A Multicommodity Flow-Based Detailed Router With Efficient Acceleration Techniques

Xiaotao Jia, Yici Cai, *Senior Member, IEEE*, Qiang Zhou, *Senior Member, IEEE*, and Bei Yu, *Member, IEEE*

***Abstract*—Detailed routing is an important stage in very large scale integrated physical design. Due to the extreme scaling of transistor feature size and the complicated design rules, ensuring routing completion without design rule checking (DRC) violations becomes more and more difficult. Studies have shown that the low routing quality partly results from nonoptimal net-ordering nature of traditional sequential methods. The concurrent routing strategy is always based on an NP-hard model, thus is at a disadvantage in runtime. In this paper, we present a novel concurrent detailed routing algorithm that routes all nets simultaneously. Based on the multicommodity flow model, detailed routing problem with complex design rule constraints is formulated as an integer linear programming. Some model simplification heuristics and efficient model solving algorithms are proposed to improve the runtime. Experimental results show that, the proposed algorithms can reduce the DRC violations by 80%, meanwhile can reduce wirelength and via count by 5% and 8% compared with an industry tool. In addition, the proposed algorithm is general that it can be adopted as an incremental detailed router to refine a routing solution, so the number of DRC violations that industry tool cannot fix are further reduced by 27%.**

***Index Terms*—Design rules, detailed routing, integer linear programing, multicommodity flow (MCF).**

I. Introduction

ROUTING is a very important and the most time-consuming stage in modern very large scale integrated (VLSI) circuits physical design. The top-level design may contain millions of nets in a typical hierarchical design. It generally takes days for EDA tools to finish routing, close timing, and fix design rule checking (DRC) violations on a powerful computing server with multithreading accelerations. However, due to the NP-hard nature of the routing problem, industry tools sometimes hardly guarantee total completion of the DRC-clean connections while satisfying the timing constraints, thus designers have to spend much more time on engineering change order changes to meet both timing and DRC closures.

To effectively reduce the complexity of routing problem, industry tools use divide-and-conquer mechanism, where a global routing (e.g., [1]), combined with or followed by a track assignment (e.g., [2]), is used to obtain a coarse solution (tile-to-tile paths) on a global routing graph for all nets optimizing some particular objective functions. After global routing and track assignment, routing segments are assigned into different routing tracks, then a detailed routing is to further connect net terminals satisfying all physical constraints and timing requirements.

Submicron technologies generally use multilayer routing, where metal wires are manufactured in six to twelve routing layers [3]. Each routing layer has a preferred routing direction and adjacent layers are connected by vias. A layout is partitioned into many subregions after global routing and track assignment. Detailed routing works on the 3-D routing space inside each subregion to connect all the nets. The primary task for a detailed router is to make sure there are no layout versus schematic (LVS) errors, while satisfying all the complex design rules. The secondary task for the detailed router is to optimize circuit performance, without detouring timing critical nets while preserving the nondefault spacing requirements for delay.

In modern VLSI physical design, routing faces severe challenges. As the prediction of Moore's law, the number of transistors per die grows exponentially in the last years. The scale of detailed routing becomes larger and larger. As the decreasing of feature size, more and more design rules are introduced to guarantee performance, manufacture, and yield [4]. Furthermore, the layout structure is more complicated because of the hierarchy level and complex design modules. The explosion of problem scale and design rule number make detailed routing more difficult. Assigning routing resource for every net reasonably becomes more and more important.

A. Related Works

Routing algorithms can be classified into sequential and concurrent approaches. On one hand, sequential approach routes all the nets one-by-one in a prefixed order, thus is fast but may suffer from net-ordering problem which has great impact on routing quality. On the other hand, with a global view, concurrent approach routes all nets simultaneously, but has a disadvantage in runtime which limits the problem scale.

Most of previous works are designed on a sequential manner in both grid-based and gridless models. For grid-based detailed routing model, detailed router finds legal routing path on the given routing grids. Lee's algorithm [5] is the most widely used grid-based algorithm to search for a shortest path for two terminals. Lee's algorithm consists of two phases of path search followed by trace back. In the first phase, breath-first-search strategy is applied to find a shortest path between source vertex and target vertex in manner of wave propagation. After a shortest path has been found, the second phase of trace back works to find the optimal routing solution based on waves. Line-search algorithm [6] is developed to speedup Lee's algorithm. Differing from Lee's algorithm, line-search algorithm performs a depth-first-search approach to find a routing path using line segments. Another efficient detailed routing algorithm, proposed by Hart [7], is called A*-search. Compared with Lee's algorithm, A*-search is smarter because

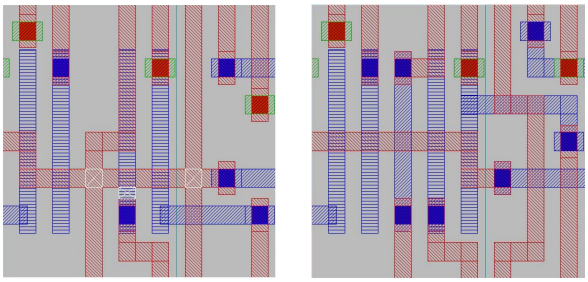
Manuscript received October 15, 2016; revised January 23, 2017; accepted March 3, 2017. Date of publication April 12, 2017; date of current version December 20, 2017. This paper was recommended by Associate Editor L. Behjat. *(Corresponding author: Yici Cai.)*

X. Jia, Y. Cai, and Q. Zhou are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: caiyc@mail.tsinghua.edu.cn).

B. Yu is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2017.2693270



(a) (b)

Fig. 1. Influence of net ordering on routability. (a) Industry tool result. (b) Our result. White rectangle is DRC marker; blue geometry is wire/pin on metal1; red and green geometry is wire on metal2 and metal3, respectively.

it selects vertex with minimum cost to propagate. The cost of a vertex is calculated by a function $f(v) = g(v) + h(v)$, where $g(v)$ is the path cost from source vertex to vertex v, while $h(v)$ is predicted cost from vertex v to target vertex. Some gridless detailed routers are proposed (e.g., [8] and [9]). In recent years, Zhang and Chu [10] proposed RegularRoute for detailed routing. Later, Zhang and Chu [11] proposed GDRouter in which FastRoute [12] and RegularRoute [10] are interleaved. Gester *et al.* [13] presented a detailed router using two efficient data structures of shape grid and fast grid. Ahrens *et al.* [14] proposed a new general multilabel shortest path algorithm, which is used to compute design rule clean paths or nontrivially colored paths in situations where a standard shortest path algorithm does not find good solutions.

To remedy the deficiencies, sequential routing often applies a heuristic net ordering and conducts a rip-up and reroute process to further refine the solution. Here, are some common net-ordering schemes: 1) the ascending order of pin number; 2) the ascending order of bounding-box area; and 3) the descending order of DRC number. In the rip-up and reroute process, the net order may change due to current DRC violation count and historical penalties. The routing engine works on a single net and other nets are fixed as routing obstacles when searching for the new path for current net. Even though different ordering strategies and several iterations of rip-up and reroute are employed, it has been found both in industry and academia that the worse routing solution partly arises from nonoptimal net-ordering nature of traditional sequential methods.

However, the aforementioned heuristic net-ordering strategies may fail to achieve valid routing solution. The routability issue becomes more severe in practical routing problems, due to increasingly problem scale and complex design rules. As shown in Fig. 1(a), the routing result is generated by a commercial EDA tool, Encounter v10.10 [15]. After several optimization iterations, there are still three DRC violations that can not be removed. In Fig. 1(b), a concurrent routing result without any DRC violations is generated by considering routing nets in the region simultaneously. We can observe that sequential strategy is difficult to effectively search for a DRC-clean solution, while concurrent strategy can effectively resolve the net-ordering problem. Therefore, an sequential router may have to expand the routing region thus involving more nets to the rip-up and reroute iterations or negotiation-based strategy.

Thanks to the ability of resolving net-ordering issue, many VLSI design problems have been modeled and solved in a concurrent manner, such as global routing [16], escape routing [17], and layer assignment [18], [19]. Multicommodity flow (MCF) problem is a network flow problem with multiple flow demands between different source and sink nodes. Even though there are some similarities between detailed routing and global routing problems, the concurrent global routing models [20]–[22] could not be directly applied to detailed routing problems. First, the routing solutions are limited because the solution space for each net in global routing is combined by some steiner trees, which is not suitable for detailed routing. Second, no complexed design rules such as spacing are considered in global routing models since they are mostly congestion driven. Thus there will be many DRC violations if global routing models are applied to solve detailed routing problem. Jia *et al.* [23] propose the first concurrent detailed router based on the MCF method considering complex design rules. But it is very time-consuming and many DRC violations still remain unresolved. Although Han *et al.* [24] propose a concurrent detailed router considering double patterning issue, the proposed router is not applicable to general detailed routing problem (will be detailed explained in Section IV-B).

B. Our Contributions

In this paper, a novel concurrent detailed routing algorithm based on MCF method is proposed to route all the nets simultaneously. Some effective heuristic strategies are also carried out to overcome the shortcoming in runtime. The main contributions of this paper are listed as follows.

1) A concurrent detailed router, MCFRoute, is proposed based on MCF model. MCFRoute can generate paths for all nets in each routing region simultaneously while satisfying given constraints. As a result, it overcomes the net-ordering problem and can get an optimal grid-based solution in each routing region.[1]
2) Besides fixing LVS errors, our detailed router supports some complex spacing rules. Some other design rules, like minarea rules are also considered in our routing flow in post-routing stage. Our router is proven to be effective in supporting 28 nm above technology nodes.
3) Several strategies are proposed to optimize the MCF model in order to reduce the runtime. An equivalent transformation is carried out to convert the detailed routing model from integer nonlinear programming formulation to integer linear programming (ILP) formulation. A heuristic strategy is proposed for speedup by reducing the scale of ILP problem by removing redundant constraints.
4) We also propose a solving algorithm to reduce the runtime. The solving algorithm starts with a prerouting stage in which a good start point is generated for ILP problem based on track assignment results. Then the ILP problem is relaxed into an LP problem and solved by an LP solver.
5) A multithreaded framework is implemented for the MCF routing engine and it can get close to linear speedup ratio on the multicore computing server. And the MCF routing engine can be embedded easily to a maze-based detailed router to do a post-routing DRC cleanup.
6) Experimental results show that compared with the state-of-the-art industry tool, MCFRoute could reduce the DRC violation count and improve the routing quality. Moreover, it can also be used as an incremental router to fix some hard DRC violations which the industry tool cannot fix.

The remainder of this paper is organized as follows. Section II provides preliminaries and problem formulation.

[1]The MCF model requires all wires and vias are placed on routing grids. For gridless cell pins, it is connected through on-grid pin-access-virtual-vias.

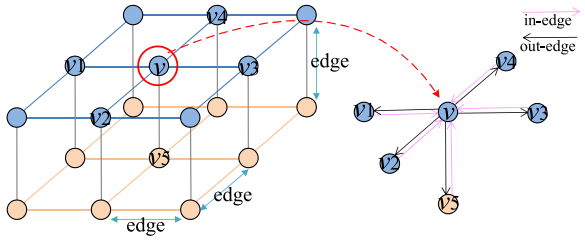


Fig. 2. Detailed routing graph.

Section III describes the overview flow of the proposed algorithms. Section IV gives the basic MCF model formulation. Section V discusses how some complex design rules are added to the basic model. Section VI describes several strategies to simplify MCF model. Section VII discusses how we solve MCF model in details. Section VIII lists experimental results, followed by conclusion in Section IX.

II. Problem Formulation

A. Preliminaries

Due to the large scale, the layout of modern chip is divided into many small routing regions after track assignment. A net in netlist is generally divided into many subnets located at different small regions (hereafter, we will refer to net in netlist as *logic net* and subnet in small routing region as *net*). The intersection point of segment and the boundary of a small routing region is usually named as *crosspoint*. Both the terminals of logic net which we refer to as *pins* below, and *crosspoints* in a small routing region are the *components* of the net.

Based on MCF theory, detailed routing problem is modeled as a path finding problem on a 3-D *routing graph* which is named as $G = (V, E)$. Here, $V = \{v_1, v_2, \ldots, v_n\}$ denotes a set of n vertices of graph G, and $E = \{e_1, e_2, \ldots, e_m\}$ denotes a set of m edges of graph G. Vertices of graph G on each layer are the intersection points of routing grids on current layer with grids projected from neighbor routing layers. Then routing grids are divided as edges of graph G by vertices. In MCF model, graph G is defined as a directed graph, which means there are two edges between adjacent vertices v_{j_1} and v_{j_2}; one is from v_{j_1} to v_{j_2}, while the other one is from v_{j_2} to v_{j_1}. We regard these two edges as *brother edges*. Fig. 2 shows a simple routing graph with two routing layers, 66 directed edges and 18 vertices.

On detailed routing graph, each vertex v_j is a terminal point of a set of edges E_{v_j} which contains no more than 12 elements as shown in Fig. 2. Each set E_{v_j} could be divided into two subsets, labeled as $E_{v_j,\text{out}}$ and $E_{v_j,\text{in}}$ with the same scale. All the edges in $E_{v_j,\text{out}}$ start from v_j and are named as out-edges of v_j, while all the edges in $E_{v_j,\text{in}}$ end at v_j and are named as in-edges of v_j. In Fig. 2, vertex v has five in-edges that are marked with a pink color and five out-edges with black color. From the assumption given above we can reach the following conclusion:

$$|V| = \sum_{l=1}^{L} R_l C_l$$

$$|E| = 2 \times \left(\sum_{l=1}^{L} (2R_l C_l - R_l - C_l) + \sum_{l=1}^{L-1} R_l C_l \right).$$

Here, L is the total number of routing layer; R_l and C_l is the row count and column count of layer l, respectively.

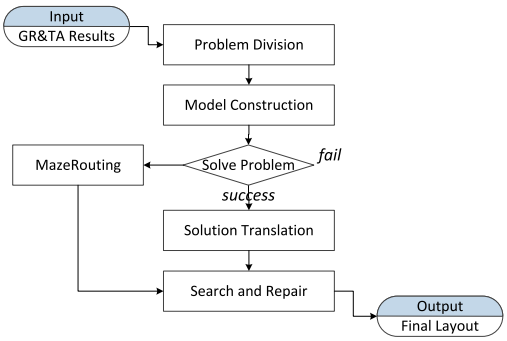


Fig. 3. Overall flow of MCFRoute.

B. Problem Definition

Generally, detailed routing is executed after global routing and track assignment in a subregion. With the guidance of global routing and track assignment results, detailed routing could access the exactly crosspoint position with layer information. In our implementation, each global routing cell (GRC) is a routing region. We define route constraints as follows.

Definition 1 (Route Constraints): A clean detailed routing result should obey the following three constraints: 1) all components of each net are connected by one path; 2) there are no intersection segments or points between any two paths or between path and routing blockage; and 3) there are no design rule violations.

Based on above definitions we achieve the following definition of detailed routing.

Problem 1 (Detailed Routing): Given the following information: 1) cross point information; 2) pin and routing blockage information; and 3) routing region, i.e., GRC, the detailed routing problem searches for connections for all nets $N = \{n_1, n_2, \ldots, n_K\}$ to minimize total routing cost, meanwhile route constraints are satisfied.

III. Overall Flow

The overall flow of the proposed detailed router is illustrated in Fig. 3. The proposed detailed router uses the results of global router and track assignment as inputs. Due to the large scale of VLSI circuits, the first step is dividing the whole routing region into many subregions. In our experiments, the partition is based on GRC. For each subregion, we first formulate the detailed routing problem into an ILP problem by three steps: 1) basic model formulation (Section IV); 2) design rule formulation (Section V); and 3) model simplification (Section VI). Then solve the ILP problem by a solver with efficient solving algorithm (Section VII). Then 0-1 solution will be translated into detailed routing solution. However, the ILP problem solving may fail because of the following two reasons.

1) There are no DRC-clean solution for the detailed routing problem.
2) The solver is terminated because it does not get a solution in the given time.

Under the second circumstance, maze routing will be executed in this region trying to find a DRC-clean solution even though it may still fail. A search and repair stage based on maze routing is implemented to handle the leaving unsuccessful detailed routing problem by adjusting the routing region. After concurrent routing, there may be some difficult regions that concurrent router can not find routing solution in given

TABLE I
NOTIONS OF MCF MODEL

notion	meaning
$E/V/N$	edge/vertex/net set (indexed by $i/j/k$)
E_{v_j}	adjacent edges set of vertex v_j
$E_{v_j,out}$	edges set whose start point is vertex v_j
$E_{v_j,in}$	edges set whose end point is vertex v_j
$E_{v_j}^{\eta}$	via edges set of vertex v_j
$u(e_i)$	binary variable, the capacity of edge e_i
$c(e_i)$	positive variable, the cost of edge e_i
$d(k, v_j)$	the flow command of vertex v_j with value of -1, 0, or 1 that net n_k demands
$f(k, e_i)$	binary variable, flow of net n_k by edge e_i

time. These regions will be routed in search and repair stage by maze routing using rip-up and reroute strategy. In the first routing stage, all nets must be connected together even though there are some violations. Then the search and repair stage will try to remove the leaving DRC violation by changing the center coordinate of routing region and expanding the region area. Three net-ordering strategies are applied in search and repair stage: 1) ascending order of net component count; 2) ascending order of net bounding box area; and 3) descending order of DRC count. The routing region grows larger as the iteration number increases. Maze routing algorithm with rip-up and reroute strategy is also used to finish the routing task in new routing region. The output is a final detailed routing result.

IV. MULTICOMMODITY FLOW MODEL

In this section, we will introduce our MCFRoute which is designed to solve detailed routing problem. Inspired by [16], the basic model described in this section is capable of finding routing solution without opens and shorts based on MCF theory.

A. Special Case

For the ease to discussion, our basic MCF model considers a special case first with the following two assumptions.

Assumption 1: Each component only covers one vertex.

Assumption 2: Each net only has two components.

Naturally, each net in N is treated as a commodity with command of unit flow in MCF model. For each net, one component is treated as source and the other one is treated as target and the unit flow is shipped from source to target. Some basic notions related to MCFRoute are given in Table I. We use these variables to model detailed routing problem through MCF theory.

Detailed routing requires that each edge of routing graph is occupied by one object at most, so edge capacity should be 1 if it is not occupied by routing blockage; otherwise, is 0.

Edge cost can be any positive real number and the cost values are different with different edge types. All edges of routing graph G fall into two categories, namely, *via-edge* and *wire-edge*. *Via-edge* connects two vertices on different layers and *wire-edge* connects two adjacent vertices on the same layer. Considering the preferred direction of routing layer, *wire-edge* is further divided into *prefer-edge* and *nonprefer-edge* based on edge direction.

The value of $d(k, v_j)$ is determined by vertex type. If vertex v_j is occupied by a component of net n_k and the component is treated as the source, then $d(k, v_j) = 1$. If vertex v_j is a component of net n_k while the component is treated as the target, then $d(k, v_j) = -1$. Otherwise, $d(k, v_j) = 0$. It is noteworthy that the assignment of source and target will not bring any effects to the result because of the symmetry of routing graph. In the experiments, the first component read in from database is assigned as source and the other one is assigned as target.

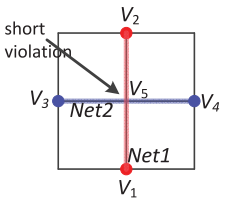


Fig. 4. Short example: two nets with two components ($V_1 - V_2$; $V_3 - V_4$).

$f(k, e_i)$ is the decision variable with value range of 0 and 1. It indicates whether an edge e_i belongs to net n_k; i.e., $f(k, e_i)$ equals to 1 if edge e_i is occupied by net n_k and equals to zero otherwise. All the constraints and objective function are formulated by these variables.

Based on the formulation of detailed routing problem in Section II, our MCF based detailed routing algorithm is introduced as follows.

First, we give connectivity constraints on all vertices for each net based on flow conservation theory to satisfy Route Constraints (1). Flow conservation theory requires that the difference of total flow of vertex v_j that net n_k stream out and stream in must be equal to the flow command of vertex v_j that net n_k demands, i.e., $d(k, v_j)$. The connectivity constraint based on flow conservation theory is formulated as (1). The first item and second item in (1) indicate the total flow of vertex v_j that net n_k stream out and stream in, respectively

$$\sum_{e \in E_{v_j,\text{out}}} f(k, e) - \sum_{e \in E_{v_j,\text{in}}} f(k, e) = d\big(k, v_j\big). \tag{1}$$

Then, edge capacity constraint and vertex capacity constraint are introduced to satisfy Route Constraints (2) which can guarantee the routing solution given by MCF model without short violations. The capacity of each routing edge is at most one in detailed routing problem and the flow each net demands is also one. So (2) can ensure edge capacity constraint

$$\sum_{k=1}^{K} (f(k, e_i) + f(k, \bar{e}_i)) \leq \min\{u(e_i), u(\bar{e}_i)\} \tag{2}$$

where e_i and $\bar{e}_i$ are brother edges.

Even though edge capacity constraint is constructed, routing solution with short violations still exists in the solution space of MCF model. Fig. 4 is a simple example. Red net (Net 1) goes through vertex v_5 using two vertical direction edges, at the same time, blue net (Net 2) goes through the same vertex using two horizontal direction edges. All edges are only used once, but there is a short violation at vertex v_5. To avoid this kind of violations, vertex capacity constraints as (3) are added to MCF model

$$\sum_{k=1}^{K} \sum_{e_i \in E_{v_j}} f(k, e_i) \leq 2. \tag{3}$$

Combining (1)–(3), the following conclusions can be obtained.

Lemma 1: $\sum_{e_i \in E_{v_j}} f(k, e_i) = 2$ or 0, $\forall n_k \in N$.

Lemma 2: $\sum_{k=1}^{K} \sum_{e_i \in E_{v_j}} f(k, e_i) = 2$ or 0.

The conclusions could be explained as that each vertex is either unused or used by a pair edges.

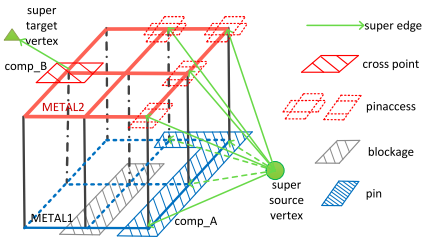


Fig. 5. Pinaccess and super vertex.

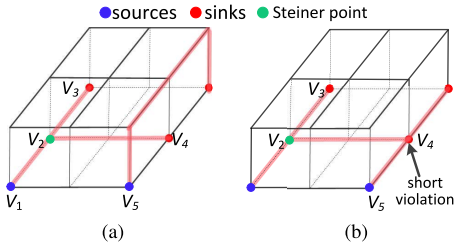


Fig. 6. Routing example from [24]. (a) Routing solution illustrated in [24]. (b) Routing solution which is feasible for [24] but is illegal.

The total cost of detailed routing problem can be calculated by

$$\text{Cost} = \sum_{k=1}^{K} \sum_{e \in E} (f(k,e) \cdot c(e)). \tag{4}$$

With different edge cost assignments, we can achieve different routing goals. In our model, prefer-edges are encouraged to be used by assigning smaller cost, and assigning larger cost to nonprefer-edges and via edges in order to minimize nonprefer-edge count and via count.

So far, our basic MCF model of detailed routing problem is completed. Connectivity constraints ensure there are no open nets, and capacity constraints eliminate the solutions with short violations. All the solutions in solution space of basic MCF model are reasonable if only open and short design rules are considered. The best routing result with minimal routing cost is selected by the objective function.

With the formal description of constraints and objective above, the detailed routing problem can be formulated as the following ILP formula:

$$\textbf{min.} \quad \sum_{k=1}^{K} \sum_{e \in E} (f(k,e) \cdot c(e)) \tag{5a}$$

$$\textbf{s.t.} \quad \sum_{e \in E_{v_j,\text{out}}} f(k,e) - \sum_{e \in E_{v_j,in}} f(k,e) = d\big(k, v_j\big), \quad \forall v_j \in V, \ \forall n_k \in N \tag{5b}$$

$$\sum_{k=1}^{K} f(k,e_i) \le u(e_i) \qquad \forall e_i \in E \tag{5c}$$

$$\sum_{k=1}^{K} \sum_{e \in E_{v_j}} f(k,e) \le 2 \qquad \forall v_j \in V \tag{5d}$$

$$f(k,e_i) \in \{0,1\} \qquad \forall e_i \in E, \ \forall n_k \in N. \tag{5e}$$

B. General Case

Now we need to consider the general cases without Assumptions 1 and 2.

1) Super Node Method: It is common that several vertices are covered by the same component, especially if the component is a pin of net. Furthermore, in order to handle the complexity caused by irregular pins, *pinaccesses* are introduced in our model to guide the access point of pins. As Fig. 5 illustrates, component A (blue polygon) is a pin and locates at metal1, pinaccesses of this pin are created on metal2 with determined via rotation. During path finding, when the path hits pinaccess, a via with determined rotation is created automatically. Here, via rotation means via enclosure extension directions on two metal layers. On each routing layer, via enclosure can be in either horizontal or vertical direction, so there are four rotations for a via. By default, via enclosure direction is the same as the preferred direction of routing layer in this paper except for pinaccess.

Fig. 5 describes a detailed routing example on 3-D routing graph. Component A is a pin which covers four vertices and has four pinaccesses while component B is a crosspoint on boundary that covers one vertex. Without loss of generality, component A is regarded as source and component B is target. Based on MCF theory, one commodity flow must outflow from any one vertex which is covered by Component A or it is pinaccess, and inflow to the vertex that Component B covers.

A strategy of super vertex is proposed to make our basic model capable of handling this general case. First we make some extension to routing graph. Two virtual vertices SV_{com_A} and SV_{com_B} are added to the routing graph that are regarded as the *super source vertex* and the *super target vertex*, respectively. Super source vertex and super target vertex are denoted by green circle and triangle in Fig. 5, respectively. These two super vertex are connected with corresponding graph vertices through *super edges* denoted by green arrow line. With the extensional routing graph, we generalize the basic model. Assuming that V_{com_A} denotes the set of vertices relevant to component A and SE_{com_A} denotes the set of super edges relevant to super vertex SV_{com_A}. Each super edge $e \in \text{SE}_{\text{com}_A}$ introduces a binary variable $g(e)$ to basic MCF model. Combining with connectivity constraints, the constraint shown in (6) ensures that the stream outflow from one and only one vertex in SV_{com_A}

$$\sum_{e \in \text{SE}_{\text{com}_A}} g(e) = 1. \tag{6}$$

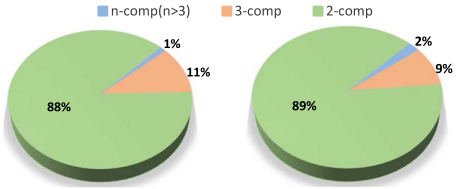
2) Component Count Aware Routing Strategy: In a detailed routing problem, not all the nets are 2-component. The model constructed above is only capable to route 2-component nets. A component count aware routing strategy is proposed in this section to handle multicomponent nets.

Recently, [24] proposes a concurrent detailed routing formulation that could handle multicomponent net. But the formulation presented in the third section of [24] can not be applied to our problem because edge capacity constraints are not enough to avoid short violations. Fig. 6(a) shows a two-net example and the corresponding routing solution cited from [24]. Fig. 6(b) shows another routing solution which is also feasible even optimal for the ILP formulation in [24]. But a short violation is carried out on vertex V_4. That is, even though the formulation in [24] is able to connect all nets together, the routing solution may not be DRC-clean.

Furthermore, the formulation considering multicomponent nets makes the ILP problem very difficult to solve, because of the following two reasons. First, the decision variable can be any integer which makes it to be a general ILP problem, which is with larger solution space compared with a 0-1 ILP problem. Second, the elements of constraint matrix contain other integers except for $-1, 0$, and 1, which makes the problem more complex to solve. Table II is a comparison of two formulations. One is the formulation in [24]. The other one is the formulation in

TABLE II
COMPARISON OF DIFFERENT FORMULATIONS

TestCase		OptRouter [24]			ours		
name	#Net	#Var.	#Constr.	Time	#Var.	#Constr.	Time
case0	2(1)	153	177	0.04	136	60	<0.01
case1	1(0)	1188	1647	0.03	594	324	0.02
case2	2(1)	2673	2997	0.17	2673	810	0.04
case3	3(2)	3564	4347	1.24	4158	1296	0.08
case4	4(2)	4752	5697	1.32	4752	1458	0.15



(a) (b)

Fig. 7. Survey results on component distribution. (a) Industrial benchmarks. (b) Academic benchmarks.

Section IV, and 3-pin nets are decomposed into three subnets using steiner point. The first column ("name") is testcase name. The second column ("#Net") is total net count and the value in brackets is 3-component net count. The results are compared in variable count ("#Var."), constraint count ("#Constr.") and runtime("Time") in *seconds*. Columns 3–5 are the results of [24], and columns 6–8 are the results of our formulation. Both formulations are solved by GUROBI [25] on the same machine and terminated until optimal solution of each formulation is found. Case0 is the example shown in Fig. 6. The formulation in [24] takes 0.04 s to get a solution as shown in Fig. 6(b), while our formulation only takes less than 0.01 s to get the optimal solution as shown in Fig. 6(a). Cases 1–4 are four examples on a routing graph with nine horizontal routing tracks and 18 vertical routing tracks. From the formulation in [24] we can find that indicator $e_{i,j}^{k}$ which indicates whether arc $a_{i,j}$ is used in the routing of net n_k is calculated by two constraints. Those indicators will introduce lots of constraints to ILP formulation and make the ILP problem much more difficult to solve. And the experimental results also show that our formulation has less constraints and spends less time to get more reasonable solution.

Except for the above analysis and the experiments, we made a survey about components count distribution with different scale of routing window in several academic and industry VLSI circuits. The statistical results in Fig. 7 show that nets with two components account for about 88% of total nets and nets with three components are about 10%. Nets with more than three components (hereafter, referred to as *multicomponent nets*) only account for 2%.

Guided by experiments and statistics, 3-component net is decomposed into three subnets in our router to accommodate the ILP model in Section IV-A. But it is unreasonable if only the steiner point is selected as the decomposition point. Three examples of failure or poor result are listed here.

1) The steiner point locates at a vertex which is occupied by the pin geometry of another net. The routing fails because of the short violation.
2) The steiner point locates at a vertex which is occupied by the pin geometry of the same net. Actually, all the vertices covered by this pin geometry can be served as decomposition point.
3) The fixed position of steiner point may lead to detour of other nets.

The main reason of these failure or poor result is the uniqueness and fixity of decomposition point. Furthermore, we find that it spends too much time to find optimal routing solution for these routing cases. So we try to provide several decomposition candidates to MCFRoute as shown in Algorithm 1. Decomposition candidates are selected based on the steiner point and the geometries around it. After the decomposition candidates are determined, a super node associated with them will be added to the model.

Algorithm 1 Decomposition Point Selection

Require: Routing graph.
Require: Components position.
1: Find steiner point;
2: Check the geometries around steiner point;
3: If the steiner point locates at the pin geometry of another net, choose the upper layer point as the steiner point and the adjacent vertices of new steiner point as decomposition point candidates;
4: If the steiner point locates at the pin geometry of the same net, choose all the point on the geometry and all the pin access of the pin as decomposition point candidates;
5: Otherwise, choose the adjacent vertices of steiner point as decomposition point candidates;

It is worth to note that even though the number of multicomponent net is small, it does not mean that they are not important. Actually, most of these nets are critical. However, the steiner point of multicomponent net is difficult to find. It is not impractical to decompose them into 2-component subnets. In our routing flow, multicomponent nets are not considered by basic MCF model. They are routed first by multisource multisink maze routing, then other nets are routed by MCFRoute using remainder routing resource. Even though this strategy can not guarantee global optimal routing in routing region, a valid solution without DRC violations can be found in less time. Assigning routing priority to multicomponent net which may be critical is also reasonable.

V. COMPLEX DESIGN RULE MODELING

In deep submicron technology nodes, there are many other complex design rules besides the LVS checking. The routing paths generated by the detailed router have to pass the DRC as well. These design rules need to be modeled correctly and efficiently in the basic MCF model. In this section, we enumerate some of these design rules' modeling in MCF model.

A. Spacing Rule

There are different types of spacing rules in typical 45/28 nm technology nodes, including metal spacing, end-of-line (EOL) spacing, cut to metal spacing, cut to cut spacing, etc. Most of them can be modeled in the MCF problem using similar method.

To describe spacing rule constraints, we define two disjunction variables $\varphi(k, v_j)$ and $\eta(k, v_j)$ as (7) and (8), respectively, for all $n_k \in N$ and $v_j \in V$

$$\varphi(k, v_j) = \begin{cases} 1, & \text{if } \exists e_i \in E_{v_j} \text{ and } f(k, e_i) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$\eta(k, v_j) = \begin{cases} 1, & \text{if } \exists e_i \in E_{v_j}^{\eta} \text{ and } f(k, e_i) = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Here, $E_{v_j}^{\eta}$ denotes a set of via edges of vertex v_j. $\varphi(k, v_j)$ indicates whether net n_k occupies vertex v_j. If net n_k occupies vertex v_j, $\varphi(k, v_j) = 1$, otherwise, $\varphi(k, v_j) = 0$. similarly, $\eta(k, v_j)$ indicates whether net n_k occupies the via edge of vertex v_j.

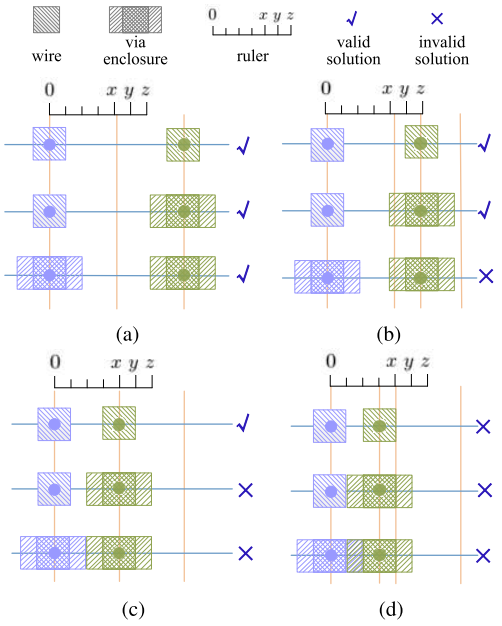


Fig. 8. Four cases of $\Gamma(v_{j1}, v_{j2})$. The x, y and z on the ruler are $\alpha_m + \gamma_m$, $\alpha_m + \beta + \gamma_m$, $\alpha_m + 2\beta + \gamma_m$, respectively.

For two vertices $v_{j1} \in V$ and $v_{j2} \in V$ on the same metal plane, let (x_1, y_1, z) and (x_2, y_2, z) be the coordinate of v_{j1} and v_{j2}. The distance of them is defined as

$$\Gamma\big(v_{j1}, v_{j2}\big) = \max(|x_1 - x_2|, |y_1 - y_2|).$$

1) Metal Spacing Rule: Basic spacing rule specifies the minimum distance allowed between two geometries of different nets on metal layer [26].

On metal layer, there are two type geometries of metal wire and via enclosure. Let α_m, β, and γ_m be the value of metal wire width, via enclosure length and metal spacing specified in the technology file, respectively. Given any one vertex pair $(v_{j1},\ v_{j2}) \in (V \times V)$ $(j1 \neq j2)$ on the same layer, $\Gamma(v_{j1}, v_{j2})$ can be classified as following four cases.

Case 1 (No Violation): If the distance of v_{j1} and v_{j2} satisfies

$$\alpha_m + 2\beta + \gamma_m \leq \Gamma\big(v_{j1}, v_{j2}\big).$$

v_{j1} and v_{j2} can be occupied by either geometry type and no spacing violation will emerge, as shown in Fig. 8(a). So we do not need to design any constraints for these vertex pair.

Case 2 (Enclosure-to-Enclosure Violation): If the distance of v_{j1} and v_{j2} satisfies

$$\alpha_m + \beta + \gamma_m \leq \Gamma\big(v_{j1}, v_{j2}\big) < \alpha_m + 2\beta + \gamma_m$$

v_{j1} and v_{j2} cannot be occupied by two vias at the same time, otherwise, there will be a spacing violation, as shown in Fig. 8(b). So constraint (9) should be added to MCF model to prohibit this routing solution

$$\eta\big(k_1, v_{j1}\big) + \eta\big(k_2, v_{j2}\big) \leq 1,\ \ \forall n_{k_1}, n_{k_2} \in N, k_1 \neq k_2. \quad (9)$$

Case 3 (Wire-to-Enclosure Violation): When the distance of v_{j1} and v_{j2} satisfies

$$\alpha_m + \gamma_m \leq \Gamma\big(v_{j1}, v_{j2}\big) < \alpha_m + \beta + \gamma_m$$

if v_{j1} and v_{j2} are used at the same time, the type of both geometries should be metal wire as Fig. 8(c) shows. Constraints (11) and (10) need to be introduced to MCF model

$$\varphi\big(k_1, v_{j1}\big) + \eta\big(k_2, v_{j2}\big) \leq 1,\ \ \forall n_{k_1}, n_{k_2} \in N, k_1 \neq k_2 \quad (10)$$

$$\eta\big(k_1, v_{j1}\big) + \varphi\big(k_2, v_{j2}\big) \leq 1,\ \ \forall n_{k_1}, n_{k_2} \in N, k_1 \neq k_2. \quad (11)$$

Case 4 (Wire-to-Wire Violation): When the distance of v_{j1} and v_{j2} only satisfies

$$\Gamma\big(v_{j1}, v_{j2}\big) < \alpha_m + \gamma_m$$

even if these two vertices are occupied by two metal wire of different nets, there will be a short violation as shown in Fig. 8(d). In this case, MCF model disallow v_{j1} and v_{j2} to be used by different net simultaneously with constraint

$$\varphi\big(k_1, v_{j1}\big) + \varphi\big(k_2, v_{j2}\big) \leq 1,\ \ \forall n_{k_1}, n_{k_2} \in N.k_1 \neq k_2. \quad (12)$$

2) EOL Spacing Rule: The EOL spacing rule ensures that optical proximity correction (OPC) can be performed without interference between the OPC shapes added at the EOLs [26]. The line-end of metal wires (EOL-wires) generally require larger spacing to separate themselves from the neighbor wires. If the line-end is within some distance to via cut, it becomes an EOL-via and has another spacing requirement. The spacing requirements for EOL-wires and EOL-vias are modeled in our MCF problem as well. Given a pair of vertices v_{j1} and v_{j2}, similar to metal spacing rule, designing EOL spacing also need to check which case v_{j1} and v_{j2} falls into based on $\Gamma(v_{j1}, v_{j2})$, α_m, β, and γ_{eol}, here γ_{eol} is the EOL spacing defined in the technology file. Then relevant constraint(s) are introduce to MCF model.

3) Cut-to-Cut Spacing Rule: Cut spacing rule specifies the minimum spacing allowed between different via cuts [26]. Let α_c and γ_c be the cut width and cut spacing value in technology file, respectively. If the distance of the given vertex pair (v_{j1}, v_{j2}) makes the following inequality to be true, constraint (9) will be added to MCF model:

$$\Gamma\big(v_{j1}, v_{j2}\big) < \alpha_c + \gamma_c.$$

4) Other Spacing Rules: As the decreasing of feature size, more and more spacing design rules are designed to guarantee chip performance, manufacture, and yield. For example, width length dependent spacing rule, contact spacing rule for specific shape and same net spacing rule are very common in 45/28 nm technologies. In this paper, these conditional spacing rules are not supported. If there are conditional spacing rule violations in MCFRoute results, they will be processed in search and repair stage by MazeRouting.

B. Minarea Rule

Minarea is another important design rule, which specifies the minimum metal area required for polygons on the routing layer [26]. To satisfy this rule, the routing path segment on each layer needs to be longer than some thresholds. Modeling this rule in the MCF problem directly will generate too many constraints which makes the runtime unacceptable. As a result, we handle this rule separately in a post-processing stage. Since the wire width on each routing layer is fixed, the Minarea rule is converted into the minimum length rule in this paper.

After MCF model decides the routing paths for all nets, the router creates the layout (wires and vias) on each layer based on the result net by net, during which it checks whether the length of polygon on each layer is larger than the minimal length requirement. If not, the router will extend that polygon to meet the minimum length requirement. However, we observe that the sequential manner of layout generation introduces some DRC violations. For instance, Fig. 9 shows two routing paths which are stack vias [see Fig. 9(a)]. In Fig. 9(b), layout of via B is first generated and the polygon is extended to meet the minimum length. However, when layout of via A is generated, a spacing violation is generated even if without polygon extension. In order to eliminate the blindness of

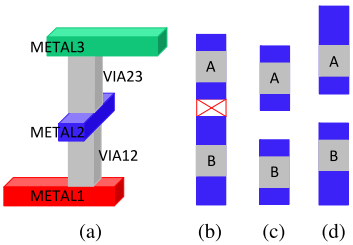


Fig. 9. Two-pass strategy for layout generation. (a) Stack via layout. (b) Layout with DRC violation. (c)–(d) Layout generated by two-pass strategy after the first pass and second pass.

polygon extension, we propose a two-pass strategy for layout generation: in the first pass, layout is generated net by net without polygon extension no matter it meets minimum length requirement or not as Fig. 9(c) shows. In the second pass, the routing engine processes the Minarea violation net by net. Since the utilization of routing resource around the polygon is much more clear when processing Minarea violation, it is conducive to reduce DRC violation. Fig. 9(d) shows the layout generated by our two-pass strategy. Based on the experiments on ISPD 2014 benchmarks, 93.2% Minarea violations could be resolved by the proposed two-pass strategy.

VI. MCF Model Simplification

A. Linearize MCF Model

The spacing constraints in Section V-A introduce many logic expressions which make the basic model very complex and nonlinear. Several methods are introduced here to optimize those constraints with better attributes and the same results by equivalent transformation.

Theorem 1: Constraint (12) is equivalent to constraint

$$\sum_{k=1}^{K}\left(\sum_{e_i\in E'_{v_{j1}}} f(k,e_i)+\sum_{e_i\in E'_{v_{j2}}} f(k,e_i)\right)\le 2 \tag{13}$$

where

$$E'_{v_{j1}}=E_{v_{j1}}\backslash\{(v_{j1},v_{j2}),(v_{j2},v_{j1})\}$$
$$E'_{v_{j2}}=E_{v_{j2}}\backslash\{(v_{j1},v_{j2}),(v_{j2},v_{j1})\}.$$

The proof is provided in Appendix A.

Theorem 2: Constraints (10) and (11) are equivalent to constraints (14) and (15), respectively

$$\sum_{k=1}^{K}\left(\sum_{e_i\in E^{\eta}{}_{v_{j1}}} f(k,e_i)+\sum_{e_i\in E'_{v_{j2}}} f(k,e_i)\right)\le 2 \tag{14}$$

$$\sum_{k=1}^{K}\left(\sum_{e_i\in E'_{v_{j1}}} f(k,e_i)+\sum_{e_i\in E^{\eta}{}_{v_{j2}}} f(k,e_i)\right)\le 2. \tag{15}$$

Theorem 3: Constraint (9) is equivalent to constraint

$$\sum_{k=1}^{K}\left(\sum_{e_i\in E^{\eta}{}_{v_{j1}}} f(k,e_i)+\sum_{e_i\in E^{\eta}{}_{v_{j2}}} f(k,e_i)\right)\le 2. \tag{16}$$

The proofs of Theorems 2 and 3 are similar to Theorem 1 thus are omitted here.

Compared with old formulation, new spacing constraints (13)–(16) have two advantages.

1) It is linear. Different from old formulation, new constraints have linear expressions rather than logical expressions.
2) It is concise. To describe the spacing rule requirement between two vertices, new method uses less constraints than old method. Old method need to construct a spacing constraint between any two nets. So there are $K(K-1)/2$ constraints totally. But new method only need one constraint.

B. Reduce the Scale of MCF Model

The complexity of the ILP problem depends heavily on the total count of variable, constraint and nonzero in the model. Reducing the scale of MCF model can make the problem easier to solve.

First, we have the following theorem.

Theorem 4: Capacity constraint for brother edges e_i and $\bar{e}_i$ could be removed from MCF model if $u(e_i)=1$ and $u(\bar{e}_i)=1$.

The proof is provided in Appendix B.

Second, we restrict the routing layers that a net can use. For a two-component net, assuming that two components locates at layer z_1 and layer z_2 ($z_1\le z_2$), respectively. The final routing resource this net can use is from layer $z_{\min}(=\max\{z_1-\Delta,0\})$ to layer $z_{\max}(=\min\{z_2+\Delta,L\})$. This restriction makes a trade-off between routing quality and runtime. Larger Δ means better quality and worse runtime while smaller Δ means worse quality and better runtime. In our experiment, if $z_{\max}\le 2$, Δ is 2; otherwise Δ is 1.

C. Constraints Relaxation

In a highly congested design, detailed router will search and repair DRC violations in routing regions with different size and center point by many iterations. It must create a unbroken path for each net even if there are some DRC violations in initial stage. This indicates that some solutions outside MCF model solution space arising from violating capacity and spacing rule constraints are also acceptable. In our model, penalty method is utilized to relax capacity and spacing constraints. In our experiments, one penalty variable which must be positive integer is added to every capacity and spacing constraints, and all penalty variables multiplied with large penalty factor are added to the objective function. Taking vertex capacity constraint for example, constraint (3) is substituted by

$$\sum_{k=1}^{K}\sum_{e_i\in E_{v_j}} f(k,e_i)-o(v)\le 2 \tag{17}$$

and $o(v)\cdot p(v)$ is added to objective function (4). Here, $p(v)$ is a large penalty factor. This relaxation makes routing solution with short violations such as Fig. 4 acceptable by setting $o(v)=2$ but the objective value will be very large. By this way, all the problems have feasible solution and DRC-clean solution is preferred.

VII. Solving MCF Model

Given MCF model for the detailed routing problem, an ILP/LP solver GUROBI [25] is used to solve MCF problem. Many efforts are done to reduce the runtime. A solving algorithm is proposed based on the input data and MCF model characters.

A. Prerouting Based on Track Assignment Results

In this section, a prerouting stage based on track assignment results is proposed to reduce the runtime. The results of this stage could provide a good start point for ILP problem.

A start point of an ILP/LP problem has an effect on the iteration/search time. The good start point often does help the ILP/LP solver to reduce the number of iterations to optimality. The closer the start point is to the optimal point in theory, the less time the solver spends.

In routing system, global router, track assignment engine, and detailed router are closely linked and maintain consistency. The former step tries its best to provide well routing guidance for the later step. So the track assignment results is the best guiding information we can get before detailed routing.

The prerouting stage aims to make full use of the information provided by track assignment. After capacity constraints and spacing constraints are relaxed in Section VI-C, a feasible solution for ILP problem is that all components of every net are connected together. In this stage, all segments of each net generated in track assignment stage are connected through wire stretch and via creation. Horizontal (vertical) segments on routing layer and vertical (horizontal) segments on neighbor routing layer(s) belonging to the same net will be stretched to have intersection and then vias are created on the intersection point. When there are no open nets, the routing solution is translated into 0-1 solution based on the routing graph which is served as the start point of ILP problem. Experimental results in Section VIII-C show that this strategy can reduce the runtime by 5% averagely.

B. LP Relaxation

As described in Section I-A, global routing has been modeled and solved in a concurrent manner in several works. Generally, the routing problem is formulated as an ILP problem based on MCF method, and solved by an ILP solver or some heuristic algorithms. The high time complexity greatly limits the feasible problem size because of the NP-complete attribution of ILP problem. There are two popular strategies to reduce the time complexity in global routing problem. The first one is dividing the routing region into subregions to reduce the problem size such that the routing problem can be handled subregion by subregion. The second one is relaxing the ILP problem into LP problem, then the fractional solution is transformed to integer solutions through rounding such as randomized rounding [27]. This strategy can greatly reduce the runtime because LP problems can be optimally solved in polynomial time. However, it should be noted that the conventional randomized rounding is not practice in detailed routing problem, because detailed routing algorithm requires exactly layout solution. If a fractional solution is transformed into 1, a short violation may be introduced; and if it is transformed into 0, an open violation may be introduced.

Inspired by the speedup approach in concurrent global routing algorithms, we try to relax the ILP problem into an LP problem. Rather than utilizing randomized rounding strategy, we can directly obtain 0-1 integer solutions by solving LP problem.

In general, for multicommodity network flow problems with more than two commodities -nets in our case- the solution of the LP problem is not necessarily that of the corresponding ILP problem because its constraint matrix is not totally unimodular [28], [29]. However, some works (e.g., [30] and [31]) have reported that in practice most of the linear program relaxation of the MCF problem often provides integer optimal solutions. The work of [30] explains that 0-1 MCF problem almost always yields a binary solution from relaxed LP formulation. Furthermore, each entry in the constraint matrix of our formulation is 0, $+1$, or -1 which is the requirement of totally unimodular matrix. This characteristic may improve

Algorithm 2 ILP Problem Solving Algorithm

```
1:  pre_routing();
2:  solveLP() → result_lp;
3:  if result_lp has no fractional solution and DRC clean then
4:      solutionTranslation();
5:  else if result_lp has DRC violations then
6:      rip-up and reroute in search and repair stage;
7:  else
8:      initial ILP problem by result_lp;
9:      solveILP() → result_ilp;
10:     if result_ilp has no DRC violation then
11:         solutionTranslation();
12:     else if Solver is terminated by time limitation then
13:         MazeRouter();
14:     else
15:         rip-up and reroute in search and repair stage;
16:     end if
17: end if
```

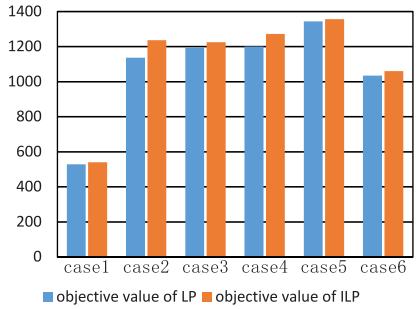


Fig. 10. Objective value gap between LP formulation and ILP formulation.

the success probability. Experimental results show that about 94.3% routing regions can yield a binary solution by solving an LP problem. Even if some routing regions fail to get binary solution, the fractional solution still provides useful information to ILP solver in our solving algorithm. First, a binary solution is produced based on the fractional solution using a rounding method, which is served as the start point of ILP problem. Second, the LP problem provides a lower bound of ILP problem. Fig. 10 shows objective value gap between LP formulation and ILP formulation of six routing cases derived from the benchmark of "chip1" in Table III. In many routing cases, we find that it takes much runtime to prove a feasible solution is optimal. Based on the LP objective value, a reasonably terminate condition could be made. In our experiments, the ILP solver is terminated if the gap between current objective value and LP objective value is less than 5%. This approach may lead to nonoptimal solution, but it can further reduce the runtime. Combined with track assignment guidance (TRG) strategy, the detailed solving algorithm is shown as Algorithm 2.

First, the start vector of ILP problem is generated based on TR results (line 1). Then, the ILP problem is relaxed into an LP problem by substituting [0, 1] boundary constraint for {0, 1} boundary constraint and solved by an LP solver (line 2). If an optimal 0-1 solution without DRC violations is achieved by LP solver, it will be translated to routing solution (lines 3 and 4). If the optimal solution is not DRC-clean (it can be detected by the optimum value), the routing problem will be rip-up and reroute in search and repair stage (lines 5 and 6). If the optimal solution has fractional value, it will be treated as the new start value of ILP problem. We will solve the ILP problem (lines 7–9). If the ILP solver could get

Algorithm 3 Task Allocation Algorithm of Four-Stage Strategy

Require: Routing region set *regions* and its scale(M rows and N columns);
Require: routing stage: S;
Ensure: routing task queue: $task_S$
1: **for** $row \leftarrow 1; row \leq M; row \leftarrow row + 2$ **do**
2: **if** $Row = 1$ && $S \geq 2$ **then**
3: $row \leftarrow 2$;
4: **end if**
5: **for** $col \leftarrow 1; col \leq N; col \leftarrow col + 2$ **do**
6: **if** $col = 1$ && $S\%2 = 1$ **then**
7: $col \leftarrow 2$;
8: **end if**
9: $task_S$.pushback($regions[row][col]$);
10: **end for**
11: **end for**

an optimal 0-1 solution, routing solution is generated based on it (lines 10 and 11). If the ILP solver does not find a DRC-clean solution in given time, this routing problem is rip-up and rerouted by a MazeRouter in the same region (lines 12 and 13). Otherwise, the problem will be rip-up and reroute in search and repair stage (lines 14 and 15).

C. Multithread Strategy

As multicore or many-core architecture has become the mainstream of CPU design, the development of EDA algorithms on multicore platform can boost performance and quality in solving EDA problems [32]. Multithread is a common and effective speedup strategy that has been implemented on some physical design problems, such as placement [33] and routing [34], [35]. Most of them are net-based or partition-based, but they may be hard to achieve high speed-up ratio. For instance, the multithread algorithm in [34] could only achieve 2.71× speedup on a quad-core machine, while the work in [35] can have 7.1× speedup on a eight-core machine. The main reason of the low speed-up ratio is that runtime is wasted in task conflict judgment and task waiting. In this paper, even though the detailed routing is executed on each subregion independently, the utilizing of routing resource near region boundary is influenced by the adjacent regions. To utilize the independence of subregions, a multithread strategy based on MapReduce [36] is developed. In order to decrease the boundary influence on routing quality, a four-stage routing strategy is proposed. Algorithm 3 shows the details of task allocation algorithm of four-stage strategy. The input of this algorithm is routing region set with its scale and routing stage, while the output is routing task queue of this routing stage. All the routing regions are partitioned into four task queues based on their row parity and column parity as Fig. 11 shows. This strategy could guarantee that two adjacent subregions will not be solved at the same routing stage. In each routing stage, any two routing tasks are absolutely independent. Comparing the CPU time and real time, we can observe that the multithread strategy we implemented achieves a near-linear speedup ratio.

VIII. Experimental Results

The MCF based detailed router, MCFRoute, is implemented with C++ language on Linux server with 2.4 GHz Intel Xeon CPU and 24 GB memory. The proposed router could handle some complex design rules, so the DRC violation count is a main index to evaluate routing quality. As far as we know, there are few academic works reporting the DRC violation count. In order to show the effectiveness of the proposed algorithm, we compare it with Cadence router, Encounter,[2] on 45 nm technology benchmarks. GUROBI is used as the LP/ILP solver. The runtimes of MCFRoute and Encounter are measured by *second* with eight threads and single thread, respectively.

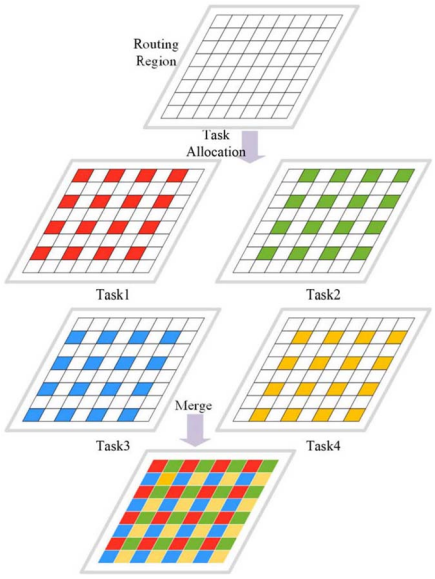


Fig. 11. Illustration of four-stage strategy.

A. Benchmark Information

Our experiments are executed on two series benchmarks. The first one is five industrial benchmarks and the second one is six academic benchmarks derived from ISPD 2005 contest [37] and ISPD 2014 contest [38]. ISPD 2005 benchmarks are transformed from bookshelf format to LEF/DEF format by a publicly available conversion tool.[3] All benchmarks are mapped to a library with 45 nm design rules. Using the APIs from OpenAccess (OA) system, we convert the LEF/DEF inputs to an OA database. Both routers work on and exchange data through the OA database. Table III shows the detailed information of our benchmarks. The first five benchmarks are derived from industrial design. The following four benchmarks are derived from [37], the last nine benchmarks are derived from [38]. In Table III, the second column (name) is the name of each benchmark. The third column (#Net) is the net count. The fourth column ("Size") denotes the GRC size, i.e., the subproblem count. Both the height and width of each GRC is equal to the height of standard cell. It gives the scale of every benchmarks. The fifth column ("#Layer") shows the routing layer count.

B. Effectiveness of Model Simplification

In Section VI-B, two optimization methods are proposed to optimize our MCF model. We take some experiments to make a comparison between original model and optimized model on total count of variable (#Var.), constraint (#Constr.), and nonzero ("#Non-zero") within several regions of different size. In Table IV, the first column lists the region size by vertical track count and horizontal track count. The second column is the net count in each region. The following six columns show number ($\times 10^5$) of variable, constraint and nonzero in original

[2]Routing command is "`routeDesign`" with parameter "`setNanoRouteMode-drouteAutoStop false`."
[3]PlaceUtil: developed by University of Michigan.

TABLE III
BENCHMARK INFORMATION

source	name	#Net	Size	#Layer
Industry	chip1	36760	297x309	6
	chip2	52612	352x367	6
	chip3	100479	463x484	6
	chip4	90905	446x466	6
	chip5	100473	462x483	6
ISPD 2005	bigblue1	280281	964x965	5
	bigblue2	560340	1558x1566	5
	adaptec1	215621	964x965	5
	adaptec2	255730	1268x1268	5
ISPD 2014[1]	matrix_mult_wt	158527	550x550	5
	matrix_mult_nt	158527	275x275	5
	pci_bridge32_1	30835	124x124	5
	pci_bridge32_2	30835	122x122	5
	edit_dist_1	133223	361x361	5
	edit_dist_2	133223	348x348	5
	des_perf_2	112878	230x230	5
	matrix_mult	158527	275x275	5
	fft	33307	133x133	5

[1] des_perf_1 is not shown here because even Encounter can not get a reasonable routing results (more than 20000 DRC violation left in final layout).

TABLE IV
EFFECTIVENESS OF MODEL SIMPLIFICATION

size	#net	#Var.		#Constr.		#Non-zero	
		orig	opt	orig	opt	orig	opt
10×10	29	1.14	0.72	0.25	0.16	0.86	0.55
20×10	35	2.78	1.86	0.58	0.39	2.09	1.41
20×20	52	8.3	5.42	1.66	1.1	6.54	4.22
40×20	90	28.7	20.3	5.51	3.96	22.5	15.5
40×40	135	86.4	65.6	16.21	12.4	69.4	48.4

model and optimized model, respectively. Experimental results in Table IV show that three metrics are reduced by 31%, 30%, and 33%, respectively. The reduction of MCF model scale could make the model easier to solve and save some runtime.

C. Effectiveness of ILP Problem Solving Algorithm

The experimental results in this section show the effectiveness of the proposed ILP problem solving algorithm in Section VII. Figs. 12 and 13 show the runtime comparison of MCFRoute without and with the proposed acceleration techniques on industrial and ISPD 2005 benchmarks, respectively. The results are divided into three patterns: the first one is "pure ILP" that solves the ILP model by ILP solver (lines 9–16 in Algorithm 2). The second one is "LP + ILP" that solves the model by LP solver first; if LP solver fails, solves it by ILP solver based on LP solution (lines 2–17 in Algorithm 2). The third one is "TRG" that executes a prerouting step based on track assignments results to provide start point for LP model first before LP + ILP flow (Algorithm 2). Note that detailed routing results are demonstrated in Section VIII-D. The exact runtimes are also labeled in the figures. From Fig. 12, we can observe that compared with pure ILP, LP + ILP flow can reduce the runtime by 29.4%. Besides, compared with LP + ILP flow, the one with prerouting stage (TRG) can further reduce the runtime by 4.8%. From Fig. 13, we can see that compared with pure ILP, LP + ILP flow can reduce the runtime by 21%. In addition, compared with LP + ILP flow, the one with prerouting stage (TRG) can further reduce the runtime by 6%. In summary, the results can demonstrate the effectiveness of our proposed solve flow in runtime reduction.

Fig. 14 demonstrates the speedup ratio of the proposed multithread algorithm on the basis of TRG solve flow. The horizontal ordinate represents the thread count and the vertical ordinate represents the speedup ratio which is calculated by

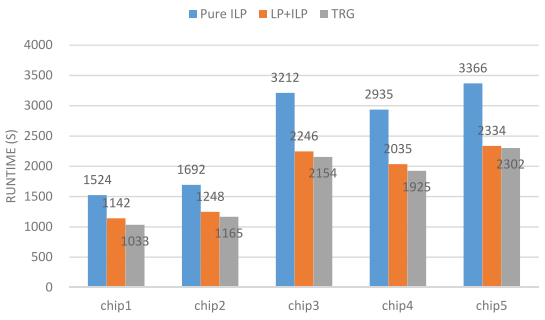


Fig. 12. Runtime improvement on industrial benchmarks.

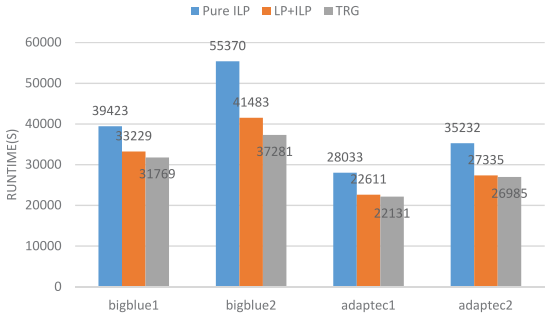


Fig. 13. Runtime improvement on academical benchmarks.

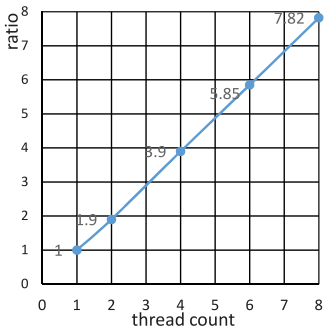


Fig. 14. Speedup ratio of multithread algorithm.

(CPU Time/Elapsed Time).[4] The experimental results show that the multithread algorithm can get a closed to linear speedup ratio.

D. Improvement of Routing Results

Besides the ILP problem solving algorithm, we also propose three effective strategies to improve the routing quality and reduce the runtime in this paper. In this section, comparisons are made among Encounter v10.10,[5] the work in [23] and the proposed MCFRoute adopting the TRG solving flow. The comparison results could demonstrate the effectiveness of decomposition point selection algorithm, two-pass layout generation strategy and four-stage task allocation algorithm. In the experiments, we try to let Encounter run longer until the improvement is negligible and the runtime is measured in second with one thread. The comparison is made on four aspects: 1) wirelength (WireLen); 2) via count (#Via, $\times 10^3$); 3) DRC violation count (#Vio.); and 4) runtime (Runtime) as shown

[4] "Elapsed time" means the amount of time that passes from the beginning of routing to its end.

[5] Note that there are newer version of Encounter and other industrial detailed routers that may conduct better performance, but v10.10 is the most updated academic version we can access to. It is by no means that our proposed detailed router to beat all state-of-the-art industrial detailed routers.

TABLE V
COMPARISON ON INDUSTRIAL BENCHMARKS

name	Encounter v10.10				MCFRoute in [23]				MCFRoute Full Flow			
	WireLen.	#Via	#Vio.	Runtime	WireLen.	#Via	#Vio.	Runtime	WireLen.	#Via	#Vio.	Runtime
chip1	716	406	267	1680	664	390	282	2140	647	370	61	1033
chip2	861	436	533	1320	833	419	281	2191	827	401	94	1165
chip3	1750	843	922	2520	1706	824	503	4857	1674	780	166	2154
chip4	1599	760	795	2280	1564	744	510	4426	1536	704	206	1925
chip5	1785	846	910	2880	1742	826	537	4822	1709	780	164	2302
Comp.	1.05	1.08	4.96	1.24	1.02	1.06	3.06	2.15	1	1	1	1

TABLE VI
COMPARISON BETWEEN ENCOUNTER AND MCFROUTE ON ISPD 2014 BENCHMARKS

name	Encounter v10.10				MCFRoute in [23]				MCFRoute Full Flow			
	WireLen.	#Via	#Vio.	Runtime	WireLen.	#Via	#Vio.	Runtime	WireLen.	#Via	#Vio.	Runtime
pci_bridge32_1	274	160	0	181	288	159	7	1469	269	159	0	385
pci_bridge32_2	280	160	0	188	294	159	10	1274	275	159	0	425
edit_dist_1	4130	876	0	673	-	-	-	-	4102	873	0	3590
edit_dist_2	4004	878	2	1116	4074	884	11	16219	3978	875	0	3736
des_perf_2	1438	665	7	1950	1502	658	416	10548	1427	661	8	3034
matrix_mult	2080	885	2467	4992	2163	879	6187	18853	2055	881	2305	7414
fft	533	228	4049	4368	556	229	10931	15843	527	228	3117	6849
Comp	1.008	1.004	1.202	0.53	1.040	1.002	2.238	3.096	1	1	1	1

TABLE VII
COMPARISON BETWEEN ENCOUNTER AND MCFROUTE ON ISPD 2005 BENCHMARKS

name	Encounter v10.10				MCFRoute in [23]				MCFRoute Full Flow			
	WireLen.	#Via	#Vio.	Runtime($10^3 s$)	WireLen.	#Via	#Vio.	Runtime($10^3 s$)	WireLen.	#Via	#Vio.	Runtime($10^3 s$)
bigblue1	167	3117	0	1.97	163	2920	0	42	163	2920	0	32
bigblue2	246	5179	0	2.8	241	5023	0	50	241	5023	0	43
adaptec1	129	2420	0	1.51	125	2321	0	31	125	2321	0	22
adaptec2	155	2829	0	1.85	151	2655	0	36	151	2655	0	27
Comp.	1.03	1.05		0.07	1.00	1.00		1.28	1	1		1

in Tables V–VII. The wirelength is measured in *millimeter*, and runtime is measured in *second* with eight threads. The first column shows chip name. Columns 2–5, columns 6–9, and columns 10–13 show the wirelength, via count, violation count, and runtime of Encounter v10.10, the work in [23] and this paper, respectively.

Table V shows the effectiveness of the proposed MCFRoute in industrial benchmarks. For these complex industrial benchmarks, the proposed algorithm can achieve a desirable results. Compared with [23], the wirelength, via count, DRC violation count and runtime in this paper are reduced by 2%, 5%, 67%, and 53%, respectively. The reduction of wirelength and via count is benefited from decomposition point selection algorithm. All of the three strategies lead to the reduction of DRC violation count and runtime. Compared with Encounter, MCFRoute can reduce the DRC violations by 80%. At the same time, the wirelength and via count are also reduce by 5% and 8%, respectively. The effective multithread strategy make the runtime only 0.8× of Encounter.

The effectiveness of MCFRoute in ISPD 2014 benchmarks is demonstrated in Table VI. ISPD 2014 benchmarks are similar to industrial designs and could be used to do detailed routing task directly.[6] Experimental results show that the proposed MCFRoute could reduce the wirelength, via count, DRC violation count and runtime by 3.87%, 0.17%, 55.32%, and 67.7%, respectively, compared with the work in [23]. Table VI also shows that MCFRoute could reduce the wirelength, via count, DRC violation count by 0.83%, 0.41%, and 16.78%, respectively, compared with Encounter v10.10. And the runtime is 1.89× of Encounter. As reported in Encounter, the density of these designs are up to 0.83 (except for edit_dist_1 and edit_dist_2), it indicates that MCFRoute is capable to handle high density design routing task.

[6]MCFRoute does not support off-grid vias, so we restrict the off-grid via generation of Encounter by "setNanoRouteMode -drouteOnGridOnly via."

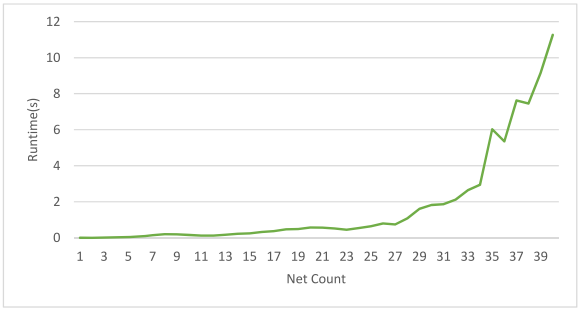


Fig. 15. Scalability of MCFRoute: relation between runtime and net count.

The routing results comparison on four ISPD 2005 benchmarks are shown in Table VII. Experimental results show that not only can our algorithms get a routing result without DRC violations but also reduce wirelength and via count by 2.2% and 4.6%, respectively. Compared with our preliminary version [23], the runtime could be reduced by 22%. However, we also observe that the runtime is a great disadvantage when dealing with larger scale benchmarks.

Fig. 15 demonstrates the relation between runtime and net count based on ISPD 2014 benchmarks. The routing window is a GRC with 10 × 10 grids. Fig. 15 shows that if the net count in routing region is greater than 27, the runtime increase exponentially as the increase of net count.

E. Utilized as Incremental Routing Tool

We have mentioned that in traditional sequential methods, such as MazeRoute, A*-search algorithm generally results

TABLE VIII
RESULTS OF INCREMENTAL DETAILED ROUTER

name	Encounter v10.10	MCFRoute	
	#Vio.	#Vio.	Runtime
chip1	267	174	36.13
chip2	533	394	51.93
chip3	922	677	93.8
chip4	795	628	83.42
chip5	910	675	90.13
matrix_mult_wt	59	39	130
matrix_mult_nt	103	43	231
Comp.	1	0.73	

in low quality routing results because of their nonoptimal net-order nature. However, they are usually faster than concurrent method. To take both advantage of sequential method in runtime and advantage of concurrent method in quality, the proposed algorithm is implemented as an incremental detailed router. A detailed router of sequential method routes the nets first, then MCFRoute executes incrementally to refine the solution with DRC violations adopts the TRG solve flow. When working as an incremental router, MCFRoute selects the routing region automatically and ripups part of local nets close to DRC violation in the region, then routes these nets simultaneously. If the DRC has not been fixed, routing region will be expanded. The number of nets to be ripped up depended on the number of DRC violation count in each routing region. Averagely, 6–20 nets are ripped up. In this experiment, routing results of Encounter v10.10 are fed as input of MCFRoute. As Table VIII shows, the experiments are executed on five industry benchmarks and two ISPD 2014 benchmarks that Encounter could not obtain DRC-clean solutions. Column 1 is the name of each benchmark, columns 2 lists the total DRC violation count of Encounter, while columns 3 lists the total DRC violation count after the refinement by MCFRoute. The last column is the runtime of MCFRoute for refinement. Experimental results show that the number of DRC violations is reduced by 27% after the refinement of MCFRoute. The runtime is also acceptable. Fig. 1 is an example from "matrix_mult_wt." It is obviously that MCFRoute is capable of handling some difficult detailed routing problems which can not be solved by sequential strategy.

IX. CONCLUSION

Based on MCF method, we present a novel concurrent detailed routing algorithm which takes complex design rules into account in this paper. Since all nets are routed simultaneously by the proposed router, net-ordering problem which directly affects the routing quality in traditional rip-up and reroute methods no longer arises. More reasonable resource assignment improves the routing quality. Several useful strategy are introduced to optimize MCF model. And a effective ILP problem solve flow is proposed to reduce the runtime. Experimental results show the effectiveness and potential of the proposed algorithm. In the future, we will carry on our research on the following aspects. First, runtime is still the major issue of this paper, and in the future we plan to study other methodologies to achieve further speed-up. Second, we will try to consider more complex design rules in our model. Thirdly, relax the fixed steiner point(s) by rectilinear steiner minimal tree is also a desirable research point. Furthermore, we plan to incorporate more objective into the proposed model such as timing, double via insertion and design for manufacture.

APPENDIX A
PROOF OF THEOREM 1

Let $F(k) = \sum_{e_i \in E'_{v_{j1}}} f(k, e_i)$ and $F'^{(k)} = \sum_{e_i \in E'_{v_{j2}}} f(k, e_i)$. (12) $\Rightarrow$ (13): if routing solution satisfies constraint (12), there are three valid cases.

Case 1: Neither of two vertices are occupied, i.e., $\varphi(k, v_{j1}) = 0$ and $\varphi(k, v_{j2}) = 0$ $\forall n_k \in N$, which mean $F(k) = 0$ and $F'^{(k)} = 0$, so (13) is true.

Case 2: Either v_{j1} or v_{j2} is occupied. Without loss of generality, we suppose v_{j1} is occupied by net $n_{k'}$, i.e., $\varphi(k', v_{j1}) = 1$ and $\forall n_k \in N, k \neq k', \varphi(k, v_{j1}) = 0, \varphi(k, v_{j2}) = 0$. In this case, $\sum_{k=1, k\neq k'}^{K} (F(k) + F'^{(k)}) + F'^{(k')} = 0$ and $F(k') = 2$, (13) is also true.

Case 3: v_{j1}, v_{j2}, and edge (v_{j1}, v_{j2}) are occupied by the same net, i.e., $\exists k', \varphi(k', v_{j1}) = 1, \varphi(k', v_{j2}) = 1$, and $\forall n_k \in N, k \neq k', \varphi(k, v_{j1}) = 0, \varphi(k, v_{j2}) = 0$. In this case, $\sum_{k=1, k\neq k'}^{K} (F(k) + F'^{(k)}) = 0$, and $F(k') + F'(k') + 2f(k', (v_{j1}, v_{j2})) = 4$. As $f(k', (v_{j1}, v_{j2})) = 1$, (13) is still true. (12) $\Leftarrow$ (13): if constraint (13) is satisfied, the values of $\sum_{k=1}^{K} F(k)$ and $\sum_{k=1}^{K} F'(k)$ fall into four cases.

1) $\sum_{k=1}^{K} F(k) = 0$ and $\sum_{k=1}^{K} F'(k) = 0$. In this case, both vertex v_{j1} and v_{j2} are unused. Constraint (12) is correct obviously.
2) One and only one of the left item in constraint (13) is 1. Based on the connectivity constraint, there is no routing solution belonging to this situation.
3) Either $\sum_{k=1}^{K} F(k)$ or $\sum_{k=1}^{K} F(k)$ takes the value of 2. Without loss of generality, we suppose $\sum_{k=1}^{K} F(k) = 2$ and $\sum_{k=1}^{K} F(k) = 0$. Based on Lemma 2, edge (v_{j1}, v_{j2}) is unused. It can be concluded that $\exists n_{k'} \in N$, $F(n_{k'}) = 2$, $\forall n_k \in N, k \neq k'$, $F(n_k) = 0$ and $\forall n_k \in N, F'(n_k) = 0$. The conclusion means that varphi$(k', v_{j1}) = 1$ and $\varphi(k_2, v_{j2}) = 0$. So constraint (12) is correct.
4) Both $\sum_{k=1}^{K} F(k)$ and $\sum_{k=1}^{K} F'(k)$ take the value of 1. Based on Lemma 2, edge (v_{j1}, v_{j2}) is used, and v_{j1} and v_{j2} is occupied by the same net and only by this net. So constraint (12) is correct. ■

APPENDIX B
PROOF OF THEOREM 4

Given two adjacent vertices v_{j1} and v_{j2}, let e_1 be edge (v_{j1}, v_{j2}) and $\bar{e_1}$ be edge (v_{j2}, v_{j1}). Both $u(e_1)$ and $u(\bar{e_1})$ are 1. Without losing its generality, assuming that there is unit flow of commodity $n_{k^\star}$ streaming from v_{j1} to v_{j2} by edge e_1.

1) *No Short*: Connectivity constraint (1) on vertex v_{j2} ensures that the commodity will stream out from one edge in $E_{v_{j2},\text{out}}$. Vertex capacity constraint (3) on vertex v_{j2} ensures that other commodity can not go through vertex v_{j2}. So if we do not add capacity constraint for edge e_1 and $\bar{e_1}$, there will not be any shorts.
2) *No Detour*: The only remaining problem we need to pay attention to is whether commodity $n_{k^\star}$ makes a circle, i.e., this commodity streams out from v_{j2} by edge $\bar{e_1}$. Absolutely, this problem will not occur. As the solution has a detour, the objective function (4) will reject it and select a better one. ■

REFERENCES

[1] M. Burstein and R. Pelavin, "Hierarchical wire routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 2, no. 4, pp. 223–234, Oct. 1983.

[2] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, "Track assignment: A desirable intermediate step between global routing and detailed routing," in *Proc. ICCAD*, San Jose, CA, USA, 2002, pp. 59–66.
[3] C. J. Alpert *et al.*, "What makes a design difficult to route," in *Proc. ISPD*, San Jose, CA, USA, 2010, pp. 7–12.
[4] D. Abercrombie, P. Elakkumanan, and L. Liebmann, "Restrictive design rules and their impact on 22 nm design and physical verification," in *Proc. Electron. Design Process. Symp.*, vol. 143. 2009.
[5] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Trans. Electron. Comput.*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961.
[6] K. Mikami and K. Tabuchi, "A computer program for optimal routing of printed circuit conductors," in *Proc. IFIP Congr.*, vol. 2. 1968, pp. 1475–1478.
[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.
[8] S.-Q. Zheng, J. S. Lim, and S. S. Iyengar, "Finding obstacle-avoiding shortest paths using implicit connection graphs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 1, pp. 103–110, Jan. 1996.
[9] J. Cong, J. Fang, and K.-Y. Khoo, "An implicit connection graph maze routing algorithm for ECO routing," in *Proc. ICCAD*, San Jose, CA, USA, 1999, pp. 163–167.
[10] Y. Zhang and C. Chu, "RegularRoute: An efficient detailed router with regular routing patterns," in *Proc. ISPD*, 2011, pp. 146–151.
[11] Y. Zhang and C. Chu, "GDRouter: Interleaved global routing and detailed routing for ultimate routability," in *Proc. DAC*, San Francisco, CA, USA, 2012, pp. 597–602.
[12] Y. Xu, Y. Zhang, and C. Chu, "FastRoute 4.0: Global router with efficient via minimization," in *Proc. ASP-DAC*, Yokohama, Japan, 2009, pp. 576–581.
[13] M. Gester *et al.*, "Algorithms and data structures for fast and good vlsi routing," in *Proc. DAC*, San Francisco, CA, USA, 2012, pp. 459–464.
[14] M. Ahrens *et al.*, "Detailed routing algorithms for advanced technology nodes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 4, pp. 563–576, Apr. 2015.
[15] (2017). *Cadence SOC Encounter*. [Online]. Available: http://www.cadence.com
[16] E. Shragowitz and S. Keel, "A global router based on a multicommodity flow model," *Integr. VLSI J.*, vol. 5, no. 1, pp. 3–16, 1987.
[17] M. M. Ozdal, "Detailed-routing algorithms for dense pin clusters in integrated circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 3, pp. 340–349, Mar. 2009.
[18] B. Yu, D. Liu, S. Chowdhury, and D. Z. Pan, "TILA: Timing-driven incremental layer assignment," in *Proc. ICCAD*, Austin, TX, USA, 2015, pp. 110–117.
[19] D. Liu, B. Yu, S. Chowdhury, and D. Z. Pan, "Incremental layer assignment for critical path timing," in *Proc. DAC*, Austin, TX, USA, 2016, pp. 1–6.
[20] R. C. Carden, J. Li, and C.-K. Cheng, "A global router with a theoretical bound on the optimal solution," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 2, pp. 208–216, Feb. 1996.
[21] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," in *Proc. ISPD*, San Diego, CA, USA, 2000, pp. 19–25.
[22] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 5, pp. 622–632, May 2001.
[23] X. Jia *et al.*, "MCFRoute: A detailed router based on multi-commodity flow method," in *Proc. ICCAD*, San Jose, CA, USA, 2014, pp. 397–404.
[24] K. Han, A. B. Kahng, and H. Lee, "Evaluation of BEOL design rule impacts using an optimal ILP-based detailed router," in *Proc. DAC*, San Francisco, CA, USA, 2015, pp. 1–6.
[25] Gurobi Optimization Inc. (2017). *Gurobi Optimizer Reference Manual*. [Online]. Available: http://www.gurobi.com
[26] (2017). *LEF/DEF Language Reference*. [Online]. Available: ftp://ftp.sitsemi.ru/pub/Cadence/lefdefref.pdf
[27] P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.
[28] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*. Hoboken, NJ, USA: Wiley, 2011.
[29] W. S. Jewell, "Multi-commodity network solutions," Dept. Oper. Res. Center, Univ. California Berkeley, Berkeley, CA, USA, Tech. Rep. ORC-66-23, 1966.
[30] D.-S. Choi and I.-C. Choi, "On the effectiveness of the linear programming relaxation of the 0-1 multi-commodity minimum cost network flow problem," in *Computing and Combinatorics*. Heidelberg, Germany: Springer, 2006, pp. 517–526.
[31] D. B. C. Faneyte, F. C. R. Spieksma, and G. J. Woeginger, "A branch-and-price algorithm for a hierarchical crew scheduling problem," *Naval Res. Logistics*, vol. 49, no. 8, pp. 743–759, 2002.
[32] S. Sapatnekar *et al.*, "Reinventing EDA with manycore processors," in *Proc. DAC*, Anaheim, CA, USA, 2008, pp. 126–127.
[33] P. Banerjee, M. H. Jones, and J. S. Sargent, "Parallel simulated annealing algorithms for cell placement on hypercube multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 1, no. 1, pp. 91–106, Jan. 1990.
[34] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "Multi-threaded collision-aware global routing with bounded-length maze routing," in *Proc. DAC*, Anaheim, CA, USA, 2010, pp. 200–205.
[35] Y. Shintani, M. Inagi, S. Nagayama, and S. Wakabayashi, "A multithreaded parallel global routing method with overlapped routing regions," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Los Alamitos, CA, USA, 2013, pp. 591–597.
[36] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[37] (2005). *ISPD 2005 Placement Contest*. [Online]. Available: http://archive.sigda.org/ispd2005/contest.htm
[38] V. Yutsis, I. S. Bustany, D. Chinnery, J. R. Shinnerl, and W.-H. Liu, "ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement," in *Proc. ISPD*, Petaluma, CA, USA, 2014, pp. 161–168.

Xiaotao Jia received the B.S. degree in mathematics from Beijing Jiao Tong University, Beijing, China, in 2011, and the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, in 2016.

His current research interests include very large scale integrated circuits physical design and optimization algorithms with a focus on detailed routing.

Yici Cai (M'04–SM'10) received the B.S. degree in electronic engineering and the M.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 1983 and 1986, respectively, and the Ph.D. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2007.

She has been a Professor with the Department of Computer Science and Technology, Tsinghua University. Her current research interests include design automation for very large scale integration integrated circuits algorithms and theory, power/ground distribution network analysis and optimization, high performance clock synthesis, and low power physical design.

Qiang Zhou (M'04–SM'10) received the B.S. degree in computer science and technology from the University of Science and Technology of China, Hefei, China, the M.S. degree in computer science and technology from Tsinghua University, Beijing, China, and the Ph.D. degree in control theory and control engineering from the Chinese University of Mining and Technology, Beijing, in 1983, 1986, and 2002, respectively.

He has been a Professor with the Department of Computer Science and Technology, Tsinghua University. His current research interests include very large scale integration layout theory and algorithms.

Bei Yu (S'11–M'14) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong. He has served in the editorial boards of *Integration*, the *VLSI Journal* and *IET Cyber-Physical Systems: Theory & Applications*.

Dr. Yu was a recipient of three Best Paper Awards at SPIE Advanced Lithography Conference 2016, International Conference on Computer Aided Design (ICCAD) 2013, and Asia and South Pacific Design Automation Conference (ASPDAC) 2012, three other Best Paper Award Nominations at Design Automation Conference (DAC) 2014, ASPDAC 2013, ICCAD 2011, and three ICCAD Contest Awards in 2015, 2013, and 2012, the European Design and Automation Association Outstanding Dissertation Award in 2014, the Chinese Government Award for Outstanding Students Abroad in 2014, the SPIE Scholarship in 2013, and the IBM Ph.D. Scholarship in 2012.