2000.

[3] M. Cho, D. Z. Pan, H. Xiang, and R. Puri, "Wire Density Driven Global Routing for CMP Variation and Timing," *In Proc. of ICCAD*, pages 487–492, 2006.

[4] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0: Architecture and Implementation of a Hybrid and Robust Global Router," *In Proc. of ICCAD*, pages 503–508, 2007.

[5] J. Cong, M. Xie and Y. Zhang, "An enhanced multilevel routing system," *In Proc. of ICCAD*, pages 51–58, 2002.

[6] FGR: A Fairly Good Router. http://vlsicad.eecs.umich.edu/BK/FGR/

[7] J.-R. Gao, P.-C. Wu, and T.-C. Wang, "A New Global Router for Modern Designs," *In Proc. of ASPDAC*, pages 232–237, 2008.

[8] M. Heins, "In the Eye of the DFM/DFY Storm," *EE Times*, May 25, 2007.

[9] ISPD 2007 Global Routing Contest. http://www.sigda.org/ispd2007/rcontest/

[10] ISPD 2008 Global Routing Contest. http://www.ispd.cc/contests/ispd08rc.html

[11] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," *In Proc. of ISFPGA*, pages 111–118, 1995.

[12] M. D. Moffitt, "MaizeRouter: Engineering an Effective Global Router," *In Proc. of ASPDAC*, pages 226–231, 2008.

[13] M. Ozdal, "ARCHER: A History-Driven Global Routing Algorithm," *In Proc. of ICCAD*, pages 488–495, 2007.

[14] M. Pan and C. Chu, "IPR: An Integrated Placement and Routing Algorithm," *In Proc. of DAC*, pages 59–62, 2007.

[15] M. Pan and C. Chu, "FastRoute 2.0: A High-quality and Efficient Global Router," *In Proc. of ASPDAC*, pages 250–255, 2007.

[16] J. A. Roy and I. L. Markov, "High-Performance Routing at the Nanometer Scale," *In Proc. of ICCAD*, pages 496–502, 2007.

[17] H. Zhou and D. F. Wong, "Global Routing with Crosstalk Constraints," *In Proc. of DAC*, pages 374–377, 1998.