Rethinking Global Routing for Modern VLSI Design: Congestion Reduction and Multi-Objective Optimization

By
Hamid Shojaei

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy
(Electrical Engineering)

at the
UNIVERSITY OF WISCONSIN-MADISON
2012

Date of final oral examination: 05/21/12

The dissertation is approved by the following members of the Final Oral Committee:

Azadeh Davoodi, Assistant Professor, Electrical and Computer Engineering
Yu Hen Hu, Professor, Electrical and Computer Engineering
Jeffrey T. Linderoth, Professor, Industrial and Systems Engineering
Parameswaran Ramanathan, Professor, Electrical and Computer Engineering
Kewal K. Saluja, Professor, Electrical and Computer Engineering
Twan Basten, Professor, Electrical and Computer Engineering, Technical University of Eindhoven, the Netherlands

Abstract

The high volume and complexity of cells and interconnect structures are causing serious challenges to routability in modern VLSI design. 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., reservation 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. In addition, interconnects now play a significant role in impacting the performance metrics of a design including power, speed and area. Global routing, as the first stage in which the interconnects are planned, is now of significant importance in determining the performance metrics and the routability of the design. However, the standard model of global routing considers minimization of wirelength with a simplified model of routing resources which ignores these objectives and complicating factors.

To address the above challenges, this dissertation has three contributions in rethinking global routing for modern VLSI design. First, we present a framework for congestion *analysis* for quick prediction of the locations of highly-utilized routing regions. The fast framework is suitable for integration in the design flow, for example

as an integration within a routability-driven placement procedure. Second, we offer two contributions in order to estimate and manage the congestion caused by local nets which are ignored in a standard model of global routing. It allows *optimizing* congestion directly within global routing by treating global and detailed routing in a more holistic manner. In addition, many of the above-mentioned factors contributing to congestion are accounted for in both our congestion analysis and optimization framework. Finally, we present a fast procedure for multi-objective global routing which is able to simultaneously optimize multiple performance metrics beyond wirelength. The framework is a collaborative one which receives as input multiple global routing solutions created by single-objective procedures.

First, we propose CGRIP which is our framework for congestion analysis. It is able to quickly identify and rank the regions on the layout which contain overflow of routing resources. CGRIP is based on a proposed Integer Programming model which minimizes overflow inside a set of routing regions defined by an input resolution parameter. A resolution lower than the global routing graph model allows building a congestion map faster. CGRIP also features several new ideas for a practical realization of the mathematical model in order to handle large industry-sized benchmark instances. It also accounts for varying wire sizes and spacings, routing blockages and virtual pins.

Second, this work expands CGRIP to account for congestion caused by local nets which is ignored in a standard model of global routing. Specifically, we present two complementary techniques which respectively *approximate* and *manage* the local nets during global routing. The latter technique is based on changing the global routing graph in a way to reduce the number of local nets and thus the error associated

with their approximation. Overall, our techniques expand the standard model of global routing into a congestion-aware model which comprehensively captures all major sources of congestion in modern VLSI design. CGRIP has also been released as a software package and a variation of it was used as a golden router to judge the ISPD 2011 routability-driven placement contest.

Finally, this dissertation presents a procedure for multi-objective global routing. The procedure is collaborative and takes as input multiple global routing solutions generated by multiple (and possibly single-objective) global routing procedures and then performs multi-objective optimization. It then generates multiple global routing solutions that make a tradeoff between the considered objectives. In our simulations, we demonstrate the effectiveness of our procedure using five modern and recently-released academic global routers while optimizing for metrics including wirelength, routing resource utilization, and interconnect power.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Global routing is a critical step after placement during which interconnect structures are planed. Consequently it is increasingly gaining importance to combat obstacles such as timing, power, and congestion when realizing complex Integrated Circuits. In advanced technology nodes, global routing has also become a more complicated procedure due to the higher number of metal layers, complex design rules and growing size of the design [3]. Furthermore, in modern designs variation in the wire sizes and spacings of the metal layers, routing blockages, *virtual* pins in higher metal layers, and high pin density all contribute to make the routing process more difficult. Moreover, with the advent of Moore's law into nanometer era, interconnect performance has become the dominant factor in design performance, and the global routing solution directly impacts a multitude of aspects such as power and manufacturability.

To improve a design's routability, one set of techniques focus on *routability-driven placement* procedures that rely on mechanisms for predicting routing congestion [6], [9], [21], [30], [43], [47], [54], [59], [61]. From this sense, a fast and flexible congestion

estimation framework that can quickly identify the highly-utilized routing regions of the layout with respect to the new features of modern designs is highly desirable. Another avenue for improving routability is to adjust the global routing procedure to account for as many modern complicating factors.

In addition, in the nanometer era, not only is global routing required to optimize the fundamental objectives (e.g. wirlength and routability), but it also needs to address other important design factors (e.g. power). Therefore, developing an effective multi-objective modeling for global routing is in great demand.

In this chapter, we first discuss the challenges and motivations behind routing congestion analysis, routability improvement, and multi-objective optimization at the global routing stage in the context of nanometer technology. Then we summarize the contributions of this dissertation for rethinking global routing to address these challenges.

1.1 Challenges and Motivations

1.1.1 Lack of Accurate Congestion Estimation for Modern Designs

Challenge: The high volume and complexity of cells and interconnect structures in modern designs are causing serious challenges to routability. In modern designs, several new factors contribute to routing congestion including more layer stacks and varying wire sizes and spacings among different layers to achieve higher performance, various forms of routing blockages that are reserved for embedded IPs or memories on the die, and virtual pins that reside on a metal layer above the placement region

to provide interconnection between different levels of hierarchical design [56]. Additionally, modern designs often have high pin density and inter-layer vias, which further contribute to congestion [55]. These factors were inadvertently ignored in the traditional global routing procedures. The existence of these new factors in modern designs makes the routability issue more and more challenging at the lower stages of the design, and in turn complicates achieving design closure and prolongs the design cycle [3], [4].

Motivation: Congestion analysis (identification of highly-utilized routing regions) is now desired to help the designer estimate the locations of the routing hotspots at the early stages of design so that different design configurations can be explored to improve routability [3], [4], [35], [43], [47].

Many of the complicating factors in modern designs can be accurately modeled with a flexible model of global routing. The work [56] explains modeling of these factors in the recently-released industrial benchmark instances of the ISPD 2011 contest [2] which are recently released as challenging instances to achieve routability. These benchmark instances are direct translations from actual industrial designs which contain multi-million nets, and model 9 metal layers with different wire widths and spacings ranging from 1x to 4x, non-rectangular cells and routing blockages, and virtual pins. Using the above model allows for more accurate routability consideration by utilizing a flexible and fast global routing model for congestion analysis. However, the existing academic global routers are not designed to effectively and comprehensively handle this model [4].

Similar to a global router, a congestion analysis tool should complete a route for each net. Unlike a global router, the process is expected to run very quickly, typically

in a few minutes. Moreover, the analysis tool is expected to run incrementally such that subsequent iterations (e.g., as a result of perturbations due to physical synthesis) also run very quickly. Given the restricted time budget that can be spared on congestion analysis, the resulting routing solution may still contain many units of overflow of routing resources. This is because the analysis tool may be working in conjunction with placement to mitigate the hotspots. However, a good congestion analysis tool should be able to provide a heatmap that can identify the problematic regions for better guiding of a routability-driven placer [21], [23], [30]. The accuracy of predicting the congested regions is limited if it is done based on traditional minimization of total overflow of routing resources which also results in creating "scenic" wires.

While academic global routing procedures have been continuously improving in the past few years [8], [14], [25], [64], [65], [66], these tools may not be suitable for congestion analysis. They are too slow for congestion analysis and the analysis is typically required to take only a fraction of a typical global routing runtime [4]. Additionally, the primary objective of all the existing procedures is to minimize the overflow of routing resources.

1.1.2 Congestion Impact of Local Nets

Challenge: In the traditional usage of global routing, the chip layout is divided into equal-sized tiles (global cells) and global routing solution of each net consists of adjacent (contiguous) global cells connecting its terminals. A local net has all its terminals inside one global cell and is traditionally ignored during global routing.

Modern designs typically have high pin density, often translating into a high number of local nets, which significantly contribute to local congestion. In many instances,

a significant portion of the nets to be routed are local. For example, for the ISPD 2011 benchmarks [2], using the winning placement solutions from the ISPD 2011 contest on routability-driven placement, on average 31.20% of the nets are local. These nets not only consume wiring tracks, but can also block access to pins inside global cells. Consequently, ignoring the local nets at the global routing stage will make serious routability issues at the lower stages.

Motivation: The existing global routing procedures [8], [14], [25], [64], [65], [66] simply ignore local congestion at the global routing stage. Some techniques [6] use a straightforward model that accounts for the local congestion inside the global cell by reducing the number of routes that can pass from each of its boundaries based on the pin density of the global cell. Depending on the locations of the pins inside the global cell, this approach may be too pessimistic or optimistic to account for local congestion. Some other techniques [3] propose to decrease the size of global cells uniformly which turns into decreasing the number of local nets. However, if a global cell is too small, the dimensions of the corresponding global routing graph become too large and the complexity of the global routing procedure becomes intractable.

It is desirable to have an accurate model that captures the impacts of the local congestion while controling the complexity of the global routing procedure. Routers that can accurately model these effects are likely to significantly reduce the effort at the detailed routing stage.

1.1.3 Interconnects as Critical Determinant of Various Performance Metrics

Challenge: In the context of complex modern designs, the problem of intercon-

nect performance improvement also grows in significance, since on-chip interconnect becomes an increasingly critical determinant of performance, manufacturability and reliability in modern VLSI designs. Interconnects are initially planed at the global routing stage. The main focus of the traditional global routing procedures is to address the primary objective of minimizing wirelength and via usage of interconnects, preferably without generating any overflow of routing resources. While wirelength is an important metric to justify the effectiveness of a global router, new factors are also becoming important that impact performance and routability of the design.

Motivation: Ever since the release of large industrial benchmarks in the ISPD 2007 contest [1], new global routers have emerged which have pushed the boundaries to obtain higher solution quality in terms of total wirelength and routing overflow, and faster runtime [5], [7], [8], [11], [12], [14], [16], [22], [25], [33], [34], [39], [42], [46], [63], [64], [65], [66]. These global routers are inherently different in the underlying procedures to tackle the routing problem. Each one provides exclusive features which result in distinct routing solutions compared to the other one. This motivates us to develop a collaborative framework that uses the strength of each router and provides a fast and robust global routing procedure. Combining various routing procedures is very difficult, if not impossible, since the procedures are highly incompatible. However, it is possible to only combine the solutions, which are reflective of these procedures in a systematic way and generate a new one which is always better than all the solutions.

Moreover, global routing is a stage where many factors contributing to interconnect performance, including the interconnect topologies, loading, and wire size and spacing can be approximated in a fairly accurate manner. Therefore, beside the traditional objective of wirelength minimization, opportunities lie in optimizing other

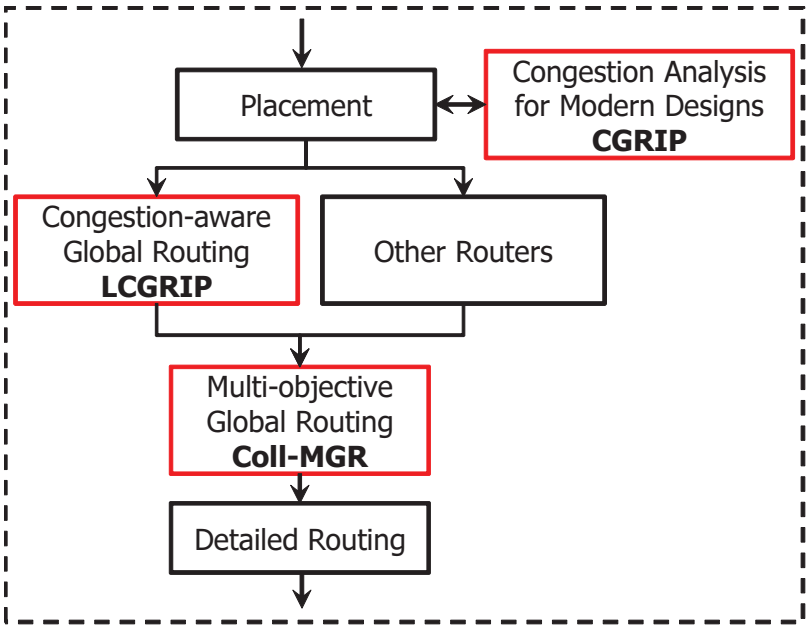

Figure 1.1: Our contributions for rethinking global routing to account for congestion reduction and multi-objective optimization.

interconnect-dependent performance objectives at this stage. Optimizing for other objectives can follow in subsequent calls of global routing based on several global routing solutions initially-generated by other (and possibly single-objective) global routing procedures. Hence, multi-objective optimization is another aspect that we have considered in our research.

1.2 Contributions

The contributions of this dissertation are categorized into three groups. First, we propose CGRIP as a fast and flexible framework for congestion analysis at the global routing stage. Next, LCGRIP is proposed as a congestion-aware global router that utilizes a novel and efficient approach to plan for local net congestion in global routing. Finally, we propose Coll-MGR as a fast and collaborative framework for multi-

objective global routing. Figure 5.1 shows how these three parts fit into a typical physical design flow.

1.2.1 Fast Congestion Analysis for Modern Designs

We present CGRIP, a fast and flexible congestion analysis framework which aims to accurately and quickly predict the congested regions on the layout. The framework utilizes the flexible model of global routing that captures many necessary modern design features.

To accurately identify and rank the congested regions of the layout in a small time-budget, this work proposes minimizing overflow of routing resources inside the regions on the layout specified by an input *resolution* parameter. 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 traditional overflow of routing resources, contains new variables to model the total overflow and maximum overflow for each region induced by the resolution parameter.

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 ripup and re-route procedures used by many global

routing frameworks.

Furthermore, this work speeds up the conventional ripup and re-route global routing procedures by performing multiple ripups and a single, simultaneous re-routing of "equivalent" nets. It also provides congestion-aware multi-terminal net decomposition to further improve the quality of the routing solution. Moreover, it utilizes a congestion-aware layer assignment procedure that assigns routing wires to different metal layers, while accounting for varying wire size, spacing, blockages, and pins at different metal layers.

In summary, CGRIP includes the following contributions:

- An IP formulation of the congestion analysis problem that aims to *quickly find the locations of the congested regions by using a resolution parameter*;
- Several methods to achieve a practical realization of the IP for a small time-budget, including:
  - an approximation method to shrink the IP formulation and a novel way to integrate its solution with a standard ripup and re-route procedure;
  - a multiple ripup, single re-route procedure for speedup improvements;
  - a congestion based minimum spanning tree construction procedure;
  - a layer assignment procedure to handle new complicating factors.

1.2.2 Planning for Local Net Congestion in Global Routing

While CGRIP supports many complicating factors in modern designs, it does not account for local congestion effects. Furthermore, CGRIP was primarily tuned to provide a rapid estimation of congestion by imposing a small runtime budget— much smaller than typically spent during global routing.

We present LCGRIP which contains two complementary techniques to manage the impact of local nets during global routing. First, we propose to *reduce* the number of local nets by using global cells of non-uniform size. A procedure is presented to transform a given global routing instance into one with fewer local nets. The procedure keeps the number of global cells the same, but the number of global nets increases, so the effort required at the global routing stage may increase. However, more nets are routed during global routing and the effort required for detailed routing can be significantly reduced. Second, we propose to *approximate* the area required to route local nets and to adjust the areas of the affected global cells before global routing. This adjustment results in global routing solutions for which it is easier to perform detailed routing.

Using these two complementary techniques to approximate and reduce the local nets, a new graph model of global routing with non-uniform global cells is presented which more accurately honors the available routing resources with respect to local nets. Our graph model also captures other factors contributing to congestion such as varying wire size and spacing, routing blockages, and virtual pins. An IP model is presented describing this *comprehensive* congestion-aware global routing problem. Our ideas are all implemented into a practical routing framework which handles large industry-sized instances. The practical realization is based on integration with CGRIP, revising its functionality to suit the goals of this research effort.

The summary of contributions of this work are as follows:

- A procedure to generate global cells of non-uniform size to minimize the number of local nets while keeping the number of global cells fixed.
- A graph model (and an IP formulation) of a comprehensive congestion-aware

global routing problem to model resource usage by the local nets in global cells of possibly non-uniform size along with other complicating factors.

- Integration with CGRIP to obtain a practical routing framework accommodating modern design features and issues.

1.2.3 Collaborative Multi-Objective Global Routing

To address design performance improvement at the global routing stage, we present Coll-MGR, a multi-objective procedure for global routing. The procedure is collaborative and takes as input multiple global routing solutions, which are generated independently (e.g., by one router that runs in different modes concurrently, or by different routers running in parallel). It then performs multi-objective optimization based on Pareto algebra [18] and quickly generates multiple global routing solutions with a tradeoff between the considered objectives. The user can control the number of generated solutions and the degree of exploring the tradeoff between them by constraining the maximum allowable degradation in each objective.

Specifically, this work identifies practical variations of multi-objective global routing. We consider the following three multi-objective case studies: minimization of interconnect power and wirelength, minimization of routing congestion and wirelength, and minimization of wirelength with respect to the (finite-capacity) routing resources. The maximum allowable degradation in wirelength is specified in all cases. Our multi-objective procedure runs in only a few minutes for each of the ISPD 2008 benchmarks [27], even for the unroutable ones, which imposes a tolerable overhead in the design flow using modern academic global routing procedures. In our simulations, we demonstrate the effectiveness of our procedure using five modern and recent

academic global routers.

The contributions in this work are summarized below.

- We introduce the concept of multi-objective and collaborative global routing which combines the routing solutions generated by different routing tools and in practice always obtain an improved solution in a very short amount of time. The framework generates multiple global routing solutions to provide a tradeoff between the considered objectives, thereby allowing the user to optimize one objective with a controlled degradation in other objectives.
- We formulate the collaborative global routing as the multi-dimensional multiple-choice knapsack problem (MMKP) [38], and propose a Pareto-algebraic heuristic to quickly solve MMKP instances with high quality.
- We study different variations of collaborative routing to minimize wirelength, minimize the highly-utilized regions on the layout under wirelength constraints, and to jointly minimize wirelength and a metric to minimize interconnect power. We propose a signal power model at the global routing stage which estimates cross-coupling capacitance based on the routing resource utilization. This model accounts for varying edge capacities, wire sizes, and wire spaces.

1.3 Thesis Overview

The remainder of the document is organized into the following chapters. Chapter 2 describes preliminary terms and the literature review about global routing and routing congestion estimation. In Chapter 3 we discuss CGRIP, our congestion analysis framework. Our approach for addressing local congestion in global routing is

presented in Chapter 4. Chapter 5 describes the formulation of multi-objective and collaborative global routing, and a procedure to effectively solve the multi-objective and collaborative routing formulation. The summary and conclusion remarks are offered in Chapter 6.

Chapter 2

Preliminaries and Literature Review

This chapter gives an overview of procedures for global routing and routing congestion analysis. We first review a terminology and a set of techniques for global routing. Next, we give an overview of existing techniques for congestion estimation.

2.1 Introduction to Global Routing

Within modern physical design flow a routing algorithm uses a two-stage approach as shown in Figure 2.1. First, global routing decides coarse-grained paths for all nets with respect to available routing resources. Next, detailed routing determines the exact tracks and vias for the nets using the paths generated by global routing. In detailed routing all the design rules and constraints are checked to ensure design and manufacturability closure. As shown in Figure 2.1, global routing may also be used to provide a more accurate wirelength and congestion estimation in an iterative

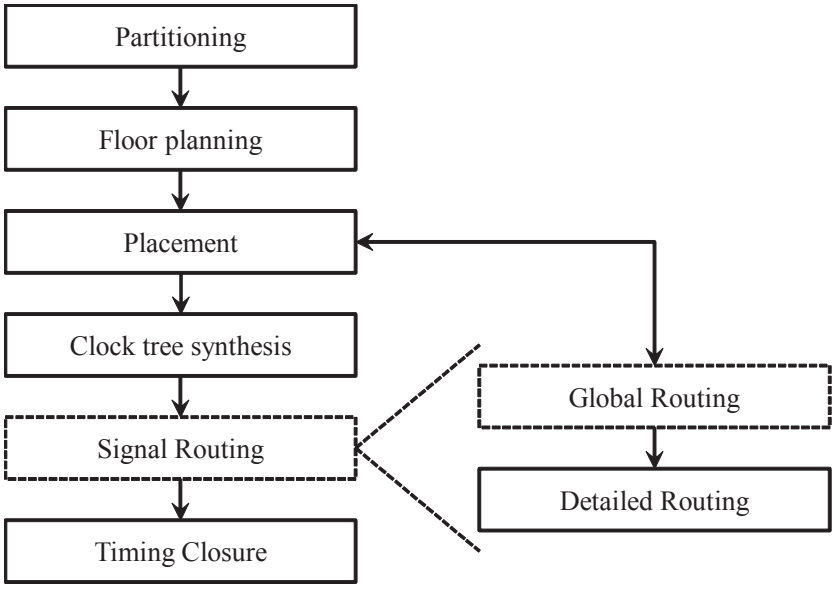

Figure 2.1: The major steps in a standard physical design flow.

integration with placement in order to enhance routability of the design.

The coarse-grained routes generated at the global routing stage also guide the detailed routing stage. Therefore, global routing can be considered as a critical step of the flow because it directly impacts both detailed routing and placement stages.

In Figure 2.2 we show placement, global routing, and detailed routing solutions of a design with three cells and five nets. Figure 2.2(a) shows the design after placement. As can be seen in this figure, after the placement phase, the exact locations of cells and the corresponding pins of each cell are known. In addition to the placement information, a netlist describing the connectivity between the cells is also provided in the figure as follows. The terminals of a net all are represented by the same net index at their corresponding locations. For instance net `n1` has two terminals which are connected to cell `C1` and another terminal which is connected to cell `C3`. The routing algorithm needs to find a path on the layout to connect all the pins with the same net indices.

Figure 2.2(b) shows the design after applying the global routing procedure. Global

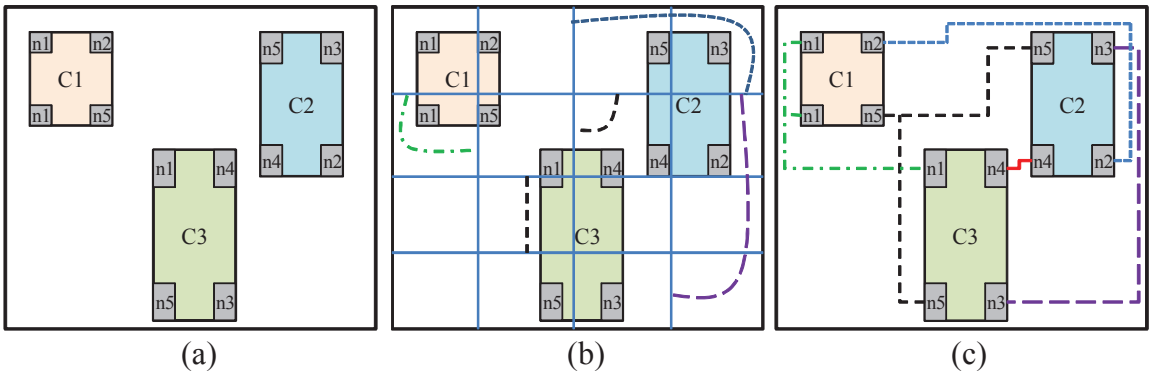

Figure 2.2: (a) Design after placement, (b) Design after global routing, (c) Design after detailed routing.

routing first divides the chip area into equal-sized *tiles* or *global cells* (gcells). A global cell is a region that further contains multiple horizontal/vertical *routing tracks* defined by the detailed routing grid. A routing track is a horizontal or vertical path that a wire can pass. For routing a net, a sequence of horizontal and/or vertical tracks are used. Usually, each routing layer includes only vertical or horizontal tracks, and the adjacent vertical and horizontal tracks on different layers are connected by inter-layer *vias*.

After defining the global cells, nets are categorized into two different groups: *global nets* and *local nets*. A local net has all its terminals inside a global cell and is traditionally ignored during global routing. In Figure 2.2, `n4` is a local net because it has all its terminals inside a single global cell. A global net is a net that has at-least two terminals in two different global cells. A global routing procedure provides coarse routes for all global nets. All nets except `n4` in Figure 2.2 are global nets. The global routing procedure may use different parts of the chip to find coarse routes for all global nets while minimizing the total wirelength. These routes need to satisfy routing resource constraints and should avoid routing blockages. A *routing blockage* is

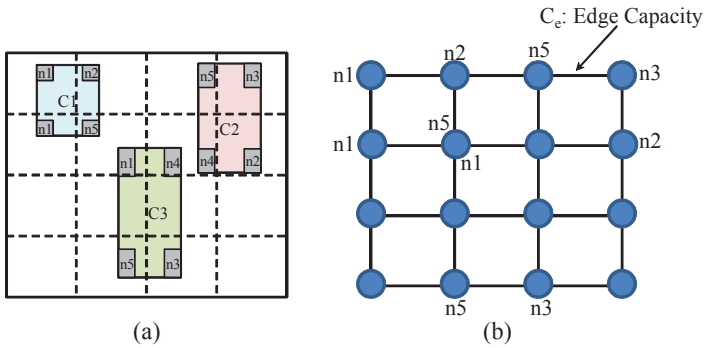

Figure 2.3: The global-routing grid graph: (a) Each layer is divided into uniform global cells, (b) Each global cell is represented as a vertex of the global routing grid graph and the connection between different global cells is represented using edges.

a part of chip that can not be used for routing. For instance parts that are reserved for power-grid, clock network, or IP blocks in an SoC are examples of routing blockages. Traditionally, routing blockages are considered as rectangular shapes. However, in modern designs this assumption is not true in practice. In modern designs, it is required to consider blockages that have non-rectangular shapes; that we refer to them as *irregular* routing blockages.

In a graph model of global routing, each node represents a global cell and each edge represents the connection between two adjacent global cells. This creates a grid-graph as shown in Figure 2.3. Furthermore, *actual capacities* are associated with the edges to represent the available routing resources between two adjacent global cells. In addition to the actual capacity, for each edge a *normalized capacity* is also defined that represents the maximum number of wires that are allowed to pass the edge. The normalized capacity of each edge depends on the wire width and minimum spacing requirements of the corresponding layers. Consider the actual capacity of edge e to be c_e, and wire width and spacing of the corresponding layer to be w_l and s_l, correspondingly. The normalized capacity of edges e is equal to $b_e = \frac{c_e}{(w_l+s_l)}$. If

an edge is utilized more than its capacity, the edge has *overflow* of routing resources. Consider the number of wires that pass edge e to be n_e. The *normalized overflow* of edge e is computed as $o_e = max(0, n_e - b_e)$. This value is multiplied by $w_l + s_l$ of the corresponding layer to show the *actual overflow* of the edge.

All terminals inside a global cell are mapped to the corresponding node of the grid graph. For each net, the router must determine a tree on this graph that connects its corresponding terminals.

The global routing grid graph is a three-dimensional graph, where each dimension represents a layer and the corresponding nodes on different layers are connected using vias. Figure 2.3 depicts a routing metal layer that is divided into uniform global cells and represented as a grid graph.

The aim of the global router is to find for each net, a tree on the routing graph, connecting the vertices corresponding to its terminal pins. This is done using graph-search algorithmic procedures. The common approaches used as graph-search algorithms in global routing are introduced in Section 2.1.1. These algorithms find a path on the routing graph connecting two nodes of the graph. However, the nets often have more than two pins. A technique, known as net decomposition, is used to convert a multi-pin net into multiple 2-pin subnets. We discuss net decomposition techniques in Section 2.1.2. In Section 2.1.3, we show how all of these techniques are utilized inside a modern global routing flow and compare the modern global routers from various aspects.

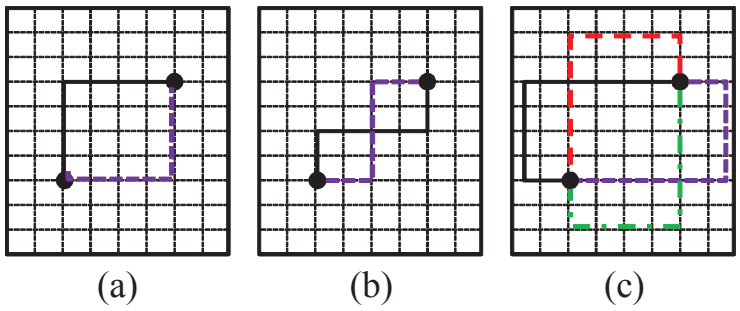

Figure 2.4: Pattern routing for two-pin nets: (a) L-shaped, (b) Z-shaped, (c) U-shaped.

2.1.1 Single Net Routing Procedures

In this section we briefly describe four popular graph-search techniques, namely **pattern**, **maze**, **A*-search**, and **monotonic** routing techniques. These techniques are commonly used for single net routing.

The graph that is considered in the single-net routing procedure, is usually a *weighted* graph. Each edge in this graph has a weight reflecting, for example, its utilization in terms of number of wires that pass from it for the current solution. We will discuss the edge weights in Section 2.1.3.

Pattern Routing: Pattern routing is the fastest method for routing two-pin nets. Instead of using sophisticated methods for finding a path between the two nodes, a small number of patterns is searched. This improves the runtime of the routing, because search is done through a small number of patterns. The topologies of the patterns are L-shape, Z-shape, and U-shape routes. For each case, different variations exist as shown in Figure 2.4. In addition to the fast runtime, pattern routing algorithms usually provide routes that have a smaller wirelength and fewer number of bends.

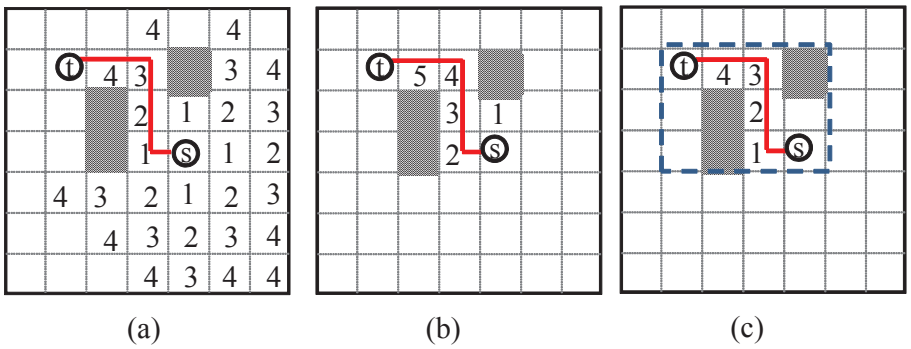

Figure 2.5: (a) Maze Routing (exploring 28 nodes), (b) A* search algorithm (exploring 6 nodes), (c) Monotonic routing algorithm (exploring 5 nodes) [28].

Maze Routing: Maze routing or Lee's algorithm [32] is a technique based on the breadth-first-search (BFS) traversal of the global routing graph. This approach has two steps. Similar to a wave propagation, in the first step, the nodes are progressively visited and labeled with respect to their distances from a "source" node until the "target" node is reached. Then in the second phase, the labels are retraced from the target node in a decreasing order until the source node is reached, and the path connecting them is identified. There may be multiple shortest paths connecting the source and the target nodes. The path with the minimum number of bends is preferred, because each bend requires inserting a via. The maze routing algorithm always finds the shortest path between two points even in the presence of obstacles. However, the time and memory complexity of the algorithm is high. We refer the reader to [32] for more details about the complexity of the maze routing algorithm.

A*-search routing: A*-search algorithm [20] is a fast search algorithm in which the weight function includes the distance of the current node from the source node and an *estimated* distance from the current node to the target. These distances are computed based on the edge weights which are based on the . This technique expands only the

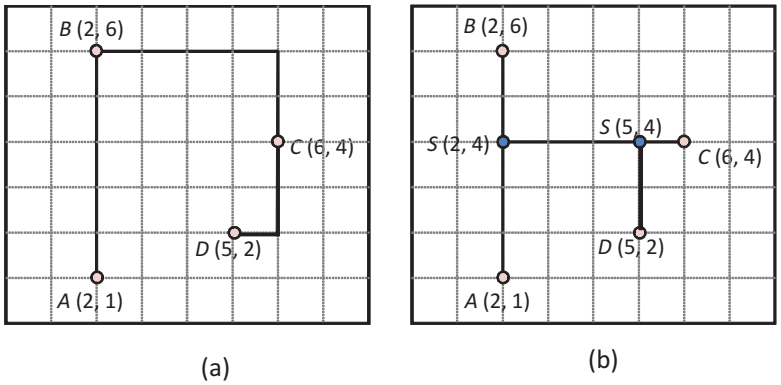

Figure 2.6: (a) Rectilinear Minimum spanning tree (RMST) , (b) Rectilinear Steiner minimum tree (RSMT)

most promising nodes. Consequently, this "best-first" search strategy eliminates a large portion of the solution space without loosing optimality.

Figure 2.5 shows how the A* search algorithm is different from the maze routing algorithm. The two algorithms are applied to the same example. The A* search algorithm reaches the target node after exploring 6 nodes, while the maze routing algorithm visits 28 nodes before it visits the target node. In this example, both algorithms find the same final route.

Monotonic routing: Monotonic routing provides a trade-off between maze routing and pattern routing so that the quality can be better than pattern routing, but the runtime is close to it [66]. A monotonic routing path from a *source* to a *target* node is a path on the routing grid from *source* to *target* which always directs toward the target. Figure 2.5(c) shows a monotonic routing path from `s` to `t`. Note that all monotonic routing paths will remain inside the bounding box of `s` and `t`.

2.1.2 Net Decomposition

Both Maze routing and A* search algorithms are typically used for finding the shortest path between two pins on the routing graph. However, the nets may have more than two pins. A common practice is to apply net decomposition [28], [58], by which a multi-pin net is decomposed into multiple 2-pin *subnets*. Each subnet is then routed independently. During this process the "**shared resources**" [28] are defined and eliminated. This means if two subnets of a net share parts of their corresponding routes, the common parts are considered only once for computing the resource utilization of the net.

Two popular techniques are widely used for net decomposition: Rectilinear Minimum Spanning Tree (RMST), and Rectilinear Steiner Minimum Tree (RSMT) of the pins of each net.

The RMST of a net is a minimum-length tree connecting all the pins according to their Euclidean distance. The RMST can efficiently be computed in polynomial time using the Kruskal's [31] or Prim's [44] algorithms. Figure 2.6(a) depicts an example 4-pin net decomposed by a rectilinear RMST, where each segment runs horizontally or vertically.

A more natural method to route multi-pin nets is using the Steiner-tree based approach. Specifically, a rectilinear Steiner minimum tree (RSMT) is used for routing a multi-pin net with the minimum wirelength. The difference between the Steiner tree problem and the minimum spanning tree problem is that in the Steiner tree problem intermediate *virtual pins* are added to the net. As a result, the spanning tree with the additional virtual pins will be shorter. These virtual pins are known as the **Steiner points**. Rectilinear Steiner Minimum Tree problem (RSMT) is a variant

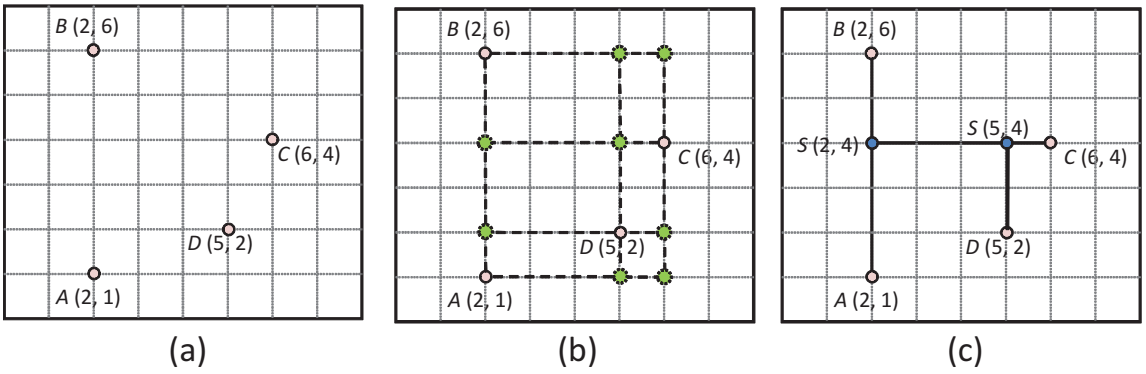

Figure 2.7: Finding the Hanan grid and the Steiner points of an RSMT.

of the geometric Steiner tree problem in the plane, in which the Euclidean distance is replaced with the rectilinear distance. Figure 2.6(b) shows the rectilinear Steiner minimum tree for our example.

The RSMT is an NP-hard problem even for a single net. Similar to the other NP-hard problems, common approaches to tackle it are heuristic procedures. Maurice Hanan [19] proved that for finding the Steiner points, it suffices to consider only the points located at the intersections of the vertical and horizontal lines that pass through terminal pins. The *Hanan grid* consists of the lines $x = x_p$, $y = y_p$ that pass through each pin location (x_p,y_p). The Hanan grid contains at most $m \times n$ candidate Steiner points, where m and n are number of distinct horizontal and vertical locations of the pins, correspondingly. Consequently, it significantly reduces the solution space for finding an optimal RSMT [28]. Figure 2.7 shows the Hanan grid and the Steiner points for a 4-pin net.

FLUTE [13] is a fast and popular technique for RSMT construction and it is widely used in the global routing procedures that use Steiner minimum tree technique for decomposition. It finds optimal RSMTs for nets with less than ten pins and produces

near-optimal RMSTs for the nets with higher number of pins.

2.1.3 Modern Global Routing

Recently, many new global routing procedures were developed to handle realistic and large-sized instances [5], [7], [8], [11], [12], [14], [16], [22], [25], [33], [34], [39], [42], [46], [63], [64], [65], [66]. These tools are inherently different in the underlying procedures to tackle the routing problem. The category of concurrent techniques consider simultaneous routing of all the nets [5], [39], [63], while the sequential ones impose an ordering to route the nets [22], [46], [66]. Furthermore, the sequential techniques apply drastically different procedures for different steps such as net ordering, layer assignment, net decomposition and route generation.

Figure 2.8 shows a standard flow used by modern global routers. We describe the different steps of this flow throughout this section.

The routing instance is provided for the global router as a connectivity list of all the nets and the placement of the pins [1], [27], [56]. It also includes information about the grid size, the vertical and horizontal capacities for each metal layer, the routing blockages, and the inter-layer vias.

The runtime of the global routing algorithms for a three-dimensional global routing grid graph is intractable. Therefore, most of the global routers convert the three-dimensional routing instance into a two-dimensional problem to consider a smaller-sized problem. **The 2D projection phase** in the global routing flow (Figure 2.8) translates a three-dimensional routing instance and projects it into a two-dimensional one. Specifically, for each edge in the two-dimensional space, the capacity is summation of the corresponding edges in the three-dimensional space for all the metal

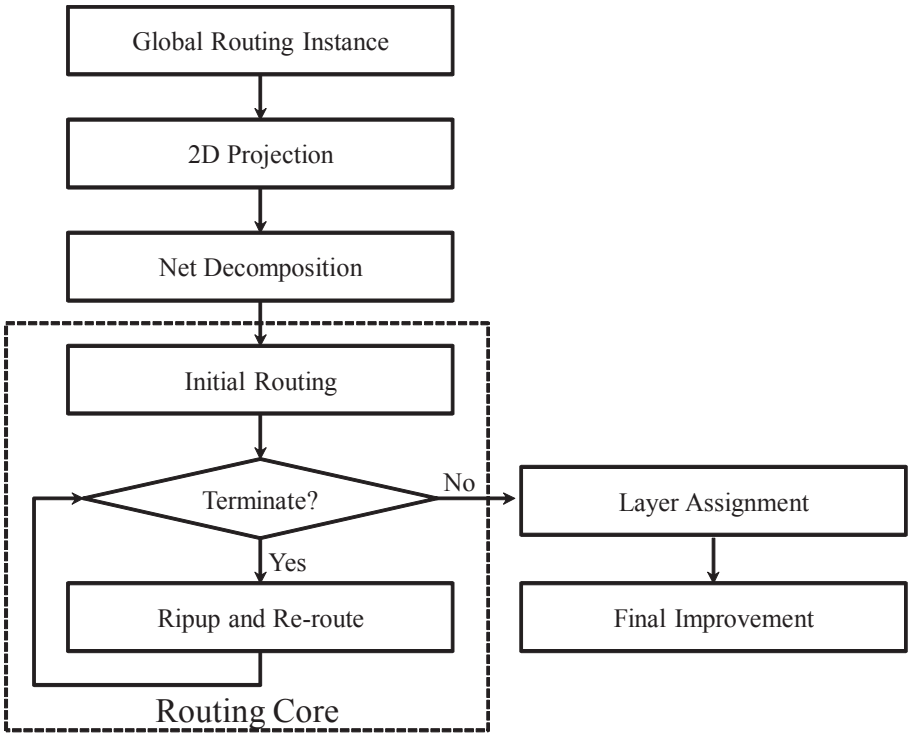

Figure 2.8: Global routing flow.

layers. If an edge in the three-dimensional space is located inside a blockage, it will get reflected in the aggregated capacity of its projected two-dimensional edge.

Next in Figure 2.8, the **net decomposition phase** decomposes all the multi-pin nets into two-terminal subnets using RMST or RSMT.

In Figure 2.8, the **initial routing phase** generates a route for each net on the two-dimensional grid. These routes are considered as the starting point for the next phase of the global routing. The routing solution is *legal*, if it does not cause any overflow of the routing resources. Otherwise, it is an *illegal* solution, and the overflow on each edge represents the units of usage above its normalized capacity.

Ripup and re-route (RRR) is a ubiquitous strategy employed by many routing tools [8], [9], [12], [25], [46], [66]. The RRR is repeated in many iterations during which

the nets with overflow are re-routed, until a violation-free solution is obtained or a runtime bound is reached. First, the initial routing solution that contains overflow is passed to the RRR procedure. At each iteration of RRR, the current routing solution is re-routed to reduce overflow. The nets that pass the edges with overflow are detected. The RRR procedure iterates over the list of such nets in a specific order and rips them up and re-routes them. During the ripup procedure, all routing resources corresponding to the ripped routes are released. Re-routing is then applied through the less congested regions, for example using a maze routing procedure. Most of the global routers use the same ripup and re-route framework. The major differences between various implementations include the order in which the nets are visited for ripup and re-route and the cost associated with each edge for maze routing [8], [9], [12], [25], [46], [66], which will be explained later.

One popular method for net ordering is based on the net *bounding box area*. The minimum-area rectangle which contains all the pins is defined as the bounding box of a net. The nets are ordered in the ascending/descending order of the areas of their bounding boxes [46]. Another method determines the high-congested areas of the grid, based on the current edge utilizations during the RRR procedure. For each net the minimum distance of its bounding box from the congested areas is computed, and defined as its criticality. The nets are then ordered in the ascending order of their criticalities [25], [66]. Recently, [14] proposes a circular fixed-ordering (CFOMR) algorithm for net ordering. The approach orders nets in a decreasing order of wirelength. Consequently, the ordering will not be fixed between two consecutive iterations of RRR.

The modern global routers consider the cost function of edge e to be expressed

as $c_e = (b_e + h_e)p_e$, where b_e is a "base cost", h_e is an added cost reflecting the congestion history, and p_e is the penalty for the current congestion. Different global routers consider different variations in setting b_e, h_e, and p_e. For example in FGR [46] and BFGR [25], the history term h_e in the expression is initially 1 and incremented by a constant (h_{inc}) at the beginning of each round of RRR if edge e has violation. The choice of h_{inc} affects convergence time and solution quality: higher values lead to faster convergence but a high wirelength [25].

The above procedure is repeated until a termination condition is met. The termination can be set to when there is no overflow of routing resources, or a time bound specified by the user is reached.

Layer assignment is a procedure in which three-dimensional routes for each net are reconstructed from solutions obtained from the projected two-dimensional graph. The work [46] shows that there exists a three-dimensional routing solution that has the same total overflow as the two-dimensional solution. Note that this is true if all metal layers have equal wire sizes and spacings. In the context of modern designs where wire sizes and spacings vary among different metal layers, the overflow of the three-dimensional solution can be larger than that of the projected two-dimensional solution. Consequently, the layer assignment procedure needs to consider overflow minimization beside the traditional objective of via minimization. Several methods to assign the routes to layers have been proposed, including the ones based on Integer Linear Programming (ILP) [11], dynamic programming [8], [66], greedy heuristics [25], [46], as well as Congestion-Constrained Layer Assignment (COLA) [34], and Congestion-Relaxed Layer Assignment (CRLA) [14].

The order in which the nets are considered in layer assignment procedure is also

important for determining the topology of the route generated for each net.

After layer assignment, some routers perform an **optional clean-up pass** on the three-dimensional graph to further minimize wirelength or the total overflow of the global routing grid (Figure 2.8).

Most of the recent global routers [8], [9], [12], [25], [46], [66] use the above mentioned procedure and route the nets in a sequential manner. In this approach the nets are sorted according to a specific criteria. The global router then visits the nets sequentially in this sorted order, and generates a path for each net.

The main drawback of the sequential approach is that the quality of the routing solution strongly depends on the procedure for net ordering. In fact, no matter which order is considered, the nets that are routed at the end of each RRR iteration are usually difficult to route. The reason is that the nets routed earlier consume a major part of the available routing resources. Also there are some cases that even if the design is routable, the global router fails to find a feasible solution due to the wrong net ordering. The optimality of the solution also depends on the net ordering. Different net orderings may result in different solution qualities. These issues can be addressed using a concurrent approach that considers all the nets simultaneously. The concurrent router replaces the core RRR procedure in Figure 2.8.

The known techniques for concurrent global routing are based on Integer Programming [12], [24], [63], [64]. The global routing problem can be formulated as an integer linear program (ILP). As an example in the ILP formulation provided in GRIP [64], for each net i, different ways of routing the net are identified (e.g. using pattern routing or maze routing). Suppose that there are n possible Steiner trees t_1^i, t_2^i, ..., t_n^i to route net i. For each tree t_j^i, a binary variable x_{ij} is defined. $x_{ij} = 1$, if

net i is routed using tree t_j^i, $x_{ij} = 0$ otherwise. The ILP formulation must enforce that only one tree must be selected for each net (e.g. $\sum_{j=1}^{n} x_{ij} = 1$).

A variable o_e is also defined for each edge of the global routing grid. This variable measures the amount of overflow of the corresponding edge. For each edge e, the number of routes that pass it, can not exceed the summation of its normalized capacity and the corresponding o_e value.

Typically, the objective of the ILP formulation is defined to minimize a linear combination of the total wirelength and the total overflow of the nets.

Integer programming is in the class of NP-complete problems [41]. The global routers that are based on Integer Programming, typically utilize a hierarchical, divide-and-conquer strategy. BoxRouter [12] is a global router that relies on the concept of box expansion and progressive integer linear programming. It spreads out the congestion from the most congested regions to the less congested ones by incrementally solving the ILP formulation. GRIP [64] is another concurrent global router that divides the global routing problem into many sub-problems. It solves each sub-problem individually, and finally connects them. It demonstrates significant wirelength improvement compared to the existing global routers, but it suffers from the long runtime. PGRIP [63], a parallel version of GRIP, further reduces the runtime of GRIP by solving the sub-problems in parallel while maintaining the large wirelength improvements, but in turn it requires a large number of processors.

Table 2.1 compares different global routers from various aspects. Even though the sequential routers follow an identical flow as shown in Figure 2.8, each may have a specific feature compared to other routers. Among these recent modern routers in Table 2.1, GRIP [64] and PGRIP [63] are the only routers that use a purely concurrent

approach. Most recently, the work [65] proposes a multi-level global router with significant improvements both in wirelength and runtime.

Table 2.1: Comparison between different global routers.

Router	**Net Decomposition**	**Two-terminal Routing**	**Net Ordering**	**Layer Assignment**	**Sequential/ Concurrent**	**2D/3D**
NTHU-R 2.0 [8]	RSMT	Pattern A* Routing	Current Congestion	COLA [34]	Sequential	2D
BFGR [25]	RMST	A* Routing	Bounding Box Current Congestion	Greedy [25]	Sequential	2D
FastRoute [66]	RSMT	Pattern A* Routing	Bounding Box	Dynamic Programming	Sequential	2D
MGR [65]	RSMT	Pattern 3D & 2D Maze	Bounding Box	Dynamic Programming 3D Routing	Sequential	2D
NCTU-GR [14]	RSMT	Monotonic A* Routing	CFORM [14]	CRLA [14]	Sequential	2D 3D
GRIP [64]	RSMT	A* Routing	–	–	Concurrent	3D
PGRIP [63]	RSMT	A* Routing	–	–	Concurrent	3D

2.1.4 Shortcomings of the Existing Procedures

Routing has become increasingly challenging with each technology and in recent modern designs, several new factors complicate the routing 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. None of the above-mentioned global routers, including the most recent one (MGR [65]) is designed to effectively and comprehensively handle this model. For example, just considering the layer assignment step, the internal procedures of [8], [9], [11], [46], [66] all require the net terminals to be on the first metal layer.

Furthermore, while many global routers have been developed and each tool has a unique strength, there is a lack of a fast and robust global router that has a consistent

operation on all the industrial benchmarks. For example, FastRoute [66] uses a greedy form of RRR and is orders of magnitude faster than other routers. However, for a subset of the ISPD 2008 [27] benchmarks, the runtime of FastRoute is significantly high [66]. NTHU-Route [16] uses a new history cost function, and compared to other sequential global routers reports the best wirelength for most of the ISPD 2008 benchmarks. However, it does not consistently have the best wirelength for all the benchmarks. The work [25] compares the routing quality of the ISPD 2008 contest winners using a new set of benchmarks. The new routing benchmarks were generated using the same netlist (from the ISPD 2008 benchmark suites [27]) but with a different placement density factor. It shows that the top routers failed to generate a violation-free solution with a slight change in placement density of the same netlist. In contrast, BFGR [25] successfully routes all the new instances without violation, while for the ISPD 2008 benchmarks, it generates solutions with higher wirelength than NTHU-Route and FastRoute [25].

Additionally, all of the recent academic global routers target minimizing the total wirelength of the routes under routing resource constraints. As feature size in advanced VLSI technology continues to shrink, other factors such as timing, power, and manufacturablity are becoming even more important.

2.2 Routing Congestion Prediction

Two distinct categories of congestion estimation methods have been proposed in the past to help the designer get an estimate of the locations of the routing congestion regions at the early stages of the design flow which are discussed in this section.

2.2.1 Metric-based Methods

These methods estimate routing congestion without performing global routing, and generate a map of the congested regions quickly. One variation is the work [10] which performs congestion estimation using a bounding-box model for each net. Specifically, the routing demand of each net is computed using its bounding box length. While this approach is fast, the work [60] showed that it fails to capture the routers' behavior accurately. Another set of approaches are based on probabilistic metrics for estimating congestion. For example, the approaches in [29], [37], [48], [61] assume that all nets are routed with their shortest length. They consider a subset of possible routing solutions and assign a probability of occurrence for each one. Then the route with the highest probability is selected and an expected horizontal and vertical usage is computed for each track with consideration of routing blockages. The work [37] presents a net-based stochastic model for 2-pin nets for probabilistic congestion estimation.

The work [67] estimates congestion based on modeling using Rent's rule to estimate the peak and regional routing demands. Pin density is another metric that has been used for congestion estimation. This metric may account for congestion estimation inside each global cell, but cannot model the congestion on the boundaries of adjacent global cells. The work [6] combines pin density with a probabilistic analysis-based model for more accurate congestion estimation.

It was empirically shown that metric-based methods are not able to effectively guide a routability driven placer [43] due to a high mismatch between what these approaches predict as routing usage and the actual routing usage after global routing. Furthermore, these techniques ignore the new complicating factors in modern designs that significantly contribute to routing congestion. For instance, they assume equal

wire sizes and spacings among different metal layers. However, modern designs have varying wire sizes and spacing for different metal ayers. Moreover, these metrics are not able to accurately capture the problem of local net congestion and irregular routing blockages. To further complicate the matter, these techniques consider all pins to be on the lowest metal layer and miss to model the congestion contribution of virtual pins residing on higher metal layers.

2.2.2 Estimation Using a Fast Global Router

These methods apply fast global routing, for example by performing only a few iterations of ripup and re-route in sequential routers, in order to generate a congestion map and are consequently slower than metric-based methods.

Academic global routing procedures have been continuously improving in the past few years [5], [7], [8], [11], [12], [14], [16], [22], [25], [33], [34], [39], [42], [46], [63], [64], [65], [66]. Recent advances both in runtime and quality of the global routers, allow designers to use global routing for congestion estimation. For example in [47], the authors integrate NTHU-Route [8] into a routability-driven placement tool that report significant routability improvement. In this work, NTHU-Route is called in one iteration of ripup and re-route to estimate the congestion.

The work [15] integrates global routing and placement to improve routing congestion and wirelength. After a short run of global routing, this technique uses wirelength-reduced cell shifting and cell rearrangement by bipartite matching to minimize both wirelength and routing congestion of the design. The global routing procedure that is utilized in this study, works on two-dimensional instances of global routing.

Other works that have considered integrating global routing and placement procedures for routability improvements include ROOSTER [45] and IPR [43]. ROOSTER [45] uses rectilinear spanning minimal trees as its routing congested model during global placement, and IPR [43] integrates FastRoute 2.0 [42] into FastPlace [57] and performs global routing during placement to achieve better routability results.

None of the above procedures captures contributors to routing congestion such as varying wire sizes, local congestion and other new factors. Furthermore, they all are based on traditional objective of total overflow minimization. In the next chapter we will show that when there is a tight runtime budget, overflow minimization can not provide an accurate congestion estimation.

In [51] we present CGRIP as a fast and flexible congestion estimation tool that can quickly and accurately identify the locations of highly-utilized regions on the layout. Furthermore, CGRIP considers many complicating factors that contribute to the routing congestion in modern designs, a few of them being, a large number of metal layers with distinct wire sizes and spacings, irregular routing blockages, and virtual pins on higher metal layers. Moreover, CGRIP is based on our new congestion model that aims to identify problematic regions and provide quick feedbacks for routability-driven placement procedures in a short amount of time. In the next chapter we will present CGRIP in details.

Chapter 3

CGRIP: A Framework for Routing Congestion Analysis in Modern Designs

In this chapter we present CGRIP, our framework for congestion analysis in global routing using Integer Programming. CGRIP is a fast and accurate congestion analysis framework which relies on a new and flexible model of global routing, formulated as an Integer Programming problem, that aims to find problematic routing regions on the layout. CGRIP captures many complicating factors that are major sources of routing congestion in modern designs, such as varying wire sizes and spacings, irregular routing blockages, and virtual pins.

We start with a motivational example in Section 3.1 that describes the benefits of regional minimization of overflow for more accurate congestion estimation. Next, we present our formulation and congestion analysis framework in Sections 3.2 and

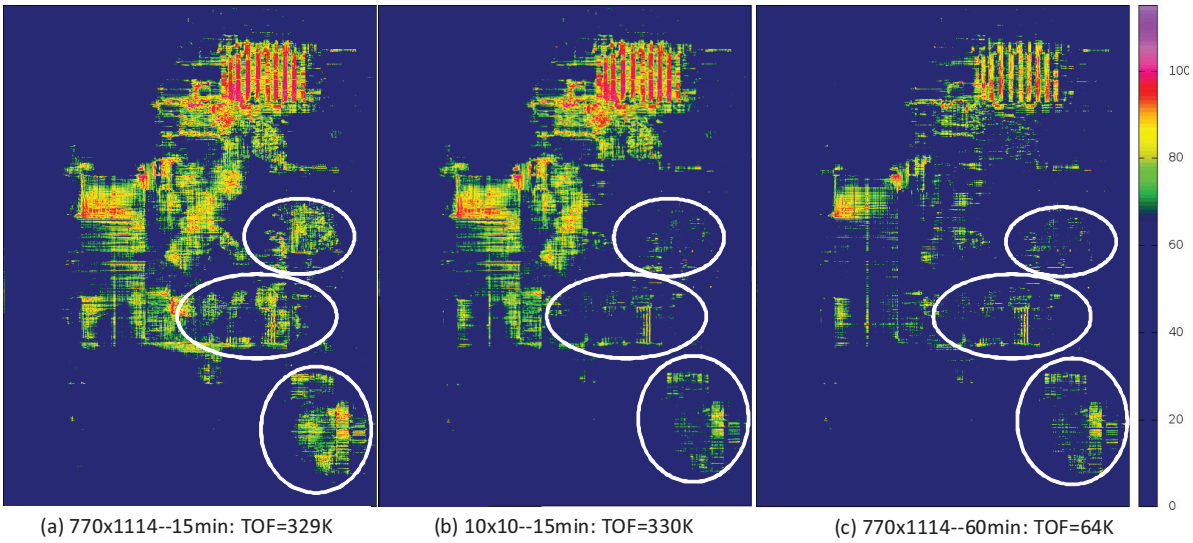

Figure 3.1: Different routing congestion maps corresponding to the same placement of benchmark instance `superblue2`, obtained from the tool Ripple [2] for (a) minimizing the total overflow on the global routing grid in 15 minutes, (b) regional minimization of overflow on a coarser grid in 15 minutes, and (c) minimizing the total overflow on the global routing grid in 60 minutes (reference case).

3.3, respectively. In Section 3.4, we present coalesCgrip, a simpler variation of our framework which was used as the reference router to judge the ISPD 2011 contest on routability-driven placement [2]. In Section 3.5, the computational experiments are offered to verify the effectiveness of CGRIP for accurate identification of congestion hotspots, using ISPD 2011 benchmarks [2]. In addition we also evaluate the solution quality obtainable by our framework when minimizing the total overflow of routing resources. Section 3.6 offers concluding remarks.

3.1 Motivational Example

Consider Figure 3.1 which shows three different congestion utilization maps of the `superblue2` benchmark instance [56]. This is corresponding to the (same) placement

instance generated by the tool Ripple [21] which was downloaded from the ISPD 2011 contest web site [2]. Each map is a 2D projection which 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. The maps are created by CGRIP, 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 identification of the regions whose overflow can be reduced using a routability-driven placer [47] rather than by introducing long detours. The total overflow (TOF) of each map is also reported.

The reference case is map (c) which is generated by minimizing the total overflow and running the router for 60 minutes, after which the map does not change drastically. Map (a) is obtained by running the router to minimize the total overflow but for a shorter runtime budget of 15 minutes. Map (a) has mismatch with map (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 ripup 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.

Map (b) is generated by running the router for 15 minutes when defining the regions using a 10x10 grid. It matches better with map (c). Due to more accurate

matching of map (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 spots for a 15 minutes time budget. This is despite similar overflow values in maps (a) and (b).

3.2 Problem Formulation

The example presented in Section 3.2 suggests that a congestion analysis tool cannot focus solely on minimizing total overflow as its optimization objective for a short runtime budget. Here we present our analysis tool which breaks the routing area into regions and uses an Integer Programming model that considers total overflow and maximum overflow found within each region.

3.2.1 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 $v_x \in \{1, \ldots X_G\}$ and $v_y \in \{1, \ldots Y_G\}$. 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

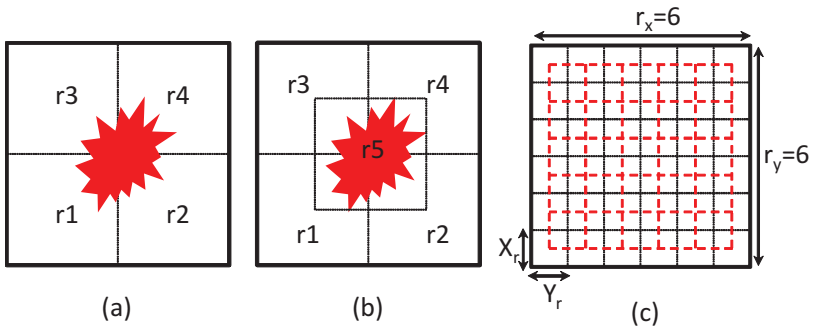

Figure 3.2: Overlapping and non-overlapping regions.

closest regions). In the case that r_x does not evenly divide X_G or r_y does not evenly divide Y_G, smaller regions may be created on the top and right boundaries of G.

Furthermore, in our framework, we provide an option to extend the definition of the regions to the case when they may be overlapping with each other. To understand the benefits of defining overlapping regions, consider the example in Figure 3.2. In Figure 3.2(a), four non-overlapping regions (r1 to r4) are initially defined. The area shown in red is a congested area of the chip. As this example shows, all the four regions have overlap with the congested area and are equally-ranked in terms of the degree of their congestion. Now consider 3.2(b) in which we allow defining overlapping regions. A new region, r5, is defined which has equal overlap with regions r1 to r4. In this case, the congested area is completely located inside r5. Consequently, r5 is reported as the most congested region and regional overflow is captured more accurately.

Assume the non-overlapping regions are defined by a grid-graph with lower-left coordinate of $x = y = 0$ and width and height of x_r and y_r for each region. Then the overlapping regions are added by defining a new grid-graph with the same region size and the lower-left coordinate of $x_r/2, y_r/2$ as shown in Figure 3.2 by the red and dashed lines.

3.2.2 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 $\mathcal{N} = \{T_1, T_2, \ldots, T_N\}$ (with $T_i \subset V$) and normalized edge capacities b_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 Programming formulation for congestion analysis can be written as:

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

$$
\begin{cases}
\sum_{t \in \mathcal{T}_i} x_{it} = 1 & \forall i \in \mathcal{N} \\
\sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} a_{te} x_{it} \leq b_e + o_e & \forall e \in E \\
\sum_{e \in E(r)} o_e \leq \tau & \forall r \in \mathcal{R} \\
q_r \geq o_e & \forall r \in \mathcal{R}, \quad \forall e \in E(r) \\
x_{it} = \{0, 1\} & \forall i = 1, \ldots, N, \forall t \in \mathcal{T}_i \\
o_e \geq 0 & \forall e \in E \\
\tau \geq 0 & \\
q_r \geq 0 & \forall r \in R.
\end{cases}
$$

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 normalized overflow of the normalized capacity b_e on edge e. (The normalized edge capacity b_e is the maximum number of routing tracks that can pass e without causing overflow. Defining the value of b_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 q_r for each region $r \in \mathcal{R}$ to be the maximum individual normalized overflow of any edge $e \in E(r)$. The objective function is a convex combination of the maximum total normalized-overflow variable τ and the sum of the individual maximum-normalized-overflow variables q_r.

For the special case of maximum resolution ($|\mathcal{R}| = |E|$), the variable q_r is simply the normalized overflow of each edge, and τ is the maximum normalized overflow of all the edges. In this case, by setting $\kappa = 0$, a formulation similar to that of GRIP [64] 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 maps in Figure 3.1, the value $\kappa = 0$ was used in all cases. The maps (a) and (c) are generated at maximum resolution $|\mathcal{R}| = |E|$. Consequently, these two cases simplify to minimizing the total normalized overflow. For map (b), the parameters $r_x = r_y = 10$ were used.

Note, formulation (IP-CA) may take an arbitrary definition of the regions $\mathcal{R}$ which can be overlapping or non-overlapping.

3.3 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 [64]) or a large number of parallel CPUs (e.g., a few hundred in PGRIP [63]). In this section, we present a practical implementation of (IP-CA) suitable for running in a tight runtime budget.

Figure 3.3 gives a graphical overview of our framework. First, a 2D-projection of the routing instance together with an initial 2D solution is created. A ripup and re-route procedure is iteratively applied until a no-overflow solution is found, or a time limit is reached, or the improvement in normalized overflow is less than 5% in 3 consecutive iterations. A congestion-aware layer assignment procedure is performed in the last step.

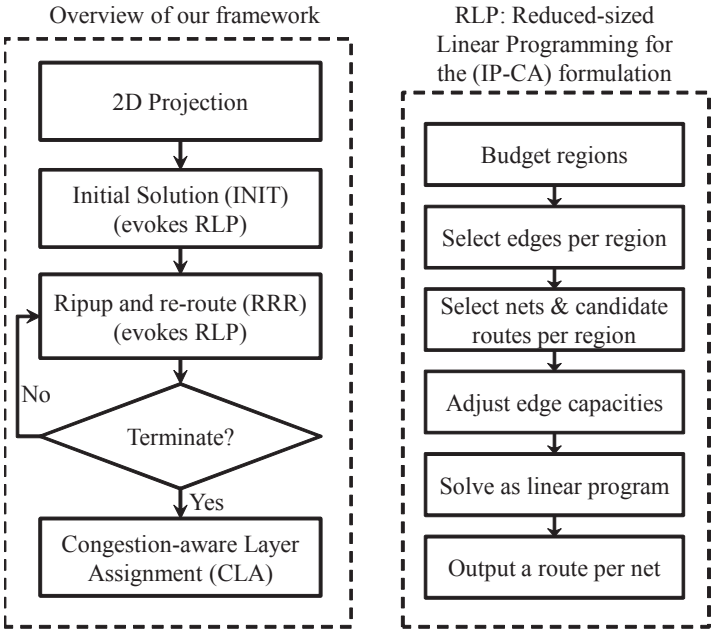

Figure 3.3: Congestion analysis framework

From a high-level of abstraction, our routing framework can be viewed as a variation of a standard ripup and re-route (RRR) framework similar to [8], [9], [11], [46], [66]. The novel aspects of our framework can be categorized into three sets of features. First, for a set of regions defined by the resolution parameter, we approximately solve the IP formulation (IP-CA) using a procedure referred to as reduced-sized linear programming (RLP). It approximates the original IP using a smaller and relaxed formulation. We further introduce a technique to use this reduced-sized formulation in order to create an initial solution and to integrate it within the current iteration of RRR. Second, we introduce various new techniques as extensions which can improve the speed and reduce the overflow in a standard RRR framework. Third, our procedure allows handling varying wire size and spacing on different metal layers, routing blockages, and virtual pins on the higher metal layers. This consideration is specifically made in creating a 2D routing instance and in our congestion-aware layer

assignment phase.

In the next subsections, we first describe the RLP step which is used in multiple steps of our frameworks. We then describe the new features in various steps of the framework in detail.

3.3.1 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. Note, the utilization of an edge e in our 2D projected graph model is defined as the *number* of routes passing from e. Specifically, we use the optimal dual variables π_e^* associated with the second set of constraints

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

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 normalized edge capacity b_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 used in a standard RRR procedure.

To obtain a practical analysis tool that works with a runtime-budget of a few minutes, each instance of RLP needs 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 3.3 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 normalized overflow q_r inside a region, as computed from the input solution. (The normalized overflow of an edge e in our 2D projected graph model is computed as the number of routes passing from e which exceeds its normalized capacity.) An important feature of this assignment is that regions 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 normalized 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 RRR. 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 3.3.3 and 3.3.4. 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. Non-critical nets are routed in a greedy fashion. Specifically, each non-critical net $T_i \in \mathcal{N} \setminus \mathcal{M}$ is routed using previously generated route for T_i that induces the least amount of additional normalized overflow up to that point. The routing

solution is used in 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 $\frac{\pi_e^*}{b_e}$, where π_e^* is the optimal dual value of its corresponding normalized edge capacity constraint in the reduced-sized LP. Further discussion of the weights is delayed to the discussion of RRR in Section 3.3.4.

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 are highly congested. We attribute this good performance to 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.

3.3.2 2D Projection

To create a 2D-projected grid-graph, for each edge in the 3D grid-graph, we compute its normalized capacity (i.e., b_e in (IP-CA)). This is done by dividing the routing capacity of the edge (i.e., length of the corresponding tile boundary after accounting for possible blockage) by the summation of the wire size and spacing for that layer. (Our definition of routing capacity is the same as given in the work [2].) In the 2D projection, the b_e of each edge e is the summation of the normalized capacities of the

corresponding edges in the 3D graph over different metal layers.

3.3.3 INIT: Initial Solution Generation

To generate an initial solution for our congestion analysis tool, we first apply maze routing for each two-terminal subnet obtained by decomposing the nets $\mathcal{N}$. The decomposition of a multi-terminal net is by finding the minimum spanning tree found on a graph with edges between each pairs of the net terminals. The weight of an edge is equal to the Manhattan Distance between its two corresponding terminals, similar to other works such as [9], [25], [46]. Maze routing is done by applying the A^* algorithm [20] 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. The utilization of a routed net is computed as the summation of the utilization of edges included in its current route, and the utilization of an edge is measured in terms of the number of routes (from various nets) which pass from the edge.

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 routes found by pattern routing similar to [24]. 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 for which 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 $T_i \in \mathcal{N}$.

3.3.4 RRR: Ripup and Re-route

A typical RRR iteration starts by updating the edge utilization (and consequently the edge weights which depend on them) using an input routing solution, either from the previous RRR iteration or from the initial solution. Next, the nets are decomposed in two-terminal subnets and then visited in a particular order. Every time a net is visited, its current route is removed and the subnet is re-routed using maze routing on the weighted graph.

Figure 3.4 shows the variation of RRR framework introduced in this work. It contains various new components which help to significantly speed up and effectively reduce the overflow. In our RRR framework, first, the utilization of the edges are updated using reduced-sized linear programming (RLP) discussed in the previous subsection. We show in our simulation results that RLP allows significant reduction in the overflow. To do ripup and re-route, we employ a new technique which we refer to as Multiple Ripup Single Re-route (MRSR). It allows ripping out multiple "equivalent" nets and re-routing them simultaneously using a single re-route opera-

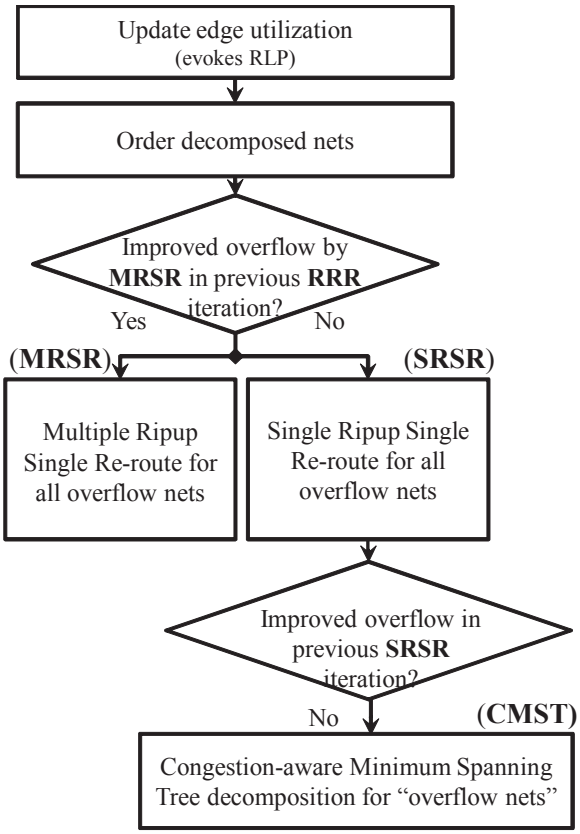

Figure 3.4: Overview of ripup and re-route

tion, and thereby significantly speeding up the process. MRSR is employed in each iteration of RRR until no improvement can be obtained to reduce overflow. At that point ripup and re-route is done by re-routing a single net at each time, as done in traditional RRR frameworks and is labeled by Single Ripup Single Re-route (SRSR) in the figure. When overflow can not be reduced further using SRSR, we change the subnets by changing the decomposition of multi-terminal nets based on the latest routing solution and employ a (one-time) congestion-aware minimum spanning tree (CMST) decomposition technique. The RRR iterations then continue with this new decomposition using SRSR until the termination condition has reached. Next, we explain these new components in detail.

Updating the Edge Utilizations

Each RRR iteration starts by applying the RLP procedure to create an updated routing solution and edge utilization values. Here we discuss how an RLP instance is created for a given resolution which requires discussing two aspects. First, a routing solution should be given as input to RLP. This is obtained either from the initial solution or from the previous RRR iteration. Specifically, for each net, the route is obtained as the union of the routes of its subnets. Second, we require as input a set of candidate routes for the selected critical nets which are used in the reduced-sized formulation. In the first RRR iteration, the candidate routes are the same as the ones used to generate the initial solution (i.e., the union of routes for the decomposed subnets of the nets). In the subsequent RRR iterations, we keep track of the routing solutions from the previous RRR iterations. Specifically due to memory restrictions, up to 10 candidates are stored for each net. These are stored from the latest RRR iterations. For the first RRR iterations the number of candidate routes will be smaller than 10. After running RLP a new routing solution is obtained and the edge utilizations are updated.

Net Decomposition and Ordering

The nets are next decomposed and ordered. To decompose a multi-terminal net into two-terminal subnets, first the route generated for the net using RLP is considered. Next, the Steiner points in this route are extracted and two-terminal subnets are defined for this net according to its route between its terminals and Steiner points. Note, since the route of a net may change in subsequent RRR iterations, its decomposition into subnets may also vary.

After obtaining the subnets they are ordered by considering the route generated for each two-terminal subnet using RLP one more time. For each subnet, we compute the total normalized overflow divided by the number of edges that have overflow on its route. Using this metric, subnet are ordered in ascending order so easy-to-fix nets with overflow are re-routed first and allowing more flexibility to reduce overflow on the nets with high overflow on their routes.

Setting up the Edge Weights

After the nets are ordered, we first compute 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.

For a critical edge $e \in D$, the penalty term is computed as p_e=pow(5, $\frac{-\pi_e^*}{b_e}$), where b_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 4.3.1. For a non-critical edge $e \in E \setminus D$, if it has overflow, then p_e=pow(5, $\frac{u_e - b_e}{b_e}$) and otherwise, $p_e = \frac{u_e}{b_e}$. 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 [25], [46], but our edge weight penalty function depends on the edge utilization values generated by RLP. Furthermore, our history term h_e is also updated *within* an RRR iteration, as the nets are traversed. Specifically, if an edge is found to have overflow for the first time, or to have an increased normalized overflow compared to its previous RRR iteration, h_e is incremented by h_{inc}. In contrast in [25] and [46] the edge histories are not changes during an iteration.

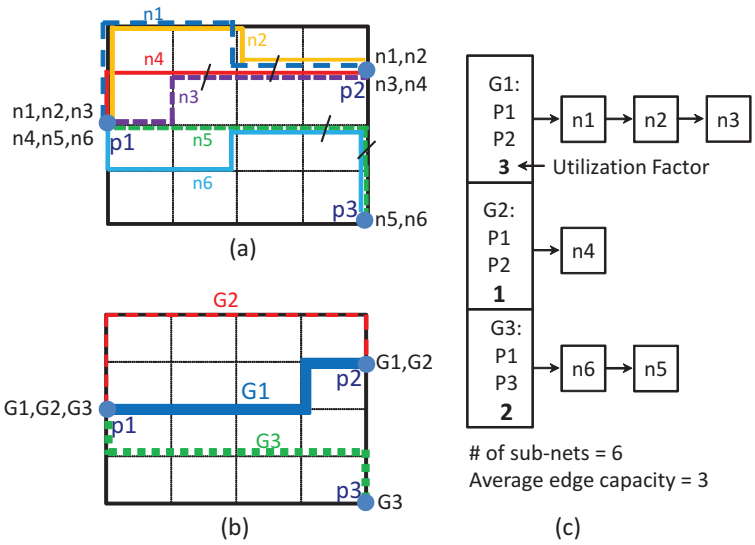

Figure 3.5: Multiple Ripup Single Re-route (a) subnets before MRSR, (b) subnets after MRSR, (c) equivalent subnet grouping

Multiple Ripup Single Re-route

When traversing the nets by the described ordering, nets that contain overflow in their RLP-generated route may be re-routed. Re-routing is done either by a multiple ripup, single re-route (MRSR) process, or a single ripup, single re-route (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 3.4.) 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 using the defined edge weights.

If both MRSR and SRSR fail to provide any overflow improvement, we change the net decomposition with respect to the current congestion estimated from the last RRR iteration. This is done for all (multi-terminal) nets that their corresponding routes

pass from the edges that have overflow. 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 b_{avg}, the average normalized capacity of the edges in the 2D grid.

For example, in Figure 3.5(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 $b_{avg} = 3$, 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 3.5(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 re-routed, all subroutes in the group are simultaneously removed, updating the available normalized edge capacities accordingly. A single re-route 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 3.5(c), shows 3 re-route steps for groups G_1, G_2 and G_3. The routes are shown with different thicknesses in Figure 3.5(c) to represent their varying contributions to the edge utilizations. In this example, overall we ripup 6 subnets but re-route 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 sig-

nificantly 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 ripup single re-route for the remainder of the RRR iterations.

Congestion-aware Multi-Terminal Net Decomposition

Using the SRSR procedure, if normalized overflow does not improve in two consecutive RRR iterations, we apply a one-time and new decomposition step on multi-terminal nets and change the creation of two-terminal subnets based on the estimation of congested areas. We refer to this step as congestion-aware minimum-spanning tree (CMST) -based decomposition. After this one-time decomposition, RRR proceeds until either a time-limit has reached, or a zero-overflow solution is found, or the improvement in overflow in 3 consecutive RRR iterations is less than 5%.

To explain our congestion-aware decomposition procedure, we first define a congestion metric p_e for each edge of the grid graph as below:

$$p_e = \max(1.0, \frac{u_e}{b_e}). \tag{3.1}$$

where u_e and b_e are utilization and normalized capacity of the edge in the 2D projected graph, respectively. The utilization of each edge is computed assuming each net is routed using the latest RRR iteration. According to Equation (3.1), if an edge does not contain overflow, then its weight will be equal to 1. However, the cost will be higher than 1 if the edge contains overflow.

For each net that its latest route generated by RRR contains an edge with overflow a new net decomposition is performed. Specifically, we consider its latest route with

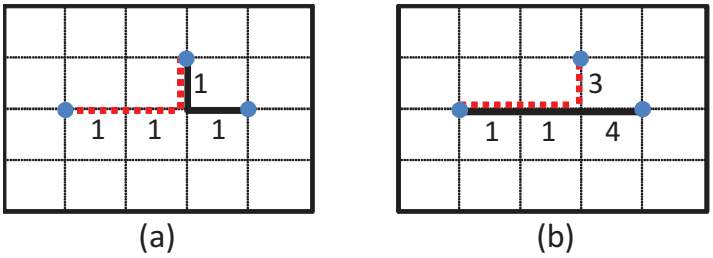

Figure 3.6: Net decomposition based on edge utilization

the edge weights defined using Equation (3.1). The decomposition is found by solving for the minimum spanning tree on a fully-connected graph with each terminal of the net as one vertex. The weight of an edge is the summation of the congestion metrics of the edges in the path connecting the two terminals in the route of the net.

As an example, Figure 3.6 shows the decomposition of a net for a routing tree. Figure 3.6(a) shows the decomposition into two subnets (shown using bold and dashed lines) when the edge weights are 1 unit. This decomposition results in subnets which have the small Manhattan Distance from each other. Figure 3.6(b) shows the minimum spanning tree using the defined edge weights. The subnets are not necessarily close to each other with respect to Manhattan Distance, and instead with respect to consideration of routing congestion.

Congestion-driven Steiner tree construction has been considered in the previous global routing procedures. For instance in [42] the authors first generate the Steiner minimal trees for all the nets using FLUTE [13], and convert all the multi-terminal nets into two-terminal subnets. Then, they generate a quick congestion map by applying pattern routing for all the subnets. Finally, they change the Steiner tree of the nets that go over congested regions so that the edges that have overflow of routing resources are avoided. The congestion map in this approach is not accurate and the Steiner trees are fixed during iterations of RRR. The work [46] uses a similar

approach, but instead of Steiner trees, this work generates minimum spanning trees and changes them based on the estimated congestion. Compared to these works, we apply our congestion-aware net decomposition after several iterations of RRR where we have more accurate congestion map. Furthermore, our technique provides more flexibility for congestion improvement by starting with minimum spanning tree and expanding it to Steiner minimal trees.

3.3.5 CLA: Congestion-Aware Layer Assignment

The congestion-aware layer assignment (CLA) step converts the generated 2D route of each net into a 3D route. In Chapter 2, we reviewed the existing approaches for layer assignment. An assumption in these techniques is that the wire size and spacing are equal for different metal layers. Hence, the objective is to assign layers to different segments such that the overflow does not change and the wirelength is minimized. However, when wire sizes and spacings are different the overflow after layer assignment may not be equal to the overflow of the 2D routing solution.

We introduce a new technique for layer assignment which considers wire size and spacing and minimizes wirelength and also results in low overflow. Our layer assignment procedure assigns the wire segments as much as possible to the lowest layer since the wire sizes are the least in these layer and consequently the imposed overflow due to additional wires assigned beyond the normalized capacity of the boundary of the two global cells is lower. To achieve such an assignment, for each edge in the 2D routing grid-graph, we first increase the capacity of the edge on the 3D graph which is located on the lowest layer. The increase is such that for each edge the 2D solution becomes a feasible solution.

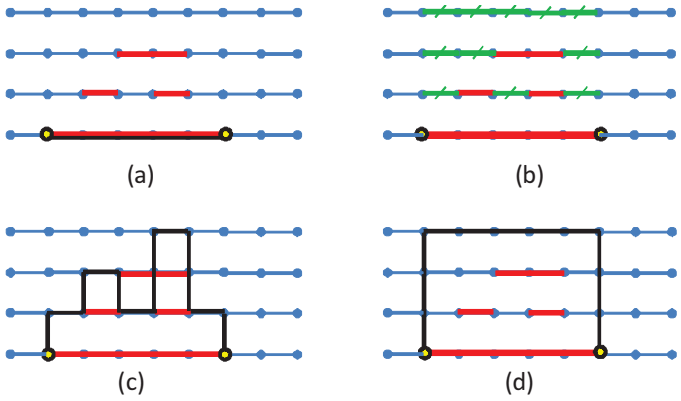

Figure 3.7: Layer assignment.

Next, the Steiner points of each route are identified to break the route into non-overlapping segments. These segments are sorted in ascending order of the number of bends in their corresponding routes and then visited in this order. Each segment is then assigned to one (or more) layers using the following strategy and based on solving a shortest path algorithm for each wire segment. First, all horizontal and vertical edges included in the segment are identified in the 2D graph. For each identified edge e, all the corresponding 3D edges from different layers which have edge e as their 2D projection and do not contain overflow are identified next. This allows considering a small subset of relevant edges in the 3D graph without considering the 3D graph in its entirety. The two end-points in the segment are then connected using a shortest path algorithm in which the remaining (un-identified) edges will be associated with a cost of infinity and thus avoided during the procedure.

Figure 3.7 shows an example of identifying the reduce graph and applying the shortest path algorithm for layer assignment of one wire segment. In Figure 3.7(a) we show a flat horizontal segment and the generated 2D route is shown as a black line. Assume that there are four horizontal layers in the 3D routing graph. The red

lines in this figure represent the edges that have no more capacities to use for routing. In Figure 3.7(b), the green edges are all the edges that can be considered for the layer assignment of the considered segment. Figure 3.7(c) shows the layer assignment using a greedy approach and Figure 3.7(d) shows the result of layer assignment using our technique. As this figure shows our technique results in less number of vias after layer assignment.

If a segment includes a terminal located in a layer other than M1, then the shortest path algorithm starts from that layer. If two terminals are on the same location but on different metal layers, a via connection will be made to connect the two terminals. As the segments are assigned to the layers, the procedure updates the corresponding edge utilizations by accounting for layer-specific wire sizes and spacings. During the CLA step, all the edges that are designated as blockage with a 0 capacity are avoided.

3.3.6 Data Structures

The route of each net is represented and stored in two ways: First, for each subnet a union of edge indexes representing the corresponding route fragment is stored. Second, the route of each net is represented as a union of edge indexes of all its subnets. While it is possible that an edge is shared among multiple subnets of the same net, it will only appear once using the second representation. Note that an important consideration needs to be taken during ripup and re-route of a multi-terminal net. Consider an edge that is shared in the route of more than one subnet in which the subnets all belong to the same net. Since such subnets are ripped up and re-routed independently, the same edge may be incorrectly ripped up for multiple times and its utilization incorrectly updated more than once. In our implementation,

we ensure a common edge in the routes of the subnets of the same net is ripped up only for the last subnet which includes the edge. Furthermore, when updating the edge utilizations, various subnets of the same net together may not contribute more than one unit to the utilization of each edge.

This data structure is similar to the branch-free representation in [25]. However, please recall that our subnets may have Steiner points as a terminal. Furthermore, we enhance the implementation of the above data structure to make it suitable in our framework. In the case when the routes of the subnets of a net overlap in an edge, the edge index is represented in each subnet separately. This repetition greatly increases the memory requirement in our framework in which up to 10 candidate routes are stored per net from the previous RRR iterations during the RLP step. To fix this problem, we only use the branch-free data structure to represent the latest route generated for a net during RRR. The other *candidate* routes of a net are each stored only as a list of edge indexes.

3.4 coalesCgrip

The congestion analysis framework presented in Section 3.3 was implemented as a tool which we refer to as CGRIP. A simplified variation of this framework with fewer features, called coalesCgrip was used as the reference router to evaluate the ISPD 2011 contest [2]. A major difference between CGRIP and coalesCgrip is the use of an external tool for generating the initial solution instead of the INIT step which was explained in Section 3.3.3. Specifically, to generate an initial solution, coalesCgrip uses a modified version of FGR [46]. We modified the source code of FGR to handle

the ISPD 2011 benchmark requirements. In addition, the RLP step in coalesCgrip is only used when formulation (IP-CA) is solved for the case of maximum resolution to minimize the total overflow as the objective. Handling of lower resolutions in coalesCgrip follows a different procedure than CGRIP. In the case of lower resolutions, the Integer Programming formulation in coalesCgrip does not change, but the critical edges are selected such that they are distributed among different regions. For ripup and re-route (RRR), it uses a net ordering method based on the bounding boxes of the decomposed nets. Also the history term of an edge weight is not updated within an RRR iteration in coalesCgrip. It also lacks the MRSR and CMST steps described in Section 3.3. It has a layer assignment step based on greedy assignment of wire segments to layers without exhaustive overflow consideration. The greedy assignment algorithm is similar to [25].

3.5 Simulation Results

Both the tools CGRIP and coalesCgrip were implemented in C++ and evoked 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 [56]. These benchmark instances include different wire sizes and spacings for 9 metal layers, irregular routing blockages, and net terminals located at any metal layers.

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

3.5.1 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 [2] and then applied our global routers. We make a comparison between CGRIP and coalesCgrip used in the ISPD 2011 contest.

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.

In Table 3.1 we compare the wirelength (WL scaled to 10^{-5}) and total overflow (TOF). We note again that the solutions generated by these tools avoid routing on the specified blockages. Even though routing on the blockages and counting it as overflow can yield to less overflow, using blockages is against the blockage definition. (A solution that uses blockages still passes the ISPD 2011 evaluation script.) So our TOF numbers are generated by avoiding blockages and are reported for the generated 3D routing solution using the contest evaluation script. The TOF value for each benchmark reflects the routing resource usage and not the number of violated tracks. So assigning routes to lower layers is a good strategy to reduce the total overflow.

The results are reported in Table 3.1. Columns 2 and 3 report the grid size and

Table 3.1: Comparison when minimizing the total overflow

Benchmark	X_GxY_G	#Nets	Tool	coalesCgrip		CGRIP	
				TOF	WL	TOF	WL
superblue1	704x516	822744	SimPLR	0	150.24	0	151.14
superblue2	770x1114	990899	Ripple	797898	307.73	115332	330.14
superblue4	467x415	567607	Ripple	85538	108.57	1414	114.38
superblue5	774x713	786999	Ripple	126186	172.86	20252	180.56
superblue10	638x968	1085737	RADIANT	616742	250.16	59306	267.84
superblue12	444x518	1293436	SimPLR	415428	228.85	35298	243.16
superblue15	399x495	1080409	Ripple	125936	179.11	12832	192.17
superblue18	381x404	468918	mPL11	31440	98.44	0	104.34
average				**9.0X**	**0.9X**	**1.0X**	**1.0X**

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 of coalesCgrip is 9.0X higher with respect to CGRIP. In terms of wirelength, coalesCgrip has slightly lower wirelength. On average over the benchmarks, the relative wirelengths of CGRIP and coalesCgrip are 1.0X and 0.9X, respectively.

3.5.2 Impact of Different Features

We proposed several new features for a practical realization of the IP formulation for industry-sized benchmark instances. Our second experiment is to show how these features individually impact the quality of the solution and runtime when minimizing the total overflow. Specifically, we evaluate the impact of each of these features: RLP, MRSR, and CMST.

Table 3.2 shows the impact of different techniques. For this experiment, we first disabled the three features and ran CGRIP. We refer to this case as CGRIP-all. We compare the TOF and WL with the case when all the features are enabled which is referred to as CGRIP+all which is the same as CGRIP. We compare these two cases

Table 3.2: Impact of Individual Features of CGRIP

Benchmark	CGRIP-all		CGRIP+RLP		CGRIP+MRSR		CGRIP+CMST		CGRIP+all	
	TOF	WL	TOF	WL	TOF	WL	TOF	WL	TOF	WL
superblue1	0	153.37	0	151.38	0	151.16	0	151.16	0	151.14
superblue2	723346	320.27	361042	324.06	499872	320.83	499872	320.83	115332	330.14
superblue4	35942	112.77	4076	115.19	15066	112.46	16694	112.29	1414	114.38
superblue5	63794	178.94	25678	181.00	37540	178.55	37996	178.38	20252	180.56
superblue10	1035144	260.27	206516	261.51	386206	261.12	386206	261.12	59306	267.84
superblue12	385182	245.00	184484	250.48	177548	244.81	177548	244.81	35298	243.16
superblue15	120140	187.72	36836	192.14	44644	187.21	37526	187.14	12832	192.17
superblue18	1404	104.30	0	104.97	0	104.32	200	104.05	0	104.24
average	**1.0X**	**1.0X**	**0.3X**	**1.0X**	**0.5X**	**1.0X**	**0.5X**	**1.0X**	**0.1X**	**1.0X**

with three other cases when each feature is enabled individually. These three cases are CGRIP+RLP, CGRIP+MRSR, CGRIP+CMST when only RLP, or MRSR, or CMST is enabled in each case and the other two features are disabled. For all the above five cases, we impose a 15-minute runtime budget as the stopping criteria.

The results show that RLP, MRSR, and CMST can improve the total overflow on average by 0.3X, 0.5X, and 0.5X respectively compared to CGRIP-all. When we use the three features jointly, the total overflow is reduced on average by 0.1X compared to CGRIP-all.

Note that for some benchmarks, when RLP, MRSR and CMST are disabled (i.e., CGRIP-all), the total overflow is still less than coalesCgrip which was given in Table 3.1. The overflow improvement in these cases are a result of other improvements such as data structure enhancements, new net ordering, and edge history update in an RRR iteration.

3.5.3 Identifying and Ranking the Regions with Overflow

Our next 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 $r_x \times r_y$ = 10x10 for a time-budget of 15 minutes.

When a lower resolution than the grid size is considered, two cases are compared:

- **NOV**: Overlap between regions is not allowed. Consequently regional minimization of overflow is performed for 100 regions when $r_x \times r_y$ = 10x10.
- **OV**: Overlapped regions are considered as discussed in Section 3.2.1. In this case when $r_x \times r_y$ = 10x10, 181 regions are defined.

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. Consequently the TOF values in this experiment are higher than the previous experiment. The blockages are similarly not allowed to be used for routing. The purpose of this experiment is to measure how closely solutions obtained with a short run-time budget match the reference solution. We measure for the following two cases.

Table 3.3: Evaluation of CGRIP for identifying and ranking the overflow regions

Benchmark	Non-overlapping regions						Overlapping regions					
	maxRes60 (ref)		maxRes15		lowRes15		maxRes60 (ref)		maxRes15		lowRes15	
	TOF	$\lvert\mathcal{R}_c\rvert$	TOF	%Err	TOF	%Err	TOF	$\lvert\mathcal{R}_c\rvert$	TOF	%Err	TOF	%Err
superblue1	504	2	514	0.00	514	0.00	504	5	514	0.00	514	0.00
superblue10	35764	72	198480	27.43	202234	8.12	35764	141	198480	13.43	286544	2.02
superblue12	29380	80	183568	26.95	186356	10.02	29380	145	183568	13.85	190432	9.80
superblue15	3694	68	41552	36.20	52658	15.12	3694	133	41552	16.18	52898	10.40
superblue18	158	29	10046	66.67	11426	31.39	158	55	10046	35.69	12444	22.78
superblue2	64366	69	329090	25.53	330044	8.18	64366	131	329090	12.38	335882	7.12
superblue4	584	41	16918	54.50	24204	28.46	584	80	16918	36.20	25322	21.09
superblue5	1296	43	48902	58.05	72740	20.37	1296	79	48902	27.47	78924	14.20
average	**16968**	**51**	**103634**	**36.92**	**110022**	**15.21**	**16968**	**96**	**103634**	**19.40**	**122870**	**10.93**

Non-Overlapping Regions

We first run lowRes15 when overlap between regions is not allowed and store the routing solution. We also run maxRes15 and maxRes60 for which the maximum resolution is used and as a result, the total overflow is minimized. After obtaining a routing solution in each of the above three cases, we super-impose a 10x10 grid-graph identical to the regions defined by the 10x10 resolution parameter and rank the resulting 100 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 actual regions with more congestion, as specified by the reference case, and are sorted with respect to the descending order of TOF in each region. We refer to these regions as the critical regions and denote them by the set $\mathcal{R}_c$.

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 $r \in \mathcal{R}_c$ 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 $Err = \frac{\sum_{r \in \mathcal{R}_c} |\text{rank}_{r,maxRes60} - \text{rank}_{r,maxRes15}|}{|\mathcal{R}_c|}$.

Overlapping Regions

We run lowRes15 again and this time we allow for overlap between regions as discussed in Section 3.2.1. Note, it is not required to run maxRes15 and maxRes60 since they are ran in the maximum resolution and we use the routing solutions for these cases from the previous run. Then for each case we impose a 10x10 resolution on the grid graph and allow overlap between regions. This results in 181 regions in each case. We again sort each list in descending order based on the total overflow inside the regions. For maxRes15 and lowRes15 we compute and report the *Err* metric as defined in the previous case.

Analysis of Results

The results are reported in Table 3.3. The number of critical regions $|\mathcal{R}_c|$ defined by the reference case is reported for each of the NOV and OV cases. For maxRes15 and lowRes15, we report the *Err* metric for the NOV and OV cases. For each case, we also report the total overflow (TOF) of the generated routing solution. Note that in lowRes15, the total overflow in NOV is different from that in OV, because the number of regions considered in RLP is different which results in different objective expressions used during this step. For maxRes15 and maxRes60 the TOF is equal in NOV and OV. The reason is that the solutions for the maximum resolution cases are obtained when minimizing the total overflow, and the *same* solution is evaluated for two types of overlapping and non-overlapping regions.

The results show that for both maxRes15 and lowRes15 cases, the *Err* metric is

lower when overlapping regions are considered, indicating a more accurate ranking of regions with overflow with respect to the reference case. For example in maxRes15 for benchmark instance `superblue2`, the *Err* metrics are 12.38% for OV and 25.53% for NOV which is more than twice the OV case. On average for all the benchmarks, the *Err* metrics in maxRes15 are 19.40 and 36.92 for OV and NOV cases, respectively. The number of regions with overflow on average are 51 and 96 in NOV and OV cases respectively.

The table also shows the *Err* metric in lowRes15 for the NOV and OV cases. Despite the fact that overlap may or may not be allowed, lowRes15 always provides a better raking compared to maxRes15 in ranking the regions with overflow. The average *Err* metric for lowRes15 for the NOV and OV cases are 15.21% and 10.93% respectively, while for maxRes15 these numbers are 36.92% and 19.40%, respectively. For benchmark instance `superblue10`, the average *Err* metric in maxRes15 for the NOV case is 27.43%. For the OV case, this number is reduced to 13.43%. For this benchmark, for lowRes15, the *Err* is only 2.02%.

The computational results indicate that for the same runtime budget of 15 minutes, the combination of lowRes15 and OV always offers a significantly lower value of *Err* than other cases and thus can achieve a more accurate ranking of regions with overflow. 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 regions containing congestion. With similar TOF between lowRes15 and maxRes15, we conclude that the results of lowRes15 is more valuable since it can more accurately rank the regions containing overflow.

3.5.4 Evaluation of Other Congestion Metrics

In this section we report other congestion metrics recently mentioned in [4]. The first metric is based on defining a "global cell congestion" metric in order to evaluate the congestion of each routed net. The second and third metrics evaluate the congestion of each routed net without considering the congestion of the global cells that are passed by the routed net. These metrics are defined below:

Metric 1) avg 20%: In this metric, we first define a congestion number for each global cell using which we define a congestion number for the route of each net. For each benchmark, we report the average of the congestion numbers of the top 20% nets (which have the highest congestion numbers).

Specifically, we first compute a congestion metric for each global cell. This is done by considering the edges in the grid-graph connecting to the vertex representing the global cell. For each edge we compute its utilization normalized to its capacity, representing the "edge congestion". The "global cell congestion" is then defined to be equal to the maximum congestion among the edges connecting to it.

Next, for each net we define its congestion as the maximum of the congestion of the global cells which are contained in its route. We then consider the nets with highest value of this "net congestion" metric and report the average value of the net congestion for the top 20% of the nets.

Metric 2) N100: This metric represents the number of nets with corresponding routes passing from the edges in the grid-graph which have a utilization above their capacities.

Metric 3) N90: This metric represents the number of nets in which a net has the corresponding route passing from at least one edge utilized more than 90% of its

Table 3.4: Evaluation of congestion metrics

Benchmark	maxRes60			maxRes15			lowRes15					
				%diff.			%diff. in NOV			%diff. in OV		
	N100	N90	avg 20%	N100	N90	avg 20%	N100	N90	avg 20%	N100	N90	avg 20%
superblue1	380952	436796	100.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00
superblue2	526680	585962	104.81	0.88	0.85	10.14	0.00	0.85	0.60	1.43	1.48	0.13
superblue4	275367	305327	100.22	0.00	0.24	3.39	-0.03	0.24	1.04	0.44	0.66	-0.09
superblue5	388842	437555	100.28	0.30	0.23	5.30	-0.01	0.23	0.72	-0.58	0.45	0.22
superblue10	605745	673386	103.84	1.40	1.44	10.31	2.51	1.44	0.49	5.85	3.08	-0.47
superblue12	869753	928233	101.43	0.85	0.56	4.28	1.15	0.56	1.62	1.15	3.60	0.64
superblue15	528972	573975	100.77	0.24	0.35	3.14	2.73	0.35	-0.16	4.46	2.14	-0.06
superblue18	266773	288023	100.08	0.82	0.85	1.75	7.13	0.85	0.91	12.82	5.92	0.01
average	**480386**	**528657**	**101.44**	**0.56**	**0.56**	**4.79**	**1.68**	**0.56**	**0.65**	**3.20**	**2.17**	**0.05**

capacity.

In Table 3.4 we report the above congestion metrics for the maxRes60 as the reference case. We then consider other cases of maxRes15 and lowRes15. For lowRes15 we consider two cases of no overlap (NOV) and overlap (OV). In these cases, for each metric, we report the percentage of difference with respect to this reference case.

First, considering the first metric (avg 20%), Table 3.4 shows that lowRes15 always results in a lower value of this metric compared to maxRes15, indicating a closer match with respect to the reference case. Overlapping regions help to further reduce this metric in the lowRes15 case. However, the second and third metric (N100 and N90) are lower in the maxRes15 case. This is justified because the goal of our framework is identifying the locations of the highly-congested regions on the chip which correlates better with the first metric defined based on congestion of a global cell. However, the second and third metrics which are defined based on number of nets do not target matching in terms of location of the congested regions.

3.6 Concluding Remarks

In this chapter we presented CGRIP, a new routing congestion analysis tool. CGRIP operates on a flexible model of global routing that captures various factors that contribute to routing congestion in modern designs. We proposed an Integer Programming formulation that targets routing congestion analysis. It includes new constraints and a new objective in order to find and rank congested regions on the layout according to an input resolution parameter. In addition to the new model, CGRIP includes several new ideas to account for routing in modern designs and for large industrial benchmarks. These ideas include a reduced-size linear programming, a multiple ripup single re-route procedure, congestion-aware net decomposition, and congestion-aware layer assignment. In the computational experiments, we verified the claim of more accurate identification of congestion hotspots in a small analysis time budget, when a low resolution is used to define the regions. Specifically, we showed that given a 15-minute runtime budget, the most-congested regions can be estimated more accurately using our framework when the regions are defined using a low resolution (of 10x10), compared to the case when they are defined with maximum resolution corresponding to minimizing the total overflow as done in standard global routing procedures. In addition, our framework can also be used as a standard global router to minimize the total overflow while considering modern design factors. In this case, we used our framework to minimize the total overflow and made comparison with coalesCgrip, a simpler variation of the framework which was used as the reference router to judge the ISPD 2011 contest on routability-driven placement and was also discussed in this chapter. CGRIP achieves roughly an order-of-magnitude improvement in the total overflow when compared to coalesCgrip.

Chapter 4

LCGRIP: Planning for Local Net Congestion in Global Routing

In Chapter 3, we presented CGRIP, a fast congestion analysis tool that handles many complicating factors in modern designs. However, CGRIP ignores the impact of local congestion caused by local nets that are located completely inside a global cell. This chapter offers two contributions in order to estimate and manage the local nets at the global routing stage. First, a procedure is given to generate global cells of non-uniform size in order to *reduce* the number of local nets and thus the cumulative error associated with ignoring or approximating them. Second, we *approximate* the resource usage of local nets at the global routing stage by introducing a capacity for each global cell in the global routing graph. With these two complementary approaches, we offer a mathematical model for the congestion-aware global routing problem that captures local congestion with non-uniform global cells along with other complicating factors of modern designs including variable wire sizes, routing blockages, and virtual pins.

In this chapter, we present a binning procedure to reduce the number of local nets by creating global cells of non-uniform sizes in Section 4.1. Approximation of local nets and the generalized graph and Integer Programming models are given in Section 4.2. The details of our routing framework are given in Section 4.3. Simulation results are presented in Section 4.4. Section 4.5 offers concluding remarks.

4.1 Non-Uniform Global Cells

In Chapter 2, we mentioned that global routing divides the chip area into global cells. The nets which are located completely inside a single global cell are defined as local nets. These nets are ignored in a standard global routing procedure. However, modern designs typically have high pin densities within a global cell which creates a large number of local nets. As a result, lack of handling the local nets at the global routing stage, results in significant increase in effort at the detailed routing stage to create a routable design. To reduce the error due to ignoring or approximating the local nets at the global routing stage, we propose to define the global cells in a non-uniform manner in order to reduce the number of local nets. Reducing the number of local nets increases the number of global nets (since the total number of nets is constant) and thus creates a more difficult global routing instance. However, the effort required for detailed routing can be significantly reduced. We will give computational results in Section 4.4 indicating that the extra effort applied during global routing pays off, generating improved overall designs with less total computing time.

In this section, we first provide mathematical notation for describing local nets in

the presence of global cells of unequal size. We then present our non-uniform global cell generation procedure.

4.1.1 Notation and the Binning Problem

To more accurately account for local effects in global routing, we revisit how instances for global routing arise out of the design process. At the most fundamental level, we are given a 3-dimensional *placement grid*

$$P = \{0, 1, \ldots X\} \times \{0, 1, \ldots, Y\} \times \{1, \ldots L\}.$$

In most instances, like the ISPD 2011 benchmark, there are $L = 9$ layers in the grid [56].

As mentioned in Chapter 2, the design problem also consists of a set of nets $\mathcal{N} = \{T_1, T_2, \ldots, T_{|\mathcal{N}|}\}$, where each net $T_n \in \mathcal{N}$ consists of a set of *pin locations* on the placement grid P. As a first step, each net with multiple pins is decomposed into two-terminal subnets based on a minimum spanning tree connecting the pins. Therefore, after decomposition, each net consists of two *pin locations*: $(T_n = \{p_1^n, p_2^n\})$. The pin locations are coordinates on the placement grid $(p_k^n = (x_k^n, y_k^n, \ell_k^n) \in P)$, and typically all pin locations are in the first layer $(\ell_k^n = 1)$. However virtual pins outside of the first layer are also possible. For example, in the ISPD 2011 benchmarks, virtual pins are located at layer 4.

To create an instance that can be solved by global routing software, this very detailed placement grid P is replaced with a less-refined version, where coordinates of the placement grid are aggregated into global cells.

Each layer $\ell \in \{1, \ldots, L\}$, consists of a number of *global cells* (gcells),

$$\mathcal{C}_\ell = \{C_{ij\ell}\}_{i=1,\ldots,N_x^\ell, j=1,\ldots,N_y^\ell},$$

where each global cell $C_{ij\ell}$ is a rectangular region on a fixed level of the placement grid P. In our work (and in all benchmark instances), we will assume that a global cell $C_{ij\ell}$ is characterized as a pair of intervals

$$C_{ij\ell} = ([\alpha_i^\ell, \alpha_{i+1}^\ell], [\beta_j^\ell, \beta_{j+1}^\ell]), \tag{4.1}$$

and the intervals have the property that they partition individual coordinate axes of the placement grid, i.e.

$$0 = \alpha_1^\ell < \alpha_2^\ell < \ldots < \alpha_{N_x+1}^\ell = X \tag{4.2}$$

$$0 = \beta_1^\ell < \beta_2^\ell < \ldots < \beta_{N_y+1}^\ell = Y \tag{4.3}$$

We define the selection of the intervals $([\alpha_i^\ell, \alpha_{i+1}^\ell], [\beta_j^\ell, \beta_{j+1}^\ell])$ for the global cells as the *binning problem*. In most global routing instances, the global cell intervals have a uniform size. For example in the ISPD 2011 benchmark instances, all global cell intervals are of length 40. In Section 4.1.2 we discuss advantages of creating instances whose global cells are of different sizes.

4.1.2 Binning Procedure

When selecting intervals that define global cells, we do not wish to change the total *number* of global cells, just their individual sizes. Specifically, N_x^ℓ and N_y^ℓ will re-

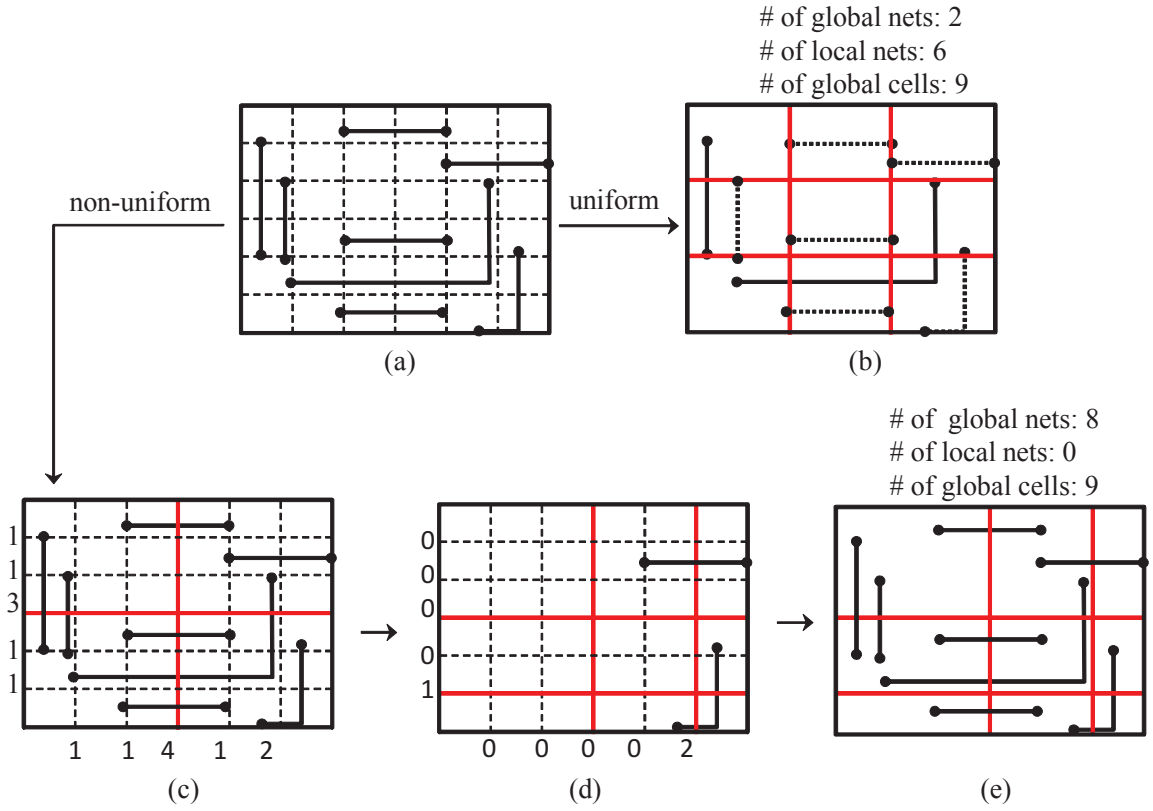

Figure 4.1: Binning to create non-uniform global cells.

main unchanged in Equations (4.2) and (4.3), but the cell starting locations $\alpha_i^\ell, \beta_j^\ell$ in Equation (4.1) will be adjusted by the binning procedure. For purposes of making a 2D-projection of the instance straightforward (see Section 4.3.1), the binning procedure will keep all global cells to be of uniform size between layers, i.e. $\alpha_i^{\ell_1} = \alpha_i^{\ell_2}$ and $\beta_j^{\ell_1} = \beta_j^{\ell_2} \quad \forall i, j, \ell_1, \ell_2$.

We explain the procedure for global cell definition using the example shown in Figure 4.1. We assume that each multi-terminal net is decomposed into two-terminal subnets based on its Minimum Spanning Tree (MST). The MSTs are shown on the placement grid in (a). The standard instance has 9 equal-sized global cells specified by 3x3 grid shown with bold red lines (b). This global cell configuration results in

2 global nets and 6 local nets. Our procedure for defining global cells is based on an iterative bi-partitioning. At each iteration, both a horizontal and vertical cut are made to maximize the number of nets that are cut. For example, in (c) the maximum number of nets cut in the horizontal and vertical directions are 3 and 4, respectively, and the corresponding cuts are shown. After each iteration, the nets that are cut are removed from consideration (as shown in (d)). The process completes when the number of intervals N_x^ℓ and N_y^ℓ are of the requisite size. By allowing for global cells of non-uniform size, the number of local nets is reduced to 0, but the number of global nets is increased to 8, as shown in (e).

The binning procedure reduces the number of local nets by maximizing the number of nets that pass a vertical or horizontal cut at each step of the algorithm. It is also computationally useful to consider a parameterized version of this algorithm that controls the tradeoff between the increase in global nets and the decrease in local nets. In the parameterized version of the procedure, when deciding the location of a vertical or horizontal cut at each iteration, we select the cut location that results in the number of cut nets to be closest to ηN_{max}, where N_{max} is the maximum number of nets that can be cut at that iteration. The user-specified parameter η is between 0 and 1 and allows for more fine-tuned control of the eventual number of local nets. A higher value of η results in a larger reduction in the number of local nets.

Post-Processing for Local Congestion Balancing: Another important consideration in the binning problem is to generate global cells with "balanced" local congestion. Therefore, after generating non-uniform global cells using the above procedure, we post-process the intervals, perturbing the boundaries of the global cells in order to balance the local congestion ratios of the global cells. For each global cell C_{ijl}, we

define the local congestion ratio LC_{ijl} as

$$LC_{ijl} = \frac{R_{ijl}}{A_{ijl}}, \tag{4.4}$$

where R_{ijl} denotes the routing resources consumed by local nets inside C_{ijl} and A_{ijl} is the area of global cell C_{ijl}. (We discuss a method for estimation of R_{ijl} in the next section.)

The post-processing step is a greedy heuristic with the objective to decrease the deviation of the congestion ratios among the global cells, while ensuring that the number of local and global nets remains the same. The procedure sequentially considers the impact of adjusting each cut line (e.g., up and down for horizontal cuts), computing the updated local congestion ratios and number of global and local nets if the interval boundary was changed. The new cut line location is chosen to be the one that does not change the number of local and global nets; and results in the maximum decrease in total deviation of the global cells' congestion ratios from the average. The latter helps to reduce the deviations in the local congestion ratios among the global cells as a mean to balance the local congestion.

The local congestion balancing approach can improve the quality of both global and detailed routing stages. In global routing, after binning, it is possible that a global cell has a large number of local nets, while a neighboring global cell does not contain any local net. The definition of vertex capacity in our formulation limits the number of global nets that can pass over these global cells. The local congestion balancing approach decreases the likelihood of these cases. Furthermore, in detailed routing the local nets are typically routed inside the corresponding global cell. Distributing local nets among different global cells using our local congestion balancing approach

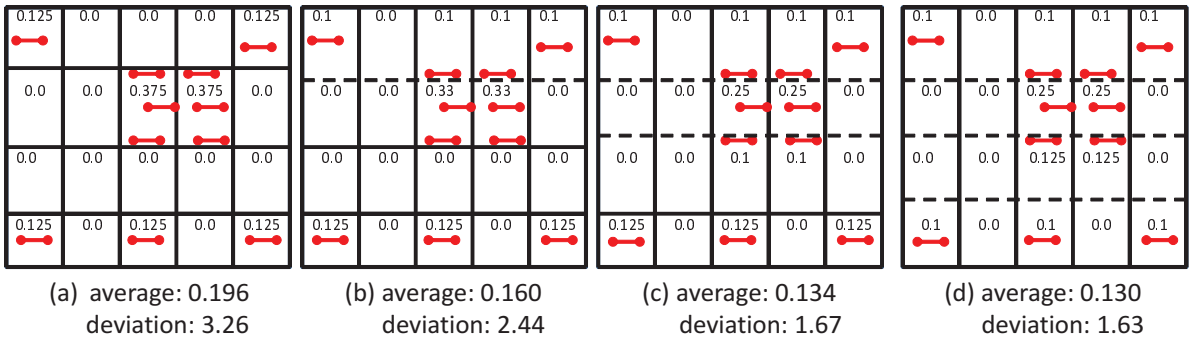

Figure 4.2: Post-processing example.

will dedicate more routing resources to local nets. Consequently, it can alleviate the routability issues corresponding to local nets at the detailed routing stage.

The effect of post-processing is depicted in Figure 4.2. In Figure 4.2(a), the congestion ratios are shown for each global cell, calculated assuming the dimensions of each global cell is 4×4, and local congestion is computed using the bounding box of each local net (which is 2 for all local nets). Figures (b) to (d) show the steps of post-processing for the horizontal cuts. In Figure (b), the top cut line (shown by the dashed line) is slightly moved down and as a result some of the local nets are moved from the bottom global cells to their neighboring top global cells. The new congestion ratios corresponding to this step are shown in the global cells and the total deviations in the congestion ratios are reduced from 3.26 to 2.17, compared to the average in each case. Similarly, in Figures 4.2(c) and (d) the remaining horizontal cut lines are visited and in the end the total deviation in the congestion ratios is reduced from 1.67 to 1.63 in Figures (c) and (d), respectively.

Recall, an important impact of the binning procedure is increase in the number of global nets. Another important consideration when applying binning is the amount of routing blockage in the design. For example, in the `superblue10` benchmark

instance, over 67% of the first four metal layers are defined as routing blockages. In such a case, generating bins of non-uniform size can significantly complicate the global routing procedure and add to its complexity. Therefore, in our framework, we only apply the binning procedure if the amount of routing blockage compared to the total chip area does not exceed a threshold. In such a case, we only apply the less-invasive post-processing step which does not change the number of global nets and balances the local congestion by re-distributing the existing local nets.

4.2 Local Congestion Modeling

In this section we describe mathematical models for approximating the routing resources consumed by local nets while additionally considering factors such as non-uniform wire size, wire spacing, and routing blockages. These models are embedded into an Integer Programming (IP) formulation to describe a congestion-aware global routing problem.

4.2.1 Local Congestion and Graph Models

To create a global routing instance, the global cell information is used to construct a graph $G = (V, E)$, where

$$V = \cup_{\ell \in \mathcal{L}} \mathcal{C}_\ell,$$

with one vertex $v = (i, j, \ell) \in V$ for each global cell $C_{ij\ell}$. Edges $e \in E$ are created between adjacent global cells. In a given layer, all wiring tracks are oriented in one direction. Additionally there are *vias* to allow for routes to move between adjacent layers. Therefore, each edge $e = ((i_1, j_1, \ell_1), (i_2, j_2, \ell_2)) \in E$ is of one of the following

three types:

- (EW): if $\ell_1 = \ell_2 = \ell$, $j_1 = j_2$, and $i_2 = i_1 + 1$ or $i_2 = i_1 - 1$,
- (NS): if $\ell_1 = \ell_2 = \ell$, $i_1 = i_2$, and $j_2 = j_1 + 1$ or $j_2 = j_1 - 1$,
- (VIA): if $i_1 = i_2, j_1 = j_2$ and $\ell_2 = \ell_1 - 1$ or $\ell_2 = \ell_1 + 1$.

We assume without loss of generality if ℓ is odd, edges are of (EW) type, and if ℓ is even, edges are of (NS) type.

Pin locations for each net $T_n \in \mathcal{N}$ are mapped to global cell locations. Specifically, if pin $p_1^n \in C_{ij\ell}$ and $p_2^n \notin C_{ij\ell}$, then vertex $v = (i, j, \ell) \in V$ is a terminal of T_n that must be connected in the global routing instance. A net T_n is a *local net* for global cell $C_{ij\ell}$ if $p_1^n, p_2^n \in C_{ij\ell}$. As explained in Chapter 2, local nets are not explicitly considered by the global routing instance, which may result in difficulty at the detailed routing stage.

In the global routing instance, (EW) edges and (NS) edges $e = (i_1, j_1, \ell), (i_2, j_2, \ell) \in E$ have a normalized capacity b_e that depends on the length of the interval defining the global cell:

$$b_e = \begin{cases} \frac{\beta_{j_2}^{\ell} - \beta_{j_1}^{\ell}}{w^{\ell} + s^{\ell}} & \text{if } e \text{ is a EW edge} \\ \frac{\alpha_{i_2}^{\ell} - \alpha_{i_1}^{\ell}}{w^{\ell} + s^{\ell}} & \text{if } e \text{ is a NS edge} \end{cases} \tag{4.5}$$

where w^{ℓ} and s^{ℓ} denote the wire size and spacing in layer ℓ. By dividing the interval with $w^{\ell} + s^{\ell}$, the normalized capacity b_e reflects the number of wiring tracks that can pass between the boundary of adjacent global cells. The first layer $\ell = 1$ is not available for global routing in ISPD 2011 benchmarks, so the capacity is set to 0 for these edges. VIA edges are given a capacity $b_e = \gamma$ that is determined by the

technology. For the ISPD 2011 benchmark instances, $\gamma = \infty$.

To account for local effects, we also consider that each vertex $v = (i, j, \ell) \in V$ has a normalized *vertex capacity* r_v. The vertex capacity is designed to limit the total number of routes that can pass through a vertex (global cell). To capture local effects in a global routing instance, the vertex capacity should be reduced based on the area required to route local nets contained in global cell $C_{ij\ell}$. We estimate the area required to route local net $T_n = \{(x_1^n, y_1^n, \ell), (x_2^n, y_2^n, \ell)\}$ in level ℓ by multiplying the wire width by a length equal to the net's half-perimeter bounding box in the lowest possible levels:

$$d_n^\ell = \begin{cases} w^\ell |x_1^n - x_2^n| & \text{if } \ell = 3 \\ w^\ell |y_1^n - y_2^n| & \text{if } \ell = 2. \end{cases}$$

Let $\mathcal{A}_{ij} \subset \mathcal{N}$ be the set of nets local to cell C_{ij1}. The routing resources consumed by local nets at level ℓ is then

$$R_{ij\ell} = \begin{cases} \sum_{n \in \mathcal{A}_{ij}} d_n^\ell & \text{if } \ell = 2, 3 \\ 0 & \text{otherwise.} \end{cases}$$

Note that we make a practical assumption that the local nets of global cell C_{ij1} only impact the capacities corresponding to global cells C_{ij2} and C_{ij3}. Other models for approximating the routing usage of local nets can be incorporated in the above equations. If there are routing blockages inside a global cell C_{ijl}, the blockage area should be added to $R_{ij\ell}$.

The capacity of vertex $v = (i, j, \ell)$ is the area of the corresponding global cell reduced by the estimated area required for routing local nets. The capacity is normalized by the length of the appropriate global cell interval (depending on whether

v is in an (EW) layer or a (NS) layer), so that r_v becomes a measure of the number of routes that may enter or leave the global cell. Specifically, we let

$$r_v = \frac{A_{ij\ell} - R_{ij\ell}}{\lambda_{ij\ell}}, \tag{4.6}$$

where $A_{ij\ell} = (\beta_{j+1} - \beta_j)(\alpha_{j+1} - \alpha_j)$ and $\lambda_{ijl} = (\alpha^{\ell}_{i+1} - \alpha^{\ell}_i)$ if ℓ is even, and $\lambda_{ijl} = (\beta^{\ell}_{j+1} - \beta^{\ell}_j)$ if ℓ is odd.

Due to the normalization of the edge capacity b_e, the lengths of the intervals $\lambda_{ij\ell}$ should be divisible by the quantity $w^{\ell} + s^{\ell}$. Hence, in our binning procedure, at each iteration of the bi-partitioning, we may slightly perturb the index of the cut point (in the placement grid P), to ensure that all interval lengths remain divisible by $w^{\ell} + s^{\ell}$.

4.2.2 Integer Programming Model

Input to the Integer Programming model consists of the grid-graph $G = (V, E)$ and a set of multi-terminal *global* nets denoted by $\mathcal{N} = \{T_1, T_2, \ldots, T_N\}$ with $T_i \subset V$, created as described in Section 4.2.1. Let $\mathcal{T}_n$ be the *candidate global routes* for net T_n—the collection of all Steiner trees connecting the terminals of T_n. We define the parameters $a_{te} = 1$ if Steiner tree t contains edge $e \in E$, and $a_{te} = 0$ otherwise. Similarly, we define parameters $z_{vt} = 1$ if Steiner tree t contains vertex $v \in V$, and $z_{vt} = 0$ otherwise. The model contains binary decision variables x_t that are equal to 1 if and only if tree $t \in \mathcal{T}_n$ is used to route net T_n. An Integer Programming (IP) describing the global routing with the normalized vertex capacity is given as follows:

$$\min_{x,o,s} \sum_{n\in\mathcal{N}} \sum_{t\in T_n} p_t x_t + M_1 \sum_{e\in E} o_e + M_2 \sum_{v\in V} s_v \qquad \text{(IP-LC)}$$

$$\sum_{t\in\mathcal{T}_n} x_t \geq 1 \qquad \forall n \in \mathcal{N} \quad (\lambda_n)$$

$$\sum_{n\in\mathcal{N}} \sum_{t\in\mathcal{T}_n} a_{et} x_t - o_e \leq b_e \qquad \forall e \in E \quad (\pi_e)$$

$$\sum_{n\in\mathcal{N}} \sum_{t\in\mathcal{T}_n} z_{vt} x_t - s_v \leq r_v \qquad \forall v \in V \quad (\mu_v)$$

$$x_t \in \{0,1\} \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}_n$$

$$o_e, s_v \geq 0 \quad \forall e \in E, \forall v \in V$$

The parameter p_t is the cost of global route t, computed as the *lengths* of its corresponding edges: $p_t = \sum_{e\in E(T)} p_e$. For the non-uniform binning case, the length of each edge is computed as the distance between the centers of the corresponding global cells. In formulation (IP-LC), the first set of inequalities enforces the routing of each net using one of its candidate trees. In the second set of inequalities, o_e is a variable that measures the normalized overflow of the normalized capacity b_e on edge e. In the third set of inequalities, s_v is a variable that measures the normalized overflow of the normalized vertex capacity r_v. The objective is a linear combination of total global routed length, total normalized edge overflow, and total normalized vertex overflow. The two parameters M_1 and M_2 are large numerical constants, so that edge and vertex overflow are minimized.

Example: Figure 4.3 depicts a global routing instance that demonstrates the utility of vertex capacities. Subfigure (a) shows five global cells with the middle one having high local congestion. The number inside each global cell is an approximation of total number of routes that can cross its four boundaries without causing overflow.

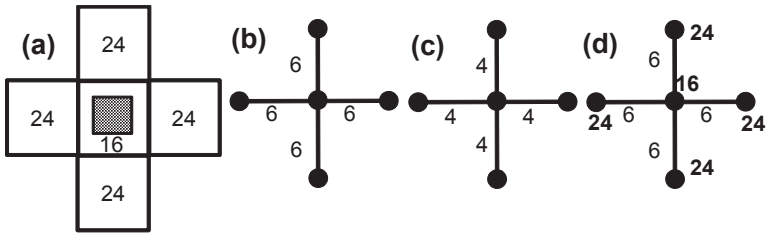

Figure 4.3: Different global routing graph models.

A conventional graph model would ignore the local congestion, resulting in the instance depicted in Subfigure (b). In this case, each edge is given a normalized capacity of 6, reflecting the maximum number of routes that can pass the boundary of the two corresponding global cells. Vertex capacities are not included in this model.

Subfigure (c) shows a model that accounts for the local congestion inside the middle global cell by reducing the number of routes that can pass from each of its boundaries to 4. This case accounts for the local congestion better than (b), but it does so by reducing edge capacities. However, accounting for local congestion by reducing edge capacities may be too restrictive. For example, a solution to the instance depicted in Subfigure (a) can have 6 routes crossing the top boundary, and this solution is excluded by reducing the edge capacities in (c).

Subfigure (d) depicts the alternative presented in this work, which uses the same edge capacities as (a). In addition, a vertex capacity of 16 is placed on the middle global cell. These combined constraints most accurately model the true instance, accounting for local congestion effects without unduly restricting the search space.

4.3 LCGRIP: Local-Congestion-Aware Routing Framework

In this section we describe our routing framework, which is an extension of CGRIP, described in Chapter 3. We refer to this extension of CGRIP to consider local con-

gestion as LCGRIP. By extending from CGRIP, this framework can also account for the same complicating factors—varying wire size and spacing at different metal layers, routing blockages, and virtual pins—as CGRIP. Note that CGRIP was designed for rapid congestion analysis. It relies on an input *resolution parameter*. In this extension, we set this parameter to be of maximum resolution.

4.3.1 Extensions

We now discuss how individual steps in CGRIP are revised to handle the new Integer Programming formulation (IP-LC) and local congestion models in the presence of non-uniform global cell sizes.

2D Projection

The 2D projection is done similar to CGRIP. It is done in a straightforward manner using our mathematical notations (presented in Section 4.2.1) and because even with non-uniform global cell sizes in each layer, we require uniformity across the layers in our global cell generation procedure. In our 2D projection, cells $C_{ij\ell}$, for a fixed location (i, j), whether of uniform or non-uniform size, are aggregated so that there is one global cell at each location (i, j) representing all the layers. Specifically, the 2D graph is $G_{2D} = (V_{2D}, E_{2D})$, where $V_{2D} = \{(i, j)\}_{i=1,\ldots,N_x, j=1,\ldots,N_y}$, and there is an edge $e = ((i_1, j_1), (i_2, j_2)) \in E$ if $i_2 = i_1 + 1$, $i_2 = i_1 - 1$, $j_2 = j_1 + 1$ or $j_2 = j_1 - 1$. Normalized capacity for edges $e = ((i_1, j_1), (i_2, j_2)) \in E_{2D}$ are computed by summing normalized capacity defined in Equation (4.5) over the metal layers:

$$b_e = \sum_{\ell=1}^{\mathcal{L}} b_{(i_1,j_1,\ell),(i_2,j_2,\ell)}.$$

Similarly, vertex capacity for $v = (i, j) \in V_{2D}$ is obtained by summing vertex capacity (Equation 4.6) over the metal layers:

$$r_v = \sum_{\ell=1}^{\mathcal{L}} r_{i,j,\ell}. \tag{4.7}$$

Initial Solution Generation

To create an initial solution, multi-terminal nets are decomposed into two-terminal subnets according to their Minimum Spanning Tree (MSTs), as in CGRIP. We note the same decomposition is used during the procedure to generate non-uniform global cells which was explained in Section 4.1. Otherwise accounting for local and global nets will be inconsistent between the generated global cells and the routing framework.

To generate an initial solution, the reduced-sized linear program (RLP) for the modified Integer Programming given by formulation (IP-LC) is solved to identify an initial solution from a set of pre-defined candidate routes obtained using maze and pattern routing. This procedure is similar to CGRIP except in the way the RLP is formed which will be explained next.

Reduced-Sized Linear Programming (RLP)

We form a reduced-sized version of (IP-LC) by writing the formulation only for a subset of optimization variables $o_{\hat{e}}$, $s_{\hat{v}}$ and $x_{\hat{t}}$ and the corresponding constraints that include these variables. Variables are identified by first identifying a set of *critical edges*, based on current utilization estimates. (The procedure for identification of critical edges is the same as CGRIP.) For each identified critical edge $\hat{e} = (\hat{v}_1, \hat{v}_2)$, the variable $o_{\hat{e}}$ and two variables $s_{\hat{v}_1}$ and $s_{\hat{v}_2}$ are added. This is because vertex capacities

are also considered in formulation (IP-LC) while they were ignored in CGRIP.

If a critical edge is used in the current solution to route a net T_n, then this net is a *critical net*, and a sufficiently large subset of routes $\hat{t} \in \mathcal{S}_n$ corresponding to candidate routes ($\mathcal{S}_n \subset \mathcal{T}_n$) will have their decision variables $x_{\hat{t}}$ included in the RLP. We note, this process to define the reduced version of formulation (IP-LC) depends on whether global cells are of uniform size. This is because both vertex and edge capacities are computed based on the global cell dimensions. We note however that our non-uniform global cell generation procedure ensures the size of the grid-graph (in term of number edges and vertices) remain the same as the uniform case.

Ripup and Re-Route (RRR)

The RRR procedure rips up nets with high normalized overflow and re-routes them. The nets to ripup are based on the computed normalized overflow in the current solution. The re-route step solves a weighted shortest path problem. In contrast to CGRIP where there are weights only on edges, the shortest path problem in our extension also has weights on vertices. This is because we consider capacity of edges and vertices.

Specifically, in CGRIP, each edge e has a weight of $1+f(\frac{g_e}{b_e})$, where f is an exponential function of an estimate of utilization of edge e, denoted by g_e, and the normalized capacity of e. Similar to CGRIP, if e is a critical edge, then the utilization g_e is the dual value corresponding to the normalized edge capacity constraint for e (π_e shown in (IP-LC)). Otherwise, g_e is computed as the number of global routes that use edge e in the most current routing solution.

In our framework, each edge $e \in E_{2D}$ has an edge weight of $l_e + f(\frac{g_e}{b_e})$, where l_e is

the length of edge e, computed as the distance between the centers of the two global cells that edge e connects. The role of l_e in this edge weight expression is to account for the used wirelength associated with each edge. In contrast, CGRIP uses $l_e = 1$ in its edge weight expression. This is because the cell sizes are equal in CGRIP and the relative contributions of the edges in terms of wirelength are equal to each other.

In our extension, during the re-routing, we also have weights for each vertex $v = (i, j) \in V_{2D}$. The vertex weights are $f(\frac{h_v}{r_v})$, where f is the same function used for the edge weight, h_v is an estimate of the vertex utilization, and r_v is the normalized vertex capacity (4.7). If v is an endpoint of a critical edge, then its utilization h_v is taken from the dual value corresponding to the vertex capacity constraint for v, denoted by μ_v in formulation (IP-LC). Otherwise, the utilization h_v is taken to be the sum of the edge utilizations for all edges incident to vertex v: $h_v = \sum_{e \in \delta(\{v\})} g_e$.

Congestion-Aware Layer Assignment

The RLP and RRR procedures are iterated for a pre-specified time limit, or until no additional overflow improvement is identified. The routes in G_{2D} are then converted into routes on G using a congestion-aware layer assignment procedure. CGRIP uses a greedy procedure for layer assignment which accounts for virtual pins, routing blockages and varying wire sizes and spacing. Here we extend the CGRIP procedure in two ways: (1) we use updated edge capacities which account for local congestion using our model which assumes the local nets are routed at the two lowest layers; and (2) computation of routing resource utilization in the greedy procedure of CGRIP is extended to account for non-uniform global cell dimensions. Specifically, the routing resource of an edge e in G is computed using an estimated length l_e and the corre-

Table 4.1: ISPD 2011 benchmark info

Bench	Placer	Grid Size	#Nets	#2T-Nets
superblue1	SimPLR	704x516	822744	2038444
superblue2	Ripple	770x1114	990899	2237446
superblue4	Ripple	467x415	567607	1316401
superblue5	Ripple	774x713	786999	1713307
superblue10	RADIANT	638x968	1085737	2579974
superblue12	SimPLR	444x518	1293436	3480633
superblue15	Ripple	399x495	1080409	2736271
superblue18	mPL11	381x404	468918	1395388

sponding wire size for that layer. The length l_e is computed as the distance between the centers of the two global cells corresponding to the two vertices of edge e.

4.4 Simulation Results

The routing framework described in Section 4.3 and the binning procedure of Section 4.1 to create global routing (GR) instances with non-uniform cells were both implemented in C++ and integrated with the CGRIP congestion analysis tool. For the binning procedure, the parameter η was set to 0.9, giving significant weight to reducing the number of local nets in the instance. For solving linear programs, CPLEX 12.2 [26] was used. ISPD 2011 benchmarks were used to validate our framework, and global routing instances were created from the winning placement solutions of the ISPD 2011 contest [2]. For each benchmark, Table 4.1 shows the placement solution used, the grid size, and the number of nets before and after terminal decomposition. These benchmarks consider non-uniform wire size and spacing, routing blockages, and virtual pins. They are specifically designed to be challenging instances for routability. For benchmark `superblue10` only the post-processing step was applied to generate non-uniform global cells because in this benchmark the amount of routing blockage in the first four metal layers was 67%.

Implemented Global Routing Variations: To demonstrate the effectiveness of our techniques, we implemented four global routing variations:

1. **U-E** (Uniform-Edge): Uniform grid with edge capacity only, without any adjustment for local congestion factors;
2. **U-AE** (Uniform-Adjusted-Edge): Uniform grid with adjusted edge capacity for local nets and no vertex capacity;
3. **U-AV** (Uniform-Adjusted-Vertex): Uniform grid with unadjusted edge but adjusted vertex capacity based on global cell congestion;
4. **NU-AV** (Nonuniform-Adjusted-Vertex): Non-uniform grid with unadjusted edge capacity and adjusted vertex capacity.

The method U-E is identical to CGRIP. Methods U-AV and NU-AV are the ones proposed in this work. Specifically, NU-AV includes all the proposed features and is same as LCGRIP. They both consider vertex capacity based on local congestion as well as (unadjusted) edge capacity. The only difference between them is whether or not global cells are of uniform size. The method U-AE adjusts the edge capacity b_e to account for local nets similar to Figure 4.3(c). Specifically, for each edge $e = (i, j)$ of type EW and NS, an adjusted number of routes that can pass the boundary is computed by calculating normalized vertex capacities r_i and r_j for its endpoints using Equation (4.6). The adjusted edge capacity is computed by replacing the numerator in Equation (4.5) by $\min(r_i, r_j)$.

The above tool variations are used to create four different global routing solutions for each benchmark. The termination criterion of the tool in all cases was set to

be when no additional improvement in overflow was obtained during the ripup and re-route (RRR) phase of the algorithm.

Detailed Routing (DR) Emulation: To measure the impact of different global routing solutions on the detailed routing stage, we (obviously) require a mechanism for performing detailed routing (DR). In this work, we had to implement our own *detailed routing emulator* to perform this evaluation. We would prefer to use an actual detailed routing tool, but at the time of this writing, there were no detailed routing tools, having an interface that takes a global routing solution as input, available to us.

We did obtain a binary of the detailed routing tool `RegularRoute`, kindly shared by the authors of [69]. `RegularRoute` is one of the most recent, competitive academic detailed routing tools. However, at this phase of its development, and similar to other detailed routing tools, `RegularRoute` does not account for the complicating factors introduced in the ISPD 2011 benchmarks, such as zero metal-1 capacity, virtual pins, and non-uniform wire sizes and spacings. Even with these restrictions, we attempted to use `RegularRoute` to perform detailed routing on the global routing solutions obtained by our techniques. To do this, we simplified the ISPD 2011 benchmarks by removing the zero-capacity on metal 1, removing the virtual pins, and making all wire sizes and spacings equal to each other. For these simplified ISPD 2011 instances, we generated four global routing solutions, corresponding to our four global routing variations. Each of these solutions was given to `RegularRoute` for detailed routing. Unfortunately, our binary of `RegularRoute` crashed due to the large ISPD 2011 benchmark sizes compared to the smaller-sized benchmarks used in the experiments of [69].

We also carefully evaluated the use of Cadence's `wroute` to perform detailed routing. Unfortunately for our purposes, `wroute`, similar to other commercial detailed routers that we considered, does not have an interface that takes a global routing solution as input. Rather, it accepts a *placement instance* as input. This same issue is also mentioned in the paper [69].

Thus, we created our own detailed routing emulator, designed to illustrate the impact of considering local congestion during global routing. The goal of the tool is to provide a relatively-accurate surrogate measure for the impact of detailed routing on given global routing solutions. We will show that the impact is significantly different among the global routing variations that were tested.

Our emulator uses a projected detailed routing instance that can be used to estimate the overflow occurring during detailed routing. (We note other published detailed routers such as [69] also work with a projected model and then apply a layer assignment step.) Specifically, our detailed routing emulator works on a two-layer grid, with one layer containing NS edges and one containing EW edges connected with VIAs. Each route in a given global routing solution is then projected to this two-layer grid. Each global cell in the projected instance has a detailed routing grid DRG superimposed upon it, which gives the same resolution as the placement grid P in the ISPD 2011 instances, ensuring that pins of all nets (global and local) are vertices of the detailed routing grid. The capacity of each NS/EW edge in the DRG is equal to the number of the NS/EW layers above that edge that do not contain an obstacle at that location.

Global cells are individually routed over the DRG in a sequential, breadth-first manner, starting from bottom left. When visiting global cell c, first a track assignment

is made for all global routes mapped to c in the current global routing solution. Some track assignments are imposed by neighboring (previously-visited) global cells. For global routes that connect to c through a VIA edge in the global routing model, a utilization of one unit of the DRG grid edge inside c is used to reflect this VIA usage. Once the track assignment is made for c, ripup and re-route (RRR) is applied to route the generated subnets of each global route and all local nets inside c. In our emulator, we do only one iteration of RRR, so that our emulator shows the immediate impact of the translation of a global routing solution into a detailed routing solution. When doing RRR, the same net ordering as in the global routing procedure is used. Each net inside c is routed using its shortest path after updating the routing resource usage by the previously-routed nets, similar to the global routing framework. Each net, however, is restricted to be routed within a bounding box of its terminals.

Evaluation Metrics: For each generated global routing solution, the overflow (denoted by GR-OF) is measured using the (unadjusted) edge capacities that are used in the U-E method. The wirelength of each case (denoted by GR-WL) is also measured. In NU-AV, the wirelength is computed while accounting for non-uniform global cells for fair comparison. For example, an edge in NU-AV which is twice than an edge in U-E due to non-uniform global cells is counted as 2 units of wirelength. For each global routing solution, the total overflow computed by the detailed routing emulator (denoted by DR-OF) is computed.

Comparison of Evaluation Metrics: Table 4.2 shows comparison of GR-OF, DR-OF, and GR-WL for each tool variation. The results confirm the following:

- Methods U-AE, U-AV, and NU-AV, which account for local nets, result in a reduced DR-OF of 0.9X, 0.7X, and 0.4X, respectively compared to U-E.

Table 4.2: Comparison of wirelength and total overflow at various stages

Benchmark	**U-E (CGRIP)**			**U-AE**			**U-AV**			**NU-AV(LCGRIP)**		
	GR-OF	**DR-OF**	GR-WL	GR-OF	**DR-OF**	GR-WL	GR-OF	**DR-OF**	GR-WL	GR-OF	**DR-OF**	GR-WL
superblue1	0	**23142**	153.36	0	**23020**	154.20	0	**12740**	154.65	0	**806**	154.67
superblue2	3168	**18880**	335.80	14496	**18506**	335.36	10526	**13154**	344.46	7756	**9140**	350.33
superblue4	228	**28696**	114.40	2024	**27476**	115.12	880	**13296**	119.20	420	**888**	119.80
superblue5	0	**10878**	184.74	322	**9256**	187.45	0	**2588**	187.78	448	**1032**	188.80
superblue10	124	**84842**	270.32	4502	**73862**	282.65	872	**66780**	281.01	766	**65232**	281.23
superblue12	0	**44556**	256.58	274	**44416**	264.11	302	**36414**	259.03	12232	**15566**	261.21
superblue15	0	**29982**	191.81	1022	**29800**	192.32	846	**18886**	192.01	2582	**7922**	193.51
superblue18	0	**11406**	105.47	0	**11184**	106.90	0	**558**	106.70	0	**444**	107.80
average	1.0X	**1.0X**	1.00X	6.4X	**0.9X**	1.00X	3.8X	**0.7X**	1.00X	6.9X	**0.4X**	1.01X

- Methods U-AE, U-AV, NU-AV all have a significantly higher GR-OF than U-E.
- The GR-WL of methods U-AE, U-AV, NU-AV are up to 1% larger than U-E. Wirelength is increased due to detours as a result of reduced vertex or edge capacities in these methods.

The increase in GR-OF in methods U-AE, U-AV, and NU-AV is to be expected, since solutions are generated for instances with a reduced edge capacity. The resulting solution when evaluated with the original unadjusted edge capacities may have a high overflow compared to U-E. However, in most cases, this increase in the GR-OF is *more than offset* by a decrease in the DR-OF. Note that DR-OF is the total overflow of the design after detailed routing. It includes GR-OF and the overflow that results from the detailed routing stage for routing the local nets.

Consideration of vertex capacity (U-AV) results in more improvement in DR-OF compared to only reducing the edge capacity (U-AE). It allows reaching higher quality solutions which are excluded from the model of U-AE. Similarly, non-uniform binning results in more improvement in DR-OF.

Impact of Non-Uniform Binning in NU-AV: Our binning procedure is parameterized by a value η that controls the reduction of local nets when generating non-

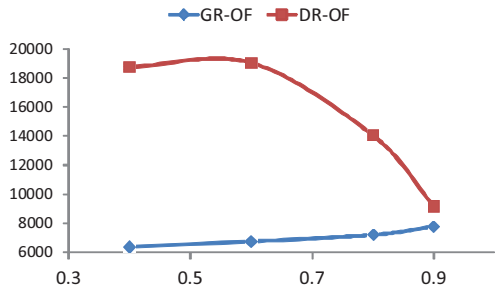

Figure 4.4: Tradeoff in DR-OF and GR-OF with η.

uniform global cells. Figure 4.4 demonstrate the impact of varying η for the instance `superblue2`. (Similar behavior was observed in the other instances.) In the figure, we see that increasing η (which decreases the number of local nets), increases the GR-OF, but decreases the DR-OF. The runtimes have a reverse trade-off: lower η (higher local nets) results in lower global routing runtime but higher detailed routing runtime. Note that our binning procedure keeps the total number of global cells the same in all the cases.

When using the binning procedure in NU-AV, we note that for all the benchmarks, DR-OF was improved both with and without the post-processing step. However, post-processing provided additional reduction in these metrics. For example in `superblue2` for $\eta = 0.9$, the DR-OF with and without post-processing were 11378 and 9180 respectively —both smaller than DR-OF of the other methods.

Comparison of Runtimes: Table 4.3 gives a runtime comparison for the different methods. The table shows that the CPU time of the detailed routing emulation is reduced on average by 0.8X, 0.6X, and 0.4X for U-AE, U-AV, NU-AV, respectively, compared to U-E. The global routing runtime on average is increased around 10% compared to U-E. Columns 2 and 3 report the percentage of local nets for uniform and non-uniform cases.

Table 4.3: Comparison of runtime (min) and local nets

Benchmark	**U**	**NU**	**U-E**		**U-AE**		**U-AV**		**NU-AV**	
	%LC	%LC	GR	**DR**	GR	**DR**	GR	**DR**	GR	**DR**
superblue1	30.8	14.1	3	**28**	7	**21**	5	**18**	7	**7**
superblue2	28.9	13.5	352	**22**	321	**17**	303	**17**	381	**16**
superblue4	35.2	16.8	180	**39**	60	**25**	201	**25**	62	**10**
superblue5	29.4	12.2	135	**42**	184	**33**	164	**24**	220	**4**
superblue10	34.1	34.0	251	**62**	341	**51**	329	**32**	342	**33**
superblue12	28.6	14.6	238	**41**	360	**42**	309	**37**	302	**22**
superblue15	34.4	15.8	212	**34**	269	**24**	259	**19**	221	**11**
superblue18	28.2	15.0	10	**32**	20	**20**	16	**15**	10	**9**
ave	31.2%	17.0%	1.0X	**1.0X**	1.1X	**0.8X**	1.1X	**0.6X**	1.1X	**0.4X**

The termination condition of CGRIP can be set by the user based on the design flow—whether the user wants a "quick-and-dirty" solution or a solution that spends higher effort to produce a high-quality global routing solution. In this work, we set the parameters of CGRIP to reduce the overflow of the GR-solution as much as possible. Specifically, the termination condition for each method is when no improvement in its objective is made for two consecutive RRR iterations. With this termination criterion, the global routing runtimes of U-E are 3min and 10min for `superblue1` and `superblue18`, respectively. However, the runtimes are multiple hours for the other instances. This longer runtime could be reduced at the expense of larger GR-OF values. For example, running CGRIP (the method U-E) for one hour results in GR-OF of 13568 for `superblue2`, and overflow exists in 6 of the 8 benchmarks. A comparison of these values to column 2 of Table 4.2 indicates that the additional effort can significantly reduce GR-OF.

Impact of Local Congestion versus Wire Sizes: Local nets and non-uniform wire sizes are two factors that contribute to congestion, but are ignored by many global routing tools. To evaluate the individual impact of these two factors, we conducted a small experiment comparing three global routing methods: 1) when local congestion

is ignored but non-uniform wire sizes are considered; 2) when non-uniform wire sizes are ignored but local congestion is considered; and 3) when both non-uniform wire size and local congestion are considered. In all cases, the global cells are of uniform size. Case 1 is same as U-E (i.e., CGRIP). Case 3 is U-AV with adjusted vertex capacity. In case 2, we assumed the same wire size and spacing in all layers to be equal to layer 1, which was done by changing the benchmark header line describing the per layer wire size and spacing values. (The routing procedures remain intact.) As a result, if layer 4 had a wire size of 2 units, after getting normalized to wire size of 1 unit in layer 1, then it passes double the number of wires on each edge. This transformation resulted in an increase in the normalized capacity of the edges in the higher layers, since these layers allow for more routing tracks to pass an edge once their wire sizes are reduced to match layer 1. This behavior is the same as the ISPD 2008 benchmarks [27].

After generating global routing and detailed routing solutions, we evaluated each solution using the original wire size using our detailed routing emulator, similar to the previous experiments. In all cases, the DR-OF, which we are using as a surrogate measure of for the goodness of the true detailed routing solution, was significantly reduced by considering both additional complicating factors. For example, for benchmark `superblue4`, the DR-OF for Cases 1, 2, 3 were 28696, 3348396, and 13296, respectively. Note that Case 2 had *significantly* higher DR-OF than the other cases. We conclude that non-uniform wire sizes and local nets are both crucial factors for routability. Our routing models and framework can account for *both* of these factors.

4.5 Concluding Remarks

In this chapter, we proposed two techniques for considering local effects during global routing. First, we introduced a technique for constructing global routing instances with global cells of non-uniform size that can decrease the number of local nets while controlling the complexity of the global routing solution procedure. Second, we proposed a model to approximate the congestion induced by local nets and incorporated this mathematical model as a vertex capacity constraint into a congestion-aware Integer Programming (IP) formulation. The IP formulation also accounts for non-uniform wire sizes, routing blockages, and virtual pins. Our computational results show that significant improvement in the quality of (estimated) detailed routing solutions can be obtained using our methods.

Chapter 5

Coll-MGR: Collaborative Multi-Objective Global Routing

In this chapter we introduce a routing procedure for collaborative and multi-objective global routing. It helps to optimize multiple interconnect-dependent objectives simultaneously which is crucial given the increasing role of interconnects. We provide details, and study different variations of our procedure. Specifically, we consider the following multi-objective case studies: minimization of interconnect power and wirelength, minimization of routing congestion and wirelength, and minimization of wirelength and utilization of routing resources.

This chapter is organized into the following sections. Section 5.1 describes the problem formulation. Section 5.2 describes our collaborative procedure to solve the formulation. Section 5.3 discusses several variations of the framework. Simulation results are presented in Section 5.4 and concluding remarks are offered in Section 5.5.

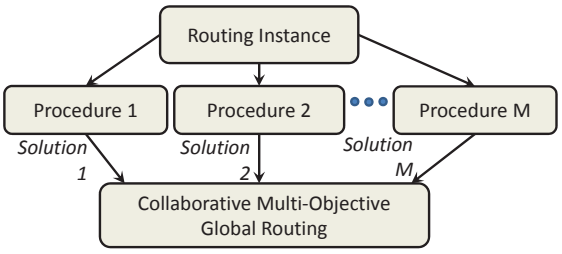

Figure 5.1: Overview of our collaborative multi-objective framework.

5.1 Problem Formulation

Our approach is collaborative and takes as input different routing solutions and conducts multi-objective optimization beyond the traditional wirelength optimization. As shown in Figure 5.1, different solutions can be obtained by running different routing procedures in parallel which could be from different tools or by concurrently running the tool from the same vendor in different modes.

In this chapter, we first discuss a mathematical description of the (single-objective) Collaborative Global Routing problem and explain the multi-objective solution procedure in the subsequent section. We are given a routing grid graph $G = (\mathcal{V}, \mathcal{E})$. Each edge e has a normalized capacity b_e, representing the number of routes that can pass from the boundary of the two adjacent global cells. A set of N nets given by $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$, (with $n_i \subseteq \mathcal{V}$), are also specified. The graph G has 3 dimensions to capture the multi-layer routing.

We use $\mathcal{T}_i$ to denote a set representing the collection of candidate routes for net n_i. These candidate routes are provided a-priori based on the flow given in Figure 5.1. Each routing procedure generates a 3D route for each net which may span multiple metal layers.

We denote by $\mathcal{T}_i(k)$ the routing solution of procedure k for net n_i so we have

$\mathcal{T}_i(k) \in \mathcal{T}_i,\ \forall k = 1, \ldots, M$, where M is the number of routing procedures. It may be possible that two procedures generate the same route for a net in their solutions (e.g., $\mathcal{T}_i(k) = \mathcal{T}_i(j)$ indicates that the same route is obtained for net n_i by the tools k and j). In this case, the set $\mathcal{T}_i$ will include this common route only once. Therefore, the number of candidate routes for n_i can be at most M: $|\mathcal{T}_i| \leq M$.

Each candidate route is a 3D Steiner tree connecting the terminals in n_i. For an arbitrary Steiner tree t, let $a_{te} = 1$ if it contains edge $e \in \mathcal{E}$; $a_{te} = 0$ otherwise. Define the binary variable x_{it} to be equal to 1 if and only if $t \in \mathcal{T}_i$ is selected for net n_i. The Collaborative Global Routing (Coll-MGR) problem is written as the following Integer Linear Programming (ILP):

$$\min_{x} \sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} p_{it} x_{it} \tag{5.1}$$

$$\begin{cases} \sum_{t \in \mathcal{T}_i} x_{it} = 1 & \forall i = 1, \ldots, N \\ \sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} a_{te} x_{it} \leq b_e + o_e & \forall e \in E \\ \sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} wl_{it} x_{it} \leq WL_{th} & \\ x_{it} \in \{0, 1\} & \forall i = 1, \ldots, N, \forall t \in \mathcal{T}_i. \end{cases}$$

where p_{it} is a constant parameter representing the cost (penalty) of route t for net n_i. The objective is to minimize the summation of the costs of the routed nets. For example, if p_{it} is the length of Steiner tree t, then the objective is to minimize the total wirelength. Cost could be alternatively defined, e.g., associated with an edge (instead of a net). For example, it can be the number of heavily-utilized routing edges, or the overall power. We elaborate when we discuss variations of the Coll-MGR formulation in Section 5.3.

In the above formulation, the first set of equations enforces selection of one can-

didate route for each net. In the second set of equations, o_e is a constant parameter. It represents the normalized overflow of edge e corresponding to the routing solution of one of the M procedures. If procedure k is the one considered as reference, then o_e is given by

$$o_e = \max(0, \sum_{i=1, t=\mathcal{T}_i(k)}^{N} a_{te} - b_e). \tag{5.2}$$

In this work we assume the selected procedure is the one which results in the smallest total overflow routing solution. So for example, in the case when one of the procedures results in a zero overflow solution for all the nets, then all the o_e parameters will be set to zero. Based on our definition of o_e, the quantity $b_e + o_e$ can be thought of as an adjusted (and possibly increased) normalized edge capacity. The second set of equations thus ensures that the selected candidate routes of all the nets do not exceed this adjusted normalized edge capacities. This definition of normalized edge capacity guarantees that a feasible solution always exists for the Coll-MGR ILP formulation.

In the third set of equations, the parameter wl_{it} represents the wirelength of route t for net n_i. The third equation bounds the total wirelength of the selected routes by a user-defined input WL_{th}. Thereby, minimization of the objective function can be done with respect to a *tolerable degradation factor* compared to an initial wirelength, for example corresponding to one of the routing tools. This is because wirelength is the prominent objective of global routing. Optimizing other objectives should be done by controlling the impact on wirelength.

The formulation of Coll-MGR is in the form of the multi-dimensional multiple-choice knapsack problem (MMKP) [38]. The MMKP is in the class of NP-complete problems. In [52], a Pareto-algebraic heuristic is given to quickly solve MMKP instances with high quality. Pareto algebra [17], [50] allows to reason about multi-

objective optimization problems in an algebraic manner. This translates to a compositional, incremental solution approach to MMKP. In this work, we extend the procedure in [52] and adapt it for the Coll-MGR problem. Furthermore, our procedure is applicable to the case when multiple objectives are simultaneously optimized. It generates multiple global routing solutions and a corresponding tradeoff set between the objectives. We elaborate the multi-objective case as we discuss our procedure in the next section.

5.2 Collaborative Global Routing Procedure

This section presents our collaborative global routing procedure. We first describe an encoding of candidate routes as *configurations* of a net to apply our procedure as an MMKP instance and then present the details of the procedure.

5.2.1 Modeling and Overview of Our Procedure

In a Pareto-algebraic description of Coll-MGR, each candidate route of a net is represented in terms of a configuration of $P + R$ dimensions denoted by $(d_1, d_2, \ldots, d_{P+R})$. For P considered cost (penalty) functions, the first P dimensions represent the corresponding costs of that configuration. For example for $P = 2$, the first and second dimensions can be the wirelength and power of the candidate route, respectively. For the global routing problem, most of the dimensions of a configuration are equal to zero so a sparse representation is used to store a non-zero dimension as a pair of the index and corresponding value.

Furthermore, R is the number of edges in the routing grid-graph, excluding those

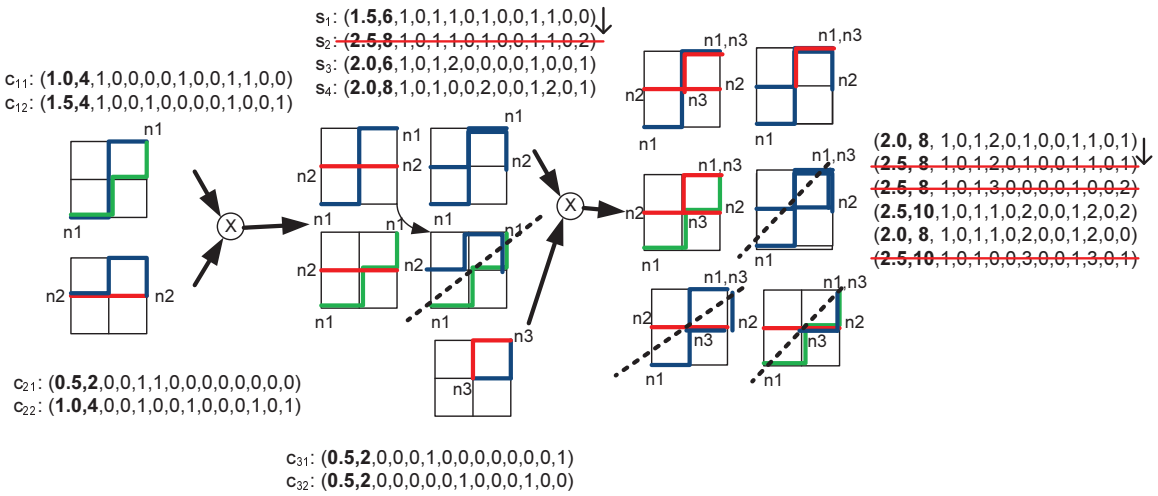

Figure 5.2: Example of our Pareto-algebraic procedure for the Coll-MGR problem.

corresponding to vias. If a candidate route is passing an edge, then the corresponding dimension will be a 1 entry in its configuration, and otherwise it is 0. Therefore the summation of the last R entries in the configuration reflects the routing resource usage of the candidate route. It could be smaller than its wirelength if the vias are assumed to have infinite capacity, similar to the ISPD global routing contest [1] and many subsequent academic routing procedures are based on this setup.

We denote the j^{th} configuration of net n_i by $c_{i,j}$. For example, in Figure 5.2 (left), we have 2 candidate routes for net n_1 and 2 for net n_2. Each route has therefore two configurations. The locations of the net terminals are marked on the 2x2 routing grid-graph. We assume $P = 2$ for two objectives of power and wirelength which are given by the first two dimensions. The remaining $R = 12$ entries reflect the utilization of the edges in the grid-graph; for a candidate route, there is an entry of 1 in a dimension if it contains the corresponding edge.

The set of candidate routes for a net n_i thus composes its *configuration set* which we denote by $\mathcal{C}_i = \{c_{i,1}, ..., c_{i,|\mathcal{C}_i|}\}$. For the Coll-MGR problem, we assume $|\mathcal{C}_i| \leq$

$M; \forall i = 1, \ldots N$, where M is the number of input routing procedures. Configuration $c_{i,j}$ is a Pareto point of set $\mathcal{C}_i$ if it is not dominated by any other configuration point. A configuration point dominates another point iff it is at least as good as that point in all dimensions and better in at least one dimension. Pareto algebra defines operations on configuration sets in order to compute Pareto points. For more details, we refer the reader to [18].

We proceed to an overview of our procedure to solve the multi-objective Coll-MGR formulation as an MMKP instance. We traverse the nets once in a specific order (discussed in Section 5.2.3). At each step of computation, we combine the configurations of the current net with an existing configuration set corresponding to all the nets processed in the previous steps. During each step we maintain a set of options for all nets processed up-to that point, each option providing a different tradeoff in the P objectives and the R resources. After each step, we thus obtain a set of compound configurations denoted by $\mathcal{S} = \{s_1, s_2, ..., s_{|\mathcal{S}|}\}$ where s_i is one compound configuration. Conceptually, each combination of two configuration sets corresponds to a product of these sets.

Each compound configuration is also represented by a tuple of $P + R$ dimensions. As before, the first P dimensions give the corresponding costs of the compound configuration after the combination. The last R dimensions give the resource usage (or utilization) of each intra-layer edge in the routing grid-graph after the combination; they are obtained by element-wise addition of the corresponding dimensions in the combined configurations. For example, when combining two configurations s_i and s_j, the last R entries in the compound configuration are simply the element-wise addition of the last R entries in s_i and s_j, indicating that the utilization of each edge will be

the summation of its utilizations in s_i and s_j. Similarly, the wirelength is always the summation of the wirelength dimensions in s_i and s_j. However, the P cost dimensions of the compound configuration may be a function of all the other dimensions of s_i and s_j in the general case. We discuss variations of Coll-MGR for different cost functions and elaborate computation of the compound configurations in Section 5.3.

After obtaining a compound configuration, a feasibility check is made and the configuration will be removed if it is infeasible. Specifically, if the wirelength (first dimension) is larger than the specified upperbound (WL_{th}) or if the utilization of an edge (any of the last R entries) is larger than its adjusted normalized capacity (o_e+b_e), then the compound configuration is infeasible and will be removed. Furthermore, configurations that are dominated in terms of all the dimensions are also removed, i.e., only Pareto points are kept.

Consider Figure 5.2 again which shows a snapshot of the procedure for a small example of nets n_1, n_2, n_3 on the 2x2 grid-graph. Each edge has an adjusted normalized capacity of 2 units. First, net n_1 and net n_2 (each with 2 configurations) are combined. Among the 4 compound configurations, s_2 is dominated by s_1 and is thus removed. Net n_3 with 2 configurations is then combined with the configuration set (of size 3) representing nets n_1 and n_2. Next, we obtain 6 configurations. The last step then removes 3 of these configurations (where 1 configuration is removed because it is dominated and 2 other configurations are infeasible due to normalized edge capacity violation).

5.2.2 Details of the Procedure

Algorithm 1 gives the details of the described procedure. Specifically, we consider the MMKP procedure given in [52] for runtime management in chip-multiprocessors, and introduce new considerations to solve the Coll-MGR problem.

For each net n_i in $\mathcal{N}$, with $1 \leq i \leq N$, we require an initial configuration set $\mathcal{C}_i = \{c_{i,1}, \ldots, c_{i,|\mathcal{C}_i|}\}$, where the configurations correspond to the routes from the M routing procedures. In addition, we take as input the adjusted normalized edge capacity vector denoted by $b_{\mathcal{E}}$, and two user-defined parameters L and WL_{th}. The former controls the execution runtime of the algorithm (as explained in Section 5.2.4) while the latter defines the wirelength threshold.

The set of compound configurations $\mathcal{S}$ is initially set to $\mathcal{C}_1$ for net n_1 (line 2). This means that element $s_j = c_{1,j}, \forall s_j \in \mathcal{S}$. The remaining nets are then traversed once. We discuss the traversal ordering in Section 5.2.3. For each net n_i (line 3), we consider each $s_j \in \mathcal{S}$, reflecting the combinations of processed nets n_1 to n_{i-1} with configurations of n_i (lines 5, 6).

First we check if the compound configuration corresponding to the combination of $c_{i,l}$ and s_j is feasible (line 7). To check for feasibility, we first compute the last R dimensions by element-wise additions of the corresponding dimensions in $c_{i,l}$ and s_j and check for satisfaction of the normalized edge capacities.

We also compute the wirelength of the compound configuration by adding the wirelength dimensions in $c_{i,l}$ and s_j and compare with WL_{th}. If the feasibility checks are successful, we proceed to combine $c_{i,l}$ and s_j which is denoted by the compound configuration s^{tmp}. To obtain s^{tmp} (line 8) we need to compute the remaining dimensions which correspond to the values of the cost functions in the configurations

Algorithm 1 Pareto-algebraic heuristic for Coll-MGR

```
1: Coll-MGR(C_i (∀n_i ∈ N), b_E, L, WL_th)
2: S = C_1 //so we have (s_1 = c_{1,1}, s_2 = c_{1,2}, ..., s_{|C_1|} = c_{1,|C_1|})
3: for all net n_i, i = 2 to |N| // Sec. 5.2.3 do
4:    S^tmp = ∅
5:    for all l = 1 to |C_i| do
6:       for all s_j ∈ S do
7:          if feasible(s_j, c_{i,l}, b_E, WL_th) then
8:             s^tmp = combine(s_j, c_{i,l}) // Sec. 5.3
9:             S^tmp = S^tmp ∪ {s^tmp}
10:            min(S^tmp) // Sec. 5.3
11:         end if
12:      end for
13:   end for
14:   S = S^tmp
15:   reduce(S, L) // Sec. 5.2.4
16: end for
```

considered for nets n_1 to n_{i-1} (given in s_j) and for net n_i (given in $c_{i,l}$).

In Section 5.3 we discuss the implementation of *combine* (line 8) for different cost functions, corresponding to different variations of Coll-MGR. After the combination, the result is added to the set $\mathcal{S}^{tmp}$, reflecting the set of feasible combinations between the configurations of n_i and the s_js processed so far. The Pareto-algebraic *min* operation is applied to $\mathcal{S}^{tmp}$ to get the Pareto points. This boils down to checking whether the newly-added configuration is dominated or whether it dominates any of the already-present configurations. Since the number of dimensions $(P + R)$ could be very large for the Coll-MGR problem, applying the *min* operation considering all the dimensions is highly time-consuming. We therefore approximate *min* by applying it in a 2-dimensional space. We discuss this in Section 5.3 for different Coll-MGR variations.

The process repeats to combine the next configuration s_{j+1} with n_i. When all s_js

are combined with the configurations of n_i, set $\mathcal{S}$ is updated to $\mathcal{S}^{tmp}$ and net n_{i+1} is processed.

For implementation efficiency, we apply the *min* operation after combining each feasible s_j and $c_{i,l}$ (instead of a one-time *min* after all the configurations are combined). This allows reduction in the number of intermediate non-Pareto configurations, which makes each iteration of the loops faster. This implementation of minimization is following the *Simple Cull* approach [68]. As shown in [17], Simple Cull is in practice the most efficient implementation.

However, the above procedure is still intractable in many cases, as the size of $\mathcal{S}$ can become very large. To speed up the computation, we check if the size of $\mathcal{S}$ becomes larger than the user-defined threshold parameter L. We only keep at-most L configurations and prune the rest (line 15). We explain this step in Section 5.2.4. In the end, a tradeoff set of the considered objectives is provided to the user. The set has at-most L points. Our implementation allows each point to be tracked back to generate the corresponding configuration of each net, and thus a complete global routing solution. The user can pick a desired point and the algorithm generates the final solution.

5.2.3 Net Ordering

Algorithm 1 relies on an ordering for processing the nets. Based on our experience, we observe that the ordering can highly impact the quality of the final solution. Selecting a suitable ordering may not be trivial since the single-objective routing procedures may use different orderings which may be incompatible with each other. We observe that for our Pareto-algebraic procedure, a good ordering is one which can

Algorithm 2 Net ordering procedure

1: NETORDERING($\mathcal{C}_i(\forall n_i \in \mathcal{N})$, $\mathcal{N}$, $\mathcal{E}, k$)
2: **for all** $e \in \mathcal{E}$ **do**
3: $\quad u_e^{approx} = \sum_{i=1,t=\mathcal{T}_i(k)}^{N} a_{te}$
4: **end for**
5: $\mathcal{N}_{order} = \emptyset$; $\mathcal{E}_{candid} = \mathcal{E}$
6: **while** $|\mathcal{N}_{order}| \neq |\mathcal{N}|$ **do**
7: $\quad e_{crit}$=arg max$_{e \in \mathcal{E}_{candid}}(u_e^{approx})$
8: $\quad \mathcal{N}_{crit}$= $\{n_i \in \mathcal{N} |$ e_{crit} included in $c_{i,j}$, $1 \leq j \leq |C_i|\}$.
9: $\quad \mathcal{N}_{order} = \mathcal{N}_{order} \circ$ sort-decreasing-wirelength($\mathcal{N}_{crit}$)
10: $\quad \mathcal{E}_{candid} = \mathcal{E}_{candid} - \{e_{crit}\}$
11: **end while**

identify and prune the infeasible and dominated solutions early in the combination steps. Otherwise, the number of Pareto points will quickly reach the threshold and the reduction step will need to be applied to remove the Pareto points.

A good ordering can be achieved by looking at the congested areas first which can be *approximated* from the solutions of the routing procedures. In a congested area, the routing grid edges are utilized to close to their capacities, and therefore it will be more likely to detect and remove infeasible and dominated configurations in the first few iterations of the main loop of Algorithm 1. In addition, the congested areas include many more nets and may contribute more to the considered cost functions such as overall wirelength or power.

Algorithm 2 gives the pseudo-code of our ordering procedure. It receives the configuration set of each net and the sets of nets and edges as input. It also receives as a parameter a routing procedure k used as reference. As shown in lines 2 to 4 of Algorithm 2, for each edge, an approximate utilization is computed as the utilization of that edge in the solution of the routing procedure k. This approximate utilization is then used to drive the net ordering.

During ordering, we keep updating an ordered subset of nets $\mathcal{N}_{order} \subseteq \mathcal{N}$ until $|\mathcal{N}_{order}| = |\mathcal{N}|$ (line 6). We first identify a critical edge $e_{crit} \in \mathcal{E}_{candid}$ with highest approximate utilization u_e^{approx}, where initially $\mathcal{E}_{candid} = \mathcal{E}$ (line 5). Next, we form the subset $\mathcal{N}_{crit} \subseteq \mathcal{N}$ of critical nets where $n_i \in \mathcal{N}_{crit}$ if any of its configurations $c_{i,j}$ includes e_{crit} (line 8). Since the configurations of these routes include e_{crit}, they are more likely to result in pruning. We then sort the subset $\mathcal{N}_{crit}$ according to decreasing order of the wirelength of the nets, with each n in N_{crit} wirelength approximated by averaging the wirelengths corresponding to all its configurations $c_{i,j}$ (line 9). The sorted subset is then concatenated to the end of the existing $\mathcal{N}_{order}$ (also in line 9 where concatenation is denoted by the $\circ$ symbol). Finally, we update the set $\mathcal{E}_{candid}$ to find the next e_{crit} while excluding the previous e_{crit} identified from the previous iteration.

Using the above-described approximate utilization significantly improves our ordering procedure compared to other methods to estimate the utilization such as by looking at the overlap in the net bounding boxes. This is because of two reasons. First, the approximate utilization is inline with the adjustments applied to the normalized edge capacities in the Coll-MGR formulation. Second, it corresponds to the utilization of one of the input routing solutions. In contrast, estimating the edge utilization in the existing sequential routing procedures is a more challenging task because they build a routing solution completely from scratch.

5.2.4 The Reduce Procedure

According to Algorithm 1, the set $\mathcal{S}$ representing the compound configurations as set in line 14 contains all the Pareto points. If the size of $\mathcal{S}$ exceeds the user-defined

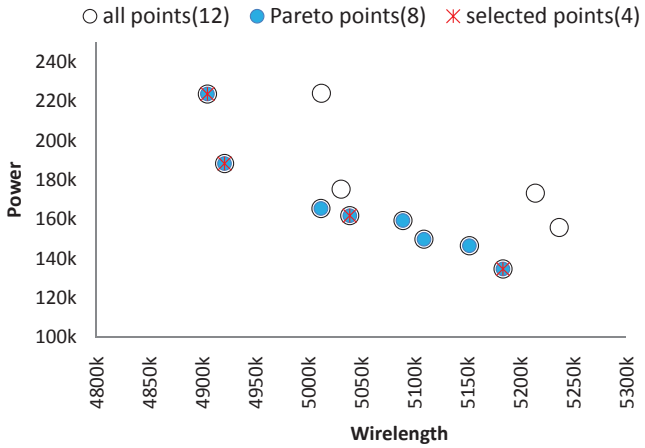

Figure 5.3: Reduce after processing 150K nets in `a1`.

parameter L, we apply a procedure to drop some of these Pareto points. Our goal is to select L Pareto points (hence L compound configurations) such that they provide a tradeoff set between the considered cost functions and resources. Since the number of cost functions and resources $(P+R)$ is very large, in this work, we only discuss the reduce procedure for specific variations of Coll-MGR (discussed in Section 5.3). In all these variations, we project the compound configurations into a two-dimensional space. We will discuss the definition of each dimension for each variation separately (in Section IV). For example, when we consider two cost functions in the multi-objective Coll-MGR, each dimension represents one of the cost functions and each configuration is represented by a point in this space. We denote dimensions 1 and 2 by d_1 and d_2.

Figure 5.3 shows a snapshot of the projected compound set $\mathcal{S}$ after processing 150K nets in benchmark `a1` for two cost functions of power and wirelength. To select L configurations, we first select two configurations, one with minimum d_1 value and the other with minimum d_2 value. We then select $L-2$ additional configurations as follows. For each configuration we determine a weight given by $\frac{d_1}{d_2}$ which characterizes

the tradeoff between the two dimensions. In the above example, for configuration i, we define the weight $w_i = \frac{power_i}{wirelength_i}$.

We then try to cover the range of weights uniformly. We first select $L-2$ "reference values" according to the following formula: $\min_i(w_i) + \frac{\max_i(w_i)-\min_i(w_i)}{L-1} \times k,\ \forall k = 1, \ldots, L-2$. To select the remaining $L-2$ configurations, we visit these $L-2$ reference values one at a time. For reference value k, we select the configuration with a corresponding weight closest to this reference value. We remove the selected configuration and continue until all the reference values are considered. Therefore for $L-2$ reference values, we select $L-2$ configurations with weights which are closest to them. Together with the two initially-selected configurations, we obtain L configurations.

Figure 5.3 shows the selection of $L = 4$ in the above example. The procedure attempts to quickly and fairly approximate the case when none of the Pareto points are dropped.

5.2.5 Runtime Complexity

For M collaborating routers with up to L stored solutions and N nets, the runtime complexity of the Coll-MGR procedure is $O(N(ML \times M^2L^2 + M^2L^2log(ML))) = O(NM^3L^3)$. This is because for each of the N iterations of the algorithm (excluding the first one which can be ignored), the combination of L compound configurations representing the processed nets is considered with M configurations of the currently-considered net. For each combination, a Pareto minimization (line 10) is applied which is of $O(M^2L^2)$ for an input size of $O(ML)$. Then the *reduce* procedure requires a sorting and a walk through the compound configuration S (line 15) to pick L config-

urations. Therefore, given that the size of any of the compound configuration sets S is bounded by M^2L^2, the complexity of the *reduce* step is $O(M^2L^2log(ML))$. Without the *reduce* phase and Pareto minimization, the size of the compound configuration set exponentially grows and is of $O(M^i)$ for iteration i.

In practice, the runtime of the algorithm never shows its worst-case behavior. The Pareto minimization maintains a list of Pareto points observed *so far* every time one of the M configurations of the current net is combined with the current compound configuration set (line 10). As a result, the size of S^{tmp} is significantly reduced compared to its upper bound.

5.3 Variations of Collaborative Global Routing

In this section we study a few multi-objective variations of Coll-MGR. For each variation, we discuss the specific implementations of *combine*, *min* and *reduce* in Algorithm 1. In Section 5.4, we present the simulation results for each case.

5.3.1 Power and Wirelength Minimization

Power of signal interconnects (which excludes power-delivery and clocking networks) can have a significant contribution to the total power consumption in the nanometer era; for example, the contribution of the interconnect power is reported to be around 30% of dynamic power for a 45nm high performance microprocessor synthesized using a Structured Data Paths design style and about 18% of the overall power spectrum [49]. Recently, global routing has gained importance to improve power consumption at the global routing stage. For instance the work [62] and [36] address multiple

dynamic supply voltage at the global routing stage to reduce dynamic power at this stage.

To model the signal power accurately, area and fringe capacitances for each wire segment should be considered depending on its width and metal layer. Furthermore, the spacing between the wires can determine their coupling capacitance which highly deteriorates with technology scaling. The above factors (i.e. metal layer and wire width) can be approximated during the global routing stage fairly accurately. Moreover, wire spacing can also be approximated at this stage by *approximating* the utilization of each edge within the global routing procedure. Therefore, opportunities lie in signal power optimization at this stage. Here, we consider multi-objective optimization of an interconnect power metric and wirelength.

Specifically, we first adjust the normalized capacities of each edge in the global routing grid-graph to account for the wire size and spacing of its corresponding metal layer, as provided by the target technology. Typically for each layer a range of wire size options are available. In this work we assume the minimum wire size is used for each layer. This is because we use routing as the technique used for power minimization and do not consider wire sizing. After this adjustment, we consider the routes provided by the M routing procedures to be the candidate routes. We then apply Coll-MGR to optimize an interconnect power metric and wirelength.

The consideration of wire size and spacing reduces the normalized edge capacities at the higher metal layers since these quantities are typically much larger at the higher layers. To compute the number of routes that pass the boundary of two adjacent global bins in a metal layer, we divide the length of the boundary of the two global bins by the summation of wire size and spacing of that layer. This gives a

more accurate estimate of the number of routes that can pass the boundary which will be the adjusted normalized edge capacity of the global routing edge corresponding to that boundary. Note, all the edges in the same metal layer will have the same normalized capacity using this approach. We first review a power model for global routing (proposed by us in [53]).

Interconnect Power Model for Global Routing

The total interconnect power denoted by POW can be expressed as the summation of dynamic powers of the routed nets:

$$POW = V_{DD}^2 \times f_{clk} \times \sum_{i=1}^{|\mathcal{N}|} \alpha_i (C_i^{sink} + C_i^{wire}) \tag{5.3}$$

where V_{DD} is the supply voltage, f_{clk} is the frequency, and α_i is the switching activity of net i. The capacitance of a routed net i is the summation of the capacitances of its sink cells (denoted by C_i^{sink}), and of its wire (denoted by C_i^{wire}). The C_i^{wire} is expressed as the summation of all its unit wire segment capacitances and given by the equation below:

$$C_i^{wire} = \sum_{\forall j, e_j \in t_i} C_j^{unit} \tag{5.4}$$

where C_j^{unit} is the capacitance corresponding to edge e_j which is contained in route t_i. For a wire segment corresponding to edge e_j, we assume a width w_j and metal layer l_j. In the general case, the wire can be in the middle of two other neighboring segments running in parallel with spacing s_j, all of which are mapped to edge e_j, as shown in Figure 5.4. The capacitance corresponding to this wire segment is denoted

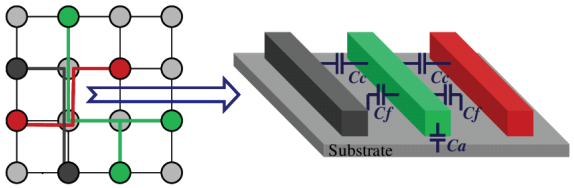

Figure 5.4: Mapping edge utilization to interconnect capacitance.

by C_j^{unit} and is given by the equation below:

$$C_j^{unit} = C_a(l_j, w_j) + 2C_f(l_j, w_j, s_j) + 2C_c(l_j, w_j, s_j) \tag{5.5}$$

where C_a and C_f are the area and fringe capacitances with respect to substrate, and C_c is the cross-coupling capacitance between the wire segment and its parallel-running wires. Figure 5.4 illustrates these three types of capacitances.

The area, fringe and coupling capacitances are a function of metal layer l_j and wire width w_j corresponding to edge e_j. In addition, the fringe and coupling capacitances also depend on spacing s_j on edge e_j. For a given metal layer, wire width and spacing, the corresponding unit-length capacitance can then be computed typically using a lookup table provided by the technology library. For example, for the 45nm technology model used in this work [40], we observed that with decrease in spacing, C_c significantly increases while C_f slightly decreases. Also, with increase in wire width, C_a drastically increases. The wire width is also known from the library for each metal layer and a higher metal layer requires a higher wire width. The spacing depends on the routing congestion. Specifically, to compute the spacing at edge e_j at the global routing stage, we use the following formula:

$$s_j = \max(s_{min_j}, \frac{length_j}{\#wires_j} - w_j) \tag{5.6}$$

where $length_j$ denotes the length of the boundary between the two adjacent global bins that edge e_j represents, $\#wires_j$ denotes the number of wires that pass from e_j, w_j denotes the corresponding wire size, and s_{min_j} represents the minimum wire spacing for the layer that has e_j which is obtained from the lookup library. According to the above formula, we determine the spacing s_j such that the maximum distance between the wires that pass from e_j is obtained. This strategy minimizes the fringe and cross-coupling capacitances which makes sense if no spacing constraints are provided. On the other hand, if additional spacing constraints are provided, they can be incorporated for a more accurate estimate of wire spacing. For example, the adjustment may be a reduced normalized capacity due to the presence of a fixed number of short (local) nets which completely fall inside a global bin.

Note, in the presence of overflow when edge e_j is utilized higher than its capacity, the spacing will be set to the minimum spacing s_{min_j} which is he smallest spacing that can be used for the edge according to the library model.

In summary, we estimate the spacing in the boundary of two adjacent global bins from the utilization of the corresponding edge in the global routing grid-graph. (During the global routing procedure, this utilization is further estimated as the nets are sequentially routed.) Together with the wire size and metal layer for that edge, we then look up the capacitance components from the library and use the presented equations to estimate the total signal power over all the nets. Therefore, to compute the power metric during global routing, the only requirement is the knowledge of the utilization of each edge in the grid-graph (given that the wire size and spacing are provided at each metal layer).

Details of the Coll-MGR Procedure

We assume the configuration of each net is represented by $2+R$ dimensions. The first dimension is an *estimate* of total power. In this work, we assume that each configuration of each net has for its first dimension as a value the total power estimated using the edge utilizations obtained from the solution of the reference routing procedure which was also used during net-ordering in Algorithm 2. The second dimension is the corresponding wirelength which is the length of the candidate route of that net. Similarly, in the compound configuration the first dimension is a more accurate estimate of total power (where the increased accuracy comes from taking into account the actually selected routes for the nets processed so far) and the wirelength is the length of the corresponding candidate routes represented by that compound configuration.

In Algorithm 1, to apply the *combine* function to configuration $c_{i,l}$ and compound configuration s_j, we first update the last R fields by element-wise addition. We also compute the corresponding wirelength by adding the wirelength fields of $c_{i,l}$ and s_j. To compute the power field, we first obtain an *estimate* of the utilization of each edge by adding the utilization from the first i nets processed so far, with an estimate of the utilization for the remaining $N-i$ unprocessed nets. More specifically, for each edge e, we consider a utilization contributed by the processed nets n_1 to n_i, obtained by adding the corresponding fields in $c_{i,l}$ and s_j that represent edge e. We then estimate the remaining utilization of e contributed by nets n_{i+1} to n_N as estimated using configurations $c_{j(>i),k}$ (i.e., provided by the reference routing tool k). Once the utilization of each individual edge in the grid-graph is estimated, the power is computed as discussed before. Please note in this multi-objective variation, an estimate of the route corresponding to each unprocessed net is necessary in order to

compute the utilization for each edge e_j and consequently its wire spacing according to Equation (5.6) for power estimation.

After the combination, to apply the *min* operation, each compound configuration in s^{tmp} is projected to a two-dimensional point with wirelength and power as the two dimensions. The Pareto minimization operation is then applied. Similarly, during the *reduce* procedure, we assume d_1 to be wirelength and d_2 to be power consumption.

5.3.2 Congestion and Wirelength Minimization

We define a variation of Coll-MGR to minimize the utilization on those edges in the global routing grid-graph that are utilized close to or higher than their normalized capacities. An edge is highly utilized if its utilization is higher than 70% of its normalized capacity. We refer to such an edge as a HUT edge (i.e., highly-utilized). The goal is to minimize the summation of the utilization of the HUT edges simultaneously with wirelength. These two cost functions can be traded off with each other. According to [4], the HUT edges are likely to cause routability problems later during detailed routing. Minimizing the utilization of the HUT edges results in spreading the routing congestion and in turn increasing wirelength. The user is also allowed to define WL_{th} to control the maximum increase in wirelength.

In this variation, the configuration of each net has $2 + R$ dimensions where the first dimension is the total utilization of the HUT edges, and the second dimension is wirelength. Initially, for each net, the first field is 0 because the normalized capacity of each edge is typically a large value and the rest of the nets are not considered. However, when considering the compound configurations, it is possible that some of the edges will become highly utilized. So the first field will likely have a non-zero entry

in the subsequent steps of the algorithm. The second field represents the wirelength of a compound configuration.

In the *combine* function, we first compute the last R fields of the combination of s_j and $c_{i,l}$ by element-wise addition. To compute the utilization of the HUT edges, we visit the last R fields after the combination. Recall each field represents the utilization in one of the intra-layer edges which can be compared against the normalized capacity of that edge to determine if it is a HUT edge. We then add the utilizations of the identified HUT edges. The wirelength field is also computed by adding the wirelength fields of s_j with $c_{i,l}$. Next, in Algorithm 1, to apply Pareto minimization, we project each compound configuration in S^{tmp} to a two-dimensional point. Here, the first dimension is the utilization of the HUT edges, and the second dimension is wirelength. We ignore the resource utilization (which is wirelength excluding the infinite-capacity vias) to keep the number of dimensions small in order to reduce the runtime of the algorithm for the typically-large global routing instances. In the *reduce* function, we also assume d_1 is the utilization of the HUT edges and d_2 is wirelength.

5.3.3 Routing Resource and Wirelength Minimization

In this variation of Coll-MGR, we consider minimizing wirelength as one objective and the routing resource usage as the second objective. The resource usage of each route is its wirelength excluding the vias on its path. This is because in this work, similar to others, we consider vias to have infinite capacity and hence they are not counted towards resource usage. So two routes can have the same wirelength but different resource usage, i.e., a difference in number of vias. When considering these two dimensions, a routing solution is better than another if it is not worse in wirelength

and resource usage while it is better in at least one of these dimensions.

The reason we discuss this variation is because it shows the application of our (inherently multi-objective) framework for single objective wirelength minimization. This is because our second objective of minimizing the total routing resource usage is typically correlated with the total wirelength.

More specifically, in this variation, the configuration of each net has $2 + R$ dimensions where the first two dimensions reflect the wirelength of the route and its resource usage, respectively. Resource usage is obtained by subtracting the number of vias from the wirelength. Similarly, the first two dimensions of the compound configurations reflect the wirelength and resource usage corresponding to the configurations (candidate routes) of the nets that were combined so far.

The *combine* function in Algorithm 1 (line 8) simply adds the wirelength of compound configuration s_j with the wirelength of configuration $c_{i,l}$ of net i. The last R fields are also added element-wise to obtain s^{tmp}, and then s^{tmp} is added to the existing set $\mathcal{S}^{tmp}$. Next, to apply the *min* operation to $\mathcal{S}^{tmp}$ (line 10), first each compound configuration of $\mathcal{S}^{tmp}$ is projected to a two-dimensional point: the first dimension is wirelength, and the second dimension is the aggregate resource usage (obtained by adding the last R fields of that compound configuration to get a resource usage value). Pareto minimization is then applied to find the Pareto points in this two-dimensional space. Finally, in the *reduce* procedure, we assume d_1 is wirelength and d_2 is the resource usage.

5.4 Simulation Results

We considered three input routing procedures from recent academic global routers: NTHU-Route 2.0 (denoted by NTHU-R) [8], BFGR [25] and FastRoute 4.0 (denoted by FastRoute) [66] in our first two experiments related to power and congestion. For the wirelength experiment we considered two additional routers, NCTU-GR [14] and MGR [65]. The solutions generated by NTHU-R [8] and MGR [65] were obtained from the corresponding authors (or the tool's website). The solutions of BFGR [25], FastRoute [66], and NCTU-GR [14] were created by running their binaries and verifying the results using the ISPD 2008 evaluation script provided by the ISPD contest. Using these sets of solutions as candidate routes, we implemented our Coll-MGR procedure using C++ and conducted simulations using the ISPD 2008 benchmark instances [27]. The Coll-MGR procedure requires specifying one input solution as the reference case to adjust the edge capacities in its internal procedure. In our implementation we set the reference to be the input solution with the smallest total overflow.

Table 5.1 lists the information about each benchmark for the first three routers; columns 2 to 6 list the number of nets, grid granularity, number of layers, and the default vertical and horizontal edge capacities of each benchmark instance. Columns 7 to 15 list the corresponding wirelength (WL), overflow (OF), and power (POW) of each benchmark for each procedure. The wirelength is scaled to 10^5. We explain calculation of power in detail later. All simulations ran on a machine with a 2.8GHZ Intel CPU and 12GB of memory.

Table 5.1: Information about the ISPD 2008 benchmark instances and the solutions of the input routing procedures.

Bench	#Nets	Grid	#L	Vcap	Hcap	NTHU-R [8]			BFGR [25]			FastRoute [66]		
						WL	OF	POW	WL	OF	POW	WL	OF	POW
a1	176715	324x324	6	70	70	53.44	0	318043	54.31	0	332448	53.80	0	301465
a2	207972	424x424	6	80	80	52.29	0	247900	52.32	0	283572	52.20	0	258564
a3	368494	774x779	6	62	62	131.01	0	721996	131.41	0	788712	131.20	0	701841
a4	401060	774x779	6	62	62	121.69	0	557546	121.63	0	660529	121.30	0	545372
a5	548073	465x468	6	110	110	155.38	0	839255	156.72	0	934743	155.82	0	862231
n1	270713	399x399	6	62	62	46.53	0	198935	46.80	0	177140	46.30	0	211310
n2	373790	557x463	6	110	110	75.70	0	306686	75.70	0	347852	75.20	0	319470
n3	551667	973x1256	6	80	80	106.57	31454	545813	106.40	33900	589827	108.40	31276	553274
b1	282974	227x227	6	70	70	55.95	0	358874	57.21	0	387691	56.60	0	370414
b2	576816	468x471	6	52	52	90.59	0	414968	91.14	0	453963	91.20	0	407729
b3	1122340	555x557	8	148	148	130.69	0	562362	131.79	0	648007	130.00	0	577168
b4	2228903	403x405	8	204	204	230.90	160	998194	232.01	434	1007510	230.21	138	1024810
n4	636195	455x458	6	84	84	130.46	138	647181	130.80	218	740925	130.50	136	625358
n5	1257555	637x640	6	88	88	231.58	0	1122040	233.00	0	1266330	230.90	0	1171950
n6	1286452	463x464	6	132	132	176.89	0	924324	180.12	0	1008550	177.51	0	958422
n7	2635625	488x490	8	212	212	355.22	54	1643040	352.11	606	1827920	353.40	54	1631960

5.4.1 Coll-MGR for Minimizing Power and Wirelength

This section evaluates a variation of our Coll-MGR procedure for joint wirelength and power minimization as given in Section 5.3.1. In this experiment, we consider two cases. First, we assume a maximum wirelength degradation of 0%, compared to the Min-POW case which we explain shortly. We set $L = 4$ to explore the trade off between the wirelength and power as shown in Figure 5.3. In the second case, we allow up to 2% wirelength degradation to show that our framework can trade it off with improvement in power consumption. We use the input solution of the tool with minimum power for the net ordering procedure and to approximate the edge utilization in Coll-MGR's implementation.

To compute power consumption, first, for area capacitance, the relative wire sizes were scaled with respect to the wire size of layer M1 which we extracted from a 45nm technology library [40]. Then, to model coupling capacitance from the utilization of an edge in the grid-graph, we computed the remaining (not utilized) normalized

capacity on that edge by accounting for the wire sizes. We then computed the spacing between the wires on that edge, as given by Equation 5.6, and converted the spacing to coupling and fringe capacitance numbers by looking up the values for each metal layer from our considered 45nm technology library. The via edges were excluded in power computation because in the ISPD 2008 benchmarks vias are required to have infinite capacity which prohibited us to compute a wire spacing in Equation 5.6. After obtaining the capacitance values, to compute power, we assumed each net to have an activity factor which we randomly selected to be in the range of 0.05 to 0.2.

In our experiments, we compute the power of the routing solutions of the input routers. We note this is only to allow defining a base for comparison since these input solutions were not created with power as an objective. When creating the Coll-MGR routing solution, the procedure accounts for wire size and spacing as explained before. We note in both cases, i.e., when running the Coll-MGR procedure and when running the input routers (or using their solutions), the normalized capacity of each edge–which is the maximum number of routes that can pass from it without creating overflow–remains the same. Subsequently, the computation of overflow and utilization of an edge follows the same procedure for the input routers and Coll-MGR so their comparison is under the same conditions. The only difference is that Coll-MGR also accounts for power within its internal procedure and takes in varying wire size and spacing numbers when calculating Equation 5.5.

Evaluation of power is the same in all the cases, i.e., Coll-MGR and the input solutions. Varying wire size and spacing values (which correspond to our technology library) are used to calculate Equation 5.5. Furthermore, the utilization of each edge which is used in calculation of the coupling capacitance is computed exactly according

Table 5.2: Results for minimizing power and wirelength

	Min-POW	Coll-MGR											
		WL Degredation: 0%						WL Degredation: 2%					
			%imp. Min-POW		%imp. POW				%imp. Min-POW		%imp. POW		
Bench	Tool	T	WL	POW	NTHU-R	BFGR	FastRoute	T	WL	POW	NTHU-R	BFGR	FastRoute
a1	FastRoute	2.82	0.67	8.69	8.74	15.44	8.69	2.13	0.19	9.96	10.01	16.62	9.96
a2	NTHU-R	2.04	0.21	7.41	7.41	19.02	11.25	2.64	-0.78	12.40	12.40	23.39	16.03
a3	FastRoute	5.75	0.16	4.87	7.49	15.32	4.87	5.83	-0.18	6.83	9.40	17.06	6.83
a4	BFGR	5.07	0.33	14.33	14.34	14.33	14.51	4.97	-0.85	17.98	17.99	17.98	18.16
a5	NTHU-R	6.00	0.44	8.94	8.94	18.22	11.38	5.69	-0.75	11.04	11.04	20.11	13.42
n1	NTHU-R	2.42	0.39	5.79	5.79	17.50	11.31	2.39	-1.05	10.92	10.92	21.99	16.14
n2	NTHU-R	3.41	0.33	14.86	14.86	24.94	18.29	2.42	-1.60	18.68	18.68	28.31	21.96
n3	NTHU-R	5.30	0.42	12.67	12.67	18.01	14.45	5.35	-1.28	16.36	16.36	21.47	18.06
b1	NTHU-R	2.18	0.14	2.57	2.57	9.80	5.56	2.59	-0.46	4.32	4.32	11.43	7.26
b2	NTHU-R	3.56	0.00	14.19	14.19	21.55	16.74	3.60	-0.45	16.15	16.15	23.34	18.64
b3	NTHU-R	4.47	0.45	17.50	17.50	28.36	19.60	5.29	-1.32	20.65	20.65	31.10	22.67
b4	FastRoute	8.60	0.19	8.56	8.62	19.14	8.56	7.56	-1.46	14.31	14.37	24.23	14.31
n4	NTHU-R	5.27	0.47	12.82	12.82	23.91	13.96	5.82	-0.85	16.53	16.53	27.14	17.62
n5	FastRoute	5.81	0.03	4.24	5.84	16.55	4.24	9.13	-1.05	8.80	10.32	20.52	8.80
n6	NTHU-R	7.37	0.27	5.73	5.73	13.56	9.06	6.64	-0.90	9.86	9.86	17.35	13.04
n7	NTHU-R	6.39	0.06	12.41	12.41	21.28	13.92	8.46	-0.91	15.81	15.81	24.33	17.26
ave		**4.06**	**0.29**	**9.72**	**10.00**	**18.56**	**11.65**	**5.56**	**-0.86**	**13.16**	**13.43**	**21.65**	**15.01**

to the corresponding routing solution.

The results are reported in Table 5.2. First, for each benchmark, the solution of three routing procedures which has the minimum power is designated as the Min-POW solution and the name of the corresponding router is reported in column 2. The power for each benchmark for each procedure is reported in Table 5.1. The results of Coll-MGR are reported in columns 3 to 14. Columns 4 and 5 report the percentage improvements in different metrics when we select the minimum power obtained from Coll-MGR, compared to the Min-POW case with the wirelength degradation threshold set to 0%. The power of Coll-MGR is improved on average by 9.72% compared to the Min-POW case and the wirelength is always less than or equal to that of the Min-POW case.

We also report the percentage improvement in power over each tool in columns 6 to 8. On average the savings in our power metric with respect to NTHU-R [8],

Table 5.3: Pareto points when minimizing power and wirelength.

Bench	Pareto Points(% of Imp. Over Min-POW)							
	Point1		Point2		Point3		Point4	
	WL	POW	WL	POW	WL	POW	WL	POW
a1	0.19	9.96	0.37	7.20	0.46	7.04	0.56	6.93
a2	-0.78	12.40	-0.59	10.59	-0.21	9.80	0.36	9.09
a3	-0.18	6.83	-0.09	6.14	-0.03	6.14	0.06	5.78
a4	-0.85	17.98	-0.75	16.82	-0.65	16.40	0.00	16.05
a5	-0.75	11.04	-0.69	10.68	-	-	-	-
n1	-1.05	10.92	0.06	10.32	1.14	8.79	0.62	7.99
n2	-1.60	18.68	-0.70	17.82	-0.53	17.75	0.00	0.00
n3	-1.28	16.36	-1.14	15.46	-	-	-	-
b1	-0.46	4.32	-0.09	2.14	1.13	1.24	0.13	0.85
b2	-0.45	16.15	0.76	14.92	1.53	14.30	0.21	13.68
b3	-1.32	20.65	-0.41	19.94	-0.25	19.38	0.15	18.97
b4	-1.46	14.31	-1.40	14.24	-	-	-	-
n4	-0.85	16.53	-0.66	15.48	-0.57	15.09	-	-
n5	-1.05	8.80	-1.04	8.16	-0.91	8.02	0.00	7.93
n6	-0.90	9.86	-0.89	9.07	-0.71	8.72	0.00	0.00
n7	-0.91	15.81	-0.91	15.48	-	-	-	-

BFGR [25], and FastRoute [66] are 10.00%, 18.56%, and 11.65%, respectively. The overflow of Coll-MGR is at most equal to the overflow of the tool with the smallest total overflow which can be looked up from Table 5.1.

The runtime of Coll-MGR is reported in column 3 for each benchmark. The runtime is on average 4.06 minutes.

For the case when up to 2% wirelength degradation is allowed, the results are reported from columns 9 to 14. In this case, the percentage of power improvement over the Min-POW case on average is 13.16%, while the wirelength on average is increased by 0.86%. For the benchmark `a1` we improve both wirelength and power; the power improvement is larger than in the 0% degradation case. The improvements with respect to the base cases NTHU-R [8], BFGR [25], and FastRoute [66] in this case are 13.43%, 21.65%, and 15.01%, respectively.

In Table 5.3, we also report the power and wirelength of all the Pareto points stored at the last stage of the Coll-MGR algorithm for the 2% WL degradation ex-

periment. Each Pareto point represents a different routing solution obtained using our procedure. The entries are percentage improvement numbers relative to the wirelength and power of the Min-POW case. For example for benchmark `a2` we have four Pareto points at the last stage of Coll-MGR, corresponding to four different global routing solutions. The first Pareto point results in 0.78% increase in wirelength with 12.40% decrease in power. The second Pareto point results in 0.59% decrease in wirelength with 10.59% decrease in power, and so on. Note that in benchmark `a1`, all the Pareto points result in both wirelength and power improvements, compared to the Min-POW case. The reason that the number of stored solutions is sometimes less than $L = 4$ is because after combining the compound configurations at the last stage of Coll-MGR, it is possible that infeasible configurations are eliminated and non-Pareto points are dominated. The removal of these configurations can result in the final number of stored solutions to be less than L. The availability of multiple solutions within one experiment provides designers another option to select appropriate solutions (besides the option to a-priori defining the WL threshold and overflow parameters).

5.4.2 Coll-MGR for Minimizing Congestion and Wirelength

In this section, we evaluate the variation of our Coll-MGR procedure for congestion spreading to be traded off with wirelength as given in Section 5.3.2. The objective of Coll-MGR is to simultaneously minimize wirelength and the total utilization of edges in the routing grid-graph which are utilized higher than 70% of their normalized capacity. This threshold for highly-utilized edges is mentioned in [4]. We refer to these edges as the HUT edges (i.e., highly-utilized) and denote the summation of

Table 5.4: Information about the HUT edges of the input routing solutions.

Bench	NTHU-R [8]		FastRoute [66]		BFGR [25]	
	U_{HUT}	|HUT|	U_{HUT}	|HUT|	U_{HUT}	|HUT|
a1	1273868	257960	1316280	285338	1406310	295205
a2	1134378	201647	1210836	219086	1384318	250004
a3	3362124	714015	3233446	742605	3724838	830651
a4	2210568	583105	2208114	637746	2863184	788055
a5	3496494	458074	3664166	495728	4027622	527272
n1	887098	198717	946270	213842	1037758	230537
n2	1120624	218546	1186228	249327	1373000	273598
n3	2431870	465796	2431870	481585	2904822	618780
b1	1611590	181535	1739818	196242	1698788	196805
b2	1936104	495846	1999832	558911	2158884	536065
b3	1923616	292576	2043724	312910	2502030	372292
b4	3570538	378057	3570538	386102	4307742	445552
n4	3515696	497985	2909278	520849	2909278	587634
n5	4694444	821161	5075392	908105	5615164	964734
n6	3672558	395938	3844852	424234	4053064	442132
n7	6182088	596755	6182088	614273	7399204	696069
average	**2688979**	**422357**	**2722671**	**452930**	**3085375**	**503462**

the utilization of the HUT edges by U_{HUT}. Reducing the utilization of the HUT edges allows more effective handling of issues such as crosstalk at the lower design stages [4]. Table 5.4 shows the information about the HUT edges for the different routing procedures. For each case, the first column represents the total utilization of the HUT edges and the second column shows the number of these edges. For each benchmark the underlined number shows the smallest U_{HUT} among the three different procedures. For example for `a1`, NTHU-R [8] has the smallest total utilization on the HUT edges.

We impose a bound of 0% for the maximum wirelength degradation so no wirelength degradation is allowed in this experiment. This is compared to the "Min-HUT" case which corresponds to one of the input solutions for each benchmark, as we shortly explain. We apply multi-objective Coll-MGR and among the stored solutions at the final step of the algorithm select the one with smallest U_{HUT} as the final solution. In Coll-MGR's net ordering procedure, the solution of the Min-HUT case is used.

Table 5.5: Results for simultaneous minimization of the HUT edges and wirelength

	Coll-MGR													
Bench	**%imp. for L=4**							**%imp. for L=7**						
		(Min-HUT)			U_{HUT}				**(Min-HUT)**			U_{HUT}		
	T	U_{HUT}	**|HUT|**	**WL**	**NTHU-R**	**BFGR**	**FastRoute**	**T**	U_{HUT}	**|HUT|**	**WL**	**NTHU-R**	**BFGR**	**FastRoute**
a1	1.65	3.67	1.71	0.17	3.67	12.75	6.78	2.65	3.67	1.71	0.17	3.67	12.75	6.78
a2	2.00	6.78	3.12	0.23	6.78	23.61	12.67	4.80	6.83	3.18	0.25	6.83	23.66	12.72
a3	4.95	1.99	7.96	0.26	5.74	14.92	1.99	8.58	4.03	7.98	0.27	7.70	16.69	4.03
a4	4.75	12.78	15.62	0.31	12.88	32.74	12.78	8.50	13.90	17.05	0.31	14.00	33.60	13.90
a5	5.80	5.60	2.04	0.21	5.60	18.05	9.92	12.41	8.20	1.89	0.21	8.20	20.30	12.40
n1	1.90	5.91	3.90	0.26	5.91	19.57	11.79	3.18	5.94	3.97	0.26	5.94	19.60	11.82
n2	3.05	11.46	4.23	0.33	11.46	27.74	16.36	6.02	11.49	4.22	0.33	11.49	27.76	16.39
n3	4.55	13.46	4.46	0.29	13.46	27.55	14.14	10.43	13.93	4.42	0.29	13.93	27.94	14.61
b1	2.05	1.75	0.10	0.11	1.75	8.99	6.79	4.86	9.33	0.07	0.11	9.33	16.01	13.98
b2	3.40	4.01	1.75	0.14	4.01	13.91	7.07	6.61	4.03	1.60	0.15	4.03	13.93	7.08
b3	5.05	14.08	6.69	0.41	19.13	33.95	19.13	14.52	18.96	9.93	0.41	23.72	37.70	23.72
b4	5.00	8.84	4.57	0.34	8.84	24.44	12.63	21.17	8.95	4.60	0.35	8.95	24.53	12.73
n4	5.00	5.73	6.36	0.21	5.73	21.99	8.99	13.11	9.22	6.23	0.21	9.22	24.88	12.35
n5	5.45	5.16	1.30	0.18	5.16	20.71	12.28	19.24	8.20	1.34	0.18	8.20	23.25	15.09
n6	7.00	4.60	1.41	0.23	4.60	13.56	8.88	20.32	7.39	1.43	0.24	7.39	16.08	11.54
n7	8.15	8.22	2.80	0.43	8.22	23.31	12.15	26.23	11.53	2.84	0.44	11.53	26.08	15.32
ave	**4.36**	**7.13**	**4.25**	**0.26**	**7.68**	**21.11**	**10.90**	**11.41**	**9.10**	**4.53**	**0.26**	**9.63**	**22.80**	**12.78**

In our framework the parameter L defines the maximum number of routing solutions stored at each step of our procedure. Increasing L allows creating more solutions which provide a tradeoff set in the two considered objectives. However it also increases the runtime of our procedure. We experiment with two cases of L 4 and 7.

The results are reported in Table 5.5. For a routing solution, we report U_{HUT}, and the number of HUT edges which we denote by |HUT|. For each benchmark, we first identify the tool with minimum U_{HUT} among the three routing procedures based on the information in Table 5.4 and define it as Min-HUT. For each benchmark, the underlined number in Table 5.4 represents the Min-HUT case.

We compare these metrics for the Coll-MGR solution in columns 2 to 8 for the case when $L = 4$. Columns 3 to 5 report the percentage improvement in these metrics in Coll-MGR over the Min-HUT case. On average, Coll-MGR reduces the utilization on the HUT edges by 7.13%, and the number of HUT edges by 4.25%. Interestingly,

for these benchmarks, our framework provides solutions that are always better than the original solutions both in terms of wirelength and the congestion metrics. The average improvement in wirelength is 0.26%. Also note that the total overflow does not exceed the total overflow of the tool with minimum total overflow. For most of the benchmarks (all except `n3`, `b4`, `n4`, and `n7`), all the routing procedures have 0-overflow solutions (as can be seen in Table 5.1) and no adjustment in capacity is necessary. So our solution has also 0 overflow. For example in `a4` we have a 0-overflow solution while reducing |HUT| by almost 15.62% compared to the Min-HUT case and improving the wirelength by 0.31%.

Next, columns 6 to 8 report the percentage improvements in total utilizations of HUT edges in Coll-MGR compared to each tool. The U_{HUT} is improved by 7.68%, 21.11%, 10.90% in NTHU-R [8], BFGR[25], and FastRoute[66], respectively.

The runtime of Coll-MGR (in minutes) is reported in column 2. The average runtime of Coll-MGR in this experiment is 4.36 minutes. The runtimes remain consistently in the order of few minutes even for the unroutable benchmarks.

The results for the case when $L = 7$ are also reported in columns 9 to 15. Columns 10 to 12 show the percentage of improvement for U_{HUT}, the number of HUT edges, and wirelength over the Min-HUT case. On average the utilization of the HUT edges is reduced by 9.10% and the number of HUT edges by 4.53%. The wirelength is always less than the wirelength of the Min-HUT case. Finally, the percentage improvements in total utilizations over different tools are reported in columns 13 to 15. When $L = 7$, Coll-MGR improves the utilization of the HUT edges from NTHU-R [8], BFGR [25], and FastRoute [66], by 9.63%, 22.80%, and 12.78%, respectively. The runtimes for this case are larger than the case when $L = 4$, but they are still in the order of few

minutes, consistently and even for the unroutable benchmarks. The average runtime in this case is 11.41 minutes. This increase in runtime is traded off with improvement in the solution quality; as can be seen U_{HUT}, |HUT| and WL are each improved in each benchmark compared to the corresponding case for $L = 4$.

In summary, in the congestion spreading experiment we showed that Coll-MGR can be used to significantly reduce the number and total utilization of the HUT edges with a runtime of few minutes for any of the ISPD 2008 benchmark and without any wirelength degradation.

5.4.3 Coll-MGR for Minimizing Wirelength

We implemented a variation of our Coll-MGR procedure for minimizing wirelength (together with routing resource usage) as described in Section 5.3.3. For the net ordering procedure of Coll-MGR, the input solution with the smallest total wirelength is used. The results are reported in Table 5.6 for the routable benchmarks. Columns 2 to 6 report the wirelength of the five academic routers used to generate input routing solutions to Coll-MGR for this experiment.

To verify the impact of the number of collaborating routers, we experimented with $M = 4$ and $M = 5$. (For $M = 4$, MGR [65] was excluded from the set.) To also show the impact of the number of routing solutions generated by Coll-MGR we experiment with $L = 5$ and $L = 10$. The combination creates four cases of Coll-MGR for this experiment. Table 5.6 shows the wirelength (WL) and runtime for each case. The overflow is zero in Coll-MGR.

The multi-objective optimization in Coll-MGR is considering minimization of both wirelength and routing resource usage. In this case, a route with high wirelength

Table 5.6: Results for wirelength minimization for benchmarks without overflow

Bench	WL for Different Routers					Coll-MGR							
						M=4				M=5			
						L=5		L=10		L=5		L=10	
	NTHU-R	BFGR	FastRoute	NCTU-GR	MGR	WL	T	WL	T	WL	T	WL	T
a1	53.44	54.30	53.80	53.04	52.86	52.97	2.73	52.90	4.13	52.83	2.82	52.50	6.13
a2	52.29	52.30	52.20	51.51	51.47	51.43	3.32	51.34	4.30	51.32	3.91	51.13	6.48
a3	131.01	131.40	131.20	129.24	128.91	128.91	4.30	128.87	10.85	128.85	7.75	127.79	15.99
a4	121.69	121.60	121.30	120.53	119.95	120.41	6.26	119.80	9.66	119.86	8.26	119.04	15.65
a5	155.38	156.70	155.80	153.96	153.23	153.50	6.95	153.30	11.96	152.09	9.69	151.15	18.89
b1	55.95	57.20	56.60	56.29	55.88	56.28	2.24	56.21	5.34	55.85	4.03	55.85	7.31
b2	90.59	91.10	91.20	88.66	88.91	88.55	4.22	88.35	7.33	88.85	4.41	88.71	9.71
b3	130.69	131.80	130.00	129.35	128.76	129.09	4.67	128.71	8.94	128.55	8.58	128.04	16.49
n1	46.53	46.80	46.30	45.76	45.63	45.45	3.01	45.27	4.96	45.57	3.25	45.37	6.34
n2	75.70	75.70	75.20	74.51	74.49	74.14	4.06	74.11	6.87	74.35	5.76	73.62	10.62
n5	231.58	233.00	230.90	228.68	228.01	228.15	7.44	228.03	14.40	227.83	11.23	225.90	26.83
n6	176.89	180.10	177.50	175.49	174.93	174.45	7.57	175.06	17.19	174.73	10.82	174.44	21.69
ave	**110.15**	**111.00**	**110.17**	**108.92**	**108.59**	**108.61**	**4.82**	**108.50**	**8.58**	**108.39**	**6.63**	**107.80**	**13.25**

Table 5.7: Wirelength minimization for benchmarks with overflow

Bench	WL and OF for Different Routers										Coll-MGR		
	NTHU-R		BFGR		FastRoute		NCTU-GR		MGR		M=5 , L=10		
	WL	OF	WL	OF	WL	OF	WL	OF	WL	OF	WL	OF	T
b4	230.90	160	232.00	434	230.20	138	224.83	188	228.80	130	228.14	130	7.91
n3	106.57	31454	106.40	33900	108.40	31276	104.59	31688	105.79	31276	105.43	31276	5.72
n4	130.46	138	130.80	218	130.50	136	127.35	144	129.45	136	128.98	136	6.73
n7	355.22	54	352.10	606	353.40	54	342.37	54	349.94	58	341.48	54	15.78
ave	**205.79**	**7952**	**205.33**	**8790**	**205.63**	**7901**	**199.79**	**8019**	**203.50**	**7900**	**201.01**	**7899**	**9.04**

may have low routing resource usage if many vias are used on its path. This is because vias are modeled with infinite capacity and hence not viewed as routing resource in our experiments. The consistent improvement in wirelength is observed even when both wirelength and routing resource usage are simultaneously optimized in this experiment.

As can be seen, the higher value of M and L each result in improvement in wirelength. Increase in each parameter increases the runtime of Coll-MGR. The maximum wirelength improvement is for $M = 5$ and $L = 10$ which results in an average runtime of 13.25 minutes over the benchmarks. In this case, the average

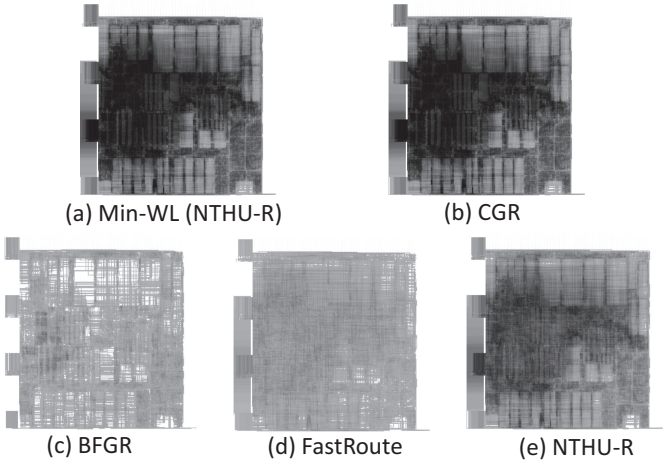

Figure 5.5: The utilization maps for `a1`.

wirelengths of MGR [65] and Coll-MGR are 108.59 and 107.80 respectively.

Table 5.7 shows the wirelength and overflow of all the tools. For Coll-MGR we only report the case when $M = 5$ and $L = 10$ which is the case resulting in the best solution quality. As can be seen, the average wirelength of MGR [65] and Coll-MGR are 203.50 and 201.01 with Coll-MGR having the equal or better overow for the unroutable benchmarks.

For a Coll-MGR solution, we can verify the amount and type of contribution by each collaborating tool. For example, Figure 5.5 shows the utilization maps for `a1` for $M = 3$ using NTHU-R [8], BFGR [25], and FastRoute [66]. The top row gives the utilization maps of Coll-MGR and of NTHU-R [8] which is the one with smallest wirelength (denoted by Min-WL). The bottom row shows the breakdown of the map of the Coll-MGR solution. For example Figure 5.5(e) shows the map corresponding to the 60.12% wirelength contributed by NTHU-R [8]. For this benchmark FastRoute and BFGR have similar wirelength contribution of about 20%. However, we observe that the map of FastRoute's contribution is spreading over the chip area, while the

Table 5.8: Layer-based contribution (in percentage) of each tool in a1

Tool	M1	M2	M3	M4	M5	M6
NTHU-R [8]	54.66	50.64	60.32	60.95	58.43	57.77
FastRoute [66]	39.90	36.27	22.87	21.33	9.19	8.46
BFGR [25]	5.44	13.08	16.81	17.73	32.38	33.78

map of BFGR's contribution is mostly concentrated in the middle-left and bottom-right of the chip. We verified that the BFGR contribution includes a higher number of shorter nets concentrated in these areas while the FastRoute contribution includes longer nets that span the chip area.

Furthermore, for the above example, Table 5.8 shows the contribution of each routing tool per layer in this example. For each tool and metal layer a percentage is reported that corresponds to the degree of routing resource usage in the Coll-MGR solution in that metal layer. (So each column adds to 100%.) Here we observe that the contribution of BFGR increases when moving to the higher metal layers (from 5.44% in M1 to 33.78% in M6), while the contribution of FastRoute decreases (from 39.90% in M1 to 8.46% in M6). The contribution of NTHU-R [8] is maximum in M3 and M4.

5.5 Concluding Remarks

We introduced Coll-MGR as a procedure for collaborative multi-objective global routing which receives as input different routing solutions. In general, for tight resource constraints, the Coll-MGR procedure may fail to find a feasible solution. However, we showed in practice a feasible solution is always attainable in our experiments. Furthermore, significant improvement is possible to optimize for other objectives in addition to the primary objectives of wirelength and overflow used in global routing.

Obtaining feasible and high-quality solutions in our experiments is due to effective realization of the Pareto-algebraic procedure of Coll-MGR for the three presented global routing variations. Key factors contributing to effective realization are summarized below.

First, in the Coll-MGR formulation, the normalized capacity of each edge is adjusted according to the overflow of the solution with the *smallest* total overflow. This helps controlling overflow even though it is not explicitly considered in the Coll-MGR procedure.

Second, due to the values of the edge capacities in the routing benchmarks (which are often higher than 1), infeasibility won't be an issue in the initial iterations of the Coll-MGR procedure because the edges are typically not utilized to their capacities.

Third, we apply a net ordering procedure which starts processing the nets passing from the congested areas first. These areas contain more overflow edges and by processing them prior to the un-congested regions, we allow for the maximum flexibility to avoid infeasibility in the later iterations. Moreover, we observed that the routes generated by the input routers were more diverse for the nets passing from the congested areas and not much different in the un-congested areas. This observation also helped with achieving feasibility combined with our net ordering procedure.

Fourth, a cost function is reasonable if it behaves monotonically with respect to the dimensions of a configuration. Since most of the cost functions considered in this work are monotonic, in the exact version of Coll-MGR (when the *reduce* procedure is excluded), a dominated configuration is guaranteed not to yield any better solution in future iterations and can be removed with certainty. Consequently, the Pareto points allow capturing the potentially interesting configurations.

Fifth, in the *reduce* phase, we select L configurations which provide a good approximation of the current configuration set. We choose configurations that cover the entire Pareto curve range uniformly. This method ensures that we always store the solution with *the lowest resource usage* as well as the one with the lowest cost and in addition some intermediate ones. This sampling method provides the highest possible flexibility in selecting fitting configurations in later steps.

When considering the selection of parameters M and L, available memory is a factor given the large size of the ISPD 2008 global routing benchmark which reflect industry-sized instances. Specifically, for each net M configurations are stored. At most L distinct routing solutions are stored at the end of each iteration of Coll-MGR. Furthermore, the number of intermediate stored solutions may exceed L when combining the configurations of a new net with the current configuration set. Moreover, the increase in parameters M and L increases the runtime of the algorithm; however it also allows more improvement in the solution quality as illustrated in the simulation results. In practice we expect M, the number of collaborating input solutions, to be a small value similar to those specified in our experiments. The parameter L can be set based on the memory size for the considered benchmark as well as the user's choice to tradeoff the solution quality with runtime of the procedure.

Overall, in this chapter, we presented a quick Pareto-algebraic procedure for multi-objective global routing and showed its benefit in practice using an industry setting and based on collaboration among recent academic global routing tools.

Chapter 6

Summary and Conclusions

With the advent of Moore's law into nanometer era, routability of a design has become extremely challenging. Modern designs typically contain many complicating features that significantly contribute to the issue of routing congestion. In addition, interconnects play a significant role in determining many crucial performance metrics of a design.

This dissertation made three contributions in rethinking global routing, as a key step in the design, in order to address the challenges associated with routability and consideration for multiple performance metrics of modern designs.

First, we introduced CGRIP, as a fast routing congestion analysis tool. CGRIP operates on a flexible model of global routing that captures various factors that contribute to routing congestion including, varying wire sizes and spacings among different metal layers, irregular shapes of routing blockages, and virtual pins. We proposed a parameterized Integer Programming formulation for the congestion analysis problem that identifies and ranks the regions on the layout, defined by an input resolution

parameter, which contain overflow. The formulation offers new constraints and objectives for congestion analysis in a small time budget which is an extension of a standard global routing formulation that minimizes the total overflow.

To engineer a practical solution approach to our formulation, we introduced several new ideas, such as working with a reduced-sized and relaxed problem model and integrating it with a traditional ripup and re-route framework, simultaneous re-routing of multiple "equivalent" nets, congestion-aware multi-terminal net decomposition, and a congestion-aware layer assignment technique.

We also described a coalesCgrip, a simpler variations of our framework with fewer features which was used to judge the ISPD 2011 content on routability-driven placement.

We have provided an open-source and user-friendly application programming interface (API) in C++ that offers all the functions needed to use by a routability-driven placer. Specifically, we have implemented a function by which incremental routing is possible. If designers make small changes, for example by moving a small subset of cells, CGRIP is able to reflect the impact of the change more quickly than running from scratch. This is an important feature of CGRIP for embedding it within a routability-driven placer. Both coalesCgrip and CGRIP are freely available online and have been downloaded by many research groups from different countries since 2011.

Next, we presented LCGRIP, our local-congestion-aware routing framework which accounts for local effects at the global routing stage in addition to the issues already considered in CGRIP. In the standard physical design flow, local nets are completely ignored at the global routing stage. However, in modern design with high pin den-

sities inside global cells, ignoring congestion effects of local wires will cause serious routability issues at the detailed routing stage.

We proposed two techniques for considering local effects during global routing. First, we introduced a technique for constructing global routing instances with global cells of non-uniform size that can *reduce* the number of local nets while keeping the number of global cells intact. Second, we proposed to *approximate* the area required to route local nets and to adjust the areas of the affected global cells before global routing. Using these two complementary techniques to approximate and reduce the local nets, a graph model of global routing with non-uniform global cells is presented which includes vertex capacity in addition to the conventionally-used edge capacity. Our graph model also captures other factors contributing to congestion such as varying wire size and spacing, routing blockages, and virtual pins. An Integer Programming (IP) model was presented describing this *comprehensive* congestion-aware global routing problem. Our computational results show that significant improvement in the quality of (estimated) detailed routing solutions can be obtained using our methods compared to CGRIP.

Finally, we introduced Coll-MGR, a collaborative and multi-objective global routing framework which combines the routing solutions generated by different routing tools and in practice always obtains an improved solution in a very short amount of time. Specifically, we discussed different variations of multi-objective routing to jointly minimize wirelength and power, minimize the highly-utilized edges in the routing grid-graph under wirelength constraints, and minimize wirelength and utilization of the routing resources. We proposed a Pareto-algebraic heuristic to solve these variations in a very fast runtime. Our farmework generates several complete routing

solutions that make a tradeoff between considered objectives. We demonstrated in our experiments that collaborative and multi-objective routing helps to significantly reduce the corresponding metrics in each variation. This is without any increase in the minimum overflow among the collaborating routing tools.

We believe our fast and accurate congestion analysis framework, CGRIP, can shed light on routability issues of the design and open the door to interesting research opportunities for improving the routability at the early stages of the design such as floorplanning and placement. Also our comprehensive global routing procedure may further improve routability issues and alleviate the design-closure problem in the design trajectory. In addition, our collaborative and multi-objective global routing procedure, Coll-MGR, offers a promising path to effective simultaneous optimization of several performance metrics, and comprehensively addresses performance shortcomings early in the design flow.

Future work includes integrating CGRIP with a routability-driven placer to provide quick and accurate feedbacks about routing congestion, and to verify the impact of our formulation and the resolution parameter in improving routability of the design.

Another direction for routability improvement in the nanometer era is to account for even more congestion issues. As mentioned in Chapter 1, modern designs usually have varying wire sizes and spacings among different metal layers. In these designs, via resource utilization does not follow traditional assumptions [3]. Typically, when a via connects two layers with different pitches, the via is as wide as the thicker one [3]. In this case, the via will consume wiring resources on the thinner metal layer. Consequently, the impact of inter-layer vias need to be considered for more accurate congestion estimation [3]. In our future work, we plan to study the impact

of inter-layer vias as another factor contributing to routing congestion.

Furthermore, our collaborative framework can be extended to consider delay as another objective. We have already considered power, congestion and wirelength metrics in our multi-objective framework. Global routing is also a suitable stage to improve the timing on the critical paths by reducing the wire delays on these paths. The delays of the wires can be approximated considering their topology, loading, as well as wire size and spacing. The above factors (i.e. metal layer and wire width) are determined during the global routing stage. Moreover, wire spacing can also be approximated at this stage. Therefore, opportunities lie in interconnect delay optimization at this stage.

Bibliography

[1] ISPD 2007 global routing contest and benchmark suite [online]http://www.sigda.org/ispd2007/rcontest/.

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

[3] C. J. Alpert, Z. Li, M. D. Moffitt, G.-J. Nam, J. A. Roy, and G. Tellez. What makes a design difficult to route. In *International Symposium on Physical Design*, pages 7–12, 2010.

[4] C. J. Alpert and G. E. Tellez. The importance of routing congestion analysis. In *DAC.COM Knowledge Center Article*, 2010.

[5] L. Behjat, A. Vannelli, and W. Rosehart. Integer linear programming models for global routing. *INFORMS Journal on Computing*, 18(2):137–150, 2006.

[6] U. Brenner and A. Rohe. An effective congestion driven placement framework. In *International Symposium on Physical Design*, pages 6–11, 2002.

[7] Y.-J. Chang, T.-H. Lee, and T.-C. Wang. GLADE: A modern global router

considering layer directives. In *International Conference on Computer-Aided Design*, pages 319–323, 2010.

[8] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang. NTHU - Route 2.0: A fast and stable global router. In *International Conference on Computer-Aided Design*, pages 338–343, 2008.

[9] H.-Y. Chen, C.-H. Hsu, and Y.-W. Chang. High-performance global routing with fast overflow reduction. In *Asia and South Pacific Design Automation Conference*, pages 582–587, 2009.

[10] C.-L. E. Cheng. RISA: accurate and efficient placement routability modeling. In *International Conference on Computer-Aided Design*, pages 690–695, 1994.

[11] 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 Transaction on Design Automation and Electronic Systems*, 14(2):32:1–32:21, 2009.

[12] M. Cho and D. Z. Pan. BoxRouter: A new global router based on box expansion and progressive ILP. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 26(12):2130–2143, 2007.

[13] C. Chu and Y.-C. Wong. FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 27(1):70–83, 2008.

[14] K.-R. Dai, W.-H. Liu, and Y.-L. Li. NCTU-GR: Efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-d global routing.

IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 20(3):459–472, 2012.

[15] K.-R. Dai, C.-H. Lu, and Y.-L. Li. GRPlacer: Improving routability and wirelength of global routing with circuit replacement. In *International Conference on Computer-Aided Design*, pages 351–356, 2009.

[16] J.-R. Gao, P.-C. Wu, and T.-C. Wang. A new global router for modern designs. In *Asia and South Pacific Design Automation Conference*, pages 232–237, 2008.

[17] M. Geilen and T. Basten. A calculator for Pareto points. In *Design, Automation, and Test in Europe*, pages 16–20, 2007.

[18] M. Geilen, T. Basten, B. D. Theelen, and R. Otten. An algebra of Pareto points. *Fundamenta Informaticae*, 78(1):35–74, 2007.

[19] M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265, 1966.

[20] P. E. Hart and N. J. Nilsson. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(37):100–107, 1968.

[21] X. He, T. Huang, L. Xiao, H. Tian, G. Cui, and E. F. Y. Young. Ripple: An effective routability-driven placer by iterative cell movement. In *International Conference on Computer-Aided Design*, pages 74–79, 2011.

[22] C.-H. Hsu, H.-Y. Chen, and Y.-W. Chang. Multilayer global routing with via and

wire capacity considerations. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 29(5):685–696, 2010.

[23] M.-K. Hsu, S. Chou, T.-H. Lin, and Y.-W. Chang. Routability-driven analytical placement for mixed-size circuit designs. In *International Conference on Computer-Aided Design*, pages 80–84, 2011.

[24] J. Hu, J. A. Roy, and I. L. Markov. Sidewinder: a scalable ILP-based router. In *International Workshop on System Level Interconnect Prediction*, pages 73–80, 2008.

[25] J. Hu, J. A. Roy, and I. L. Markov. Completing high-quality global routes. In *International Symposium on Physical Design*, pages 35–41, 2010.

[26] *IBM ILOG CPLEX V12.0, User's Manual for CPLEX*, 2009.

[27] ISPD 2008 global routing contest and benchmark suite [online]http://www.sigda.org/ispd2008/contests/ispd08rc.html.

[28] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer, 2011.

[29] A. B. Kahng and X. Xu. Accurate pseudo-constructive wirelength and congestion estimation. In *International Workshop on System Level Interconnect Prediction*, pages 61–68, 2003.

[30] M.-C. Kim, J. Hu, D.-J. Lee, and I. L. Markov. A SimPLR method for routability-driven placement. In *International Conference on Computer-Aided Design*, pages 67–73, 2011.

[31] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. 7(1):48–50, 1956.

[32] C. Y. Lee. An algorithm for path connection and its application. In *IRE Transaction on Electronic Computers, 10*, pages 346–365, 1961.

[33] T.-H. Lee, Y.-J. Chang, and T.-C. Wang. An enhanced global router with consideration of general layer directives. In *International Symposium on Physical Design*, pages 53–60, 2011.

[34] T.-H. Lee and T.-C. Wang. Congestion-constrained layer assignment for via minimization in global routing. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 27(9):1643–1656, 2008.

[35] C. Li, M. Xie, C. Koh, J. Cong, and P. Madden. Routability-driven placement and white space allocation. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 26(5):858–871, 2007.

[36] W.-H. Liu, Y.-L. Li, and K.-Y. Chao. High-quality global routing for multiple dynamic supply voltage designs. In *International Conference on Computer-Aided Design*, pages 263–269, 2011.

[37] J. Lou, S. Krishnamoorthy, and H. S. Sheng. Estimating routing congestion using probabilistic analysis. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 21(1):32–41, 2002.

[38] M. Moser, D. P. Jokanvi, and N. Shiratori. An algorithm for the multidimensional multiple-choice knapsack problem. *IEICE Transaction on Fundamentals of Electronics*, E80-A(3):582–589, 1997.

[39] D. Müller. Optimizing yield in global routing. In *International Conference on Computer-Aided Design*, pages 480–486, 2006.

[40] Nangate 45nm Open Cell Library [Online]. In *http://www.nangate.com*, 2008.

[41] E. D. Nering and A. W. Tucker. *Linear Programs and Related Problems.* Academic Press, 1993.

[42] M. Pan and C. Chu. FastRoute: A step to integrate global routing into placement. In *International Conference on Computer-Aided Design*, pages 464–471, 2006.

[43] M. Pan and C. Chu. IPR: An integrated placement and routing algorithm. In *Design Automation Conference*, pages 59–62, 2007.

[44] R. C. Prim. Shortest connection networks and some generalizations. 36:1389–1401, 1957.

[45] J. A. Roy and I. L. Markov. Seeing the forest and the trees: Steiner wirelength optimization in placement. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 26(4):632–644, 2007.

[46] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 27(6):1066–1077, 2008.

[47] J. A. Roy, N. Viswanathan, G.-J. Nam, C. J. Alpert, and I. L. Markov. CRISP: Congestion reduction by iterated spreading during placement. In *International Conference on Computer-Aided Design*, pages 357–362, 2009.

[48] M. Saeedi, M. Zamani, and A. Jahanian. Prediction and reduction of routing congestion. In *International Symposium on Physical Design*, pages 72–77, 2006.

[49] R. Shelar and M. Patyra. Impact of local interconnects on timing and power in a high performance microprocessor. In *International Symposium on Physical Design*, Invited talk http://ispd.cc/slides10/8_01.pdf, 2010.

[50] H. Shojaei, T. Basten, M. Geilen, and P. Stanley-Marbell. SPaC: A symbolic Pareto calculator. In *International Conference on Hardware/Software Codesign and System Synthesis*, pages 179–184, 2008.

[51] H. Shojaei, A. Davoodi, and J. Linderoth. Congestion analysis for global routing via integer programming. In *International Conference on Computer-Aided Design*, pages 256–262, 2011.

[52] H. Shojaei, A. H. Ghamarian, T. Basten, M. Geilen, S. Stuijk, and R. Hoes. A parameterized compositional multi-dimensional multiple-choice knapsack heuristic for CMP run-time management. In *Design Automation Conference*, pages 917–922, 2009.

[53] H. Shojaei, T.-H. Wu, A. Davoodi, and T. Basten. A Pareto-algebraic framework for signal power optimization during global routing. In *International Symposium on Low Power Electronics and Design*, pages 407–412, 2010.

[54] P. Spindler and F. M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *Design, Automation and Test in Europe*, pages 1226–1231, 2007.

[55] 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 *International Conference on Computer-Aided Design*, pages 621–624, 2010.

[56] 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 *International Symposium on Physical Design*, pages 141–146, 2011.

[57] N. Viswanathan and C. Chu. Fastplace: efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In *International Symposium on Physical Design*, pages 26–33, 2004.

[58] L. Wang, Y. Chang, and K. Cheng. *Electronic Design Automation: Synthesis, Verification, and Test (Systems on Silicon)*. Morgan Kaufmann, 2009.

[59] M. Wang, X. Yang, K. Eguro, and M. Sarrafzadeh. Multi-center congestion estimation and minimization during placement. In *International Symposium on Physical Design*, pages 147–152, 2000.

[60] M. Wang, X. Yang, and M. Sarrafzadeh. Congestion minimization during placement. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 19(10):1140–1148, 2000.

[61] J. Westra, C. Bartels, and P. Groeneveld. Probabilistic congestion prediction. In *International Symposium on Physical Design*, pages 204–209, 2004.

[62] T.-H. Wu and A. Davoodi an J. T. Linderoth. Power-driven global routing for multi-supply voltage domains. In *Design, Automation, and Test in Europe*, pages 443–448, 2011.

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

[64] T.-H. Wu, A. Davoodi, and J. T. Linderoth. Global routing via integer programming. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 30(1):72–84, 2011.

[65] Y. Xu and C. Chu. MGR: Multi-level global router. In *International Conference on Computer-Aided Design*, pages 250–255, 2011.

[66] Y. Xu, Y. Zhang, and C. Chu. FastRoute 4.0: Global router with efficient via minimization. In *Asia and South Pacific Design Automation Conference*, pages 576–581, 2009.

[67] X. Yang, R. Kastner, and M. Sarrafzadeh. Congestion estimation during top-down placement. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 21(1):72–80, 2002.

[68] M. Yukish. Algorithms to identify Pareto points in multi-dimensional data sets. *PhD Thesis, Pennsylvania State University*, August 2004.

[69] Y. Zhang and C. Chu. RegularRoute: An efficient detailed router with regular routing patterns. In *International Symposium on Physical Design*, pages 45–52, 2011.