Congestion Analysis for Global Routing via Integer Programming

Hamid Shojaei, Azadeh Davoodi, and Jeffrey T. Linderoth*
Department of Electrical and Computer Engineering
*Department of Industrial and Systems Engineering
University of Wisconsin at Madison, USA
Email: {shojaei, adavoodi, linderoth}@wisc.edu

***Abstract*— This work presents a fast and flexible framework for congestion analysis at the global routing stage. It captures various factors that contribute to congestion in modern designs. The framework is a practical realization of a proposed parameterized integer programming formulation. The formulation minimizes overflow *inside* a set of regions covering the layout which is defined by an input resolution parameter. A resolution lower than the global routing grid-graph creates regions that are larger in size than the global-cells. The maximum resolution case simplifies the formulation to minimizing the total overflow which has been traditionally used as a metric to evaluate routability. A novel contribution of this work is to demonstrate that for a small analysis time budget, regional minimization of overflow with a lower resolution allows a more accurate identification of the routing congestion hotspot locations, compared to minimizing the total overflow. It allows generating a more accurate congestion heatmap. The other contributions include several new ideas for a practical realization of the formulation for industry-sized benchmark instances some of which are also improvements to existing global routing procedures. This work also describes coalesCgrip, a simpler variation of our framework which was used to evaluate the ISPD 2011 contest.**

I. Introduction

The high volume and complexity of cells and interconnect structures in modern designs are causing serious challenges to routability. Rapid congestion analysis (i.e., identification of the congestion hotspots on the layout) is becoming crucial to fix the routability problem at the early stages of the design, for example during placement [7] and in conjunction with global routing [8], [10]. In modern designs, several new factors contribute to routing congestion including significantly-different wire size and spacing among the metal layers, sizes of inter-layer vias, various forms of routing blockages (e.g., reserved for power-grid, clock network, or IP blocks in an SoC), local congestion due to pin density and wiring inside a global-cell, and virtual pins located at the higher metal layers. However, none of the past estimation techniques such as [2], [11], [14], [15] capture these new sources of congestion comprehensively.

Many of the above factors can be accurately modeled with a flexible model of global routing in which the routing grid-graph has varying edge capacities, and the net terminals are mapped to any vertex in the (3-dimensional) grid-graph. The reader is referred to [12] that explains the importance of considering these factors for congestion analysis in modern designs and the work [13] that explains modeling of some of these factors in the recently-released industrial benchmark instances of the ISPD 2011 contest [1].

Using this new model allows for more accurate routability consideration by utilizing a flexible and fast global routing model for congestion analysis. However, the majority of the existing academic global routers are not designed to effectively and comprehensively handle this model. For example, just considering the layer assignment step, the internal procedures of [3], [4], [5], [9], [18] all require the net terminals to be on the first metal layer. Moreover, the majority of the existing academic global routers are too slow for congestion analysis and the analysis is typically required to take only a fraction of a typical global routing runtime. For example, the routing framework of [17] allows capturing some or all the new congestion factors, however it has a relatively high execution runtime.

Furthermore, the goal of a congestion analysis tool is to quickly identify and rank the congestion hotspots on the layout. This is because accurate prediction of the hotspot locations allows a routability-driven placer to effectively spread the cells outside the congested regions [8], [10]. The procedure, when applied iteratively, allows creating shorter routes to fix routability, which may translate into less buffer usage and improvement in timing and power. However, the accuracy of predicting the congestion hotspots, especially within a small time-budget is limited if it is done based on traditional minimization of total routing overflow which also results in creating long routes when detouring the nets outside the congested regions.

In this work, a fast framework is presented which aims to accurately predict the congestion hotspots. The framework utilizes the flexible model of global routing that captures many necessary modern design features. For each net, the user can also specify the degree to which it is routed in a *scenic* manner by restricting the route to lie within a specified factor of its bounding box. This feature is employed in the framework to allow identifying the hotspots that are most suitable to be removed by displacing the cells using a placer, as opposed to creating long detoured routes using a global router. In this work, the reference for "actual" congestion is the one obtained from the same framework but with a significantly longer runtime budget.

To accurately identify and rank the congested regions of the layout in a small time-budget, this work proposes minimizing overflow inside the regions on the layout specified by an input *resolution* parameter. In the maximum resolution case, the regions are simply the edges of the global routing grid-graph. Lowering the resolution creates regions which can be much larger in size than those defined by the global routing grid-graph. Region-based minimization of overflow with a lower resolution is not equivalent to working with a coarsened model for global routing and overflow is always defined with respect to the (actual) global routing grid granularity, regardless of the size of the regions. A novel contribution of this work is to show that for a small analysis time-budget, region-based minimization of overflow with lower resolution provides a more accurate identification and ranking of congestion hotspots compared to minimizing the total overflow.

The framework relies on a proposed Integer Programming (IP) formulation which, in addition to modeling the overflow of individual edges on the global routing grid, contains new variables to model the total overflow and maximum overflow for each region induced by the resolution parameter. For the highest resolution, the formulation simplifies to the one in [16] to minimize the total overflow.

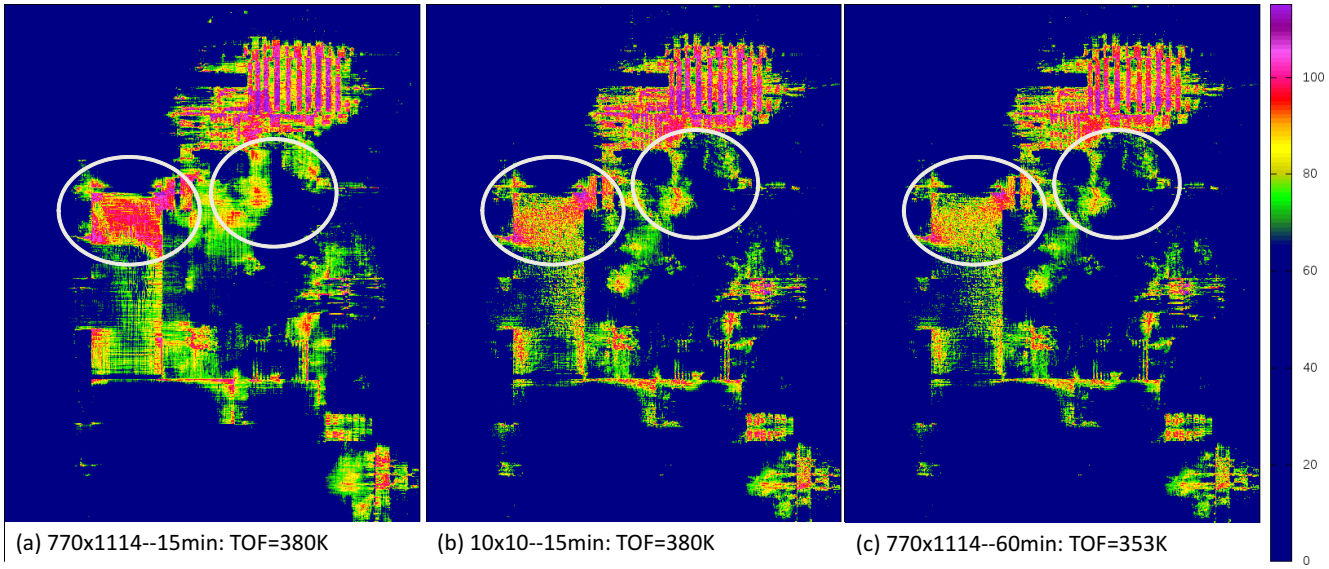

Fig. 1. Different routing congestion heatmaps corresponding to the same placement of benchmark instance `superblue2`, obtained from the tool Ripple for (a) minimizing the total overflow on the global routing grid in 15 minutes, (b) regional minimization of overflow on a coarser grid in 15 minutes, and (c) minimizing the total overflow on the global routing grid in 60 minutes (reference case).

Since solving the IP formulation to optimality is impractical for realistically-sized problem instances, this work also proposes several new ideas for a practical implementation of the framework to obtain a high quality approximate solution to the formulation. The approximation relies on a method which is referred to as "reduced linear programming" for shrinking the original IP formulation without seriously degrading its solution quality, as well as a novel way to integrate the solution of the formulation with the traditional rip-up and reroute procedures used by many global routing frameworks. Furthermore, this work speeds up the conventional rip-up and reroute global routing procedures by performing multiple rip-ups and a single, simultaneous rerouting of "equivalent" nets. It also provides a congestion-aware layer assignment procedure to account for varying wire size, spacing, blockages, and pins at different metal layers.

In the computational experiments, we verify the claim of more accurate identification of congestion hotspots for lower resolutions within a small time-budget. In addition we also evaluate the solution quality obtainable by our framework when minimizing the total overflow. For the latter case, comparison is made with coalesCgrip, a simpler variation of our framework which was used as the reference router to judge the ISPD 2011 contest on routability-driven placement and is also discussed in this work.

In summary this work makes the following contributions:

- An IP formulation of the congestion analysis problem that aims to quickly find the locations of the congestion hotspots by using a user-specified resolution parameter;
- Several methods to achieve a practical realization of the IP for a small time-budget, including: 1) an approximation method to shrink the IP formulation and a novel way to integrate its solution with a standard rip-up and reroute procedure; 2) a multiple rip-up, single reroute procedure which provides speedup improvements; and 3) a layer assignment procedure to handle varying wire size, spacing, blockage and pin locations.

In the remainder of this work, we start with a motivational example in Section II before presenting our formulation and congestion analysis framework in Sections III and IV, respectively. We briefly discuss coalesCgrip in Section V and present simulation results in Section VI, followed by conclusions in Section VII.

II. Motivational Example

Consider Figure 1 which shows three different congestion heatmaps of the `superblue2` benchmark instance [13]. This is corresponding to the (same) placement instance generated by the tool Ripple which was obtained from the ISPD 2011 contest web site [1]. Each 2D projected heatmap reflects the aggregated edge utilizations of different metal layers at each point. It is obtained by modifying the script provided on the contest web site. Any utilization less than 70% is shown in dark blue. In the remaining regions, the lighter colors correspond to a lower utilization. All the heatmaps are created by our proposed router, running in different configurations. The difference is only due to the used runtime budget and *resolution* parameter in the optimization. In all the cases, the router is required to route each net within 110% of its original bounding box. This strategy allows a more accurate identification of the congested regions that are most useful to be fixed by a routability-driven placer [10] because otherwise they may never get fixed or may require creating long detours. The total overflow (TOF) of each heatmap is also reported.

The reference is heatmap (c) which is generated by running the router for 60 minutes (after which the heatmap doesn't change drastically) and minimizing the total overflow. Heatmap (a) is obtained by running the router to minimize the total overflow but for a shorter runtime budget of 15 minutes. Heatmap (a) has mismatch with heatmap (c) in many cases in the figure, including the circled areas. This is partially because when the objective is minimizing the total overflow, there is not sufficient time to reduce the overflow in *all* the locations. The router prioritizes different regions based on its rip-up and re-route procedure in order to obtain the maximum reduction in the total overflow and may not have the chance to optimize some regions. However, this is not an indication of difficulty of routability since the same router could fix those regions in a longer runtime.

Heatmap (b) is generated by running the router for 15 minutes when generating the regions using a 10x10 grid. It matches better with heatmap (c). Due to more accurate matching of heatmap (b) with the reference one, it is more useful to a routability-driven placer which displaces cells based on the locations and ranking of the congested hotspots for a 15 minutes time budget. This is despite similar overflow values in heatmaps (a) and (b).

III. Problem Formulation

The example presented in Section II suggests that a congestion analysis tool cannot focus solely on minimizing total overflow. Our analysis tool breaks the routing area into regions and uses an integer programming model that considers total overflow and maximum overflow found within each region.

A. Region Definition

Most components of our congestion analysis tool operate on a 2D-projection of the global routing instance, whose resultant grid-graph is $G = (V, E)$. For notational purposes, each vertex $v = (v_x, v_y) \in V$ has coordinates $v_x \in \{1, \dots X_G\}$ and $v_y \in \{1, \dots Y_G\}$. Regions are contiguous, rectangular areas of the grid graph whose non-overlapping edge sets partition the edge set E. The regions form a set $\mathcal{R}$, and the number of regions is controlled by two input *resolution* parameters r_x and r_y. The parameters r_x and r_y specify the length and height of each rectangle, respectively, so that the number of regions is $|\mathcal{R}| = r_x \times r_y$. Edges on the boundary of two regions are assigned to the region whose position is closest to the left and bottom of G. (Except for the edges on the top and right corners of the grid, which will be mapped to their closest regions).

B. Integer Programming Formulation

The regions $\mathcal{R}$ defined by resolution parameters r_x and r_y are used in an integer programming formulation for congestion analysis. The formulation is given a set of (multi-terminal) nets $\mathcal{N} = \{T_1, T_2, \dots, T_N\}$ (with $T_i \subset V$) and edge capacities c_e $\forall e \in E$. Denote by $\mathcal{T}_i$ the set of all possible candidate routes, or feasible Steiner Trees, for net T_i. The user can restrict how *scenic* each net T_i is routed by providing a scaling parameter η_i, and $t \in \mathcal{T}_i$ only if all edges of t are contained in an η_i-scaled bounding box of the terminals of T_i. The parameter $a_{te} = 1$ if tree t contains edge $e \in E$, and $a_{te} = 0$ otherwise. Let $\mathcal{R}$ be the set of regions created by resolution parameters r_x and r_y, and for region $r \in \mathcal{R}$, let $E(r) \subseteq E$ is the set of edges that belong to r. Our congestion analysis model will be able to minimize the maximum total-overflow inside any region, or minimize the sum of the maximum-overflows of all the regions. (Note, this is *not* equivalent to global routing on a coarser grid-graph because overflow is always defined with respect to the (actual) global routing grid-graph.) Define the binary decision variable x_{it} that is equal to 1 if and only if net T_i is routed with tree $t \in \mathcal{T}_i$. An integer program for congestion analysis can be written as:

$$\min_{x,o,s,\tau} (1-\kappa) \sum_{r \in \mathcal{R}} s_r + \kappa\tau \qquad \text{(IP-CA)}$$

$$\begin{cases} \sum_{t \in \mathcal{T}_i} x_{it} = 1 & \forall i \in \mathcal{N} \\ \sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} a_{te} x_{it} \le c_e + o_e & \forall e \in E \\ \sum_{e \in E(r)} o_e \le \tau & \forall r \in \mathcal{R} \\ s_r \ge o_e & \forall r \in \mathcal{R}, \ \ \forall e \in E(r) \\ x_{it} = \{0,1\} & \forall i = 1, \dots, N, \forall t \in \mathcal{T}_i \\ o_e \ge 0 & \forall e \in E \\ \tau \ge 0 & \\ s_r \ge 0 & \forall r \in R. \end{cases}$$

The first set of inequalities enforce selection of exactly one route for net i from the set of its candidate routes $\mathcal{T}_i$. In the second set of inequalities, o_e is an integer variable that measures the overflow of the normalized capacity c_e on edge e. (The normalized edge capacity c_e is the maximum number of routing tracks that can pass e without causing overflow. Defining the value of c_e to account for factors that cause congestion will be explained in the next section.) The third set of inequalities defines a variable τ that takes on the value of the maximum total-overflow in any region. The fourth set of inequalities defines variables s_r for each region $r \in \mathcal{R}$ to be the maximum individual overflow of any edge $e \in E(r)$. The objective function is a convex combination of the maximum total overflow variable τ and the sum of the individual maximum-overflow variables s_r.

For the special case of maximum resolution ($|\mathcal{R}| = |E|$), the variable s_r is simply the overflow of each edge, and τ is the maximum overflow of all the edges. In this case, by setting $\kappa = 0$, a formulation similar to that of GRIP [16] is obtained that minimizes the total overflow. For $\kappa = 1$ the maximum overflow is minimized. The user has the option to decide the tradeoff between minimizing a combination of these overflow-based metrics, as well as setting the resolution parameters r_x and r_y that control the granularity of the maximum overflow calculations. To obtain the heatmaps in Figure 1, the value $\kappa = 0$ was used in all cases. The heatmaps (a) and (b) are generated at maximum resolution $|\mathcal{R}| = |E|$. Consequently, these two cases simplify to minimizing the total overflow. For heatmaps (c), the parameters $r_x = r_y = 10$ were used. Please note, in the remainder of this work we assume $\kappa = 0$.

The formulation (IP-CA) is a general formulation that may take an arbitrary (and possibly overlapping) definition of the regions $\mathcal{R}$. However, in our implementation, the size of each region is equivalent and the regions are non-overlapping. In this case, to minimize the objective expression, the solver naturally spends a smaller effort to obtain a low s_r for a region r that is not a hotspot and spends a higher effort in minimizing s_r in the difficult regions.

IV. Congestion Analysis Framework

Solving the formulation (IP-CA) to optimality is impractical for realistic instances within the time budget allowed for analysis. Attempts to accurately solve similar-sized formulations either require a significant runtime (e.g., many days in GRIP [16]) or a large number of parallel CPUs (e.g., a few hundred in PGRIP [17]). In this section, we present a practical implementation of (IP-CA) suitable for running in a tight runtime budget.

A. Overview

Our congestion analysis tool relies on two key components. The first component drives the algorithm, starting with a 2D-projection of the routing instance and the creation of an initial 2D-solution. A rip-up and reroute procedure is iteratively applied until a no-overflow solution is found, or a time limit is reached, or the improvement in overflow is less than 5% in 3 consecutive iteration. A congestion-aware layer assignment procedure is performed in the last step. The second key component of our tool, used in both the initialization and rip-up and reroute steps, is the use of a reduced-size version of the IP formulation (IP-CA).

Rip-up and reroute is a ubiquitous strategy employed by many other routing tools [3], [4], [5], [9], [18]. This work suggests a new net ordering procedure and methods that improve the speed during rip-up and reroute. Our layer assignment also accounts for irregular capacity adjustments caused by varying wire size, spacing and routing blockages, and net-terminal locations at the higher metal layers. The integration of approximate integer programming with rip-up and reroute to obtain significant improvement in solution quality in a tight time-budget is a unique contribution of our framework.

Figure 2 gives a graphical overview of the key components of our framework. To create a 2D-projected grid-graph, for each edge in the 3D grid-graph, we first compute a normalized capacity which is

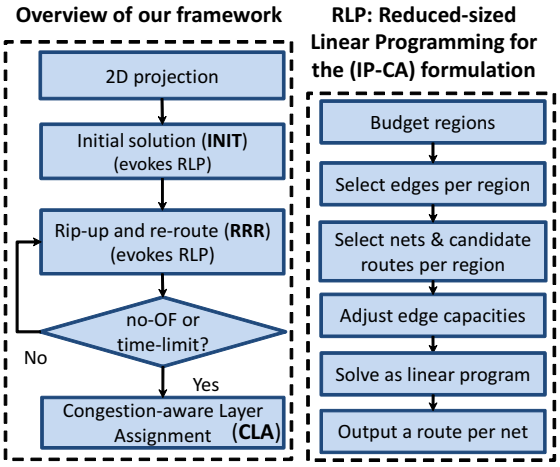

Fig. 2. Congestion analysis framework

defined as the maximum number of routing tracks that can pass from it (i.e., c_e in (IP-CA)). This is done by dividing the routing capacity of the tile boundary corresponding to the edge by the summation of the wire size and spacing for that layer. For the definition of routing capacity please refer to [1]. In the 2D projection, the c_e of each edge e is the summation of the normalized capacities of the corresponding edges in the 3D graph over different metal layers. Similarly, in this section, the reference to overflow on an edge after 2D projection is the number of routing tracks exceeding the normalized edge capacity.

Next, an initial solution is generated. The initialization uses our procedure for Reduced-sized Linear Programming (RLP) that can quickly generate useful estimates of the utilizations of the edges in the 2D grid-graph. Next, rip-up and reroute (RRR) is iteratively performed that again requires the use of the RLP procedure. RRR is iterated until a solution with 0 overflow is found, or the analysis time limit is reached, or if the overflow improvement in 3 consecutive iterations remains less than 5%. (All these stopping criteria are adjustable by the user and these values are set by default.) The final step of our algorithm is congestion-aware layer assignment (CLA).

B. RLP: Reduced-sized Linear Programming

Reduced-size linear programming (RLP) works by creating a smaller-sized formulation of (IP-CA) that contains only a subset of *critical* edges $D \subset E$ and critical nets $\mathcal{M} \subset \mathcal{N}$. For each critical net $T_i \in \mathcal{M}$ a small subset of its candidate routes $\mathcal{S}_i \subseteq \mathcal{T}_i$ are used. The formulation (IP-CA) is then restricted to the critical edges D, critical nets $\mathcal{M}$, and reduced candidate routes $\mathcal{S}_i$, $\forall T_i \in \mathcal{M}$. The number of regions defined by the resolution parameters is unchanged in RLP. The normalized edge capacities are also adjusted to account for the impact of the remaining non-critical nets.

The output of the RLP step is a selected route for each critical net and a measure for relative utilization of the critical edges which will be used to setup their edge weights during RRR, and is defined based on the dual variable values of the reduced-sized linear program. Specifically, we use the optimal dual variables π_e^* associated with the second set of constraints

$$\sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} a_{te} x_{it} \leq c_e + o_e \quad \forall e \in E.$$

The important insight is that the duality theory of linear programming states that π_e^* is the rate of change of the optimal objective value of the linear programming relaxation of (IP-CA) with respect to a unit change in edge capacity c_e. Thus, the value π_e^* encodes significant *edge-specific* information about the importance of edge e with respect to the congestion-oriented objective of (IP-CA). This is exactly the information that is utilized in the rip-up and reroute procedures.

To obtain a practical analysis tool that worked with a runtime-budget of a few minutes, each instance of RLP needed to be solved in a matter of seconds. Computational experience suggested that the solver can generate a solution in a few seconds when $|D| = 5000$, $|\mathcal{M}| = 1000$, and $|\mathcal{S}_i| \leq 10$ $\forall i \in N$ to form the reduced (IP-CA) formulation. Interestingly, this observation is true regardless of the benchmark instance.

When creating the reduced (IP-CA) formulation, we are provided an input route for each net T_i. This route is obtained from maze routing in the INIT step or from the solution of the previous iteration in RRR. The input route for each net is used to identify the critical edges D. The critical edges are used to identify the critical nets $\mathcal{M}$.

Figure 2 illustrates the steps in one call to the RLP procedure. For a given resolution that results in $|\mathcal{R}| = r_x \times r_y$ regions, we start by allocating the $|D| = 5000$ edge budget to different regions. The number of edges k_r in region r is proportional to the total overflow s_r inside a region, as computed from the input solution. An important feature of this assignment is that edges with a higher estimated overflow will be represented with more variables in the reduced formulation. Note that $\sum_{r \in R} k_r = 5000$.

To select the specific edges for each region r, we select the k_r edges with the highest estimated overflow in the region. The critical edge set D is the union of all selected edges. The next step of RLP is to compute the critical nets $\mathcal{M}$. For each net $T_i \in \mathcal{N}$, if its input route contains a critical edge $e \in D$, then T_i is a candidate to become a critical net. Note that the input route for a net may contain multiple edges $e \in D$. Each candidate critical net is sorted according to the total overflow induced by its input route. The $|\mathcal{M}| = 1000$ with the highest overflows form the set of critical nets. Candidate routes $\mathcal{S}_i$ for each critical net are either selected using pattern/maze routing or by selecting one of its candidate routes, identified in previous iterations of rip-up and reroute. To keep the number of candidate routes small, for each net, we utilize at most $|\mathcal{S}_i| = 10$ candidate routes taken from the latest RRR iterations. Details are given subsequently in Sections IV-C and IV-D. Besides from the critical nets, the same procedure is followed to maintain a list of candidate routes for each non-critical net as it may become critical in future RRR iterations.

Before solving the reduced formulation, the normalized edge capacities are adjusted. All nets not identified as critical are fixed to use their input routes, and the normalized edge capacities are reduced by subtracting the number of fixed nets passing from each edge.

The reduced formulation is solved as a linear program by relaxing the integrality requirements on the variables x. To create a routing solution and to generate an approximate integer-valued solution for (IP-CA), each critical net $T_i \in \mathcal{M}$ is routed with the candidate route $t \in \mathcal{S}_i$ whose associated linear programming solution value x_{it}^* is largest: $i^* \in \arg\max_{t \in \mathcal{S}_i} x_{it}^*$ $\forall T_i \in \mathcal{M}$. Non-critical nets are routed in a greedy fashion. Specifically, each non-critical net $T_i \in \mathcal{N} \setminus \mathcal{M}$ is routed by using the previously generated route for T_i that induces the least amount of additional overflow in the solution to that point. This routing solution is used in the subsequent RRR iteration.

Furthermore, as previously stated, the RLP also provides a measure of relative utilization for each critical edge that will be used to compute edge weights during the RRR step. Specifically, for each critical edge $e \in D$, its relative utilization is taken to be $\frac{\pi_e^*}{c_e}$, where π_e^* is the optimal dual value of its corresponding edge capacity constraint in the reduced-sized LP. Further discussion of the weights is delayed to the discussion of RRR in Section IV-D.

In practice, even though the budget of $|D| = 5000$ critical edges is significantly smaller than the total number of edges (e.g., a few hundred thousand in the ISPD 2011 benchmark instances), our analysis tool can still provide a good estimate of the regions that contain hotspots. We attribute this good performance to the the fast runtime of RLP that allows its iterative use within RRR. Note also that each time RLP is invoked within RRR, the allocation of the 5000 critical edges to different regions changes based on the edge utilizations obtained from the previous iteration of RRR. We have observed that the allocation of the edges to regions of truly highest congestion gradually increases during the algorithm.

C. INIT: Initial Solution Generation

To generate an initial solution, we first apply maze routing for each two-terminal subnet obtained by decomposing the nets $\mathcal{N}$ based on minimum spanning trees similar to [4], [9]. Maze routing is done by applying the A^* algorithm to find a smallest-weight path between each pair of terminals. The edge weight used reflects the *current* utilization of nets that are routed so far. To enforce that net T_i is routed within its η_i-scaled bounding box, a large weight is given to edges outside this box. The maze routes of the subnets of a multi-terminal net are merged to form its first candidate route.

The initial candidate route set for each net is augmented by adding variations of the route found by pattern routing. Specifically, two candidate routes are added by considering only L-shaped routes for each bend subnet of a multi-terminal net. One candidate route is obtained by merging the top-right L-shaped routes of the subnets. The other is obtained by merging the right-top L-shaped routes. Two additional candidate routes are added using Z-shaped routes for the subnets. One is by merging right-top-right Z-shaped routes of the subnets. The other is by merging the top-right-top Z-shaped routes.

Overall, we obtain five candidate routes for each net from maze routing and pattern routing. We then apply the RLP procedure where we assume the input route for each net is its first candidate route from maze routing (to determine the critical edges and nets that pass from the high-overflow edges). RLP generates a route for each net which will be the initial solution given to RRR. In our experience, this step requires no more than two minutes to complete for any of the ISPD 2011 benchmark instances using a routing box that is $\eta_i = 10\%$ larger than the original bounding box for each net $T_i \in \mathcal{N}$.

D. RRR: Rip-up and Re-route

At each iteration of RRR an input routing solution is rerouted to reduce overflow. Figure 3 gives an overview of the steps in one RRR iteration. The first step uses RLP to produce a new routing solution and a measure used to setup edge weights during rip-up and reroute. Compared to other RRR procedures like [3], [4], [9], [18] one of our contributions is effective integration with RLP. Our experience is that integrating RLP with RRR allows for a significant overflow reduction in a short runtime, even though fewer RRR iterations are done than other methods because of the overhead incurred by RLP.

After performing RLP to produce an updated routing solution and edge utilization measures, each multi-terminal net is decomposed into two-terminal subnets based on its minimum spanning tree. For each subnet a route is constructed by identifying the path connecting its two terminals on the route of its multi-terminal net. For each subnet, we compute the total overflow divided by the number of edges that have overflow on its route. Subsets are ordered in ascending order of this measure, so that easy-to-fix nets with overflow are rerouted first.

Re-routing of the subnets begins by first computing weights for each edge. Edge e is given weight $w_e = 1 + p_e \times h_e$, where p_e and h_e are the *penalty* and *history* terms, respectively.

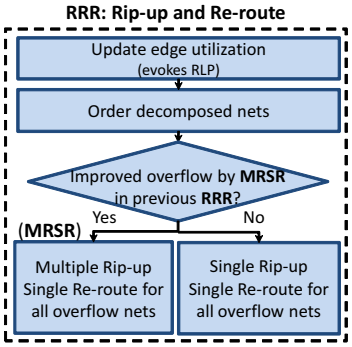

Fig. 3. Overview of rip-up and reroute

For a critical edge $e \in D$, the penalty term is computed as p_e=pow(5, $\frac{-\pi_e^*}{c_e}$), where c_e is the edge normalized capacity and π_e^* is the optimal dual value of the capacity constraint for edge e in the reduced-sized LP, as discussed in Section IV-B. For a non-critical edge $e \in E \setminus D$, if it has overflow, then p_e=pow(5, $\frac{u_e - c_e}{c_e}$) and otherwise, $p_e = \frac{u_e}{c_e}$. In the expressions of the non-critical edges, the term u_e indicates the edge utilization of the current routing solution. The history term h_e in the expression for w_e is initially 1 and incremented by a constant ($h_{inc} = 1.2$) at the beginning of each round of RRR if edge e has overflow. This edge weighting scheme is similar to that of [9], but our edge weight penalty function depends on the output of RLP (either the current routing edge utilizations u_e for non-critical edges or the dual values π_e^* for critical edges). Furthermore, our history term h_e is also updated *within* an RRR iteration, as the nets are traversed.

Using these edge weights, nets that contain overflow may be rerouted either by a multiple rip-up, single reroute (MRSR) process, or a single rip-up, single reroute (SRSR) procedure. For the RRR iterations, initially MRSR is performed at each iteration, until no overflow improvement is detected. After that SRSR is performed in the remaining iterations. (See Figure 3.) In SRSR, each net is routed by first freeing the normalized edge capacity used by the subroute generated by RLP. A standard weighted shortest path algorithm is used to connect the subnet to the remainder of the route.

Multiple Rip-up Single Re-Route (MRSR): We made the observation that decomposed subnets of different nets often had the same start and end terminals. The MRSR procedure is a novel strategy to take advantage of this fact by simultaneously removing multiple routes and routing them together. To apply MRSR after RLP, we first identify all equivalent subnets that contain overflow. Equivalent subnets for each terminal pair are aggregated into groups of size at most c_{avg}, the average capacity of the edges in the projected 2D grid.

For example, in Figure 4(a), 6 subnets pass over the overflow edges. The edges that have overflow are marked in the example. Here, subnets n_1 to n_4 are equivalent since they all connect the terminals p_1 and p_2. For $c_{avg} = 3$, we assign n_1, n_2, n_3 to group G_1 and thus n_4 will be assigned in a new group G_2. Similarly, subnets n_5 and n_6 are both connecting p_1 and p_3 and are assigned to group G_3. (See Figure 4(b).) After the equivalent subnets are aggregated into groups, the subnet-list is traversed in the previously explained order. If a subnet in a group is to be rerouted, all subroutes in the group are simultaneously removed, updating the available normalized edge capacities accordingly. A single reroute procedure is applied to route *all* the equivalent subnets. The edge utilization on the aggregate route is incremented by the number of equivalent subnets in the group.

Figure 4(c), shows 3 reroute steps for groups G_1, G_2 and G_3. The routes are shown with different thicknesses in Figure 4(c) to represent

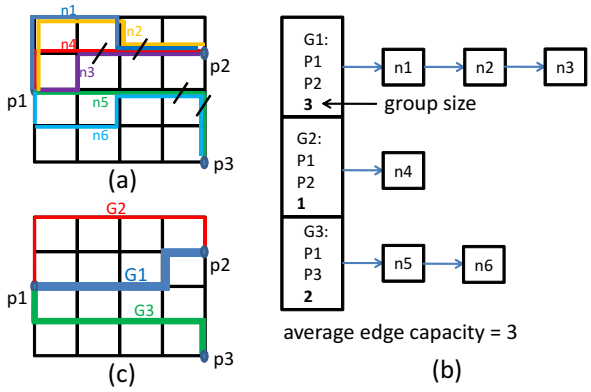

Fig. 4. Multiple Rip-up Single Re-route (a) subnets before MRSR, (b) subnets after MRSR, (c) equivalent subnet grouping

their varying contributions to the edge utilizations. In this example, overall we rip-up 6 subnets but reroute only 3 times. The re-routing process is a major bottleneck for RRR, so this strategy allows for significant speedup, especially for the larger benchmarks. We note that the route of the (undecomposed) nets that share the same subnet may still remain significantly different. Further, we apply MRSR in the first iterations of RRR, until we observe that the previous RRR iteration did not result in overflow improvement. We then switch to single rip-up single reroute for the remainder of the RRR iterations.

E. CLA: Congestion-Aware Layer Assignment

The CLA step converts the generated 2D route of each net into a 3D route. First, the Steiner points of each route are identified to eliminate the inaccuracy introduced by the overlapping subnets. The nets are sorted in ascending order of the number of bends in their corresponding routes and then visited in this order and assigned to the layers using a greedy strategy. Specifically, for each net, the corresponding 2D route is broken into branches at the Steiner points and each branch is further broken into horizontal and vertical flat segments. A flat segment is assigned to a layer if overflow is not introduced. Otherwise, the segment is assigned to multiple layers in a greedy manner to minimize the total overflow and via count up to that point. Finally, the segment will be connected using vias to the closest segment of the same net to ensure connectivity. If a segment includes a terminal located in a layer other than M1, then a via connection will be made to connect it to the terminal at the target layer. As the segments are assigned to the layers, the procedure updates the corresponding edge utilizations by accounting for their specific wire sizes and spacings. During the CLA step, all the edges that are designated as blockage with a 0 capacity are avoided.

V. COALESCGRIP

The congestion analysis framework presented in Section IV was implemented as a tool which we refer to as CGRIP. A simplified variation of this framework, called coalesCgrip was implemented as the reference router to evaluate the ISPD 2011 contest [1]. Compared to CGRIP, coalesCgrip relies on a modified version of FGR [9] to generate an initial solution and does not have the INIT step. (We modified FGR to handle the ISPD 2011 benchmark requirements.) coalesCgrip uses the RLP step and its integration in rip-up and reroute, however not based on the formulation of (IP-CA) but based on the formulation of GRIP [16] to minimize the total overflow. For rip-up and reroute, its net ordering is based on the bounding boxes of the decomposed nets. It lacks the MRSR step and does not update the history term of an edge weight within an RRR iteration. In the contest, coalesCgrip was only used with maximum resolution. Therefore, due to lack of space, we skip the details for lower resolutions.

VI. SIMULATIONS

CGRIP and coalesCgrip were both implemented in C++ and used CPLEX 12.2 for solving the reduced-sized linear programs[1].All experiments ran on a machine with a 2.8GHz Intel CPU and 12GB of memory. (Our actual memory usage was just a fraction of the 12GB memory.) Our analysis tool supports the new bookshelf format used in the ISPD 2011 benchmark suites [13]. These benchmark instances include different wire sizes and spacings for 9 metal layers, routing obstacles, and net terminals located at higher metal layers. We used the branch-free data structure [6] to represent the decomposed subnets in our implementation of CGRIP and coalesCgrip.

A. Total Overflow Minimization

The first experiment is aimed at demonstrating the effectiveness of CGRIP to generate high-quality routing solutions using only a short runtime budget. In this case, we use the formulation (IP-CA) with $|\mathcal{R}| = |E|$ regions and $\kappa = 0$, so the analysis tool seeks to minimize the total overflow on the global routing grid-graph. For each benchmark, we use the placement instance identified as "the best solution" among the contest participants which we downloaded from the ISPD 2011 contest website [1] and then apply routing. For CGRIP, we impose a 15-minute runtime budget as the (only) stopping criteria. For coalesCgrip, we run the (same) binary as used at the contest on our machine which resulted in the runtime of some benchmarks to be higher than 15 minutes. For this experiment, there is no limit on the size of the routing box for each net in either of the tools.

We compare the wirelength (WL scaled to 10^{-5}) and total overflow (TOF). We note again that the solutions generated by CGRIP and coalesCgrip both avoid routing on the specified blockages. Alternatively, routing on the blockages and counting it as overflow can yield to less overflow. But this is against the blockage definition, although the generated solution still passes the ISPD 2011 evaluation script. Furthermore, our reported TOF is for the generated 3D routing solution as given by the contest evaluation script. It reflects the routing resource usage and not the number of tracks. So if a route is assigned to a lower layer, it consumes less routing resource.

The results are reported in Table I. Columns 2 and 3 report the grid size and the number of nets for the benchmarks. The name of the used placement instance is reported for each benchmark. CGRIP performs significantly better than coalesCgrip in reducing the TOF. The TOF is improved on average by 77.29%. The CGRIP WLs are always higher than coalesCgrip but in general it can be decreased by controlling how scenic the nets are routed.

B. Identifying and Ranking the Regions with Overflow

Our second experiment is aimed at assessing the accuracy of CGRIP at identifying and ranking the overflow regions. To perform this experiment, we run CGRIP, using $\kappa = 0$ in the IP model (IP-CA), for different resolutions and runtime budgets.

The following cases are compared:

- **maxRes60**: CGRIP is ran with maximum resolution (given in column 2) which results in minimizing the TOF for a time-budget of 60 minutes. This case is also our reference case.
- **maxRes15**: CGRIP is ran with maximum resolution with a shorter time-budget of 15 minutes.
- **lowRes15**: CGRIP is ran with a much lower resolution of $r_x \times r_y$ = 15x15, resulting in regional minimization of overflow for 225 regions for a time-budget of 15 minutes.

In all the above cases, we force CGRIP to control the scenic nets, ensuring that all nets are routed within 110% of their bounding boxes.

[1]Both CGRIP and coalesCgrip tools are available for download at http://homepages.cae.wisc.edu/~adavoodi/gr/cgrip.htm

TABLE I
EVALUATION OF CGRIP FOR MINIMIZING THE TOTAL OVERFLOW, AND FOR IDENTIFYING AND RANKING THE REGIONS WITH OVERFLOW

				Total OF minimization					CGRIP ranking of regions with OF					
				coalesCgrip		CGRIP			maxRes60		maxRes15		lowRes15	
Benchmarks	**X_GxY_G**	**#Nets**	**Placer**	**WL**	**TOF**	**WL**	**TOF**	**TOF Imp.%**	**TOF**	**$\lvert\mathcal{R}_c\rvert$**	**TOF**	**Err**	**TOF**	**Err**
superblue1	704x516	822744	SimPLR	150.24	0	150.91	0	0.00	0.5K	1	0.5K	0.0	0.5K	0.0
superblue2	770x1114	990899	Ripple	307.73	797898	317.83	138544	82.64	353K	100	380K	22.1	382K	18.8
superblue4	467x415	567607	Ripple	108.57	85538	111.51	2968	96.53	23K	60	28K	12.0	32K	10.1
superblue5	774x713	786999	Ripple	172.86	126186	176.32	28676	77.27	55K	36	56K	3.2	58K	1.0
superblue10	638x968	1085737	RADIANT	250.16	616742	256.55	112720	81.72	52K	101	120K	20.4	122K	9.2
superblue12	444x518	1293436	SimPLR	228.85	415428	241.56	35954	91.35	338K	108	352K	15.2	357K	6.1
superblue15	399x495	1080409	Ripple	179.11	125936	185.19	14052	88.84	54K	97	60K	21.9	64K	10.9
superblue18	381x404	468918	mPL11	98.44	31440	102.40	0	100.00	42K	51	43K	18.1	47K	13.00
Average				**187.00**	**274896**	**192.78**	**41614**	**77.29**		**69.2**		**14.1**		**8.6**

Consequently the TOF values in this experiment are higher than the previous one. The blockages are not allowed to be used.

The purpose of this experiment is to measure how closely solutions obtained with a short run time budget match the reference solution. We measure in the following manner. After obtaining a routing solution in each of the above three cases, we super-impose a 15x15 grid-graph and rank the resulting 225 regions in descending values of the TOF at each region. We then consider the top entries in the ranked list of the reference case corresponding to the regions with (non-zero) overflow. These entries reflect the indexes of the congestion hotspots in the reference case, and sorted with respect to the descending order of "difficulty". We refer to these regions as $\mathcal{R}_c$.

We then check the accuracy of the other two ranked lists (maxRes15 and lowRes15) with respect to the reference one. To measure the accuracy of a ranked list, we compute the displacement of the rank of each region r with respect to its rank in maxRes60 and report the average over all the regions in $\mathcal{R}_c$. For example for maxRes15, we compute $Err = \frac{\sum_{r \in \mathcal{R}_c} |\text{rank}_{r,maxRes60} - \text{rank}_{r,maxRes15}|}{|\mathcal{R}_c|}$.

The results are reported in Table I in columns 10 to 15. The number of considered regions $|\mathcal{R}_c|$ defined by the reference case is reported in column 11. For maxRes15 and lowRes15, we report the ranking error (Err) as explained above. For each of the 3 cases, we also report the total overflow (TOF) of the generated routing solution. For example in `superblue10`, the average displacement in ranking of the 101 regions with overflow is 9.2 units in lowRes15 and 20.4 (more than twice) in maxRes15. On average for all the benchmarks, the ranking error is 8.6 units in lowRes15 while it is 14.1 units in maxRes15 out of 69.2 regions with overflow.

The computational results indicate that for the same runtime budget of 15 minutes, lowRes15 always offers a significantly lower error than maxRes15 and thus can more accurately rank the congested regions. This confirms our intuition that minimizing the TOF may not be the correct objective for a small time-budget if the goal is identifying the congested hotspots. Furthermore, maxRes60 has the smallest TOF due to a longer runtime and directly minimizing TOF as the objective. The TOF of lowRes15 is similar (often slightly higher) than maxRes15. However, we contend that the results of lowRes15 is more valuable to a routability-driven placer than maxRes15 since it can more accurately rank the hotspots in the order of their difficulty.

VII. CONCLUSIONS AND FUTURE WORKS

This work introduced CGRIP, a new routing congestion analysis tool. CGRIP operates on a flexible model of global routing that accurately reflects various factors that contribute to congestion. We proposed a (parameterized) Integer Programming formulation for the congestion analysis problem that identifies and ranks the most-congested hotspots for a routability-driven placer. To engineer a practical solution approach to our formulation, we introduced several new ideas, such as working with a reduced-sized linear program, integrating dual prices from a linear program with traditional rip-up and re-route procedures, simultaneous re-routing of multiple nets, and a congestion-aware layer assignment technique. Computational experiments show that our tool achieves roughly an order-of-magnitude improvement in the total overflow when compared to coalesCgrip, a simpler variation of our framework used in the ISPD 2011 contest. We also show that given a 15-minute runtime budget, the most-congested regions can be estimated more accurately with a coarse-grained (15x15) resolution of our IP model than one that minimizes the total overflow. Future work includes integration with a routability-driven placer to verify the impact of resolution in improving routability and investigating more complex region definitions.

REFERENCES

[1] ISPD 2011 routability-driven placement contest [online] http://www.ispd.cc/contests/11/ispd2011_contest.html.

[2] U. Brenner and A. Rohe. An effective congestion driven placement framework. In *ISPD*, pages 6–11, 2002.

[3] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang. NTHU - Route 2.0: A fast and stable global router. In *ICCAD*, pages 338–343, 2008.

[4] H.-Y. Chen, C.-H. Hsu, and Y.-W. Chang. High-performance global routing with fast overflow reduction. In *ASP-DAC*, pages 582–587, 2009.

[5] M. Cho, K. Lu, K. Yuan, and D. Z. Pan. BoxRouter 2.0: A hybrid and robust global router with layer assignment for routability. *ACM TODAES*, 14(2), 2009.

[6] J. Hu, J. A. Roy, and I. L. Markov. Completing high-quality global routes. In *ISPD*, pages 35–41, 2010.

[7] C. Li, M. Xie, C. Koh, J. Cong, and P. Madden. Routability-driven placement and white space allocation. *IEEE TCAD*, 26(5):858–871, 2007.

[8] M. Pan and C. Chu. IPR: An integrated placement and routing algorithm. In *DAC*, pages 59–62, 2007.

[9] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. *IEEE TCAD*, 27(6):1066–1077, 2008.

[10] J. A. Roy, N. Viswanathan, G.-J. Nam, C. J. Alpert, and I. L. Markov. CRISP: Congestion reduction by iterated spreading during placement. In *ICCAD*, pages 357–362, 2009.

[11] P. Spindler and F. M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *DATE*, pages 1226–1231, 2007.

[12] T. Taghavi, Z. Li, C. J. Alpert, G.-J. Nam, A. Huber, and S. Ramji. New placement prediction and mitigation techniques for local routing congestion. In *ICCAD*, pages 621–624, 2010.

[13] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam, and J. A. Roy. The ISPD-2011 routability-driven placement contest and benchmark suite. In *ISPD*, pages 141–146, 2011.

[14] M. Wang, X. Yang, K. Eguro, and M. Sarrafzadeh. Multi-center congestion estimation and minimization during placement. In *ISPD*, pages 147–152, 2000.

[15] J. Westra, C. Bartels, and P. Groeneveld. Probabilistic congestion prediction. In *ISPD*, pages 204–209, 2004.

[16] T.-H. Wu, A. Davoodi, and J. T. Linderoth. Global routing via integer programming. *IEEE TCAD*, 30(1):72–84, 2010.

[17] T.-H. Wu, A. Davoodi, and J. T. Linderoth. A parallel integer programming technique to global routing. In *DAC*, pages 194–199, 2010.

[18] Y. Xu, Y. Zhang, and C. Chu. Fastroute 4.0: global router with efficient via minimization. In *ASP-DAC*, pages 576–581, 2009.