A Survey on Multi-net Global Routing for Integrated Circuits *

Jiang Hu and Sachin S. Sapatnekar

{jhu, sachin}@mail.ece.umn.edu

Department of Electrical and Computer Engineering

University of Minnesota

Minneapolis, MN 55455, USA

Tel: 612-625-0025, Fax: 612-625-4583

Abstract

This paper presents a comprehensive survey on global routing research over about the last two decades, with an emphasis on the problems of simultaneously routing multiple nets in VLSI circuits under various design styles. The survey begins with a coverage of traditional approaches such as sequential routing and rip-up-and-reroute, and then discusses multicommodity flow based methods, which have attracted a good deal of attention recently. The family of hierarchical routing techniques and several of its variants are then overviewed, in addition to other techniques such as move-based heuristics and iterative deletion. While many traditional techniques focus on the conventional objective of managing congestion, newer objectives have come into play with the advances in VLSI technology. Specifically, the focus of global routing has shifted so that it is important to augment the congestion objective with metrics for timing and crosstalk. In the later part of this paper, we summarize the recent progress in these directions. Finally, the survey concludes with a summary of possible future research directions.

*This work is supported in part by the NSF under contract CCR-9800992 and the SRC under contract 98-DJ-609.

Contents

1 Introduction

Eighteen years ago, a then new journal, the IEEE Transactions on Computer-Aided Design, presented a special issue on routing in microelectronics that contained several landmark papers. Another significant publication at about the same time was a collection of papers edited by Hu and Kuh [40]. In the years that have elapsed since the publication of these two works, with the scaling of feature sizes, VLSI technology and circuits have undergone dramatic progress, as characterized by Moore's law. Particularly in recent years, this has resulted in interconnect delay accounting for a major portion of circuit clock cycle, as a result of which VLSI physical design has grown to be a critical factor. Moreover, the growing circuit complexity has enlarged the size of the design automation problems in physical design and has brought forth a new set of challenges.

Within the physical design flow, one of the most critical steps is global routing, a stage where signal nets are connected coarsely under a given placement so that wire/via spaces are allocated to each signal net. The quality of the global routing solution directly affects chip area, speed, power consumption and the number of iterations required to complete the design cycle, and hence this step plays an important role in determining circuit performance. On the other hand, global routing is a notoriously difficult problem: even the most simple version of the problem, where a set of two-pin nets is to be routed under congestion constraints, is an NP-complete problem [49].

As a result of both the importance of the problem and its difficulty, a great deal of research has been carried out on global routing during the last two decades, covering a variety of design styles including gate arrays, sea of gates, standard cell-based designs and custom circuits. Various techniques and strategies have been proposed, including rip-up-and-reroute, hierarchical methods, multi-commodity flow techniques and iterative deletion. However, even with all of these efforts, it is not entirely accurate to imply that the global routing problem has been solved satisfactorily. In particular, the newest advances in VLSI technology have raised a new set of issues to be solved and have further complicated the requirements on global routing.

The purpose of this survey is to provide a comprehensive overview of research in global routing, with specific emphasis on the problem of the simultaneous global routing of multiple nets in integrated circuits. This problem requires competent resource management as the global nets compete for a restricted set of global resources such as routing resources, crosstalk budgets and net delays. The

nature of this problem has changed remarkably over the last two decades, with several design styles gaining favor and then falling out of favor, but many of the fundamental techniques that were introduced are useful to other more modern design paradigms. In reading this survey, the reader is cautioned not to set too much store on the specific technology being discussed, but rather, to focus on the underlying algorithms in an attempt to determine how best they may be extended to the specific routing problem du jour.

There are several prior surveys that complement the material presented here. Two early surveys on global routing are presented in a paper by Kuh and Marek-Sadowska [50], and in a chapter of the book by Lengauer [56]. The books by Sherwani [78], Sait and Youssef [73] and Sarrafzadeh and Wong [75], present a more updated coverage of progress in global routing. The book by Kahng and Robins [45] and the survey paper by Cong *et al.* [16] focus on global routing issues for a single net. In most of these sources, the attention paid to the problems of simultaneously routing multiple global nets is limited, and the objective of this work is to attempt to bridge that gap. In the remainder of this paper, the phrase "global routing" will implicitly imply, unless otherwise stated, that the routing of all nets is being considered. For a survey of this type, it is appropriate to also list a set of related routing problems that have not been covered in this survey in order to limit its scope. These include MCM/PCB routing, FPGA routing, single layer routing, and parallel algorithms for global routing.

This paper is organized as follows. At the outset, the problem background and formulation are described in Section 2, and the basic techniques that are frequently used in global routing are summarized in Section 3. Section 4 introduces the first set of global routing algorithms, namely, sequential routing techniques. Edge-wise routing methods are covered in Section 5, followed by a brief review of move-based heuristics in Section 6. Next, rip-up-and-reroute approaches are described in Section 7 and multicommodity flow based methods in Section 8. Hierarchical global routing procedures and the iterative deletion technique are discussed in Sections 9 and 10, respectively. Most of the methods described until this section use conventional metrics to manage resource contention. Towards the end, in Sections 11 and 12, we present ways in which some objectives related to timing driven and crosstalk driven global routing are incorporated. Finally, Section 13 concludes this survey and suggests directions for future research.

2 Problem background and formulation

For a given location for every cell or macro-block in the layout, each set of electrically equivalent pins that belong to different cells or macro-blocks, called a net, must be wired together in a step commonly referred to as routing. Due to technological or methodological restrictions, it is often the case that some areas of the layout are prohibited from allowing wire routes, corresponding to wiring blockages. The fundamental goal of the routing step is to connect every net successfully and to resolve resource contentions. In modern VLSI design, this could be an exceedingly complex problem if it were done in a single step, since there could be millions of elements and nets integrated on a single chip. One common approach is to divide the routing procedure into two stages: global routing and detailed routing. In global routing, the chip area is divided into a set of coarsely-defined regions, and wires that cross the boundaries of these regions are allocated a coarse route that determines which regions they must traverse. Following this step, the routing within each such region is carried out in the detailed routing stage.

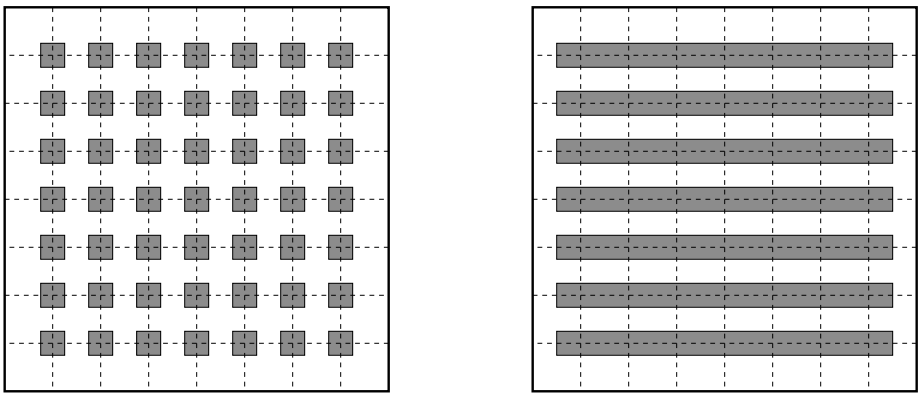

Figure 1: Tessellation for (a) gate array/sea of gates and (b) standard cell design.

Depending on the design style, different forms of routing graphs are constructed to define the coarse regions for global routing. For gate array, sea of gates and standard cell designs, the entire routing area is typically tessellated into a grid array, as shown in Figure 1. Note that the horizontal grid line usually goes through the middle of a row of cells in the standard cell design. The dual graph of this tessellation is the routing graph $G = (V, E)$, as shown in Figure 2(a). Each vertex $v \in V$ represents a grid cell and each edge $e \in E$ corresponds a boundary between two adjacent grid cells. In custom design, on the

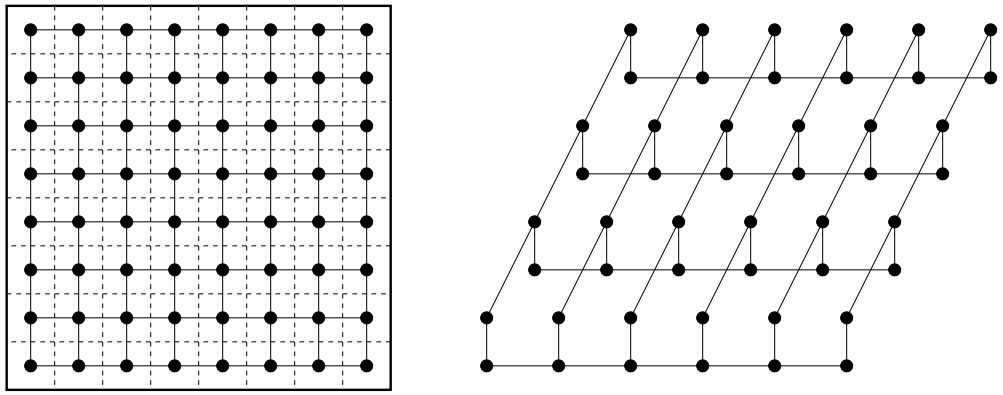

Figure 2: The construction of a routing graph from the tessellation for a two-layer routing scheme. Via constraints are explicitly modeled in (b) but not in (a).

other hand, the cell placement is not regular as in gate array or standard cell design styles. Therefore, the routing graph is more appropriately based on the floorplan of the building blocks. This is illustrated for an example floorplan in Figure 3 where each rectangle represents a building block. Like the routing graph for gate array and standard cell designs, this routing graph is also a dual to the floorplan graph.

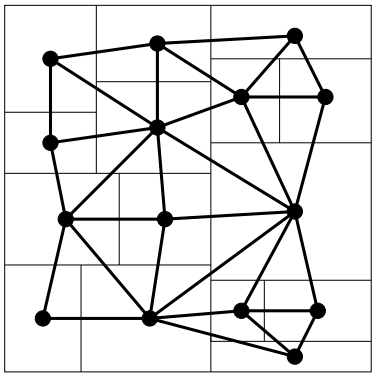

Figure 3: Routing graph for custom design.

The global routing problem requires a set of nets $\mathcal{N} = \{N_1, N_2, ..., N_k\}$ to be routed over the routing graph G. A net $N_i, 1 \leq i \leq k$ is a set of pins (or terminals) $\{v_{i,0}, v_{i,1}, v_{i,2}, ...\} \subseteq V$, among which $v_{i,0}$ is the source pin and the others are sink pins. Under the coarse assumptions made for global routing, each pin is often assumed to lie at the center of the grid cell that contains the pin. The routing problem for a

net N_i is to find an additional subset of vertices $V_{i,Steiner} \subset V$ and a set of edges $E_i = \{e_{i,1}, e_{i,2}, ...\} \subset E$ to form a rectilinear minimum spanning tree $T_i = (V_i, E_i)$, where $V_i = N_i \cup V_{i,Steiner}$.

As multiple nets are routed over the routing graph G, a common boundary between two neighboring grid cells may be crossed by wires belonging to different nets. In terms of the graph representation, this implies that the routing trees for these nets utilize a common edge $e \in E$. The number of wires that utilize an edge $e \in E$ is called the *flow* (or *demand*), $f(e)$, on the edge. The number of available routing tracks, which forms an upper bound on the allowable demand, is referred to as the *boundary capacity* (or *supply*), $u(e)$. The ratio of flow to capacity $f(e)/u(e)$ for an edge is referred to as the *density* or *congestion*, $\lambda(e)$. If $f(e) > u(e)$, then the *overflow* on the edge is defined as $\phi(e) = f(e) - u(e)$; otherwise the overflow is zero. The most fundamental statement of the global routing problem is to route all of the nets to ensure that for each edge e corresponding to a cell boundary, the number of wires across it does not exceed its routing capacity, i.e., $\phi(e) = 0, \forall e \in E$.

Another important consideration in routing global nets is to manage the number of vias on the net, and it is relatively easy to modify the above formulation to address this objective. Let us consider the routing problem under the reserved layer model, where the routing direction for all wires in a layer is identical. If a routing path makes a change in direction, i.e., a bend is induced, then a layer change is necessitated, and wires along different directions must be connected through a via at the bend position. For several reasons, such as considerations of reliability, area, and signal delay and quality, it is desirable to minimize or control the number of vias during routing. One way to do so is to limit the number of allowable vias in each grid cell. Under this scenario, the available via spaces in a grid cell may also be modeled by an edge, and we can obtain a routing graph G of the type shown in Figure 2(b) for a two-layer routing problem. This routing graph is identical to that defined earlier, except for the introduction of via edges along the grid. For a via edge e, we define $f(e)$ as the number of vias utilized in the corresponding grid cell and $u(e)$ as the number of available via spaces in the cell. In the literature, some works explicitly consider this via constraint while others do not, but these can often be extended fairly easily to do so.

The specific constraints that are of particular importance vary with the design style, but congestion management is a uniformly important objective across all design styles. For gate array/sea of gates designs, the locations of the gates are fixed, and global routing is an effort to route every net without any

overflow. The global router on this type of design is typically evaluated through metrics that measure the number of nets routed without overflow, or the total overflow with all of the nets routed. In custom design, the objective is similar, and is sometimes extended to minimize the maximum congestion $\hat{\lambda}$ over all edges $e \in E$, even if no wiring overflow occurs. The motivation for doing so is to provide a greater flexibility to a subsequent detailed routing step within a grid cell by evenly spreading out the global wires over the grid. For traditional standard cell design in a two-layer environment, only horizontal wiring channels, which are the spaces between two adjacent row of cells, are available. If a net crosses two different channels, then feedthrough cells must be inserted into each row of cells between the two channels to allow the the inter-channel connection for this net. The problem of assigning feedthroughs to improve the routing quality constitutes a separate problem that is not treated in this survey.

In addition to the congestion objective, wire length is an important concern: large wire lengths imply a larger power consumption and often, greater delays. It is not hard to see that optimizing congestion and wire length can often be competing objectives, and some global routing works have attempted to combine these objectives together. Another important consideration in the deep submicron era is that the interconnect delay, which consumes a major portion of the clock cycle today and is projected to continue to do so in the future. Therefore, timing performance must be included into the set of objectives considered during global routing. In addition, crosstalk has become an increasingly vexing issue and an attempt at crosstalk management during global routing is of great value in an overall strategy to manage coupling effects between wires.

3 Basic techniques

In this section, we summarize some of the basic techniques that are frequently used to solve subproblems in global routing. These include maze routing, Steiner tree construction, 0-1 integer linear programming, and network flows.

3.1 Maze routing

One basic subproblem that is commonly encountered in global routing is that of finding a shortest path connecting two pins in the presence of wiring blockages. Perhaps the most well known solution to this problem is the maze routing [54,66] algorithm, which works on a routing graph similar to the grid graph

in Figure 2(a). Each edge e has a cost $c(e)$ associated with it, and this cost may be different for different routing directions. If an edge e is within the wiring blockage area, then its cost $c(e) = \infty$; otherwise its cost is finite and specified. A common cost metric is the rectilinear edge length, counted in terms of the number of grid cells.

The maze routing algorithm can be regarded as an implementation of Dijkstra's shortest path algorithm [26] on the routing graph that minimizes the path cost. For a given graph $G = (V, E)$, where each edge $e \in E$ has a cost $c(e)$, Dijkstra's algorithm is able to find a minimum cost path $p(v_s, v_t)$, which is set of consecutive edges, connecting a source vertex $v_s \in V$ and a target vertex $v_t \in V$. This algorithm consists of a cost labeling step followed by a path tracing phase. In the cost labeling phase, starting from the source vertex v_s, the accumulated cost from the source to each vertex is labeled one by one in a "wave expansion" manner until the target vertex is reached. The minimum cost path is then traced back from the target vertex to the source by retrieving the bookkeeping information that is maintained in the cost labeling phase. The runtime for this algorithm is $O(|E| + |V| \log |V|)$ through a Fibonacci heap implementation, where $|E|$ is the number of edges and $|V|$ is the number of vertices.

If there is an optimal shortest path, maze routing is guaranteed to find it. However, in practice, it is found that the maze routing algorithm is slow and has large memory requirements, and many efforts [78] have attempted to improve its speed and memory usage. Another well known class of methods is the category of line probe-based algorithms (see, for example, [34]), which do not rely on a grid graph. While these methods are faster than maze routing, they are not guaranteed to find a solution, even if one exists. For a good description on maze routing and related algorithms, the reader is referred to [78].

The maze routing algorithm has been extended to find a path connecting two pins in such a way that it favors a path that passes through less congested areas [68]. Since maze routing inherently considers only one net at a time, this extension requires nets to be considered one at a time, with the consequent dependence on the order in which nets are processed. The procedure computes the edge cost $c(e), e \in E$, so that it reflects the current congestion at its corresponding cell boundary, instead of a distance metric such as the rectilinear length. There are several variations on this cost definition: for example, one

could use the density $\lambda(e)$, or $\lambda^2(e)$. Another effective cost function, in our experience, is given by

$$c(e) = \begin{cases} \frac{f(e)+1}{u(e)-f(e)} & : \quad f(e) < u(e) \\ \infty & : \quad f(e) \geq u(e) \end{cases}$$

An extension to maze routing that considers congestion costs explicitly can achieve the purpose of not only helping the path to avoid wiring blockages, but also of distributing the routing congestion, and therefore this technique is frequently used in global routing. A technique for timing driven maze routing has been proposed recently in [43, 44]

3.2 Steiner tree construction

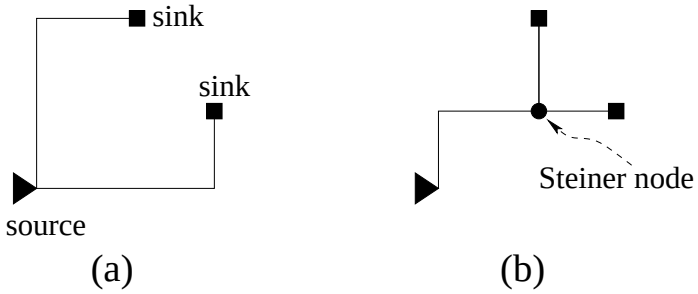

Figure 4: A three pin net connected by (a) a spanning tree and (b) a Steiner tree.

Both maze routing and line probe methods are designed for connecting two pin nets. However, in practice, nets with more than two pins are often encountered in the routing problem. A common approach in dealing with a multi-pin net is to decompose it into a set of two-pin nets. One way of performing this decomposition is to begin with constructing a minimum spanning tree (MST) over the pins, and to maze-route each pair of pins that corresponds to an edge in the MST. In Figure 4(a), a possible solution using this procedure for routing a three-pin net, decomposed into two two-pin subnets, is shown. It is easily seen that such a procedure may not lead to the best routing quality. The wire length of a routing tree can often be reduced by introducing extra nodes in addition to the given pins, and constructing an MST over all of these nodes. The added nodes are called Steiner nodes, and an illustration is shown in an example in Figure 4(b).

In most VLSI routing problems, since all of the wires are either horizontal or vertical, only rectilinear

Steiner trees (RST) are considered. One basic RST construction formulation is to minimize the wire length of the tree, i.e., to construct a rectilinear Steiner minimum tree (RSMT). RSMT construction is a well-known NP-complete problem and many approximation algorithms have been developed. Several of these are based on results that relate the wire length of the optimal RSMT to that of the optimal MST. When timing is considered, merely minimizing wire length is not adequate. As a first approximation, it can be observed that the length of the source-sink routing path length plays an important role in determining the signal delay. Several early works [5, 17] are dedicated to finding a good compromise between wire length and the maximum source-sink path length (also known as the radius). Later research efforts have directly considered delay metrics in the tree construction procedure [7, 19, 22, 36, 37, 58, 85]. For surveys on Steiner tree construction techniques for VLSI routing, the reader is referred to [11, 16, 45].

Many of the Steiner tree construction algorithms that have been proposed in the literature focus on the optimization of a single net, and do not consider wire congestion issues explicitly. Nevertheless, these algorithms can be applied to serially route the nets, with the most critical nets being routed in advance of other non-critical nets. When the edge cost is defined according to congestions, the Steiner minimum tree algorithms may be applied directly to even out congestion while simultaneously restraining the wire length [13]. Steiner minimum tree algorithms are also often employed in providing a set of candidate routes to multicommodity flow based methods [3,9] and other iterative techniques. These methods will be described in detail later in this paper.

3.3 0-1 integer linear programming

Global routing may be formulated as a special type of optimization problem, called a zero-one integer linear programming (0-1 ILP) problem [61]. For a set of candidate routing trees $\mathcal{T}_i = \{T_{i,1}, T_{i,2}, ...\}$ for net N_i, we use variable $x_{i,j}$ to indicate if tree $T_{i,j}$ is selected for net N_i. The global routing problem

can then be formulated as:

$$\begin{array}{rll} \text{Minimize} & \hat{\lambda} & \\ \text{Subject to:} & \sum_{T_{i,j} \in \mathcal{T}_i} x_{i,j} = 1, & \forall N_i \in \mathcal{N} \\ & \sum_{i,j:e \in T_{i,j}} x_{i,j} \leq \hat{\lambda} u(e), & \forall e \in E \\ & x_{i,j} = \{0,1\}, & \forall N_i \in \mathcal{N}, \forall T_{i,j} \in \mathcal{T}_i \end{array} \tag{1}$$

The first constraint, along with the restriction of the $x_{i,j}$'s to $\{0,1\}$, requires that one tree be chosen for each net. The second constraint and the objective together ensure that the maximum congestion is minimized.

One straightforward approach to this problem is to first solve the continuous linear programming relaxation, obtained by replacing the third constraint with $x_{i,j} \in [0,1]$, since practical solutions to linear programming problems can be found in polynomial time [46], but the 0-1 ILP problem is NP-complete. The fractional solution thus obtained may then be transformed to integer solutions through rounding techniques such as randomized rounding, as in [71]. An alternative approach in [83] applies an interior point method in conjunction with column generating techniques [39] to solve the 0-1 integer linear programming problem.

In practice, the global routing problem is seldom solved entirely using the 0-1 ILP formulation since the problem size can grow to be very large. More often, the ILP technique is embedded into a larger overall global routing strategy, such as solving a subproblem at one hierarchical level of a hierarchical routing procedure [8,33,41,62,81], where the complexity of the computing the optimal solution to a 0-1 ILP is manageable.

3.4 Network flow model

The objective of the global routing problem is to allocate a limited set of resources evenly to a given set of demands. Intuitively speaking, the nature of this problem is quite coherent with the problem of finding optimal flows in a network [52], and several research efforts have pursued this as solution technique.

A network is a connected graph consisting of a set of vertices and edges; for the global routing problem, this graph is a minor modification of the routing graph in Figures 2 and 3. In a basic network

flow model, there are two special vertices: one called the source and another called the target or sink. A certain amount of flow, called the demand, for one commodity must be shipped from the source to the target vertex. Each edge has a flow capacity, which represents the upper bound for the flow that is allowed to pass through the edge. One version of the problem requires the transport of as much flow as possible through the network without exceeding any edge capacity, and this is referred to as the max-flow problem. Another version assigns a cost per unit flow to each edge, and sets the problem objective to be the minimization of the total transportation cost through the network for a given flow from the source to the sink; this formulation is called the min-cost flow problem. The appealing feature of the network flow problem is that it can be solved in polynomial time to obtain an optimal integer solution when edge capacities are integers. A good description of network flow methods can be found in the book by Ahuja *et al.* [2]. While it is not possible to model the entire global routing problem as a single commodity network flow problem, this technique can be employed to solve some subproblems in global routing [14, 38, 65] and can lead to high quality solutions.

There is a special type of network flow model that can be applied directly to the global routing problem. This is the multicommodity flow problem, where many commodities must be shipped on a common network, and each commodity has its own sources and targets that may be different from that for other commodities. In the mapping to the global routing problem, each net can be treated as a commodity. An advantage of this approach is that the multicommodity flow problem can be formulated as a linear programming problem. Since linear program solvers are slow for the sizes of problems encountered in global routing, research on the multicommodity flow problem has mostly focused on heuristics and combinatorial approximation algorithms. These methods will be introduced in the following sections.

4 Sequential routing techniques

Perhaps the oldest and most straightforward strategy for routing multiple nets is to select a specific order and to then route the nets sequentially in that order. The major advantage of this approach is that the congestion information for previously routed nets can be taken into consideration while routing a given net. For example, in early algorithms that operated by decomposing multi-pin nets into two-pin nets, each net was routed using techniques such as the obstacle avoidance version of maze routing or

the line probe method. In these approaches, a cell boundary is said to be open to path searching until all of the tracks have been occupied by previously considered nets; after that point, the boundary was treated as an obstacle.

The drawback of this sequential approach is that the quality of the solution depends greatly on the order in which nets are processed, and that it is hard to find a good net ordering. Under any net ordering, it is often more difficult to route the nets that are considered later since they are subject to more blockages. Moreover, there is no feedback mechanism that permits these nets to feed information back to the nets routed earlier with directions on regions that should be left free for their routes. Early work by Abel [1] concluded that there is no single net ordering technique that consistently performs better than any other ordering method. Despite the controversial net ordering issue, there are several good research results on sequential routing that have been reported, mostly through the use of iterative loops that feed back congestion information from the later routed nets to the earlier routed nets.

4.1 Force-directed routing

In [31], the global routing problem for two-pin nets is solved by emulating a particle movement in a force field. This approach assumes that a net order is given and routes each net sequentially. For each net, a particle departs from from the source pin and moves toward the target pin under the field generated by the source pin, target pin, unrouted pins in other nets, and routed wires. The trajectory of this particle motion forms the routing path connecting its source and target pin.

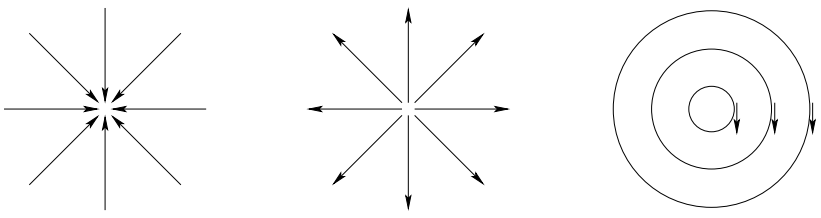

Figure 5: Various types of forces in force-directed routing.

There are three types of forces, as illustrated in Figure 5. When a particle moves to a certain position, an unrouted pin in another net will exert a repulsive force along the direction of the line joining the

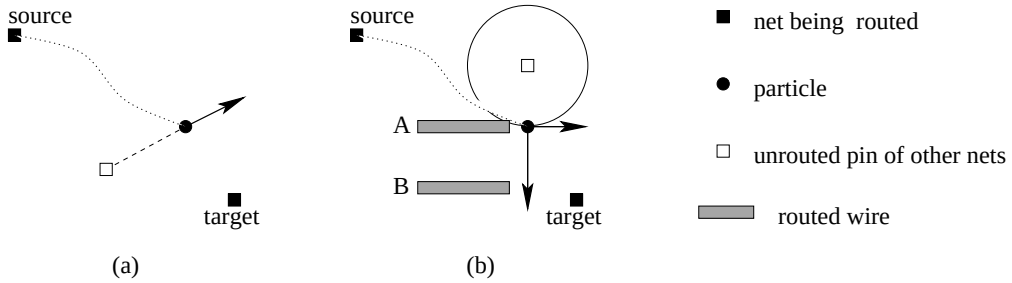

Figure 6: An example of the forces exerted on a particle: (a) a repulsive force from an unrouted pin in another net that does not lie in same row or column as the particle. (b) a circular force from an unrouted pin in another net in the same row or column as the particle, or from a routed wires in the same row or column.

particle and the unrouted pin, as shown in Figure 6(a). The magnitude of the force is $\frac{1}{pr^2}$, where p is the number of the total unrouted pins and r is the distance between the particle and the unrouted pin. The source pin generates a similar repulsive force with a magnitude of $\frac{1}{r^2}$. To a particle in motion, the only attractive force is from the target pin along the line joining the particle and the target, and the magnitude of this force is $\frac{1}{r^{1.5}}$, which is generally greater than any other kind of force, so that the particle can be guaranteed to reach the target. The direction of the combined force from the source and the target is snapped to the direction of motion of the particle direction if the two directions are not too different, so that the number of bends can be reduced.

If an unrouted pin or a segment of routed wire from a different net is in the same row or column as the article, it will exert a circular force of $\frac{1}{r^2}$. If this force is horizontal (vertical), it will take the same direction as the horizontal (vertical) component of the combined force from the source and the target, as illustrated in Figure 6(b). This force-directed method can be parallelized by letting the particle for every net move simultaneously.

4.2 Sequential routing through Steiner min-max tree construction

A Steiner min-max tree (SMMT) is a Steiner tree whose maximum-weight edge is minimized over all Steiner trees. In [12], Chiang *et al.* solve the global routing problem by constructing Steiner min-max trees for each net sequentially, defining each edge weight according to estimates of the routing congestion.

The algorithm sorts the nets according to the perimeter of their bounding box and routes one net at a time, with the nets that have a smaller bounding box being routed earlier. The routing tree construction is carried out on a routing graph of the type described in Section 2, with the edge weights on the graph being proportional to the wire congestion. For a certain net, a cell is called terminal cell if it contains a pin of this net; otherwise, it is called non-terminal cell. An optimal Steiner min-max tree that spans the net on the routing graph is obtained through a two-step procedure in polynomial time. In the first step, a minimum spanning tree over all of the cells in the routing graph is constructed. In the second step, any degree-one non-terminal cells are eliminated from this MST. It can be proven that the resulting tree is an optimal Steiner min-max tree. The edge weights are updated dynamically after the routing of each net, reflecting changes in the congestion.

Intuitively speaking, a Steiner min-max tree attempts to minimize the maximum edge weight, i.e., to minimize the wire congestion. However, the procedure does not consider wire length in constructing the SMMT. Finding a Steiner min-max tree with minimum wire length (MSMMT) is shown to be an NP-complete problem, and the authors of [12] propose a heuristic algorithm, based on the SMMT algorithm, that tries to find the MSMMT. In this heuristic, a wire length limit ratio $\rho > 1$ is defined, and the routing is performed over several iterations. Initially, the limit ratio is set to be tight, in a range that lies in the interval $(1, 2)$. In each iteration, the nets are routed one at a time using the SMMT algorithm, in the ascending order of the semiperimeter of the bounding box for the net. After this SMMT solution is obtained, its wire length is evaluated to check whether it is less than ρ times a lower bound on the wire length, given by the semiperimeter of the bounding box for the net. If not, this routing solution is rejected, and this net will be considered for routing in the next iteration. The value of ρ is increased from each iteration to the next so that every net can eventually be routed.

At the end of this iterative SMMT phase, this method iteratively reroutes each net using the minimum spanning tree algorithm in the same constant order. A rerouting result is accepted only when it is better than the solution of SMMT phase.

4.3 Minimum weighted Steiner tree

In the SMMT based global routing procedure, the primary objective is to minimize the maximal congestion and the secondary objective is to minize wire length. In [13], another sequential global routing

method is developed in an effort to simultaneously minimize congestion and wire length. This method routes the nets sequentially using a minimum weighted rectilinear Steiner tree (WRST) approximation algorithm.

The example shown in the work of [13] is based on a custom design such as that shown in Figure 7(a), where the routing area is divided into a set of small regions. Each region is assigned a weight that is defined according to the complexity and wire congestion of the region. The shaded rectangles correspond to macros that act as wire blockages, and the weights in these regions are set to ∞.

This work constructs a routing graph for a net as the union of the Hanan grid for the net and the grid formed by extending the borders of each region until they encounter a blockage or the boundary. For example, the routing graph for the design shown in Figure 7(a) is illustrated using dashed lines in Figure 7(b). For a wire that is routed along a border between two regions with different weights, it is assumed that the weight with a smaller value is chosen. The weight of an edge is the product of the edge length and the weight of the region the edge belongs to, and the objective of the procedure is to minimize the cost of the weighted rectilinear Steiner tree. It can be shown that there is always a minimum weight path connecting two nodes using edges that lie exclusively on this routing graph.

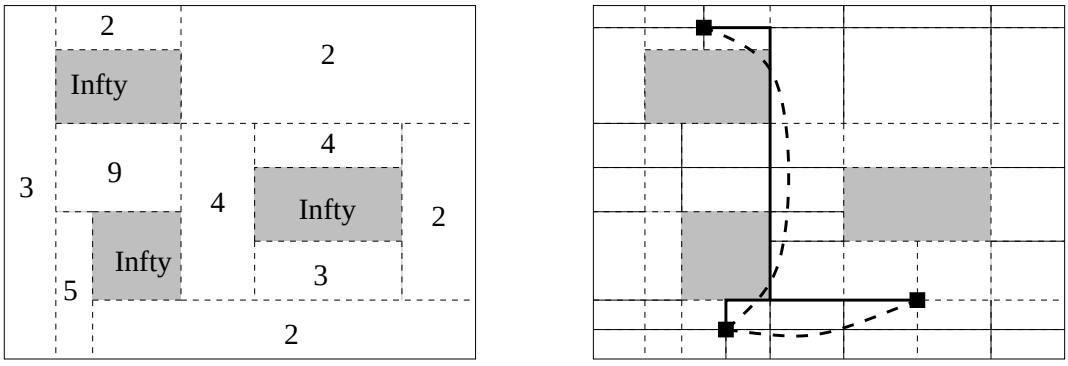

Figure 7: (a) A set of regions in a layout, each with a specified weight. (b) A routing graph and a minimum weighted Steiner tree for this layout.

As in the SMMT-based approach, the nets are first sorted in increasing order of the bounding box size and are routed sequentially in this order. The weights for all regions are updated after each net is routed.

Each net is routed by means of a minimum weighted Steiner tree over the net. Based on a proof that the weight of a minimum spanning tree (MST) is at most two times the weight of optimal WRST solution, the tree construction proceeds by beginning with an MST skeleton and heuristically attempting to maximize the overlaps in its embedding on to the routing grid. For the net in Figure 7(b) whose pins are indicated by the three solid squares, the MST is shown by a set of thickened dashed lines. After the MST is obtained, each edge in the MST is instantiated into an actual wire route one by one, with the shortest edges being embedded first. Wherever possible, a wire route is merged with already embedded wire routes from the same net to reduce the wire length. If there is more than one minimum weight path for an edge in the SMT, the one that can deliver the largest wire length reduction is chosen. Since this process can only decrease the wire length from the MST result, the final result has at most twice the wire length of the optimal WRST's in the worst case.

5 Region-wise routing

One way to avoid the net ordering problem is to route one region at a time instead of one net at a time. This alters the problem to one of determining the order in which regions should be considered, and this may be decided according to net distributions, as in the method described in Section 5.1. In an alternative approach outlined in Section 5.2, inter-row wires are routed first, and the intra-row nets are then routed in the order of horizontal wires, vertical wires and wires with both orientations.

5.1 Unique pattern first and outer rim first routing

In an investigation of the properties of net distributions, Li and Marek-Sadowska [57] made two observations. Firstly, some nets or subnets may have to take certain unique patterns, regardless of what the final feasible routing is, and it is useful to identify and route these first. Secondly, in typical layouts the wiring is crowded at the center of the chip and relatively sparse at the outer rim. Therefore, it is better to start routing from the outer rim of the entire routing area and then shift the routing toward the center step by step. Although this work was specifically directed towards gate array designs, the key ideas can be extended to other problems.

The following definitions are useful in describing the algorithm and are related to an abstraction of the problem in terms of a routing graph, G.

Definition 5.1 (Non-Pass-Through (NPT) cell) *A cell is said to be a non-pass-through (NPT) cell if the number of unconnected pins of different nets inside it is equal to or less by one than the total remaining channel capacities on its four boundaries.*

Definition 5.2 (Outermost and inner meshes) *In a particular planar drawing of the cell graph G, where the boundary of the exterior face of a planar embedding of G corresponds to cells at the boundary of the chip, the outermost mesh is a subset of nodes and edges in the routing graph G that bounds the exterior face from G, and an inner mesh is a subset of nodes and edges in the routing graph G that bounds an area where edges are missing.*

An inner mesh exists when there is a grid edge whose wiring capacity is zero. An example illustrating the idea of the outermost mesh and the inner mesh is shown in Figure 8.

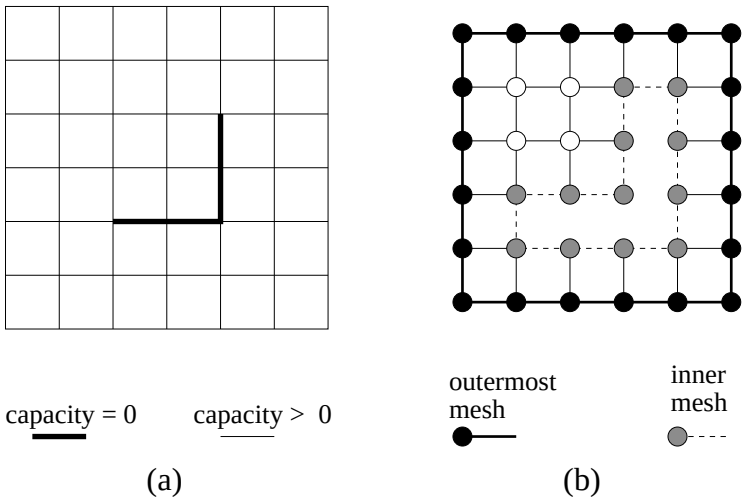

Figure 8: An example of outermost mesh and inner mesh.

The algorithm is based on the following properties when the global routing problem has a feasible solution with no wiring overflow.

Lemma 5.1 *If there is a two-pin net, N_k, with pins in two adjacent cells with non-zero wiring capacity at their common boundary, then there exists a feasible routing solution in which N_k is wired through the common boundary of the two adjacent cells.*

Lemma 5.2 *If the total wiring capacity on a closed region boundary is U, and there are P pins inside such that $P > U$, then at least $P - U$ of them must be connected within this region.*

Lemma 5.3 *If there are two adjacent NPT cells, g_a and g_b, with a non-zero wiring capacity u_{ab} on the common boundary, and there are u_{ab} or fewer nets with pins in both g_a and g_b, then there is a feasible solution in which these pins are connected by the shortest path between g_a and g_b.*

This algorithm proceeds by first finding pairs of pins in adjacent cells and connects them directly according to Lemma 5.1. Next, it identifies the NPT cells from this partial routing result and determines unique routes according to following rules:

1. Nets with pins in the neighborhood of NPT cells must find unique routes that do not pass through the neighboring NPT cells.

2. By Lemma 5.3, nets that have two pins in adjacent NPT cells are connected through the common boundary of the adjacent NPT cells.

3. If there is a net with one pin located in an NPT cell and the other in its non-NPT neighbor, they are connected directly across their common boundary.

After these unique patterns have been routed, some new NPT cells may be generated due to the consequent reduction in the wiring capacity. These cells are identified and the process is repeated until no such new cells are created.

In the next phase, the routing proceeds along the outermost mesh and the inner meshes. Any nets with more than two pins within a mesh are connected within the mesh, as long as such a connection does not lead to barriers separating the routing graph into disjoint regions. Those pins that remain unconnected are connected to neighboring non-mesh cells. After one layer of outermost mesh is routed, the outermost mesh is expanded inwards. Similarly, a routed inner mesh is expanded outward. If a routing decision at an outer mesh is found to be incorrect when the inner meshes are routed, the algorithm resorts to backtracking to correct the previously made decision. This procedure is repeated until all of the cells are routed. For a routing instance with k nets to be routed on a $p \times q$ grid, this algorithm has a complexity of $O(k^2(\log k + pq))$.

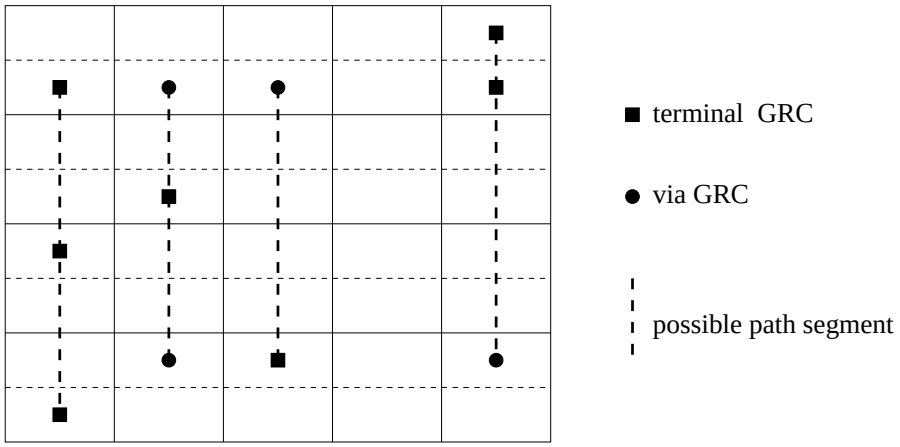

Figure 9: Candidate paths for inter-row connections in the algorithm proposed by Thaik *et al.*

5.2 Routing in order of wire orientations and in terms of rows

In [81], Thaik *et al.* designed a global routing algorithm for a sea-of-gates environment that proceeds edge-wise instead of net-wise. They consider the global routing problem for three layers such that top and bottom layers are reserved for vertical wires and middle layer is devoted to horizontal wires. The routing region is tessellated into a tile graph as shown in Figure 9, and each row is further divided into two subrows separated by the thin dashed lines.

The algorithm consists of three phases. In the first, each net that spans multiple rows is connected by a vertical path in the top layer that covers the vertical span of the entire net. The end point of the path may either be a terminal GRC (global routing cell) that contains a pin of the net, or a via GRC. For a specific net, there could be several paths that cover its vertical span, as illustrated in Figure 9, and only one such path needs to be chosen. This choice should be made while ensuring that there are no wiring capacity overflows. This decision process is implemented in [81] by means of a zero-one integer linear programming (ILP) formulation that is solved using a 0-1 ILP package.

In the second phase, connections within each row are considered. Recall that each row consists of two subrows. If two pins lie in two adjacent subrows of the same row and are in the same column, they are connected by a straight vertical edge in the lowest routing layer as long as no overflow occurs. Next, straight horizontal connections are made on the middle layer between pairs of terminal-GRC's, via-GRC's and/or metal-3 paths. These problems are also formulated and solved as 0-1 ILP problems.

After phase two, there may still be nets whose routing has not been completed due to wiring constraints. These remaining connections are completed in the third phase, which is also constrained to routing within a row. In this step, multi-bend routes are considered, and the route for each row is solved as a $2 \times N$ routing problem using a hybrid hierarchical approach, where the routing at each hierarchical level is solved as a 0-1 ILP problem.

6 Move-based heuristics

Move-based heuristics are commonly used to find the optimal solutions to computationally difficult problems. Arguably, the most well known move-based heuristic is simulated annealing, developed by Kirkpatrick *et al.* [47]. The motivation for this approach is that in solving computationally difficult combinatorial optimization problems, greedy heuristics are easily trapped at local minima because only those moves that reduce the value of the cost function are accepted. In simulated annealing, a certain amount of hill-climbing is permitted, providing an increased chance of finding the global minimum. More precisely, any cost-decreasing move is always accepted, and a cost increasing move is accepted with a probability of $e^{-\Delta C/T}$, where ΔC is the increase in the cost and T is a parameter known as the temperature. It can be seen that the less the increase on the cost, the more likely the move is accepted. The procedure works iteratively, with an outer loop starting from a high temperature where virtually all moves are accepted, and gradually "cooling" the temperature so that the likelihood of accepting cost-increasing moves is progressively diminished. At each temperature, a number of moves are carried out, which are either accepted or rejected according to the criterion outlined above. This process is said to be analogous to the cooling process during annealing of metals.

The simulated annealing technique is applied to global routing in [84], where only two-pin nets are considered and the number of bends for each net is no more than two. In the well-known TimberWolf package [76], simulated annealing is applied to both placement and global routing. In this package, a set of candidate routing trees is created for each net and one of the trees is randomly chosen as the initial solution. Each move implies a switch from one tree to another for a net. The cost to be minimized is the total wiring overflow over the entire routing graph. The net whose topology is to be changed by a move is randomly chosen from a grid cell boundary where an overflow exists.

Other move-based techniques applied to global routing include simulated evolution [10], genetic al-

gorithm [28] and tabu search [91].

7 Rip-up and reroute

Another common approach to avoid the net ordering dilemma is the rip-up-and-reroute method. This approach starts by routing each net individually without considering congestion, usually constructing Steiner minimum trees for each net. After all of the nets have been routed, the congested areas are identified, and the nets in those areas are ripped up and rerouted through less congested areas. This rerouting is often based on the maze routing algorithm. Although this method sounds very simple, it is surprisingly effective and has long been the most commonly used global routing method in industry. Moreover, it can always be combined with other global routing methods as a post-processing step to further improve the routing quality. The degrees of freedom in this approach are related to, for example, the different strategies for choosing the net to be rerouted, or the order in which boundaries with overflow are processed.

An early and influential rip-up-and-reroute global routing method was proposed by Ting and Tien [82]. Broadly speaking, their procedure first selects a set of congested grid cell boundaries and then chooses the subset of nets that pass across these boundaries to be rerouted.

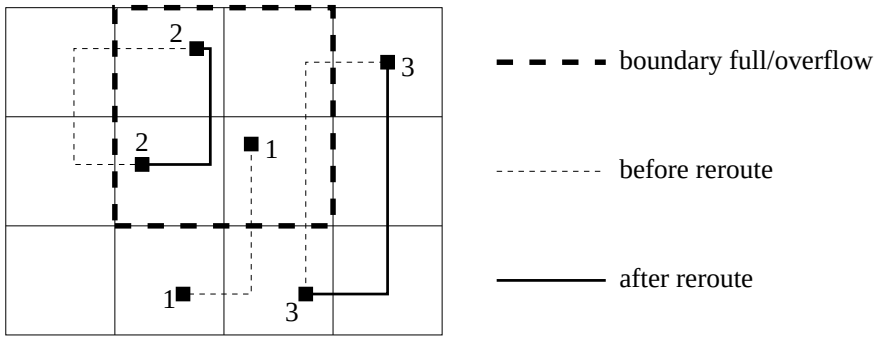

Figure 10: An example showing the overflow along a full loop. This overflow can be reduced only when there is at least one net that crosses the loop twice.

A special case corresponds to the situation where the set of saturated or oversaturated boundaries forms a closed loop, as illustrated by the thickened dashed square in Figure 10. Along this loop, each boundary is either full or has an overflow, and at least one boundary has an overflow. In this case, if a

wire crosses this loop only once, as is the case for net 1, rerouting this net will not satisfy the congestion constraints. More generally, if all of the nets pass through the loop only once, then no feasible solution exists. On the other hand, if a net crosses this loop twice and all of its pins lie either entirely within or entirely outside the loop, as is the case for nets 2 and 3, respectively, then we can reroute these nets to ensure that the devious routes taken by these double-crossing nets are altered to lie entirely within or without the loop. The first step of Ting and Tien's algorithm is to identify such loops and to reroute the two-crossing nets.

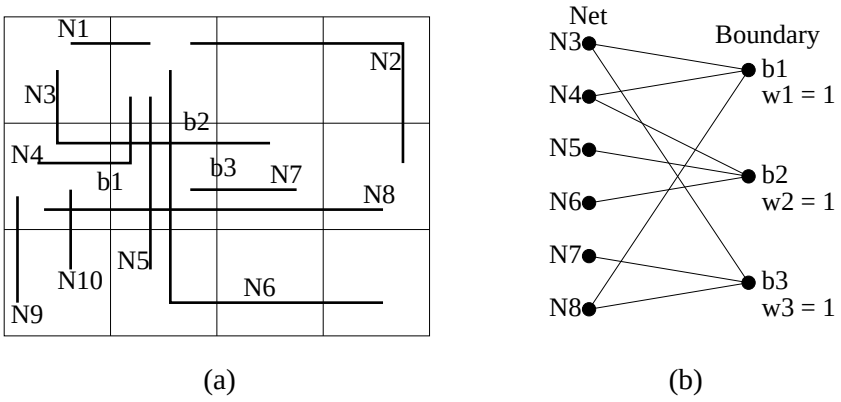

Figure 11: Mapping the problem of choosing nets to be rerouted on to the problem of finding a cover on a bipartite graph.

When no such loop exists, Ting and Tien's algorithm selects a set of $k > 1$ most congested boundaries to reroute. There are typically many nets that run across these boundaries, and the technique employed to choose the nets to be rerouted is illustrated through the example in Figure 11(a). In this example, the wiring capacity on each boundary is assumed to be two, and it is easily verified that there are three boundaries, b_1, b_2 and b_3, with overflow. A bipartite graph is then constructed as shown in Figure 11(b). The group of vertices to the right correspond to the overflow boundaries, and those to the left correspond to the nets that cross those boundaries. An arc is set up between a net vertex and a boundary vertex if the net crosses this boundary. Each boundary vertex b_i has a weight w_i equal to the overflow of the corresponding boundary. A minimum cardinality cover set from the group of vertices to the left is selected to cover every boundary vertex w_i times, and the nets in this cover set are selected to be rerouted.

Another variant of rip-up-and-reroute method is due to Nair [67]. In this procedure, *every* net is ripped up and rerouted based on the observation that a wire in a non-overflow area may be able to move further and to a less congested area to leave some room for wires in adjacent congested areas. Another feature is that every net is rerouted in the same constant order iteratively. The rationale for this is that the routes that are chosen early in an iteration are based on less accurate congestion information compared to the routes that are determined later, and therefore, these early routed nets should be corrected first in the next iteration as well.

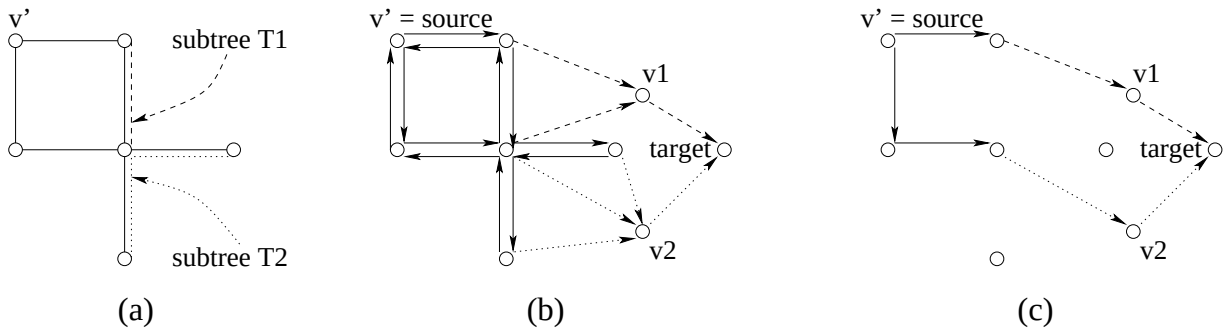

Figure 12: An example of rerouting using a network flow model.

In [65], Meixner and Lauther reroute a set of nets simultaneously using a single commodity min-cost network flow formulation, so that the net ordering problem in rerouting is avoided. The method successively considers each node $v' \in V$ for a routing graph $G = (V, E)$ and reroutes the nets that have either a sink node or a Steiner node with degree greater than two at v'. For each such net, its routing edges from v' to the next sink or Steiner node with degree greater than two are ripped up, leaving behind a set of partial routing trees. For example, two nets N_1 and N_2 may leave behind the trees T_1 and T_2, respectively, shown in Figure 12(a).

The problem is now to connect nodes at v' to subtrees T_1 and T_2 for nets N_1 and N_2 without causing a wiring overflow, and simultaneously minimizing the wire length. To solve this problem, the partial routing graph is transformed into a network flow model as shown in Figure 12(b). Each undirected edge is mapped on to a pair of directed edges whose capacity is the corresponding wiring capacity and whose cost is the edge length. Two pseudo nodes v_1 and v_2 are added for nets N_1 and N_2, respectively,

together with a target node. Every node in partial tree T_i is connected to the node $v_i, i = 1, 2$, by means of a directed edge, and finally, each v_i is connected to the target node. These edges all have a cost of zero and capacity of one. The node v' serves as the source node, and a minimum cost flow for this network is determined for a total flow of p units through the network, where p is the number of partial routing trees. Such a flow can be used to determine an optimal solution, if it exists, that satisfies wiring capacities and minimizes the wire length. A sample solution for our example is illustrated in Figure 12(c).

In addition to avoiding net ordering problem, this method has the advantage that an optimal integer solution to this single commodity flow problem can be obtained in polynomial time and no rounding procedure is required, since the single commodity network flow problem has a polynomial time optimal integer solution if each edge capacity is an integer [2]. However, the quality of the solution depends on the order in which the nodes in G are processed, though this node ordering problem is less severe than the original net ordering problem.

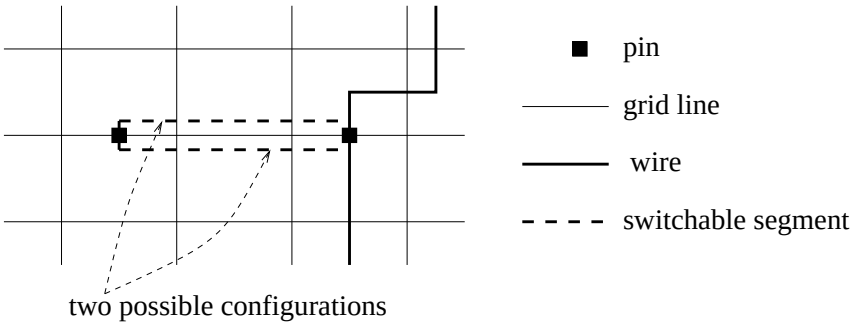

Figure 13: An example of a switchable segment.

In [55], Lee and Sechen proposed a rip-up-and-reroute strategy using a multi-stage refinement procedure, instead of applying maze routing immediately after initial tree construction. As in other rip-up-and-reroute methods, they initially construct Steiner trees for each net without considering congestion. Next, wires are ripped up and rerouted to reduce congestion in four successive stages. One scenario is related to the idea of *switchable segments*, illustrated in Figure 13. In a gate array or a standard cell design, an equivalent pair of pins at the top and bottom of a cell is treated as a single pin located at the middle of the row where a horizontal grid line goes through. In stage one of the rerouting process,

one of these two options for each switchable segment is chosen in a way such that the congestion is minimized. In stage two, the L-shaped wires in the initial Steiner trees are allowed to be rerouted as Z-shaped connections to reduce congestion; note that there is no wire length increase in either stage one or stage two. In stage three, connections that were originally straight in the initial Steiner trees are allowed to detour and become U-shaped connections, with a bound on the permissible increase in the wire length. During this stage-wise refinement, the algorithm attempts to control the wire length and the number of bends. If some overflow still remains after these three stages of rerouting, traditional maze routing is finally employed to further reduce the congestion in stage four.

In [60], a customized routing graph is constructed for global routing using multilayer macrocells, or building blocks. This routing graph is similar to the graph in Figure 7(b), except that it is in three dimensions to reflect the multilayer design. The authors extend Wang's heuristic [87] to construct Steiner trees for each net separately at the beginning. After the initial tree construction, their method starts by inspecting every grid cell to see if the number of wires (vias) that pass through it exceeds the available routing tracks (via spaces). Next, every over-congested cell is processed in order of the most congested cell first. For each such over-congested cell, a new route is found for every net in it that can avoid all the over-congested or full cells, if such a route exists. These routes are maintained in a priority queue with the top route having the minimum wire length increase compared to its corresponding old route. After all of the possible new routes have been found, the top new route is popped out from the queue and used to replace its corresponding old route repeatedly until the over-congestion problem for this cell is resolved.

8 Multicommodity flow based approach

Although the sequential, rip-up-and-reroute and other heuristics may be effective in practice, they cannot provide a certain answer as to whether or not a feasible solution exists. In other words, if they fail to find a feasible solution, it is not clear whether this is attributable of the non-existence of a feasible solution or because of shortcomings of the heuristic. Moreover, when a heuristic does find a feasible solution, it is not known whether or not this solution is optimal, or how far it is from the optimal solution.

These questions may be answered if we formulate and exactly solve the global routing as a multicom-

modity flow problem. A multicommodity flow operates on a network that is a graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is a set of n vertices and $E = \{e_1, e_2, ..., e_m\}$ is a set of m edges. Some researchers define this network as a directed graph while some others define it as a undirected graph. Generally speaking, this difference affects only the form of the problem formulation. For multi-pin net global routing, an undirected graph formulation is more convenient.

Over this network, k commodities must be transported from some vertices to other vertices. In global routing, we can treat each net as a commodity and we can say that there is a set of $\mathcal{N} = \{N_1, N_2, ..., N_k\}$ commodities that are to be shipped. For each commodity N_i, a certain amount d_i, namely the *demand*, is required to be shipped. In global routing, each N_i corresponds to a net with more than one pin and each d_i is always one. Each edge has a flow capacity $u(e)$ and cost $c(e)$, which have the same interpretation as in the global routing problem formulation. The routing can be expressed in terms of either edges or trees. In the edge-based expression, a variable $f_i(e)$ represents the amount of flow passing through edge $e \in E$. Note that $f_i(e)$ is always non-negative in an undirected network while it can be any real number for a directed network. The tree-based expression assumes that there is a set of possible routing trees $\mathcal{T}_i = \{T_{i,1}, T_{i,2}, ...\}$ for each net N_i. The binary variable $x_{i,j}$ is set to one if $T_{i,j} \in \mathcal{T}_i$ is selected for net N_i; otherwise, $x_{i,j}$ is zero. Of course, it is impractical to enumerate all of the possible trees and these trees are usually generated on the fly in practice.

Typically, there are two constraints that must be satisfied in a multicommodity flow problem. The first is the *demand constraint*, which requires that the amount of flow shipped for each commodity should be equal to its demand, and the second is the *bundle constraint*, which states that the total amount of flow $f(e)$ passing through each edge $e \in E$ should not exceed its capacity $u(e)$. In the fractional flow version, the decision variables $f_i(e)$ or $x_{i,j}$ may be any non-negative real number. In the zero-one integer flow version, there is the third *zero-one integer constraint* that regulates that $f_i(e)$ and $x_{i,j}$ must take a value of either zero or one. The global routing problem maps on to the zero-one integer version. Due to computational complexity issues related to integer programming and the large size of the global routing problem, the integer flow problem here is often relaxed to the fractional formulation, whose solution is transformed to an integer solution after a rounding procedure.

Besides these constraints, there are several variations on the objective function for the multicommodity flow problem. One formulation tries to minimize the total cost of transportation, and this is called

the min-cost multicommodity flow problem. Another formulation attempts to minimize the maximum edge density (as defined in Section 2) and is called the concurrent flow problem.

8.1 The Shragowitz-Keel algorithm

The work by Shragowitz and Keel in [79] is perhaps the first reported work on global routing using the multicommodity flow model. Unlike many subsequent efforts using this formulation, they did not employ an off-the-shelf multicommodity flow algorithm, but instead, developed their own polynomial time algorithm. Superficially, this algorithm looks similar to the rip-up-and-reroute method and the authors investigated the feasibility of convergence and the convergence rate. However, no statements about the optimality are made with the analysis.

Their approach is based on a *directed* network and is restricted to two-pin nets. For each net N_i, one of its pins $v_s \in V$ is selected as source where a unit of flow is generated, i.e., the net flow of net N_i at this vertex $d_i(v_s)$ is 1. The other pin v_t is the sink where a unit of flow dissipated, i.e., the net flow of net N_i is $d_i(v_t) = -1$. For simplicity, in this description, a directed edge from vertex v to v' is denoted by the vertex pair (v, v'). The Shragowitz-Keel formulation of the global routing problem is as follows:

$$\begin{array}{rll} \text{minimize} & \sum_{\forall e \in E} \sum_{\forall N_i \in \mathcal{N}} c(e)|f_i(e)| & \\ \text{subject to:} & \sum_{v':(v,v') \in E} f_i(v,v') - \sum_{v':(v',v) \in E} f_i(v',v) = d_i(v), & \forall N_i \in \mathcal{N}, \forall v \in V \\ & \sum_{\forall N_i \in \mathcal{N}} |f_i(e)| \leq u(e), & \forall e \in E \\ & f_i(e) \in \{0, \pm 1\}, & \forall N_i \in \mathcal{N}, \forall e \in E \end{array} \tag{2}$$

Initially, this algorithm discards the bundle constraint to obtain a min-cost solution where the edge cost $c(e)$ is defined to be the rectilinear edge length. Next, it iteratively reduces the violations on the bundle constraint while maintaining a minimum feasible cost. The first step is solved individually for each net and any shortest path algorithm may be used. As in other methods, this step corresponds to finding the minimum cost route for each net while ignoring the effects of the other nets. The algorithm then identifies the subset of edges E_ϕ with maximum overflow $\phi = \max(\phi(e), \forall \in E)$ and the subset of edges $E_{\phi,\phi-1}$ with an overflow of either ϕ or $\phi - 1$. If $\phi = 0$, then the algorithm is done. The cost of each edge in $E_{\phi,\phi-1}$ is updated to infinity and all other edges keep their cost unchanged. The subset $\mathcal{N}_\phi$ is defined such that each commodity or net in it has at least one $f_i(e) = 1$ such that $e \in E_\phi$. The

subset of $\mathcal{N}_{\phi,\phi-1}$ is defined in a similar manner. Based on the updated cost, the shortest path algorithm is executed for every commodity in $\mathcal{N}_\phi$. If there is a new path with a cost of infinity, this algorithm checks a feasibility condition which can verify whether there is a feasible solution for this problem. If the problem remains infeasible, this algorithm perturbs the flows and reruns the iterations again. If there is no new path with a cost of infinity, this algorithm chooses the commodity N_i that has the smallest increase in cost from the old path to the new path, and replaces its old path with the new path. This process is repeated until there is no overflow.

8.2 The Raghavan-Thompson rounding method

One interesting formulation for multi-terminal nets routing to multicommodity flow model is conducted by Raghavan and Thompson in [72], where the exposition of the algorithm is restricted to nets with exactly three pins, although the authors state that generalizations to multiple pins are possible. This formulation is also based on a routing graph $G = (V, E)$, and the objective is to minimize the maximum flow among all edges, i.e., minimize $\hat{f} = \max(f(e), \forall e \in E)$.

Consider a situation where each net $N_i \in \mathcal{N}$ consists of three terminals v_{i1}, v_{i2} and v_{i3}. In the formulation, Raghavan and Thompson assume that one Steiner node is needed for each net and an indicator variable $s_i(v) \in \{0, 1\}$ is used to denote whether or not each vertex $v \in V$ is the Steiner node for net N_i. The global routing problem is then formulated as a multicommodity flow problem in which $s_i(v)$ units of flow are to be shipped from each of v_{i1}, v_{i2} and v_{i3} to vertex $v \in V$.

This problem is first relaxed to be a linear programming problem and solved by any linear program solver. The next crucial task is to obtain the integer solution from the optimal linear program relaxation solution. We use $\tilde{f}(e)$ and $\tilde{s}_i(v)$ to represent the fractional results from the linear programming and denote the optimal max-flow as $\tilde{f}$. Then these values must satisfy the following conditions.

$$\sum_{v \in V} \tilde{s}_i(v) = 1, \quad \forall N_i \in \mathcal{N} \tag{3}$$

$$\tilde{f}(e) \leq \tilde{f}, \quad \forall e \in E \tag{4}$$

Here all of the $\tilde{f}(e), \tilde{s}_i(v) \in [0, 1]$. We express a set of solutions as S and start with the initial solution

S_0 as:

$$S_0 = \{\tilde{f}, \tilde{f}(e), \tilde{s}_i(v), \forall N_i \in \mathcal{N}, \forall v \in V, \forall e \in E\}.$$

The rounding procedure proceeds for k stages to get a sequence of solutions $S_1, S_2, ..., S_k$, where $k = |\mathcal{N}|$. In each stage i, the flow of one net N_i is rounded to integers and will not be changed later, and the solution proceeds from S_{i-1} to S_i. For each solution S, a potential function is defined as:

$$\Psi(S) = \sum_{e \in E} \prod_{N_i \in \mathcal{N}} [f_i(e)\omega + 1 - f_i(e)], \tag{5}$$

where the parameter $\omega > 1$ will be defined later.

Each stage consists of two phases. The Steiner node for the net being processed is determined in phase one and the integer flow to the Steiner node is obtained in phase two. In phase one of stage i, a vertex v_p is identified so that $s_i(v_p)$ is forced to 1 and $s_i(v) = 0, v \neq v_p, v \in V$. If we denote the flow from any vertex $v_{ij} \in N_i$ to vertex $v \in V$ as $f_i(e, v)$, selecting v_p as Steiner node for N_i also implies that we let $f_i(e, v_p) = \tilde{f}_i(e, v_p)/\tilde{s}_i(v_p)$ and $f_i(e, v) = 0$ for all other vertices v. This corresponds to picking paths from each pin of the net to the Steiner point. If the solution after Steiner node selection is S'_i, then the Steiner must be chosen in a way so that $\Psi(S'_i)$ is minimized. The authors have proved that this procedure ensures that $\Psi(S'_i) \leq \Psi(S_{i-1})$.

After the Steiner node is selected, the fractional flows from net N_i to vertex v_p, $f'_i(e, v_p)$, are rounded to be either 0 or 1 in phase two. Let us consider how to round the flow from one of the pins v_{i1} to v_p. Through a so-called *path stripping* procedure, the fractional flow from v_{i1} to v_p is organized into a set of paths $\{P_1, P_2, ...\}$, each with a certain amount of flow. The summation of flow over all of these paths must be one. The flow along path P_l is now selected to be one such that the solution from this choice $S'_i(P_l)$ minimizes the potential $\Psi(S'_i(P_l))$. After the flows from all three pins of net N_i have been rounded, we can obtain the solution S_i, and it is proven that $\Psi(S_i) \leq \Psi(S_{i-1})$.

If the parameter ω is chosen so that it satisfies the relation

$$[\frac{e^{\omega-1}}{\omega^{\omega}}]^{\tilde{f}} = \frac{1}{m},$$

then the upper bound of the final integer max-flow $\hat{f}$ is as follows:

$$\hat{f} \leq \begin{cases} \tilde{f} + (e-1)\sqrt{\tilde{f} \ln m} & : \quad \tilde{f} \geq \ln m \\ \frac{e \ln m}{\ln \frac{e \ln m}{\tilde{f}}} & : \quad \tilde{f} < \ln m \end{cases} \tag{6}$$

where m is the number of edges in G and e is the base of the natural logarithm.

8.3 Application of the Shahrokhi-Matula algorithm

In [9], Carden *et al.* developed the first reported global router with a theoretical bound from the optimal solution, based on a multicommodity flow algorithm. They applied Shahrokhi and Matula's two-terminal multicommodity fractional flow algorithm [77] followed by randomized rounding to obtain a multi-terminal multicommodity integer flow solution. Shahrokhi and Matula's algorithm is an ϵ-optimal approximation algorithm. In contrast with Shragowitz and Keel's approach, their method is directed towards a concurrent multicommodity flow formulation instead of a min-cost multicommodity flow formulation.

The integer linear programming formulation used here is the same as formulation (1). Its linear programming relaxation is obtained by omitting the integer constraint in (1) and we denote this relaxation as LP. The dual linear programming (DLP) of the LP is:

$$\begin{array}{rll} \text{maximize} & \sum_{N_i \in \mathcal{N}} \theta_i & \\ \text{subject to:} & \sum_{e \in E} c(e) l(e) = 1 & \\ & \sum_{e \in T_{i,j}} l(e) \geq \theta_i, & \forall N_i \in \mathcal{N}, \forall T_{i,j} \in \mathcal{T}_i \\ & l(e) \geq 0, & \forall e \in E \end{array} \tag{7}$$

The variable $l(e)$ is the dual variable corresponding to each edge e and is also referred to here as the edge weight. The variable θ_i represents the throughput of the flow from net N_i. According to the theory of duality in linear programming, a feasible solution to DLP provides a lower bound for the optimal solution of LP, and the LP solution reaches its optimum when it equals the DLP solution. Based on this property, Shahrokhi and Matula design an approximation algorithm in which the LP and DLP solutions are pushed closer after each iteration, and the final difference between the LP and DLP

solutions provides an upper bound on how far the LP solution away from the optimal solution. An ϵ-optimal algorithm implies that the resulting $\hat{\lambda}$ is at most $(1+\epsilon)$ times the optimal solution $\hat{\lambda}^*$, where the value of ϵ is a specified parameter between $(0, 1]$. The smaller the value of ϵ is, the more optimal the solution is and the longer is the computation time.

Initially, the edge weight is defined as $l(e) = 1/u(e) \forall e \in E$. The algorithm begins by constructing a minimum weight Steiner tree T_i for each net N_i to obtain a solution satisfying the first constraint in the formulation (1) without considering the bundle constraint. As a consequence, the value of $\hat{\lambda}$ may initially be inordinately large. In subsequent steps, the algorithm iteratively recomputes the Steiner trees to decrease the value of $\hat{\lambda}$ until the optimality condition is satisfied. In order to avoid iterations on trees with very small flows, a parameter σ_0 is defined as the smallest allowable fraction of flow changes. If $U = \sum_{e \in E} u(e)$ and $u_* = \min(u(e), \forall e \in E)$, then σ_0 is chosen to be $\frac{u_*^2 \epsilon}{16 \alpha U}$, where the constant α, which is a user-tunable parameter, is typically between 0.01 and 100.

Next, the weights of the edges that whose flows were altered in the last iteration are updated according to $l(e) = e^{\alpha \lambda(e)}$. Using this updated weight, new minimum weight Steiner trees are recomputed for each net. If $\theta = \frac{\sum_{e \in E} l(e) u(e)}{\sum_{\forall N_i \in \mathcal{N}} \sum_{\forall e \in T_i} l(e)}$, then ϵ-optimality is satisfied if $\theta - \frac{1}{\lambda \theta} \leq \epsilon$, If ϵ-optimality is not satisfied, then the net with the largest cost reduction from the old Steiner tree to the new Steiner tree is rerouted so that a portion of flow in the old Steiner tree is switched to the new Steiner tree, and this cost update and rerouting process is repeated until the ϵ-optimality condition is satisfied. At this point, the actual flow for each tree is scaled to $\bar{x}_{i,j} = x_{i,j}/\hat{\lambda}$. Finally, the fractional solution is integerized in to integer solution through randomized rounding, and any remaining overflow is resolved through a rip-up-and-reroute process.

8.4 Application of Garg-Könemann algorithm

The latest reported global routing procedure based on multicommodity flow algorithm is due to Albrecht [3]. This work is an application of Garg and Könemann's multicommodity flow approximation algorithm [29], which is simpler and faster than Shahrokhi and Matula's algorithm [77]. Besides congestion, wire length is also considered in this method.

Albrecht's algorithm works on a routing graph $G = (V, E)$ shown in Figure 2(b), which represents a two layer global routing problem in which one layer is reserved for horizontal wires and the other

for vertical wires. The edges between two layers represent vias and their capacities are the maximum number of vias allowed in a grid cell. The algorithm permits different wire widths, and uses the symbol $w_i(e)$ to represent the wire width of edge $e \in E$ belonging to net N_i. The resulting integer linear programming formulation is slightly different from that in (1). Assuming that all of the demands $d_i = 1, \forall N_i \in \mathcal{N}$, the formulation is:

$$\begin{array}{rll} \text{minimize} & \hat{\lambda} & \\ \text{subject to:} & \sum_{T_{i,j} \in \mathcal{T}_i} x_{i,j} = 1, & \forall N_i \in \mathcal{N} \\ & \sum_{i,j:e \in T_{i,j}} w_i(e) x_{i,j} \leq \hat{\lambda} u(e), & \forall e \in E \\ & x_{i,j} \in \{0,1\}, & \forall N_i \in \mathcal{N}, \forall T_{i,j} \in \mathcal{T}_i. \end{array} \tag{8}$$

```
1.  l(e) ← δ/u(e), ∀e ∈ E
2.  x_{i,j} ← 0, ∀T_{i,j} ∈ T_i, ∀N_i ∈ N
3.  j_i ← 0, i = 1, ..., k
4.  While (Σ_{e∈E} l(e)u(e) < 1)
5.    For i = 1 to k
6.      If j_i == 0 or Σ_{e∈Ti,j} w_i(e)l(e) > (1 + γε)θ_i
7.        Find a minimal Steiner tree T_{i,j_i} ∈ T_i
          for net N_i with edge weight w_i(e)l(e), e ∈ E
8.        θ_i ← Σ_{e∈T_{i,j_i}} w_i(e)l(e)
9.      x_{i,j_i} ← x_{i,j_i} + 1
10.     l(e) ← (1 + εw_i(e)/u(e))l(e), ∀e ∈ T_{i,j_i}
```

Figure 14: An approximation algorithm for fractional global routing.

As before, the ILP problem is relaxed to the a linear programming problem which is solved by the approximation algorithm in Figure 14. The symbols δ, γ and ϵ are constants, and k is the number of nets to be routed. In the Shahrokhi-Matula algorithm, a fraction of flow is switched from a higher weight (more congested) tree to a lower weight (less congested) tree in each iteration. In Garg-Könemann's algorithm, a flow of d_i is added to a low weight tree $T_{i,j} \in \mathcal{T}_i$ in each iteration with the previously placed flows remaining unchanged. Finally, the amount of flow on each edge is scaled back by the number of iterations. The following significant results are reproduced from [3].

Theorem 8.1 *Given any non-negative values $l(e)$ for each edge $e \in E$, the expression:*

$$\frac{\sum_{i=1}^{k} \min_{T \in \mathcal{T}_i} \sum_{e \in T} w_i(e) l(e)}{\sum_{e \in E} l(e) u(e)}$$

provides a lower bound on the optimum value of the fractional global routing problem. Moreover, there exist nonnegative values $y_e, e \in E$, such that the expression above is equal to the optimum value of the fractional global routing problem.

Theorem 8.2 *If there exists a solution for the fractional global routing problem with maximum relative congestion smaller than 1, the algorithm finds a $\rho-$approximation in at most*

$$1 + \frac{1}{\epsilon' \hat{\lambda}^*} \ln_{1+\epsilon'}(\frac{m}{1-\epsilon'})$$

phases, if $\rho = 1/(1-\epsilon')^3$ and $\epsilon' = \epsilon(1+\gamma\epsilon)$. Moreover, the variables $y_e, e \in E$, converge to the solution of the dual linear program.

9 Hierarchical methods

The first prominent hierarchical method for routing was proposed by Burstein and Pelavin in [8]. This method recursively divides the routing regions into successively smaller sub-regions, and nets at each hierarchical level are routed simultaneously and refined in the subsequent levels. The sub-regions are referred to as super cells, compared to the global routing cells (GRC) defined in the global routing formulation. This is a systematic divide-and-conquer approach and therefore transforms the large and complicated global routing problem into a series smaller and simpler sub-problems. As a result, this method is inherently faster than flat routing methods. In fact, this approach is quite versatile and can be applied to detailed routing as well. Moreover, it reduces the problem of net-ordering in sequential routing since higher level decisions are used to guide the solution at lower levels of the hierarchy. While this approach does not entirely remove the net ordering issue, this is an issue of diminished importance in comparison with flat approaches. The simplicity of this method carries with it limitations on the quality of the routing solution, since a routing decision at one hierarchical level neglects the requirements at subsequent level and may make a choice that is suboptimal for subsequent levels. Thus, a hierarchical method is often combined with other global routing techniques, such as rip-up-and-reroute.

Hierarchical methods form a rich family of global routing approaches and numerous works [8, 14, 32, 33, 41, 51, 53, 59, 62–64, 69, 70, 89] have been reported. These methods can proceed in a top-down, bottom-up or hybrid manner and have been applied to gate array/sea of gates, standard cell and custom design. Typical partitioning methods may proceed by bisection or quadrisection, and may be either uniform or non-uniform. In this section, we will describe some major representative works in this family of methods.

9.1 Top down successive refinement

The original method proposed by Burstein and Pelavin in 1983 was applicable to gate arrays, switchbox routing and channel routing. The method is based on the assumption of uniformity of the wiring substrate, a restriction removed by subsequent works. In addition to minimizing wire congestion, this method tries to minimize via congestion and wire length in terms of the number of grid cells traversed by the wires.

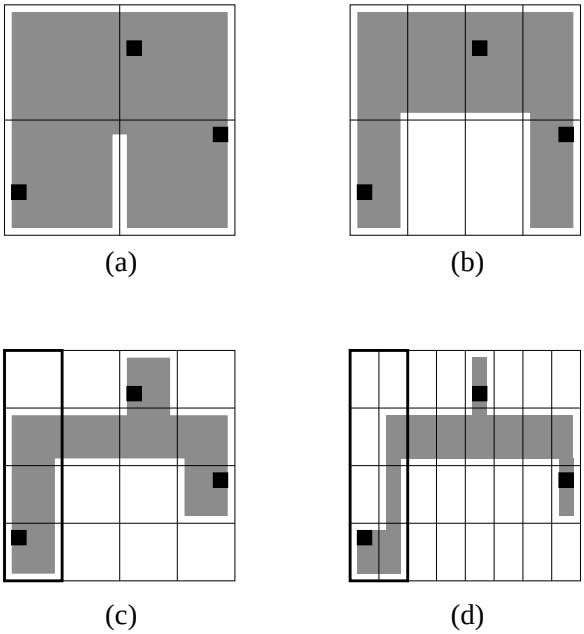

Figure 15: An example of routing by top-down hierarchical refinement.

As shown in Figure 15, the routing region is recursively bisected into smaller super cells, and at each level, the routing is performed in terms of the super cells at that level. In the example of Figure 15, the routing at each level is represented by the shaded super cells for a 3-pin net. This process is performed

in a top-down manner until the super cells reduce to the actual grid cells for global routing.

The initial routing is trivial and the critical part of this algorithm is related to refining the routing solution from one hierarchical level (a $1 \times N$ super cell) to a lower level (a $2 \times N$ (super) cell) after each bisection. For the example in Figure 15, a routing solution for the set of 1×4 cells that lie inside the thickened box in (c) is refined to a routing solution for the 2×4 cells inside the thickened box in (d) in the next step. Thus, the critical problem is to determine routes for multiple nets in a $2 \times N$ cell to minimize congestion and wire length. Burstein and Pelavin proposed two heuristics to solve the $2 \times N$ routing problem: (1) divide-and-conquer, and (2) dynamic programming.

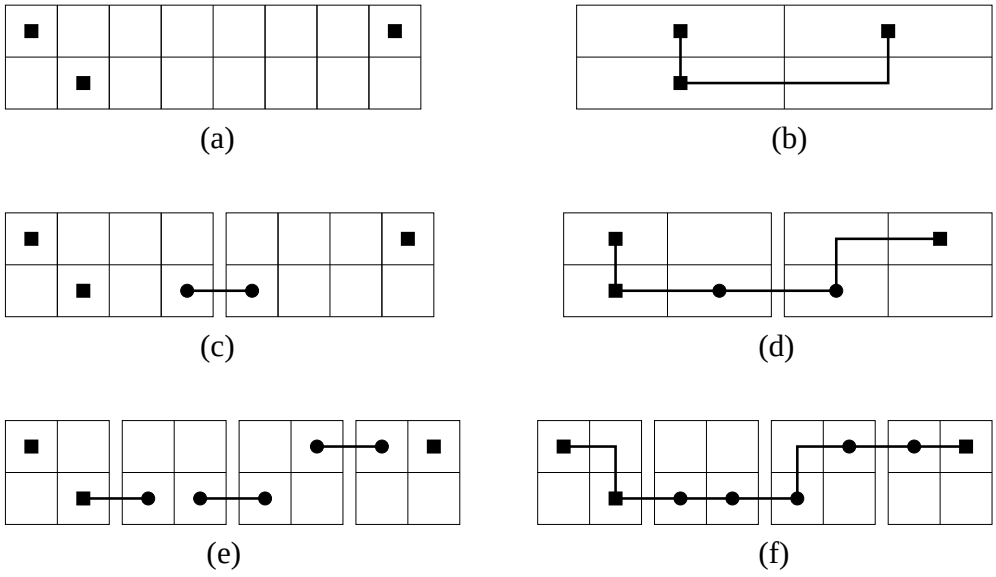

Figure 16: An example of divide and conquer in $2 \times N$ routing.

The divide-and-conquer heuristic is illustrated in the example in Figure 16. The $2 \times N$ ($N = 8$) routing problem in Figure 16(a) is first reduced to a 2×2 routing problem and solved as in Figure 16(b). Next, the cells are partitioned in halves and two pseudo pins (black dots in Figure 16(c)) are introduced at the two neighboring cells across the cutline and on the routing path of solution in (b). Now the problem is reduced to routing a $2 \times N_1 (N_1 = 4)$ and a $2 \times N_2 (N_2 = 4)$ problems, where $N_1 + N_2 = N$. The two new sub-problems can be recursively solved as a series of 2×2 routing problems as shown in Figure 16(d) – (f).

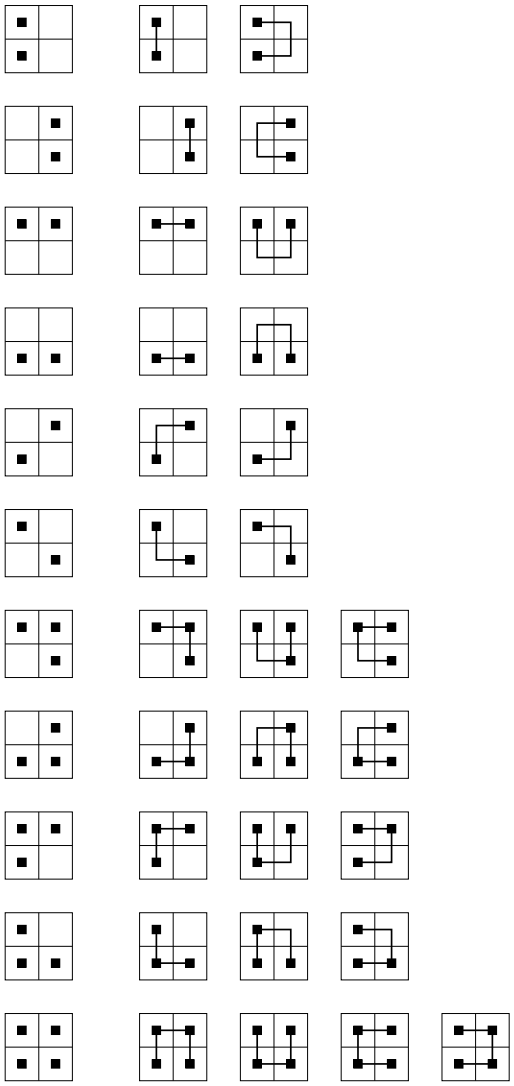

Figure 17: An enumeration of all patterns for 2×2 routing.

Through this divide-and-conquer process, the $2 \times N$ problem is reduced to a series of 2×2 routing problems, each of which is relatively simple since all of the possible routing patterns can be enumerated easily as shown in Figure 17. Therefore, the problem of routing each net in a 2×2 grid is equivalent to choose a routing pattern for each net among the patterns in Figure 17. Burstein and Pelavin formulate this problem as integer programming problem to minimize overflow and maximize wiring and via slacks; the wiring slack is defined as the available wiring tracks on a grid boundary minus the number of wires across this boundary, while the via slack is the number of via spaces in a grid cell minus the number of bends in this grid cell. The integer programming is first relaxed and solved as a linear program, after which the integer solution is obtained by rounding the fractional solutions.

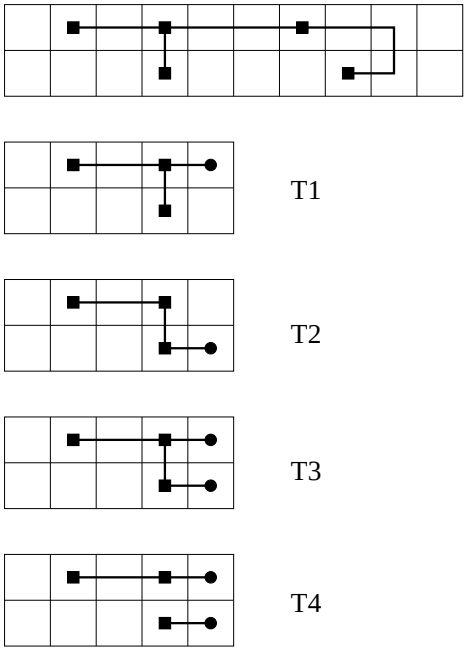

Figure 18: Examples for different tree configurations representing optimal subsolutions for a dynamic programming based solution to the problem of $2 \times N$ routing.

The second approach that was proposed is based on dynamic programming. For a set of $2 \times N$ cells, the procedure starts with the leftmost column k, with a terminal cell containing a pin, and grows a tree towards the right by incrementing k by one column at a time until the N^{th} column is reached. At each step, there are a set of optimal subsolutions of partial trees that are maintained for the $2 \times k$ cells to the left. These trees can be enumerated as:

- $T^1(k)$: a minimum cost tree connecting all of the terminal cells in the left k columns to cell $g(1,k)$.
- $T^2(k)$: a minimum cost tree connecting all of the terminal cells in the left k columns to cell $g(2,k)$.
- $T^3(k)$: a minimum cost tree connecting all of the terminal cells in the left k columns, to both $g(1,k)$ and $g(2,k)$.
- $T^4(k)$: two disjoint minimum cost trees T^* and T^{**}, $g(1,k) \in T^*$ and $g(2,k) \in T^{**}$. Each terminal cell is in either T^* or T^{**}.

An example of a set of such trees is shown in Figure 18. When the trees grow from column k to $k+1$, a new set of trees $T^i(k+1), 1 \leq i \leq 4$ are obtained from $T^i(k), 1 \leq i \leq 4$, and the minimum cost tree of

each type is stored for each value of k. The dynamic programming property arises from the fact that the optimal subsolution for a value of k can be created from the optimal subsolution trees for column $k-1$. The above $2 \times N$ grid routing procedure is performed for each net sequentially with the wiring cost dynamically updated according to the changes on congestions. After all of the nets have been routed, each net is ripped up and rerouted by repeating the same procedure, but using the complete congestion information. The computational complexity for the presented algorithms is $O(k(p+q)\log(pq))$ for k nets to be routed on a $p \times q$ grid.

9.2 Bottom up merging

In [63], a bottom-up hierarchical global routing method is described by Marek-Sadowska. This method attempts to overcome the limitation of Burstein and Pelavin's method that limits it to a uniform substrate.

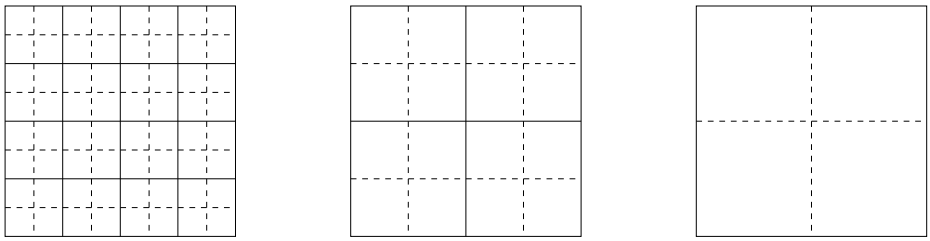

Figure 19: Routing is performed within each solid super cell in bottom-up hierarchical routing.

Initially the routing region is partitioned into a set of 2×2 super-cell arrays. In Figure 19(a), each array is separated by the solid lines from other arrays. The super-cells within an array are depicted by the dashed lines. At this lowest hierarchical level, each super-cell is a basic grid cell and the routes are restrained within each array individually, i.e., no wires are allowed to cross the solid border. The multi-net routing is performed in each 2×2 array through maze routing in order to minimize the congestion. At the next higher level, every other solid border is open to form a new set of 2×2 arrays as shown in Figure 19(b). This process is repeated until the top level only one 2×2 array is reached as in Figure 19(c). This bottom-up hierarchical method is followed by a rip-up-and-reroute process.

In [41], a linear programming based bottom up hierarchical global routing method was proposed by Hu and Shing. In contrast to the uniform quadrisections in the above hierarchical method, this technique performs recursive bisections according to the net distributions. Without loss of generality, we can assume that the initial bisection is a vertical cut line. In deciding where to make this cut, the following density ratio R is evaluated.

$$R = \frac{\text{number of nets separated by the cut}}{\text{number of available tracks across the cutline}} \tag{9}$$

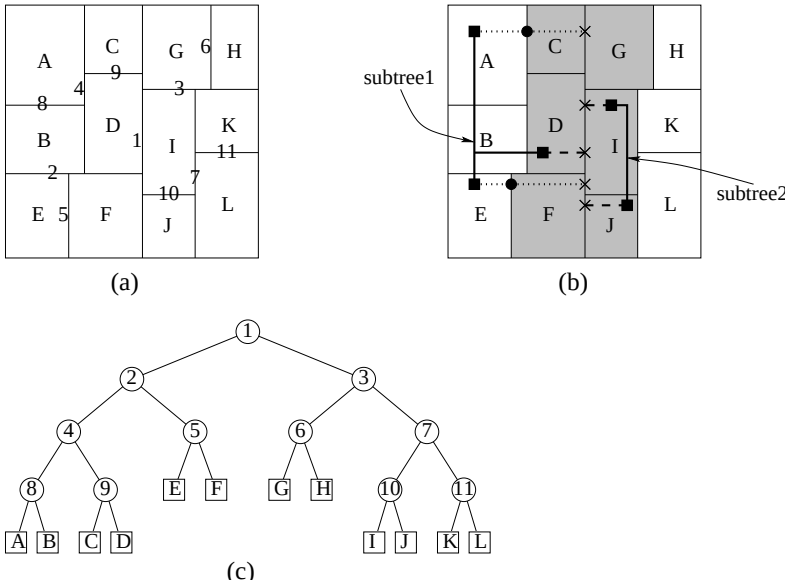

Figure 20: (a) Recursive bipartitioning according to net distributions. (b) An example of merging sub-routing-solutions (c) The resulting partition tree.

The cut line that maximizes the ratio R is chosen for this bisection. If this ratio is greater than one, then there is no feasible solution for this problem. On the other hand, even if the ratio R is no greater than one, it is likely to be close to one because this is the maximal ratio. This implies that each net should should cross this cut line once. In the example in Figure 20(a), the entire region is first bisected, by the vertical cut line labeled 1, into the left and right subregions. Then the two subregions are further bisected recursively in the same manner until the smallest subregions, called atomic regions, have a size that is manageable for integer linear programming (ILP) solvers. This recursive bisection process can be represented as a binary tree, shown in Figure 20(c). Each leaf in the binary tree corresponds to an atomic region, the root corresponds to the initial cut line and the internal nodes represent other cut

lines.

After this cutting process, the global routing within each atomic region is formulated and solved as an ILP problem. Next, the solutions for each of the atomic regions are recursively pasted back in the reverse direction of the cutting process, i.e., the subsolutions are merged together in a bottom up manner. Figure 20(b) shows an example of how subtrees on two sides of a vertical cut line are connected. Each pin in an atomic region adjacent to the cut is projected to the cutline along the dashed lines to form the set of candidate crossing points marked "X" on the cut line, corresponding to connections from pins in regions D, I and J. If a pin is not in an atomic region adjacent to the cut line, it is first projected to such an atomic region and then further projected to the cut line, as shown by the dots in Figure 20(b). The crossing point for each net is restricted to be one of the candidate points. The problem of merging multiple nets across a cut line is then solved as an ILP to ensure that no wiring overflow occurs.

9.3 Hybrid hierarchical method

A common problem for hierarchical methods is that the routing decision at one hierarchical level may be suboptimal for a subsequent level, and this is true for both top-down and bottom-up approaches. In order to overcome this problem, Hayashi and Tsukiyama proposed a hybrid hierarchical approach in [32] that combines top-down and bottom-up method into a unified approach for the problem of global routing for more than two layers.

This method first decomposes each multi-pin net into 2-pin nets, or *from-to* pairs by constructing minimum spanning trees. Next, the from-to pairs are categorized according to the hierarchical levels. At the top-level (level 1), the entire routing region is quadrisected into an array of 2×2 super cells, and in level $1 \leq i \leq n$, the routing region is divided into $2^i \times 2^i$ super cells. The lowest level is n where each super cell is equivalent to the actual routing cells in global routing. A set of from-to connections routed on level i is a subset of the set of all from-to connections such that two pins of each such connection lie in two neighboring super cells at level i with a common border or a common corner point. This set of from-to connections is denoted as $FT(i)$; note that no connection in $FT(i)$ belongs to $FT(n), FT(n-1), ...,$ or $FT(i+1)$.

The flow of the algorithm consists of two loops for the hierarchical levels, with a top-down hierarchical

loop embedded in a bottom-up hierarchical loop. The outer loop start from the lowest hierarchical level n and iterates to level 1. In each iteration of outer loop with hierarchical level $1 \leq i \leq n$, this algorithm first routes all of the connections in $FT(i)$. From any two pins located in two super cells with a common border at level i, this algorithm always connects them with a wire directly across this common border. Initially, connections are made using lower routing layers, and higher routing layers are employed once the capacity on the lower layers is fully utilized. For two pins lying in regions with a common corner point, this algorithm greedily chooses one of the L-shaped connections to minimize the congestion and preferentially attempts to use the lower routing layers.

After this initial routing at level i, a top-down loop is started from hierarchical level $i+1$ to level n to further refine the routing of connections in $FT(i)$. In each iteration of this inner loop, the route for each from-to connection in $FT(i)$ is refined through maze routing and a rip-up-and-reroute procedure, using congestion to measure the cost. Compared with pure top-down hierarchical routing and pure rip-up-and-reroute, a hybrid hierarchical method such as this one provides the potential for reduced congestion.

9.4 Hierarchical routing for custom design

The hierarchical methods introduced in sections are all based a uniform grid graph. In [62], Luk *et al.* proposed a hierarchical global routing method for custom circuit design based on a slicing floorplan, as shown in Figure 21(a), where each rectangle is a building block. The routing graph for such a design is a dual graph for the floorplan as shown in Figure 21(b). It is assumed that the detailed location of each pin has not been assigned at this stage, and therefore the pins within each building block are assumed to lie at its center, as shown in Figure 21(a) by the solid squares. In the routing graph in Figure 21(b), each node (round dot) corresponds to a building block and an edge is set up for every adjacent building blocks. Each edge has an edge length, given by the Manhattan distance between the two rectangles that this edge connects. In addition, a routing capacity is associated with each edge.

The entire routing region is recursively partitioned into a set of four super cells, called H_4. Such a partition is generated by a horizontal cut followed by one vertical cut in the top portion and one vertical cut in the bottom portion, or by a vertical cut followed by one horizontal cut in the left portion and one horizontal cut in the right portion. One such partition is shown in the thickened lines in Figure 21(c).

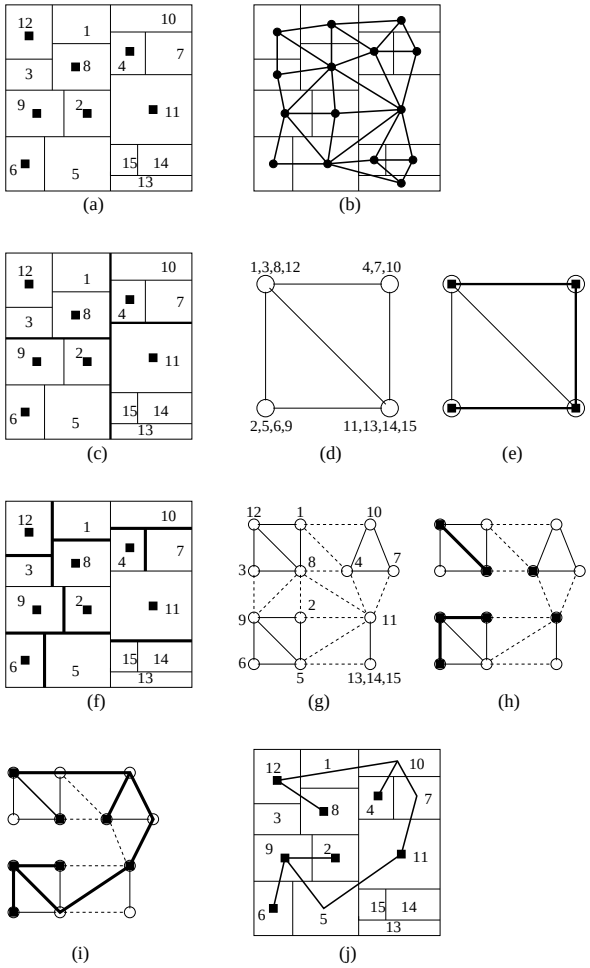

Figure 21: An illustration of hierarchical routing for custom design.

Routing at this level is performed on a routing graph corresponding to this abstraction, as shown in Figure 21(d). Each super node in this routing graph corresponds to a super cell at this level, and this routing graph is a clique on four nodes with one edge removed. When a four-partition is not possible, a three-partition (K_3) is performed to form a routing graph that is a clique of three nodes. When neither four-partition nor three-partition is feasible, a two-partition (K_2) is performed as shown by the solid line connecting the region (13,14,15) with cell 11 in the lower-right portion of Figure 21(f).

The routing patterns on an H_4 (K_3 and K_2 are degenerate versions of H_4) graph can be enumerated, and routing for multiple nets to minimize congestion and wire length is carried out on this graph, similar to Burstein and Pelavin's approach of integer programming. In [33], Heisterman and Lengauer suggested speed-up techniques that can improve the speed of this routing problem using integer programming.

For the given net in Figure 21(a), the routing at the first level is shown by the darkened edges in Figure 21(e).

At the next level, the four super cells in level one are partitioned again as shown in the darkened lines in Figure 21(f). The corresponding routing graph is shown in Figure 21(g), where the solid edges are expanded from each super node in the previous hierarchical level and the dashed edges are expanded from the super edges of the previous hierarchical level. The routing on the cells expanded from same super node is similar to the H_4 routing introduced earlier. The four super nodes in level one result in the routing network depicted in Figure 21(h). Note that for each net, the routes chosen at the previous level are transformed into a reduced routing graph as shown in Figure 21(h). A Steiner tree algorithm on this reduced routing graph is applied to join the trees in the forest to form the routing tree at the current level, as shown in Figure 21(i). This process is repeated until the hierarchy proceeds down to the level of the original building blocks. The final routing of this particular example is displayed in Figure 21(j). For the problem of routing k nets for n macros, the computational complexity of this algorithm is $O(kn^2)$.

9.5 Hierarchical bisection and linear assignment

Instead of recursive quadrisections, another class of hierarchical global routing works use recursive bisection [53,64]. This method can be applied to both regular gate arrays and irregular custom designs, and was proposed independently by Marek-Sadowska and by Lauther in 1986. We will use an example similar to that in [64] to illustrate this method.

Figure 22(a) shows a design consisting of a set of rectangles that correspond to building blocks. The darkened line shows a cut line that bisects the routing region; although a straight cut is shown here, zig-zagging cuts may also be used. Along this cutline, there are eight sections of the borders between the neighboring building blocks, and these sections are called channels. For any net that must cross this cutline, the task at the current hierarchical level is to identify a channel where the net makes the crossing, such that no wiring overflows are seen and the wire length is minimized. For example, consider the net represented by small squares in Figure 22(b) to be routed through the channels. Once a crossing channel is found for a net, a pseudo pin is inserted in the net at the channel on either side of the cut line, as shown by the dots in Figure 22(c). The original net is then broken into two subnets in the left

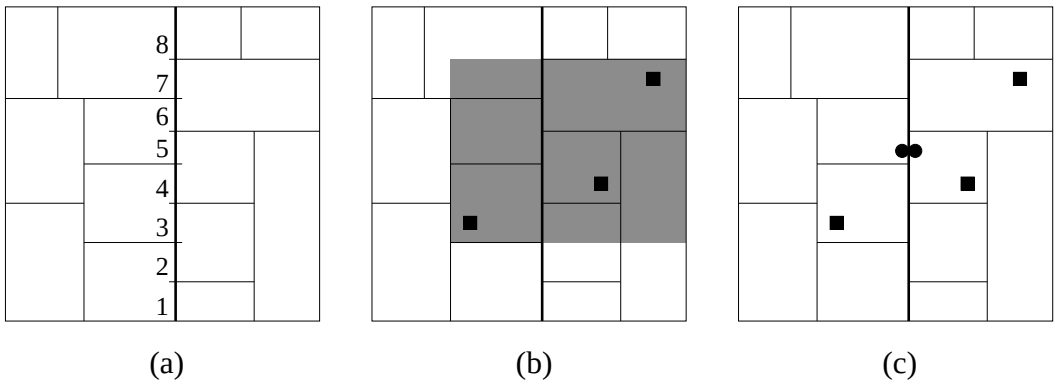

Figure 22: (a) Enumerating the channels along the thickened cutline. (b) The routing region for a three-pin net, shown by the shaded region. (c) Insertion of a pseudo pin at the channel where the net crosses through.

and right subregions, respectively. This bisection and assignment process is then performed recursively in the subregions until each subregion is at the level of a building block in custom design or a grid cell in a gate array.

The problem of finding a channel for each crossing net is formulated and solved as a linear assignment problem. The rows of the cost matrix correspond to the nets to be assigned and the columns of the cost matrix correspond to the channels along the cut line. Each entry in the cost matrix includes two terms: wire length cost and net criticality. If channel j lies within the bounding box of net i, then the wire length cost $a_{ij} = 1$; otherwise, a_{ij} increases with the increase of wire length. In Figure 22(b), the bounding box of the given net is depicted by the shaded area, and the wire length costs for this net and all of the channels are, respectively, $[3, 2, 1, 1, 1, 1, 1, 2]$.

One shortcoming of this method is that each net is allowed to cross the cutline only once. This may place unnecessary restrictions on the routing paths. In order to deal with this problem, Marek-Sadowska described a net decomposition scheme that permits a multi-pin net to cross the cut more than once. Lauther [53] derives a computational complexity of $O(kp \log p)$ for routing k nets on a $p \times p$ grid.

9.6 Four bend hierarchical routing

In [14], a hierarchical global routing method that can control the number of bends is developed. Like other hierarchical approaches, this method recursively quadrisects the routing region into successively smaller array 2×2 cells. The main algorithm considers only two-pin nets and uses the $FT(i)$ concept introduced in Section 9.3. At each hierarchical level i, only the nets of $FT(i)$ are routed, using a two-stage approach. The difference between this and other hierarchical methods is that the route for each net in $FT(i)$ is determined in terms of the actual basic grid cells at level i, rather than in terms of super cells, and therefore these routes are fixed once they are identified, and are not refined in a later stage.

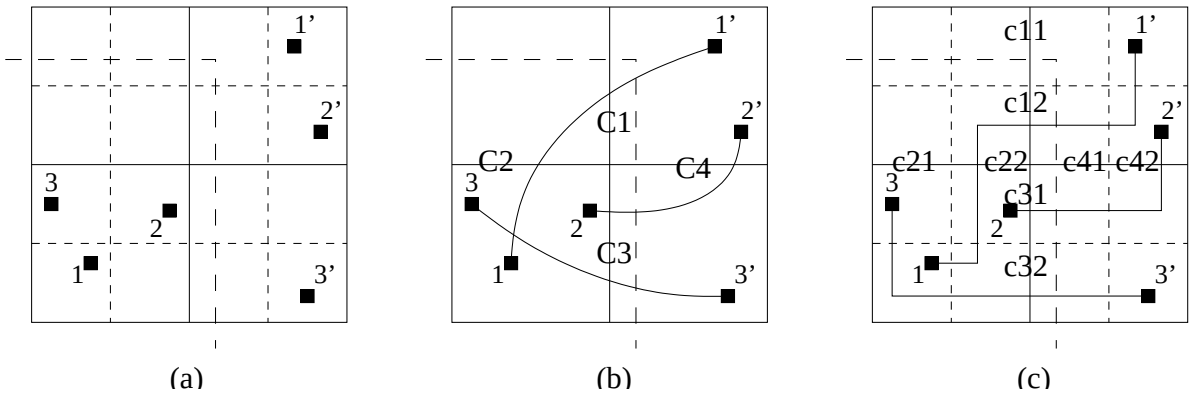

Figure 23: (a) An example of two-stage routing at one hierarchical level of the four bend hierarchical routing algorithm. The route shown by a thickened dash line is a wire routed at a higher hierarchical level. (b) The routing result of stage one, with the routes shown by the thickened solid lines. (c) The final routing result with no more than four bends.

We will illustrate routing in an array at a fixed hierarchical level by means of the example shown in Figure 23. Figure 23(a) illustrates an array of 2×2 super cells indicated by the solid grid, over a basic routing grid shown by dashed lines. The thickened dashed line represents a net routed at a higher hierarchical level. At the current level, there are three two-pin nets need to be routed: $(1, 1'), (2, 2')$ and $(3, 3')$. The borders between the four super cells are the four cutlines C_1, C_2, C_3 and C_4, as shown in Figure 23(b). The borders of neighboring basic cells along these cutlines are called channels, and correspond to $c_{11}, c_{12}, c_{21}, c_{22}, c_{31}, c_{32}, c_{41}$ and c_{42} in Figure 23(c). In the first stage, each net is routed in terms of which cutline the wires must cross, and the result of this step is shown in Figure 23(b). This problem is solved using integer programming, similar to Burstein and Pelavin's method. In the second stage, the route for each net is further refined to determine the channels that it must pass through,

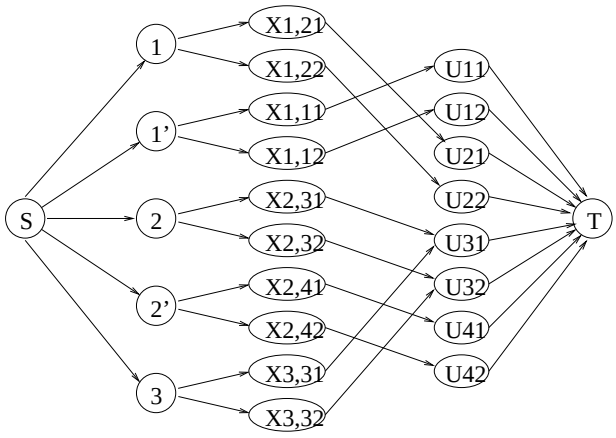

Figure 24: A network flow model constructed from Figure 23(b) to solve the second stage of the four bend hierarchical routing algorithm.

and this problem is formulated as a min cost network flow problem. The network construction for the example in Figure 23(b) is shown in Figure 24. Since net $(1, 1')$ will go through cutlines C_1 and C_2 according to the result from stage 1, the problem in stage 2 is to choose between channels c_{11} and c_{12} on cutline C_1, and between channels c_{21} and c_{22} on cutline C_2. The decision process is represented by a set of variables $X_{1,11}, X_{1,12}, X_{1,21}$ and $X_{1,22}$. If $X_{1,11} = 1$, then net $(1, 1')$ will go through channel c_{11}, and so on. The routing capacity for a channel c_{ij} is represented by the variable U_{ij}. Besides the source S and target T, there are three types of vertices in the network in Figure 24: net vertices (such as 1, $1'$...) in the second column, decision vertices in the third column and channel vertices in the fourth column. The capacity for each edge between a channel vertex and target T is its corresponding channel routing capacity, the capacities for every other edges is 1. The cost of the arcs between a net vertex and a decision vertex is proportional to the wire length that the assignment based on this decision will incur. All other arcs have cost of 0. We can see that a min-cost solution on this network will provide a routing result that satisfies the wiring capacity and minimizes the wire length. In addition, by construction, it is ensured that the number of bends for each wire never exceeds four.

10 Iterative deletion

Another technique in global routing that can avoid the net ordering difficulty is iterative deletion, due to Cong and Preas in [24, 25]. This algorithm was originally designed for standard cell design. It begins with a redundant routing and then iteratively deletes redundant edges to minimize congestion

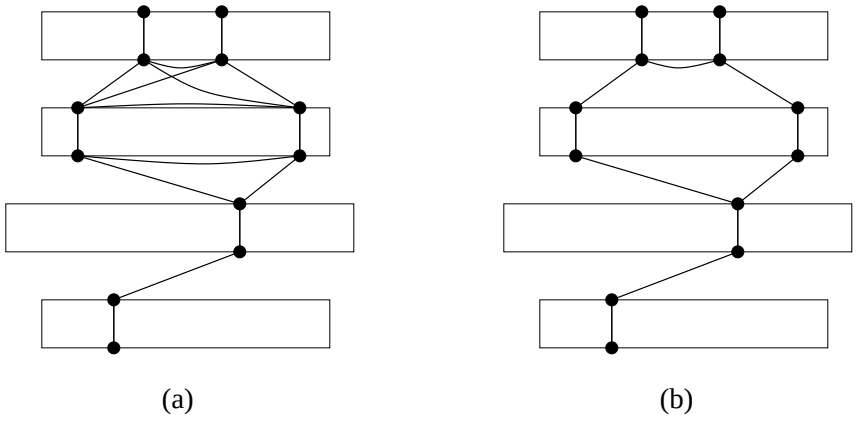

Figure 25: Example of (a) a connection graph and (b) a simplified connection graph for a net to perform iterative deletion.

while ensuring that no deletion step causes a net to be disconnected. For the example in Figure 25(a) that shows rows of standard cells separated by channels, every pair of pair of pins in the same channel and every feedthrough is connected by an edge to form a net connection graph. The weight of each edge is defined to be $w(e) = d(e)/d$, where $d(e)$ is the maximum density over the edge in the channel to which e belongs and d is the density of the channel. The maximum weight edge is iteratively deleted until each net is in the form of a tree structure. The edge weights are updated dynamically after each iteration according to the changes in the density.

The advantage of this method is that it avoids the net ordering difficulty commonly encountered in global routing. In addition, the initial redundant routings provide a method for predicting the congestion distribution, so that each deletion is guided towards reducing congestion.

The limitation of this technique is that the solution space should be restricted to ensure that it is not too large, i.e., the number of initial redundant connection edges should be limited; otherwise, this technique will become very slow. For example, the number of edges in the connection graph is $O(n^2)$, where n is the number of pins. This implies that a final $O(n)$ sized solution must be obtained through $O(n^2)$ deletion steps. Moreover, connection graph that is too dense may contain too many redundant edges so that its prediction on congestion is poor.

In order to overcome these deficiencies, Cong and Preas reduce the size of the original connection graph by enforcing the rule that no terminals are allowed between two ends of an edge for the same

net in the same channel. Such a graph is called the simplified net connection graph. The simplified connection graph for the net in Figure 25(a) is shown in Figure 25(b). This simplified graph has a size of $O(n)$ and can be constructed in $O(n \log n)$ time. If the maximal number of pins in a channel is L, the complexity of this algorithm is $O(n^2 \log L)$.

11 Timing driven global routing

As interconnect delay becomes a dominant factor that determines system performance, it is no longer adequate to consider only congestion. Several works [23, 35, 38, 42, 48, 86, 91] have been reported to incorporating timing issues into global routing. In this section, we summarize some of the prominent approaches for global routing under timing constraints.

11.1 Multicommodity flow based approach

In [35], Hong *et al.* developed a timing driven global router called TIGER, which is a multicommodity flow based approach, similar to [9]. Recall that Steiner trees need to be constructed for each net to minimize the congestion in [9]: this approach incorporates the timing issue in addition to congestion such that the path-based timing constraints can always be satisfied.

The delay model employed in TIGER uses an upper bound of the signal propagation delay in an RC tree, defined as follows. Let R_d denote the driver resistance and C_L the load capacitance for each sink. The wire resistance and capacitance per unit length are r and c, respectively. Consider a net $N = \{v_0, v_1, v_2, ...\}$, where v_0 is the source node and others are the sink nodes for which we construct a routing tree T over net N with wire length of W. For a sink $v_i \in N$, if the routing path length from source v_0 to v_i is $p_r(v_0, v_i)$, then the delay upper bound is given as:

$$t_{up}(v_i) = \beta(R_d + rp_r(v_0, v_i))(cW + C_L) \tag{10}$$

where $\beta = 2.21$, which represents the point at which the signal reaches 90% of its final value.

One of the ultimate objectives of timing-driven design is to achieve a specific clock cycle time, so that each register-to-register delay along each path is less than a constraint. This is referred to as a path-based constraint. There are typically multiple nets along a path between two adjacent stages of registers, as illustrated in the example in Figure 26, where the critical path is (v_1, v_2, v_3, v_4) and nets

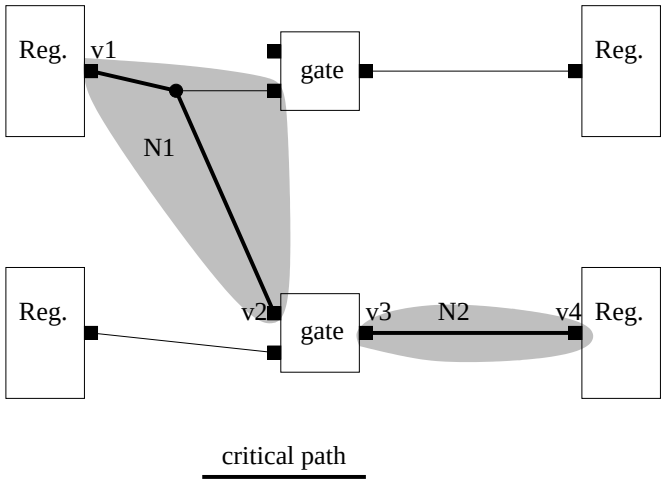

Figure 26: An example of a timing graph.

N_1 and N_2 are along that path. In contrast, in net-based approaches, the path delay constraint is decomposed into a set timing budgets assigned to the nets. If these budgets are allocated poorly, some nets located in congested areas may not have adequate timing slack to achieve a feasible routing to avoid the congestion. In TIGER, the routing tree is chosen so that the timing constraint along the critical path is always satisfied, and this provides greater flexibilities than the net-based constraints.

This timing constrained global routing problem is formulated as a multicommodity flow problem, as in [9]. In addition to the formulation for gate arrays, a formulation for standard cells is also provided. Recall that the objective of global routing in standard cell design is to minimize the chip height. If we use $f_{i,j}(e)$ to indicate whether tree $T_{i,j} \in \mathcal{T}_i$ uses edge $e \in E$, and denote the channel height at row r as $h(r)$, the LP formulation for standard cell global routing is as follows.

$$\begin{array}{rcl} \text{minimize} & \sum_{\forall r} h(r) & \\ \text{subject to:} & \sum_{T_{i,j} \in \mathcal{T}_i} x_{i,j} \geq d_i, & \forall N_i \in \mathcal{N} \\ & \sum_{N_i \in \mathcal{N}} \sum_{T_{i,j} \in \mathcal{T}_i} x_{i,j} f_{i,j}(e) \leq h(r), & \forall e \in E_{hori} \\ & \sum_{N_i \in \mathcal{N}} \sum_{T_{i,j} \in \mathcal{T}_i} x_{i,j} f_{i,j}(e) \leq u(e), & \forall e \in E_{verti} \\ & x_{i,j} \geq 0, & \forall N_i \in \mathcal{N}, \forall T_{i,j} \in \mathcal{T}_i \\ & h(r) \geq 0, & \forall \text{rows} \end{array} \tag{11}$$

The major difference from [9] is that each Steiner tree generated is always checked to ensure that

the critical path timing constraint is still satisfied, and only Steiner trees that can satisfy the timing constraint are accepted, and this principle is also adhered to in the final rip-up-and-reroute stage.

11.2 Iterative deletion based routing for standard cells

A timing driven global routing method for standard cell design was reported by Cong and Madden in [23]. This method constructs initially minimum spanning trees for each net The experimental result in this paper shows that the wire length reduction from MST to the approximated Steiner minimum tree for standard cell design is limited. If an MST cannot satisfy its timing constraint, a timing driven tree topology construction [22] and wire sizing techniques [18, 21] are applied to ensure that the timing constraint is satisfied. Next, the routing topologies of the critical nets are fixed, and a simplified connection graph is constructed for the non-critical nets. Finally, an iterative deletion process [25] is performed to minimize the congestion.

11.3 Hierarchical bisection and assignment

A net-based timing-constrained global routing approach is developed by Hu and Sapatnekar in [38]. This method emphasizes the local routing flexibilities that are available for achieving timing-constrained congestion reduction. This method starts with a timing-driven route for each individual net so that the timing constraint for each net is satisfied. The congestion is reduced through trading the flexibilities in routing these nets without affecting the timing performance.

One major timing-constrained flexibility employed in [38] is the idea of a soft edge, which is an edge connecting between source, sink and Steiner node with degree greater than two. A soft edge does not specify the precise bend nodes required to embed the wiring route into rectilinear space immediately. Instead, the precise embedding of the route is deferred to minimize congestion while the edge length is kept constant, as illustrated in Figure 27. Another timing-constrained flexibility used in [38] is the notion of a slidable Steiner node. According to [36], the optimal Steiner node for minimizing wire length subject to timing constraint may lie off the Hanan grid [30]. Such a non-Hanan Steiner node can slide along a locus of points that form a line segment, without affecting the edge length incident on it, as illustrated by the example of the Steiner node v' in Figure 28. Therefore, this idea can to provide timing-constrained routing flexibility. The third flexibility is simply to permit edges to be elongated

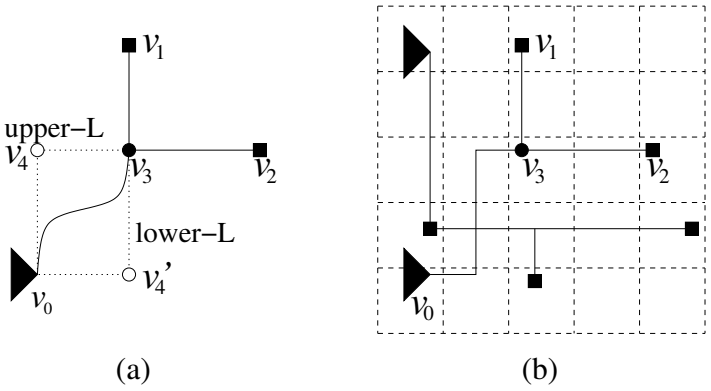

Figure 27: (a) The use of a soft edge (v_0, v_3) that does not restrict the connection to a fixed path. (b) The use of this flexibility in determining a Z-shaped path for the segment that avoids congested areas.

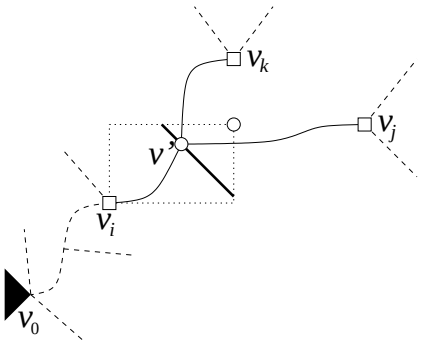

Figure 28: An example of a slidable Steiner node v'.

without causing any delay constraint violations, and this flexibility is available when there is a timing slack at a sink node.

After the initial routing, where each net is routed to meet its timing constraint without considering congestion, the congestion is reduced through a hierarchical bisection and assignment process similar to [64]. For a set of trees in Figure 29, their soft edges are routed through the channels along the cutline, as in Figure 30. The assignment in [38] is based on routing each soft edge through the channels. Due to the complexities associated with timing constraints, the routing through the channel cannot be implemented through linear assignment as in [64]. The assignment problem is solved through a

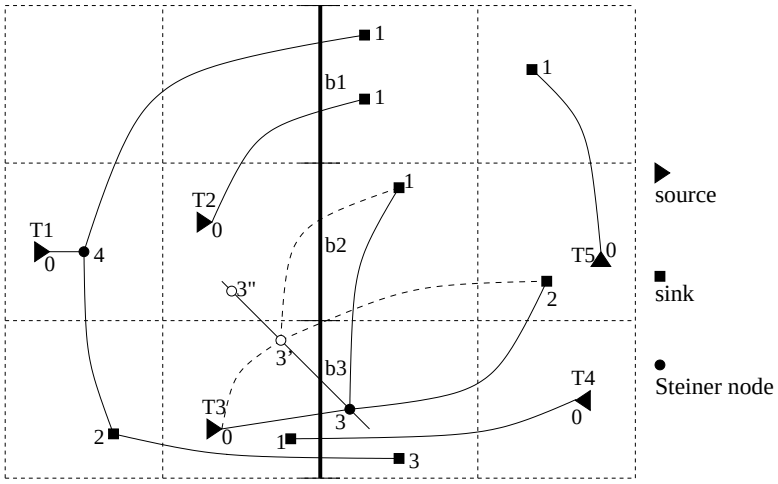

Figure 29: Assigning soft edges to each cell boundary along the thickened cutline.

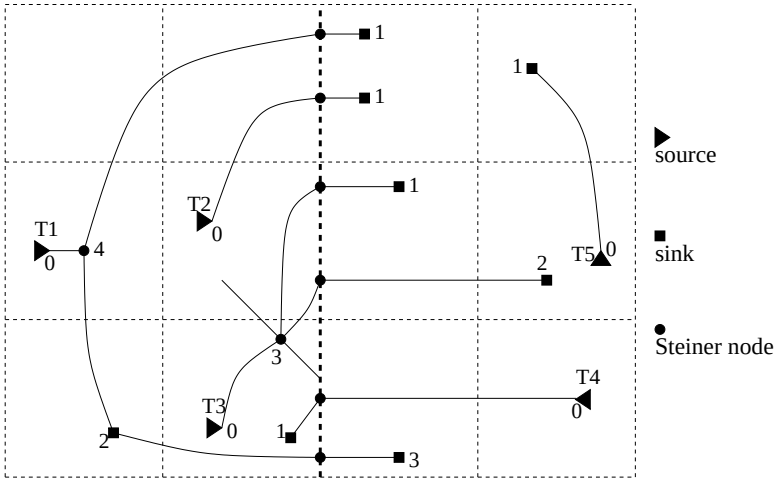

Figure 30: A sample solution to the assignment problem in Figure 29.

network flow formulation, in which only those assignments that will not cause delay constraint violations constitute the legal candidates. Some assignments may force a wire to have a detour, and the timing slack of its corresponding net will be reduced. In the network flow formulation, the cost of each assignment is defined to be proportional to the slack consumption. If a routing tree has multiple soft edges crossing the cutline, then whether a soft edge can be elongated or not depends on whether its sibling edges have been elongated and if so, how far they have been elongated, since they all share the a common timing constraint and compete for a common timing slack resource. To resolve this scenario, this work identifies the congested area along the cutline by running a max-flow algorithm on the partially constructed network. The soft edges in the multi-crossing trees are elongated to permit

an increase in the max-flow at the min-cut, by elongating them to pass through less congested areas to allow a feasible routing.

For each slidable Steiner node, two of its candidate positions, one on each side of the cutline, are considered as illustrated by the nodes v_3 and v_3' of tree T_3 in Figure 29. Since these two locations imply that a different number of soft edges will cross the cutline, they are both are incorporated into a generalized network flow model [2] as shown in Figure 31. In a generalized network flow model, each edge has a gain factor $g(e)$ besides cost and capacity, i.e., if a flow of $f(e)$ enters an edge, a flow of $g(e)f(e)$ will leave it. The generalized network flow problem is solved through Wayne-Fleischer algorithm [88]. Finally, this algorithm resolves any residual wiring overflows through a timing-constrained rip-up-and-reroute method post-processing step.

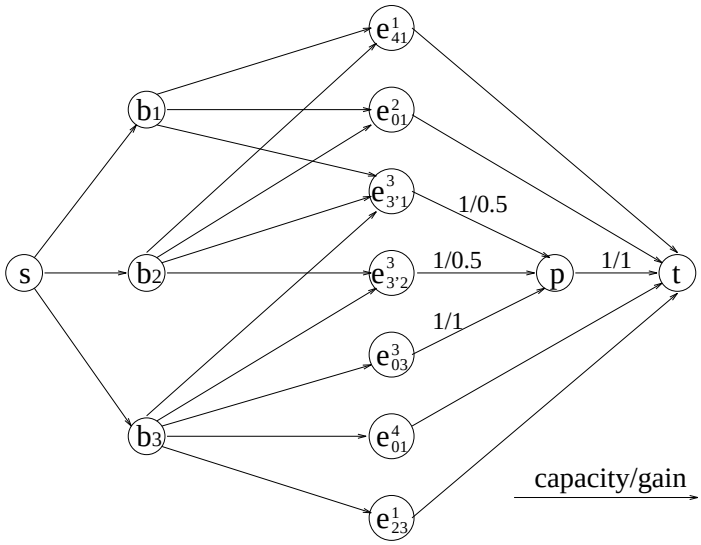

Figure 31: A generalized network flow model for solving the assignment problem in Figure 29.

12 Crosstalk driven global routing

Under deep submicron technology, VLSI circuit performance is increasingly affected by crosstalk noise due to the greater proximity between wire tracks, and the consequently increased contribution of coupling capacitances. The crosstalk noise from the coupling capacitance is usually considered and restrained in the detailed routing stage since the wire neighborhood information is required to have a reasonable estimation on coupling capacitance. However, the wiring topology flexibility at the detailed stage is limited since the route change is limited to a local small region, such as a routing channel or

a switch box. In [92, 93], Zhou and Wong consider crosstalk avoidance in global routing with the idea of exploiting the greater flexibility that is available at the global routing stage to alleviate the crosstalk problem.

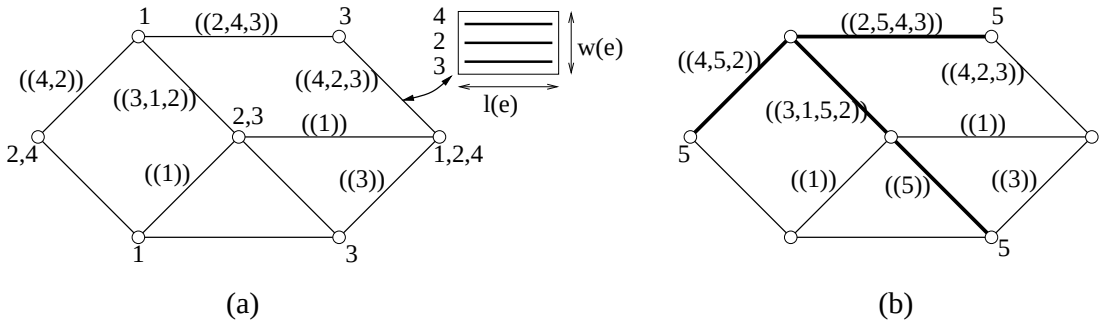

Figure 32: (a) An example showing how nets 1-4 are routed on an extended routing graph with track assignment. (b) Routing and track assignment for net 5 in on the extended routing graph in (a).

However, the wiring track adjacency information is not available in the global routing stage. To overcome this difficulty, their global routing method performs a simple layer and track assignment on the fly to obtain a rough estimation of the coupling capacitance. Therefore, their routing graph includes layer and track assignment information, and is called an extended routing graph. The global routing on this graph is called extended global routing. One example of such an extended routing graph is given in Figure 32(a). This is an undirected graph $G = (V, E)$. Each net consists of a set of pins that form a subset of the set of vertices, V. Each edge in E implies the presence of a set of routing tracks in k layers connecting adjacent vertices. An extended global routing solution is an edge labeling S such that for each edge e there is a label $S(e) = (s_1, s_2, s_3, ...s_k)$, where each s_i for $1 \leq i \leq k$ is a sequence of nets routed on layer i. The order of this sequence indicates the track assignment in this layer. Each edge $e \in E$ has a length $l(e)$ and width $w(e)$ associated with it. The example in Figure 32(a) is for one layer situation.

When the wiring adjacency information is available, a coarse estimate of the crosstalk can be made as follows. If nets a and b are adjacent on layer i of edge e, the coupling capacitance between them is computed as $C_{ab}(e) = \alpha \frac{l(e)}{w_i(e)/N_i(e)}$, where $N_i(e)$ represents the number of nets on layer i of edge e and

α is a constant. The crosstalk on net j is given by:

$$X_j = \sum_{i \neq j} \gamma_{ij} C_{ij} \tag{12}$$

where C_{ij} is the coupling capacitance between net i and j, and γ_{ij} is the crosstalk coefficient with value in $[0, 1]$. The crosstalk driven global routing problem is then formulated as:

Crosstalk-Constrained Global Routing(CCGR) Given a set of n nets, a routing graph $G = (V, E)$ and the crosstalk constraints $C_1, C_2, ...C_n$, find an extended global routing solution S such that $X_i(S) \leq C_i$, for all $1 \leq i \leq n$.

This CCGR problem is solved using a two stage heuristic in [92]. The first stage is a net-by-net sequential routing step that minimizes the total crosstalk over all of the nets. The routing procedure for each net consists of a Steiner tree construction and layer/track assignment process.

Zhou and Wong showed that the problem of assigning only layers/tracks to minimize crosstalk in one region is an NP-hard problem. This motivated the design of a heuristic that ensures that that the routed nets are kept in their layers and the order of the track assignment in one layer is not changed. For a new net to be routed in the sequential routing, a layer is chosen and the net is inserted into a track among the routed nets so that the increase of crosstalk is minimized. In Figure 32(b) a new net, net 5, is added and inserted among other routed nets. Then, the increase in the crosstalk $xtalk(e)$ on an edge e can be obtained according to this layer/track assignment information.

The minimum cost Steiner tree is heuristically constructed according to the following cost definition.

$$cost(e) = \alpha \cdot l(e) + \beta \cdot overflow(e)^2 + \gamma \cdot xtalk(e) \tag{13}$$

where α, β and γ are constant weighting factors.

In the second stage, each net with a crosstalk violation is ripped up and rerouted to satisfy the crosstalk constraint using a Lagrangian relaxation based method.

13 Conclusions and future directions

This paper presents a comprehensive survey on global routing research works, largely focusing on publications between 1983 and 2000. Due to the inherent difficulty of the problem and the increasingly complicated requirements resulting from advances in VLSI technology, global routing has been an active research area in the field of electronic design automation. Many academic and industrial researchers contributed to the numerous techniques and methods currently available. As in most cases, there is no perfect solution, and each method has its advantages and weaknesses. If one needs a simple and reasonably effective approach, perhaps rip-up-and-reroute is the best choice. When computation speed is a crucial issue, hierarchical methods can be applied. For cases that are very hard to solve, the multicommodity flow based algorithm may be the best solution technique.

Even though there is no completely satisfactory answer to the most basic congestion-driven global routing problem, the rapid progress in VLSI technology continues to impose new requirements on global routing. Currently, one of the most eminent problems is that of crosstalk avoidance in global routing. Except Zhou and Wong's work [92,93] and Xue *et al.*'s post processing method [90], few other works have been reported on crosstalk avoidance in global routing. Crosstalk avoidance in global routing has both the appealing feature of providing greater flexibility for optimization, and the difficulty of dealing with fuzzy information at the early routing stage. However, a simple observation tells us that the correlation between congestion and crosstalk may help to determine a satisfactory solution to this problem. For a specific grid cell boundary, it is generally true that greater congestion implies a greater probability of crosstalk effects. Therefore, at least we can have some probablistic estimation on crosstalk for a cell boundary. Moreover, in comparison with delay optimization, crosstalk avoidance is less contradictory to the objective of congestion reduction. However, while congestion reduction usually can help to ease the crosstalk, although the reverse is not true. Investigation on the correlation between the congestion and crosstalk may help to integrate crosstalk avoidance with congestion reduction into a unified approach.

Another interesting candidate for future research is related to the area of interconnect planning, which is the notion of helping to transform the layout at an early stage to enable wire routes that reduce congestion, crosstalk, delay and other important objectives. One manifestation of this is in buffer planning. Since buffer insertion [6] is becoming a major technique for interconnect performance optimization and numerous buffers are projected for future designs [15], buffer insertion greatly affects

global routing. In order to ensure that there is no conflict between positions where buffer insertion is required and the locations of other placed cells, since spaces for buffer insertion are limited, the global routing must let the wires pass through the buffer-allowed regions so that buffer insertion optimization can be feasibly performed. Recently, buffer planning works [20,27,74,80] have been reported, and buffer congestion and wire congestion are simultaneously handled in [4].

References

[1] L. C. Abel. On the ordering of connections for automatic wire routing. *IEEE Transactions on Computers*, G-21(11):1227–1233, November 1972.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice Hall, Upper Saddle River, NJ, 1993.

[3] C. Albrecht. Provably good global routing by a new approximation algorithm for multicommodity flow. In *Proceedings of the ACM International Symposium on Physical Design*, pages 19–25, 2000.

[4] C. J. Alpert, J. Hu, S. S. Sapatnekar, and P. G. Villarrubia. A practical methodology for early buffer and wire resource allocation. to appear in *Proceeings of the ACM/IEEE Design Automation Conference*, 2001.

[5] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng, and D. Karger. Prim-dijkstra tradeoffs for improved performance-driven routing tree design. *IEEE Transactions on Computer-Aided Design*, 14(7):890–896, July 1995.

[6] H. B. Bakoglu. *Circuits, interconnections and packaging for VLSI.* Addison-Wesley, Reading, MA, 1990.

[7] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. Near-optimal critical sink routing tree constructions. *IEEE Transactions on Computer-Aided Design*, 14(12):1417–36, December 1995.

[8] M. Burstein and R. Pelavin. Hierarchical wire routing. *IEEE Transactions on Computer-Aided Design*, CAD-2(4):223–234, October 1983.

[9] R. C. Carden, J. Li, and C.-K. Cheng. A global router with a theoretical bound on the optimal solution. *IEEE Transactions on Computer-Aided Design*, 15(2):208–216, February 1996.

[10] Y.-A. Chen, Y.-L. Lin, and Y.-C. Hsu. A new global router for ASIC design based on simulated evolution. In *International Symposium on VLSI Technology, Systems and Applications*, pages 261–265, 1989.

[11] C.-K. Cheng, J. Lillis, S. Lin, and N. Chang. *Interconnect analysis and synthesis*. Wiley Interscience, New York, NY, 2000.

[12] C. Chiang and M. Sarrafzadeh. Global routing based on Steiner min-max trees. *IEEE Transactions on Computer-Aided Design*, 9(12):1318–25, December 1990.

[13] C. Chiang, C. K. Wong, and M. Sarrafzadeh. A weighted Steiner tree-based global router with simultaneous length and density minimization. *IEEE Transactions on Computer-Aided Design*, 13(12):1461–1469, December 1994.

[14] J. D. Cho and M. Sarrafzadeh. Four-bend top-down global routing. *IEEE Transactions on Computer-Aided Design*, 17(9):793–802, September 1998.

[15] J. Cong. Challenges and opportunities for design innovations in nanometer technologies. SRC Design Sciences Concept Paper, 1997.

[16] J. Cong, L. He, C.-K. Koh, and P. H. Madden. Performance optimization of VLSI interconnect layout. *Integration: the VLSI Journal*, 21:1–94, 1996.

[17] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong. Provably good performance-driven global routing. *IEEE Transactions on Computer-Aided Design*, 11(6):739–752, June 1992.

[18] J. Cong and C. K. Koh. Simultaneous driver and wire sizing for performance and power optimization. *IEEE Transactions on VLSI Systems*, 2(4):408–425, December 1994.

[19] J. Cong and C. K. Koh. Interconnect layout optimization under higher-order RLC model. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 713–720, 1997.

[20] J. Cong, T. Kong, and D. Z. Pan. Buffer block planning for interconnect-driven floorplanning. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 358–363, 1999.

[21] J. Cong and K.-S. Leung. Optimal wiresizing under the distributed Elmore delay model. *IEEE Transactions on Computer-Aided Design*, 14(3):321–336, March 1995.

[22] J. Cong, K.-S. Leung, and D. Zhou. Performance driven interconnect design based on distributed RC delay model. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 606–611, 1993.

[23] J. Cong and P. H. Madden. Performance driven global routing for standard cell design. In *Proceedings of the ACM International Symposium on Physical Design*, pages 73–80, 1997.

[24] J. Cong and B. Preas. A new algorithm for standard cell global routing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 176–179, 1988.

[25] J. Cong and B. Preas. A new algorithm for standard cell global routing. *Integration: the VLSI Journal*, 14(1):49–65, 1992.

[26] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[27] F. F. Dragan, A. B. Kahng, I. Mandoiu, and S. Muddu. Provably good global buffering using an available buffer block plan. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 104–109, 2000.

[28] H. Esbensen. A macro-cell global router based on two genetic algorithms. In *Proceedings of the European Design Automation Conference*, pages 428–433, 1994.

[29] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of Symposium on Foundations of Computer Science*, pages 300–309, 1998.

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

[31] N. Hasan and C. L. Liu. A force-directed global router. In *Advanced Research in VLSI. Proceedings of the 1987 Stanford Conference*, pages 135–150, Cambridge, MA, 1987. MIT Press.

[32] M. Hayashi and S. Tsukiyama. A hybrid hierarchical approach for multi-layer global routing. In *Proceedings of the European Design and Test Conference*, pages 492–496, 1995.

[33] J. Heisterman and T. Lengauer. The efficient solution of integer programs for hierarchical global routing. *IEEE Transactions on Computer-Aided Design*, 10(6):748–753, June 1991.

[34] D. W. Hightower. A solution to line routing problems on the continuous plane. In *The Sixth Design Automation Workshop*, pages 1–24, 1969.

[35] X. Hong, T. Xue, J. Huang, C.-K. Cheng, and E. S. Kuh. TIGER: an efficient timing-driven global router for gate array and standard cell layout design. *IEEE Transactions on Computer-Aided Design*, 16(11):1323–1331, November 1997.

[36] H. Hou, J. Hu, and S. S. Sapatnekar. Non-Hanan routing. *IEEE Transactions on Computer-Aided Design*, 18(4):436–444, April 1999.

[37] J. Hu and S. S. Sapatnekar. Algorithms for non-Hanan-based optimization for VLSI interconnect under a higher order AWE model. *IEEE Transactions on Computer-Aided Design*, 19(4):446–458, April 2000.

[38] J. Hu and S. S. Sapatnekar. A timing-constrained algorithm for simultaneous global routing of multiple nets. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 99–103, 2000.

[39] T. C. Hu and E. S. Kuh. Theory and concepts of circuit layout. In T. C. Hu and E. S. Kuh, editors, *VLSI Circuit Layout: Theory and Design*, pages 3–18. IEEE Press, New York, NY, 1985.

[40] T. C. Hu and E. S. Kuh, editors. *VLSI Circuit Layout: Theory and Design*. IEEE Press, New York, NY, 1985.

[41] T. C. Hu and M. T. Shing. A decomposition algorithm for circuit routing. In T. C. Hu and E. S. Kuh, editors, *VLSI Circuit Layout: Theory and Design*, pages 144–152. IEEE Press, New York, NY, 1985.

[42] J. Huang, X.-L. Hong, C.-K. Cheng, and E. S. Kuh. An efficient timing-driven global routing algorithm. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 596–600, 1993.

[43] S.-W. Hur, A. Jagannathan, and J. Lillis. Timing driven maze routing. In *Proceedings of the ACM International Symposium on Physical Design*, pages 208–213, 1999.

[44] S.-W. Hur, A. Jagannathan, and J. Lillis. Timing-driven maze routing. *IEEE Transactions on Computer-Aided Design*, 19(2):234–241, February 2000.

[45] A. B. Kahng and G. Robins. *On optimal interconnections for VLSI*. Kluwer Academic Publishers, Boston, MA, 1995.

[46] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[47] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[48] T. Koide, T. Suzuki, S. Wakabayashi, and N. Yoshida. An efficient timing-driven global routing method for standard cell layout. *IEICE Transactions on Information and Systems*, E79-D(10):1410–1418, October 1996.

[49] M. R. Kramer and J. van Leeuwen. The complexity of wire routing and finding minimum area layouts for arbitrary VLSI circuits. In F. P. Preparata, editor, *Advances in computing research*, volume 2: VLSI theory, pages 129–146. JAI, Reading, MA, 1984.

[50] E. S. Kuh and M. Marek-Sadowska. Global routing. In T. Ohtsuki, editor, *Layout design and verification*, pages 169–198. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1986.

[51] M. Kuribayashi, M. Yamada, T. Mitsuhashi, and N. Goto. A hierarchical global router for macro-block-embeded sea-of-gates. *IEICE Transactions Fundamentals*, E76-A(10):1694–1704, October 1993.

[52] L. R. Ford, Jr. and D. R. Fulkerson. *Flows in networks*. Princeton University Press, Princeton, NJ, 1962.

[53] U. P. Lauther. Top down hierarchical global routing for channelless gate arrays based on linear assignment. In *Proceedings of the IFIP International Conference on VLSI*, pages 141–151, 1987.

[54] C. Y. Lee. An algorithm for path connection and its applications. *IRE Transactions on Electronic Computers*, EC-10(3):346–365, 1961.

[55] K. W. Lee and C. Sechen. A global router for sea-of-gate circuits. In *Proceedings of the European Design Automation Conference*, pages 242–247, 1991.

[56] T. Lengauer. *Combinatorial algorithms for integrated circuit layout.* John Wiely and Sons, New York, NY, 1990.

[57] J.-T. Li and M. Marek-Sadowska. Global routing for gate array. *IEEE Transactions on Computer-Aided Design*, CAD-3(4):298–307, October 1984.

[58] J. Lillis, C. K. Cheng, T. T. Lin, and C. Y. Ho. New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 395–400, 1996.

[59] Y.-L. Lin, Y.-C. Hsu, and F.-S. Tsai. Hybrid routing. *IEEE Transactions on Computer-Aided Design*, 9(2):151–157, February 1990.

[60] L. E. Liu and C. Sechen. Multilayer chip-level global routing using an efficient graph-based Steiner tree heuristic. *IEEE Transactions on Computer-Aided Design*, 18(10):1442–1451, October 1999.

[61] D. G. Luenberger. *Linear and nonlinear programming.* Addison-Wesley Publishing Company, Reading, MA, 1984.

[62] W. K. Luk, P. Sipala, M. Tamminen, D. Tang, L. Woo, and C. K. Wong. A hierarchical global wiring algorithm for custom chip design. *IEEE Transactions on Computer-Aided Design*, CAD-6(4):518–533, July 1987.

[63] M. Marek-Sadowska. Global router for gate array. In *Proceedings of the IEEE International Conference on Computer Design*, pages 332–337, 1984.

[64] M. Marek-Sadowska. Route planner for custom chip design. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 246–249, 1986.

[65] G. Meixner and U. Lauther. A new global router based on a flow model and linear assignment. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 44–47, 1990.

[66] E. F. Moore. Shortest path through a maze. In *Annals of Computation Laboratory*, pages 285–292. Harvard University Press, Cambrige, MA, 1959.

[67] R. Nair. A simple yet effective technique for global wiring. *IEEE Transactions on Computer-Aided Design*, CAD-6(2):165–172, October 1987.

[68] R. Nair, S. J. Hong, S. Liles, and R. Villani. Global wiring on a wire routing machine. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 224–231, 1982.

[69] T.-M. Parng and R.-S. Tsay. A new approach to sea-of-gates global routing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 52–55, 1989.

[70] A. M. Patel, N. L. Soong, and R. K. Korn. Hierarchical VLSI routing - an approximate routing procedure. *IEEE Transactions on Computer-Aided Design*, CAD-4(2):121–126, April 1985.

[71] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[72] P. Raghavan and C. D. Thompson. Multiterminal global routing: a deterministic approximation scheme. *Algorithmica*, 6:73–82, 1991.

[73] S. M. Sait and H. Youssef. *VLSI physical design automation: theory and practice*. McGraw-Hill, New York, NY, 1995.

[74] P. Sarkar, V. Sundararaman, and C.-K. Koh. Routability-driven repeater block planning for interconnect-centric floorplanning. In *Proceedings of the ACM International Symposium on Physical Design*, pages 186–191, 2000.

[75] M. Sarrafzadeh and C. K. Wong. *An introduction to VLSI physical design*. McGraw-Hill, New York, NY, 1996.

[76] C. Sechen and A. Sangiovanni-Vincentelli. TimberWolf3.2: a new standard cell placement and global routing package. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 432–439, 1986.

[77] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *Journal of the ACM*, 37(2):318–334, 1990.

[78] N. A. Sherwani. *Algorithms for VLSI physical design automation.* Kluwer Academic Publishers, Norwell, MA, 3rd edition, 1999.

[79] E. Shragowitz and S. Keel. A global router based on a multicommodity flow model. *Integration: the VLSI Journal*, 5(1):3–16, March 1987.

[80] X. Tang and D. F. Wong. Planning buffer locations by network flows. In *Proceedings of the ACM International Symposium on Physical Design*, pages 180–185, 2000.

[81] R. W. Thaik, N. Lek, and S.-M. Kang. A new global router using zero-one integer linear programming techniques for sea-of-gates and custom logic arrays. *IEEE Transactions on Computer-Aided Design*, 12(12):1479–1494, December 1992.

[82] B. S. Ting and B. N. Tien. Routing techniques for gate array. *IEEE Transactions on Computer-Aided Design*, CAD-2(4):301–312, October 1983.

[83] A. Vannelli. An adaptation of the interior point method for solving the global routing problem. *IEEE Transactions on Computer-Aided Design*, 10(2):193–203, February 1991.

[84] M. P. Vecchi and S. Kirkpatrick. Global wiring by simulated annealing. *IEEE Transactions on Computer-Aided Design*, CAD-2(4):215–222, October 1983.

[85] A. Vittal and M. Marek-Sadowska. Minimum delay interconnect design using alphabetic trees. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 392–396, 1994.

[86] D. Wang and E. S. Kuh. Performance-driven interconnect global routing. In *Proceedings of the Great Lake Symposium on VLSI*, pages 132–136, 1996.

[87] S. M. Wang. A multiple source algorithm for suboptimal steiner trees in graphs. In H. Noltemeier, editor, *The Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 387–396. 1985.

[88] K. D. Wayne and L. Fleischer. Fast and simple approximation schemes for generalized flow. In *Proceedings of the Tenth Annual ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 981–982, 1999.

[89] K. Winter and D. A. Mlynski. Hierarchical loose routing for gate arrays. *IEEE Transactions on Computer-Aided Design*, CAD-6(5):810–819, September 1987.

[90] T. Xue, E. S. Kuh, and D. Wang. Post global routing crosstalk risk estimation and reduction. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 302–309, 1996.

[91] H. Youssef and S. M. Sait. Timing-driven global routing for standard-cell VLSI design. *Computer Systems Science and Engineering*, 14(3):175–185, May 1999.

[92] H. Zhou and D. F. Wong. Global routing with crosstalk constraints. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 374–377, 1998.

[93] H. Zhou and D. F. Wong. Global routing with crosstalk constraints. *IEEE Transactions on Computer-Aided Design*, 18(11):1683–1688, 1999.