Industrial and
Systems Engineering

Fast and Effective Congestion Prediction Methods for ILP-based Global Routing

Logan Rakai
University of Calgary

Laleh Behjat
University of Calgary

Shawki Areibi
The University of Guelph

Tamás Terlaky
Lehigh University

Report: 08T-003

Fast and Effective Congestion Prediction Methods for ILP-based Global Routing*

ABSTRACT

Global routing is a fundamental step in physical design that determines the approximate pathways for circuit interconnect. Routing in nanometer nodes creates an elevated level of importance for low congestion routing. This paper presents several innovative methods for quickly predicting congestion in integer linear programming (ILP)-based global routing. Accurate congestion prediction before global routing can save valuable time in later stages. In this work, a new tree selection score, a new tree congestion measure, and a fast congestion map production method are presented for ILP-based routing. Results...

200 word limit

Hard-coded reference numbers in captions

could be used for placement feedback

1. INTRODUCTION

The routing phase of physical design decides exactly where the interconnect of a circuit will be located. This task is usually performed by a two phase flow consisting of a global routing phase followed by a detailed routing phase. The global routing phase of physical design determines the approximate pathways of the interconnect in a layout. As process technologies scale downward, the interconnect plays a more significant role in determining the performance of a circuit [11]. The particular importance of global routing has been acknowledged by the highly successful ISPD global routing contests in the past two years [1, 2].

Global routing is performed by overlaying a grid on the placement solution. From here, a graph abstraction is typically used, in which each grid cell becomes a vertex and each boundary between grid cells becomes an edge. The number of routing tracks that a grid cell encompasses determines the capacity of an edge. In modern designs, this capacity is further reduced by the presence of blockages, such as IP blocks. An approximate route for a net in global routing is a set of edges in the graph, or a tree, that forms a path between vertices containing pins in the net. If the number of trees using an edge, the demand for that edge, is near or above its capacity, it is said to be congested. An edge with demand in excess of its capacity is said to be overflowing. A routing solution is legal if all of the nets have a tree and none of the edges are overflowing.

Being able to relieve congestion is paramount in global routing. When edges are overflowing the layout cannot be fabricated. In addition, even if a solution is legal, congestion deteriorates signal quality and increases delay uncertainty [23]. A considerable amount of work has been done on avoiding congestion in global routing, such as [7, 15, 22, 29], and several methods exist for predicting congestion before beginning global routing [4,9,30,31,33]. It is a worthy goal to create congestion avoiding routes initially so that less routes require post-processing through rip-up and reroute or some other improvement scheme.

The first contribution presented in this paper is a new tree selection score calculation for ILP-based routing that behaves like a probability function. This prediction is used to propose a new edge demand prediction method for estimating congestion. The congestion estimation technique is used in a framework to drive additional tree generation for ILP-based routing. Then, a new quantity coined the *routing supply* is presented and shown to be superior to the routing demand quantity for avoiding overflow. In addition, a model for the global routing problem through ILPs that can be solved quickly and obtain 0-1 solutions is developed. The new model developed is then used as a way of quickly creating a congestion map for an entire circuit.

The rest of this paper is organized as follow. Relevant background information and motivating factors for this work are given in Section 2. In Section 3, score-based congestion prediction methods are proposed. In addition, a new ILP-based global routing tree selection prediction technique is proposed and used in an additional tree generation algorithm. In Section 4, a deterministic congestion map is proposed based on integer relaxation models which can be solved quickly by exploiting unimodularity. The effectiveness of the congestion maps is validated in this section. Finally, conclusions and future work are presented in Section 5.

2. BACKGROUND AND MOTIVATION

2.1 Global Routing

Sequential routing is a conventional method for perform-

9*This work is supported by NSERC, iCORE, and the Alberta Ingenuity Fund.

ISPD '09 San Diego, California USA

ing global routing. In this method, each net is routed one-by-one and is usually followed by an improvement scheme. The initial routes can be generated by any of a variety of methods, including pattern and maze routing. The quality of the initial solution is heavily dependent on the order in which the nets are routed so post-processing is required to remove that dependence. Rip-up and reroute is a popular scheme identifies problematic routes, removes them and attempts to reroute them in a better order. An iterative scheme, known as negotiation [24], provides a congestion-alleviation framework for performing rip-up and reroute. Negotiation is a part of several successful academic routers [8, 13, 25, 28].

Concurrent routing avoids the net ordering problem by routing all nets simultaneously using mathematical programming. ILP-based global routing is a classic example of concurrent routing that has been in existence for over two decades when the seminal work was performed by Burstein and Pelavin [5]. ILP-based global routers first generate a set of trees for each net that are candidates for the final route. A linear objective function is then selected based upon the desired goals for the routing solution. Then an ILP solver is used to determine which trees should be selected according to the objective while maintaining that the edge capacity constraints are not violated.

A technique that models the global routing as a multi-commodity flow problem allowing more than one tree to be fractionally selected for a net. The resulting problem is solved as a linear program (LP) followed by a rounding scheme to obtain an integer solution [3]. LPs can be solved more quickly than ILPs but the rounding causes the solution to be suboptimal. The historical criticism about ILP-based global routers being slow is becoming less valid as ILP solvers have improved. Because of these advances, several ILP-based global routers have been proposed recently. BoxRouter [8] is an ILP-based router that uses progressive box expansion to formulate a sequence of ILPs that incrementally route the circuit. BoxRouter was competitive in both the 2007 and 2008 ISPD routing contests. Another recent ILP-based global router is Sidewinder [19]. Sidewinder is a flat router that moderates size of the ILP by only selecting two candidate trees from a potentially very large group of initial candidates. ILP preprocessing techniques have also been shown to be effective at reducing runtimes without affecting solution quality [16].

2.2 Congestion Map Generation

Congestion maps are an important tool in guiding global routing. With an accurate congestion map, the global routing problem can be solved more intelligently because a priori knowledge of where overflow is most likely to occur is available. It is important for the congestion maps to be obtained quickly so that the entire process of global routing with a congestion map can be performed in a comparable amount of time to when no map is used. Several previous congestion map models in [21,23,31] relied entirely on probabilistic analysis. The method presented in [23] considers all detour-free routes assigning each an equal probability. Based on an observation that most routes are L- or Z-shaped, the method in [31] considers only these shapes for routes. The wire density estimation described in [21] uses detour-free one- to four-bend routes and an empirical formula for higher bend routes to approximate congestion with probabilities. The method also allows for detouring if detour-free regions are highly congested. These methods deal with blockages but probabilistic congestion is not the way to go for designs with many obstacles [31]. The underlying assumption of methods that assume equal probability has been noted as unrealistic when analyzing results of commercial routers on realistic circuits [28].

The main advantage of probabilistic methods for estimating congestion is that they are faster than performing global routing to get more accurate results. After calling the accuracy of probability-based methods into question in [32], a *degenerate* router, called FaDGloR, that quickly solves the global routing problem is proposed. This router uses A*-search to sequentially route the nets without performing rip-up and reroute. The algorithm is comparable in runtime to probabilistic methods. FastRoute 1.0 [26] provides less overflow in less time than FaDGloR by using edge shifting and a logistic cost maze routing.

The congestion map strategy that BoxRouter [8] uses is to first decompose each net into two-pin subnets using a Steiner tree generated by FLUTE [10]. Next it solves an ILP that has only L-shaped routes for each net with the objective of maximizing the number of nets routed. This L-shaped only ILP is the same initial congestion map strategy that is used in Sidewinder [19]. However, the congestion map is dynamically updated between ILP iterations as new candidate trees are selected. This updating is probability-based and contributes to Sidewinder's significant runtime. FastRoute [27] also uses an L-shaped only initial congestion map formulation but rather than solving an ILP, it routes sequentially. Maizerouter [25] uses the same method. NTHU-Route [13] generates a congestion map by with L-shapes as well but uses a probabilistic argument initially. Each tree is selected and has a demand of 0.5 for each of its edges, unless the two pins were collinear and only a straight, no-bend tree is produced and has a demand of 1 for each edge. It then performs edge shifting to improve the tree topology based on the initial map. The double-via insertion router presented in [6] uses a probabilistic congestion model to map the congestion. It generates all possible L- and Z-shaped tree for two-pin subnets produced by a minimum spanning tree (MST) and assumes each has an equal probability of being selected. Each tree contributes a demand equal to the inverse of the total number of L- and Z-shaped trees to the edges in the tree. FGR [28] decomposes nets with FLUTE or via an MST method and then sequentially performs an initial routing with an exponential overflow penalty rather than using only minimum length routes.

These probability-based methods, as well as other techniques [9], require the nets to be broken into two-pin subnets to reduce complexity in order to reduce runtime. This may result in inaccuracies if the global router using the map does not use the same subnet decomposition or does not decompose the nets at all.

can handle obstacles by adjusting d_{grntd} [4] put in routers part could be used for non-ILP-based routers since no solver is required

However, most of these algorithms focus exclusively on two-pin subnets and the small region within the bounding box of the two pins. This strategy is not suitable for ILP-based global routing because only a small amount of trees are generated for each net and not every possible tree needs to be considered. The existing method that considers full nets and not two-pin subnets for congestion prediction

uses a simple probability argument to estimate the congestion [4]. Probability-based congestion techniques have been criticized because of their inaccuracy [32]. This paper develops fast and accurate methods for estimating congestion in ILP-based routing.

The importance of removing overflow in global routing is paramount. Being able to predict where congested areas are likely to occur before performing routing allows a router to attempt to detour routes around areas. This can allow the router to avoid overflowing edges early on in the routing and converge to an overflow-free solution more quickly. For ILP-based routing, this can reduce the number of trees that need to be generated. Reducing the number of trees generated is beneficial because they are time consuming to generate and also increase the time required to solve the ILP. The techniques proposed in this paper are able to:

- Provide accurate congestion information
- Have minimal impact on total runtime

3. SCORE-BASED CONGESTION PREDICTION METHODS

3.1 A New Tree Selection Score Calculation

This section develops a congestion-based score calculation to express the likelihood that a tree will be selected by an ILP-based global router. These scores are used to predict edge demands or edge supplies later in the routing demand prediction method or routing supply quantity technique, so that additional trees can be constructed for congested edges. The tree construction algorithm used in this paper is based on the algorithm in [33] which uses the congestion prediction technique in [4]. In this section, an important improvement in the congestion prediction method in [4] is proposed. The flaw in the previous approach for a tree generation application did not consider a net with only one tree generated for itself. In the previous approach, the probability of each tree of a net to be chosen is assigned equally. In some case, the predicted demand for a edge is equal to one because of the sum of probability for some trees, but the edge is not certainly taken by a tree. A new tree selection score, which acts like a probability, proposed here is to remedy this problem by explicitly recognizing when a net has only one tree generated for itself.

To appreciate the amount of situations stated previously, placement data for the ISPD98 benchmarks is presented in Table 1. In Column 3 of this table reports the number of collinear, degree two nets in each circuit is reported. The percentage of two-pin nets that are collinear is tabulated in Column 4. On average, 26% of the nets in the ISPD98 benchmarks are collinear nets of degree two and therefore only have one route. The number of nets that receive only one tree using [33] will be higher than this. This is because nets with degree higher than three only receive one tree. The vast majority of nets in the benchmarks are of degree two or three but higher degree nets tend to need more routing resources as they have more pins to connect. Columns 5 and 6 contain data for the average amount of guaranteed edge demand per edge and the percentage of the edge capacity that is used up by guaranteed demand. Nets with only one tree account for 46.6% of the edge capacity on average cutting down scarce resources by almost half.

Table 1: Percentage of degree two collinear nets and nets with only one tree using in the ISPD98 global routing benchmark suite

Circuit	Nets	Collinear nets	% nets collinear	Average $d_{\text{grntd}}(\cdot)$	% capacity guaranteed
ibm01	11507	2966	25.8	7.2	51.7
ibm02	19291	4691	24.3	16.0	47.1
ibm03	21621	5644	26.1	14.0	46.8
ibm04	26163	7330	28.0	13.1	56.9
ibm05	27777	8001	28.8	24.7	39.3
ibm06	33354	9456	28.4	16.7	50.5
ibm07	44394	11032	24.9	14.6	40.7
ibm08	47944	11556	24.1	16.3	50.8
ibm09	50393	14227	28.2	12.4	44.4
ibm10	64227	14071	21.9	17.4	43.4
Avg.	-	-	26.3	-	46.6

To remedy the tree selection prediction flaw it is proposed to first generate the trees for all of the nets known to only produce one route, which in the algorithm used in this paper are collinear two-pin nets and nets of degree four and higher. A minimum length tree for a collinear, two-pin net has zero bends, whereas any alternate tree has at least two. The reason for generating only the minimum length tree for collinear, two-pin nets is to promote reduction in the total number of bends and total wirelength in the solution. Nets of degree four or higher may require much more computation to determine high quality trees than a degree two or three net. To promote fast solutions, only one tree is generated for nets of degree four or higher.

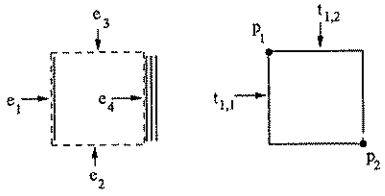
Once these trees have been routed the predicted congestions of the routing grid edges are incremented by one for each tree that passes through them since the trees routed are guaranteed to be using those edge resources. The total guaranteed demand for an edge is given by the $d_{\text{grntd}}(\cdot)$. With the deterministic routes factored in, trees for the remaining nets are generated and a new technique to calculate the tree selection score is proposed. Since it is preferable for a route to avoid congested areas, the new score calculation is constructed in a way that reflects this notion. So instead of the simplistic equal probability underlying the method in [33], the score for choosing a tree $t_{i,j}$ for a net n_i with T_i total trees is proposed to be calculated as:

$$score(t_{i,j}) = \frac{\dfrac{1}{\max\limits_{e_k \in t_{i,j}} \left(d_{\text{grntd}}(e_k)\right) + \frac{1}{|T_i|}}}{\sum\limits_{q=1}^{|T_i|} \left(\dfrac{1}{\max\limits_{e_k \in t_{i,q}} \left(d_{\text{grntd}}(e_k)\right) + \frac{1}{|T_i|}} \right)} \quad (1)$$

To explain the equation, first consider the terms in the denominator of the numerator: $\max\limits_{e_k \in t_{i,j}} \left(d_{\text{grntd}}(e_k)\right) + \frac{1}{|T_i|}$. The first term is the maximum guaranteed demand for an edge in a tree. The maximum guaranteed demand is the best congestion that a tree can hope for since it is guaranteed to be at least this high. The second term is the probability of selecting a tree when each tree is equally likely to be chosen. It is the same for all trees generated for a net. The higher the maximum guaranteed demand is for a tree the higher

the chance that it will contain a heavily congested edge in the final solution since at least one edge has high demand before routing even begins. To account for this undesirability in the probability function, the inverse of the sum of the two terms is taken. This makes trees with higher maximum guaranteed demand have a smaller probability of being chosen. The second term ensures that division by zero never occurs. This inverted term is divided by the sum of all of the inverted terms for the trees generated for a net to make the scores sum up to one. It should be noted that the tree selection probability function used in [33] is a special case of (1) when all trees have a maximum guaranteed demand of zero.

An example of how the new tree selection score is calculated is shown in Figure 1. In this example there are



$$c(e_1) = 4 \qquad d_{\text{grntd}}(e_1) = 1$$
$$c(e_2) = 4 \qquad d_{\text{grntd}}(e_2) = 0$$
$$c(e_3) = 4 \qquad d_{\text{grntd}}(e_3) = 0$$
$$c(e_4) = 4 \qquad d_{\text{grntd}}(e_4) = 3$$
$$\max_{e_k \in t_{1,1}} \left(d_{\text{grntd}}(e_k)\right) = d_{\text{grntd}}(e_1) = 1$$
$$\max_{e_k \in t_{1,2}} \left(d_{\text{grntd}}(e_k)\right) = d_{\text{grntd}}(e_4) = 3$$

Figure 1: An example for calculating the new tree selection score. Each solid lines in the left diagram represents one unit of guaranteed demand for the edge it runs parallel to.

two trees for net $n_1 = \{p_1, p_2\}$ on the right routing grid, $t_{1,1} = \{e_1, e_2\}$ and $t_{1,2} = \{e_3, e_4\}$ where the edges are labeled on the left routing grid. The capacity of all of the edges is assumed to be 4 and the guaranteed demands for each are tabulated below the diagrams in the figure. The guaranteed demand for each edge is calculated by summing the number of trees that are guaranteed to contain an edge. Each unit of guaranteed demand is shown as a solid line parallel to an edge in the left diagram in 1. The maximum guaranteed demand for an edge in each tree is also tabulated. By looking at these guaranteed demands, it can be seen that e_4 is very close to its capacity, making $t_{1,2}$ less favorable for n_1. Using (1) to find the score for $t_{1,1}$ being selected yields $score(t_{1,1}) = \frac{7}{10}$. The score for $t_{1,2}$ is similarly found to be $\frac{3}{10}$. These tree selection scores reflect the preference of a router to choose a route that is less likely to create overflow.

3.1.1 *A New Congestion Prediction Tree Generation with Routing Demand Quantity*

this subsection might be taken out. Or just routing demand quantity can be taken out. With these enhanced tree selection scores calculated, the predicted demand for an edge is calculated by summing the guaranteed demand plus all of the tree selection scores for trees containing the edge as in

$$d_{\text{pre}}(e_k) = d_{\text{grntd}}(e_k) + \sum_{t_{i,j} \in e_k,\ |T_i|>1} score(t_{i,j}). \quad (2)$$

This formula is used for forecasting which edges will be overflowing or with demands near their capacity so new trees can be generated if necessary.

The accuracy of the new predicted edge demands is verified in Table 2. The setup for this experiment was to perform global routing of the ISPD98 benchmarks using the edge capacity model (ECM) in [33] with one iteration of additional tree generation. This model is made to minimize the maximum edge demand and is a standard way to perform routing to reduce congestion. Then the actual edge demand and predicted edge demand using the method in [4] and the proposed method are calculated. The second column in Table 2 shows the actual average edge demand in the solution. The third and fourth columns list the average and standard deviation in the absolute value of the error from using the method in [4]. The average is reported as a percentage of the actual demand. The average and standard deviation of the error from using the proposed method are shown in Columns 5 and 6. The proposed method provides a more

Table 2: The proposed edge demand prediction results on the ISPD98 benchmark suite compared to the method in [1].

Circuit	Avg $d(\cdot)$	$d_{\text{pre}}(\cdot)$ by [4] avg % err	$d_{\text{pre}}(\cdot)$ by [4] stddev	Proposed $d_{\text{pre}}(\cdot)$ avg % err	Proposed $d_{\text{pre}}(\cdot)$ stddev
ibm01	7.5	15.2	1.3	11.9	1.0
ibm02	16.6	14.0	2.5	11.4	2.1
ibm03	14.6	16.7	2.7	15.3	2.5
ibm04	13.5	19.5	2.7	16.9	2.4
ibm05	25.5	11.4	3.0	8.7	2.3
ibm06	17.2	11.5	2.1	8.8	1.6
ibm07	15.0	16.1	2.7	14.2	2.4
ibm08	16.7	12.9	2.2	10.2	1.7
ibm09	12.8	16.2	2.5	15.2	2.3
ibm10	17.8	14.9	2.9	13.2	2.5
Avg.	-	14.8	2.5	12.6	2.1

accurate and precise estimate of the actual demand. The estimates appear to be quite accurate, 14.8% average error for the method in [4] on average for all of the benchmarks and 12.6% for the proposed method. The 2.2% improvement is more significant when it is considered that nearly 50% of the nets in this experiment have only one tree and their demand would be predicted exactly.

Based on the proposed predicted demand, $d_{\text{pre}}(\cdot)$, in (2) a new framework for additional tree generation with new objective in ILP based global routing is proposed. In traditional ILP formulation of global routing, the objective has been to minimize the total wirelength or the maximum edge demand. In [4], a novel ILP objective is proposed. In this model, each tree in a global routing problem is associated with a *routing demand*, $rd(\cdot)$, value that is related to the total congestion encountered by this tree. The routing demand for a tree $t_{i,j}$ is calculated as the sum of the predicted edge demands of the edges that the tree contains [4]. In this

model

$$rd(t_{i,j}) = \sum_{e_k \in t_{i,j}} d_{\text{pre}}(e_k).$$

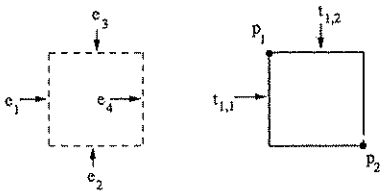
This calculation tries to foretell the amount of congestion that the tree will be a part of. In addition, the enhanced edge demand prediction method is able to more accurately guide the decision of where additional trees need to be constructed. The entire edge congestion prediction tree generation algorithm with ILP is summarized in Figure 2.

1. Generate the initial set of trees as in [33]
2. Calculate guaranteed demands for edges
3. Calculate tree selection scores with (1)
4. Determine the predicted edge demands using (2)
5. Generate additional trees for nets with trees containing edges predicted to be highly congested
6. Solving ILP problem with the objective of minimizing routing demand

Figure 2: The proposed congestion prediction-based tree generation framework.

3.2 The Routing Supply Quantity

In the previous routing demand quantity, even with an accurate edge demand prediction there is a problem.This can be seen in the example provided in Figure 3. The capacity



$c(e_1) = 4$ $d_{\text{pre}}(e_1) = 1$
$c(e_2) = 4$ $d_{\text{pre}}(e_2) = 5$
$c(e_3) = 4$ $d_{\text{pre}}(e_3) = 4$
$c(e_4) = 4$ $d_{\text{pre}}(e_4) = 3$

$rd(t_{1,1}) = 6$

$rd(t_{1,2}) = 7$

Figure 3: An example depicting the problem with the routing demand metric for use in an objective function.

of all of the edges is 4 and the predicted demand for each is given in the figure along with the routing demand for each tree. On the right, there are two trees for the net and the routed demand for each is tabulated. $t_{1,1}$ has a lower routing demand of 6 compared to $t_{1,2}$ which has a routing demand of 7. At first glance it may appear that $t_{1,1}$ is better suited for reducing circuit congestion. This may be true in some instances but not here because e_2 is actually predicted to be overflowing. The goal of minimizing overflow should take priority over all other objectives.

To avoid this shortcoming the converse of the routing demand calculation that will be referred to as the *routing supply*, $rs(\cdot)$, is proposed. The routing supply measures how much available capacity is in the edges of a tree based on the predicted edge demand. If there is an edge that is predicted to have no available capacity, a penalty is incurred. The routing supply for a tree $t_{i,j}$ is calculated through the use of the following formula

$$rs(t_{i,j}) = \sum_{e_k \in t_{i,j}} ((c(e_k) - d_{\text{pre}}(e_k)) * (1\,[c(e_k) - d_{\text{pre}}(e_k) \geq 0] + \gamma_{pnlty}\,[c(e_k) - d_{\text{pre}}(e_k) < 0])) \quad (3)$$

where $c(e_k)$ is the capacity of edge e_k, $d_{\text{predicted}}(e_k)$ is edge e_k's predicted demand, $\gamma_{pnlty} > 1$ is a shortage penalty for when the predicted demand exceeds the capacity, or "supply", and the brackets are a notational convenience adopted from [14]. The notation convention works as follows

$$[\text{true or false condition}] = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{if condition is false} \end{cases}$$

The two conditions in (3) are $c(e_k) - d_{\text{pre}}(e_k) \geq 0$, which is true if there is capacity available, and $c(e_k) - d_{\text{pre}}(e_k) < 0$, which is true when there is no capacity available. Note that the conditions are mutually exclusive so one will always evaluate to 1 and the other will evaluate to 0. These two conditions determine the coefficient multiplying the supply term $c(e_k) - d_{\text{pre}}(e_k)$. If there is available supply, the coefficient is unity, and if there is no supply, the shortage penalty will multiply the demand in excess of the capacity. The shortage penalty should be large and comparable to the edge capacity to significantly amplify the edges with overflow. An example of the routing supply of an edge is given in Figure 4. The

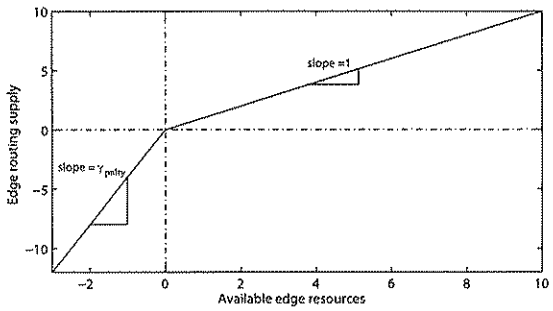


Figure 4: The routing supply of an edge. The routing supply for a tree is the sum of the routing supplies for the edges in the tree.

largest routing supply amount is when there is no demand for the edge, or when all of the edge resources are available. The slope in the region where there are routing resources available is one. When there is an excess of demand, meaning there is a shortage of available resources, the slope is γ_{pnlty}. The shortage penalty is markedly steeper since it is preferable to use several at or near capacity edges versus an edge that has an overflow of even just one.

By using the Figure 3 again, an example showing how the routing supply calculation works to avoid overflow where routing demand did not is illustrated. To calculate the rout-

ing supply for each tree (3) is used.

$$\begin{aligned} & rs(t_{1,1}) \\ = \; & (c(e_1) - d_{\text{pre}}(e_1))\,(1\,[c(e_1) - d_{\text{pre}}(e_1) \geq 0] + \\ & \qquad \gamma_{pnlty}\,[c(e_1) - d_{\text{pre}}(e_1) < 0]) \; + \\ & (c(e_2) - d_{\text{pre}}(e_2))\,(1\,[c(e_2) - d_{\text{pre}}(e_2) \geq 0] + \\ & \qquad \gamma_{pnlty}\,[c(e_2) - d_{\text{pre}}(e_2) < 0]) \\ = \; & (4-1)\,(1\,[4-1 \geq 0] + \gamma_{pnlty}\,[4-1 < 0]) \; + \\ & (4-5)\,(1\,[4-5 \geq 0] + \gamma_{pnlty}\,[4-5 < 0]) \\ = \; & 3 \; - \; \gamma_{pnlty} \end{aligned}$$

Through a similar calculation $rs(t_{1,2})$ is found to be equal to 1. In this example, if the shortage penalty is greater than 2 then the preference towards the tree not involved in overflow, $t_{1,2}$, is reflected in the calculations.

3.2.1 *Comparison between Routing Supply and Routing Demand*

To provide validation to the new routing supply quantity this section compares two global routing formulations: one with the routing demand objective and the other with the routing supply. The routing demand model formulation is as follows

$$\begin{array}{lll} \text{minimize} & \sum_{\forall i,j} rd(t_{i,j}) x_{i,j} & \\ \text{subject to} & \sum_{t_{i,j} \in T_i} x_{i,j} = 1, & \forall n_i \in N \\ & \sum_{\{i,j | e_k \in t_{i,j}\}} x_{i,j} \leq cap(e_k), & \forall e_k \in E \\ & 0 \leq x_{i,j} \leq 1, \qquad \forall n_i \in N, & \forall t_{i,j} \in T_i. \end{array} \tag{4}$$

Since the routing supply increases if the tree contains predictably uncongested edges and reduces when it contains edges predicted to be overflowing, the negative of routing supply is suitable for the routing supply model minimization. Therefore, the routing supply model is as follows

$$\begin{array}{lll} \text{minimize} & \sum_{\forall i,j} -rs(t_{i,j}) x_{i,j} & \\ \text{subject to} & \sum_{t_{i,j} \in T_i} x_{i,j} = 1, & \forall n_i \in N \\ & \sum_{\{i,j | e_k \in t_{i,j}\}} x_{i,j} \leq cap(e_k), & \forall e_k \in E \\ & 0 \leq x_{i,j} \leq 1, \qquad \forall n_i \in N, & \forall t_{i,j} \in T_i. \end{array} \tag{5}$$

In Table 3 the data for each model solved by using a linear relaxation and post-processed with a greedy round-up-the-largest scheme to obtain integer solutions is presented. The capacity constraints are relaxed enough to allow all of the nets to be routed so there is some overflow. In Table 3, Columns 2-5 report results from solving the routing demand (RD) model in (4) and Columns 6-9 report the percentage difference between the routing supply (RS) model in (5) and the routing demand model. The percentage difference values for the routing supply results are calculated using the following equation

$$\Delta_{\%} = \left(\frac{(\text{RS value}) - (\text{RD value})}{\text{RD value}} \right) \times 100\%$$

where $\Delta_{\%}$ is the percentage change. The second column, labeled "WL", lists the wirelength. In Column 3 and 4 the total number of bends in the routes and the total overflow are tabulated. Column 5 displays the total time taken to solve the problem. The last four columns report the same results for the RS model but in percentage compared to the RD model. The average percentage changes are shown in bottom row. The wirelength and number of bends increase by 0.7% and 3.4%, respectively, for the RS model but the total overflow decreases 13.7% on average. This verifies that the routing supply model is better able to reduce the amount of overflow given the same set of trees as the routing demand model. About 50% of the nets in the benchmarks only have one tree generated and would be chosen by both models. This means that the reduction in overflow is quite good given that many of the nets only have one tree to choose from.

Table 3: Percentage comparison between the routing supply and routing demand models using the ISPD98 global routing benchmark suite

	RD Model				RS Model % Comparison			
Circuit	WL	Bends	Total OV	Total TM(s)	WL (%)	Bends (%)	Total OV(%)	Total TM(%)
ibm01	60415	12962	2372	8	2.3	9.6	1.9	12.5
ibm02	166347	28039	5535	23	0.5	2.5	-6.7	0.0
ibm03	145927	22536	3193	23	0.7	3.6	1.8	0.0
ibm04	163069	26249	5623	40	0.9	4.4	-5.9	0.0
ibm05	410085	50767	133	33	0.0	0.2	-88.0	3.0
ibm06	276473	41533	5194	48	0.8	4.7	-4.8	0.0
ibm07	363836	54675	5170	100	0.2	1.2	-3.4	1.0
ibm08	403883	68447	5417	127	0.5	2.6	-5.0	7.9
ibm09	412157	62735	7840	219	0.7	3.9	-10.8	2.7
ibm10	575044	92620	8582	237	0.3	1.3	-16.0	4.6
Avg.	-	-	-	-	0.7	3.4	-13.7	3.2

4. INTEGER RELAXATION CONGESTION MAPS

The proposed strategy for producing a congestion map in this section is to use an ILP to obtain a solution with or without decomposing nets into subnets. If the router intending to use the congestion maps decomposes nets then the nets should be decomposed before generating the map. However, unlike other methods, if the router does not decompose nets the developed model will work just as well for it since it can handle either case. In order to avoid runtimes significantly higher than previous methods, the models are formulated as integer relaxations. This allows for entire nets to be routed instead of only two-pin subnets which is more suitable for routers that do not use two-pin subnet decompositions.

4.1 Proposed Integer Relaxation Model

In this section, developed integer relaxation model is for quick production of accurate congestion maps. The class of problems proposed will be used in following sections where their favorable characteristics are applied in a congestion-related application.

4.1.1 *Unimodular Integer Linear Programs*

An important property of linear programs is that a vertex in the constraint polyhedron will yield the optimal value. This was the fundamental observation that lead to the development of the Simplex algorithm for solving LPs. This idea has further ramifications for ILPs and will be applied to global routing ILP formulations in the following section.

Definition 4.1. Unimodularity - A square matrix with integer entries is unimodular if its determinant is 1, -1, or 0.

An immediate consequence of this is that for systems $\mathbf{Ux} = \mathbf{b}$ with $\mathbf{b}$ made up of integers and $\mathbf{U}$ is unimodu-

lar then if $det(\mathbf{U}) \neq 0$ the solution for $\mathbf{x}$ will also be integer. This follows from Cramer's Rule for obtaining the solution to linear systems using determinants [12].

Definition 4.2. Total unimodularity - A rectangular matrix with integer entries is called totally unimodular if all of its square submatrices are unimodular.

Theorem 4.3. (A class of unimodular matrices) *Suppose a matrix has only 1, -1, or 0 for any of its entries and each column has at most two non-zero entries. The matrix is totally unimodular if and only if the rows can be partitioned into two sets such that if a column has two non-zero elements with different signs their rows are in the same subset and if they have the same sign they are in different subsets [17].*

Consider a general LP where the variables are bounded by integers ω, υ and $\mathbf{b} \in Z^{m \times 1}$

$$\begin{array}{lll} \text{minimize} & \mathbf{c}'\mathbf{x} & \\ \text{subject to} & \mathbf{U}\mathbf{x} = \mathbf{b} & \\ & \omega \leq \mathbf{x} \leq \upsilon, & \end{array} \tag{6}$$

The matrix $\mathbf{U}$ has the properties of a totally unimodular matrix so the optimal solution is guaranteed to be made up entirely of integers. In fact, this is the only case when the solution of the LP problem will be integer as proven in a famous theorem by Hoffman and Kruskal [18].

4.1.2 Integer Relaxation Model

A global routing model proposed in this section uses unimodularity to solve the global routing problem. The main idea is to formulate the ILP such that the constraint matrix is totally unimodular. This modeling will result in the optimal LP solution, $\mathbf{x}^*$, of the relaxed unimodular ILP to contain only zeros and ones. This eliminates the need to use B&B or rounding to obtain integer solutions.

In this section, the edge capacity constraints is relaxed to obtain what will be called the integer relaxation. When the edge capacity constraints are removed the remaining exclusivity constraints $\left(\sum_{t_{i,j} \in T_i} x_{i,j} = 1,\ \forall n_i \in N\right)$ give the constraint matrix as the totally unimodular matrix, which can be verified by using Theorem 4.3 which holds since each column has only a single non-zero entry. Being totally unimodular means that the integrality constraints can be relaxed without any consequence.

This gives rise to the proposed integer relaxation of the global routing problem

$$\begin{array}{lll} \text{(Integer Relaxation)} & \text{minimize} & \mathbf{c}'\mathbf{x} \\ & \text{subject to} & \mathbf{U}\mathbf{x} = \mathbf{1}, \\ & & \mathbf{0} \leq \mathbf{x}. \end{array} \tag{7}$$

The $\mathbf{x} \leq \mathbf{1}$ constraint can be removed because it is implied by the two remaining constraints. In addition to providing binary solutions, this model can be evaluated more quickly than other models because of the reduced number of constraints. The drawback is that the capacity constraints are not considered and overflow can occur. In Section 4 an application of how the lack of capacity constraints can be advantageous is presented. It should be mentioned that the proposed integer relaxed model is solved by using Simplex in CPLEX 9.0 library [20].

4.1.3 Wirelength Integer Relaxation Congestion Maps

The first proposed integer relaxation model is a wirelength-based model known as the wirelength integer relaxation (WIR):

$$\begin{array}{lll} \text{(WIR)} & \text{minimize} & \mathbf{l}'\mathbf{x} \\ & \text{subject to} & \mathbf{U}\mathbf{x} = \mathbf{1}, \\ & & \mathbf{0} \leq \mathbf{x}, \end{array} \tag{8}$$

where $\mathbf{l} \in \Re^{(l \times 1)}$ is a vector of tree lengths where each entry is equal to the number of edges in the tree. By using a wirelength objective with the edge capacity constraints relaxed the solutions obtained from this model represent an absolute lower bound on the wirelength for a solution. This lower bound is not only the minimum for the given set of trees but also the minimum possible wirelength since each net has at least one minimum length tree produced from FLUTE.

The results for the WIR model are tabulated in Table 4. The experiments are performed using one additional iteration of tree generation using the tree generation framework proposed in Section 3. In Column 2, the maximum demand amongst all edges is given. In the third column the number of edges with overflow and the percentage of edges overflown (between parentheses) are recorded. The fourth and fifth column show the total overflow of all edges and the average amount of overflow of each overflowing edge, respectively. The rightmost column contains the runtime taken to solve the integer relaxation. The row labeled "Avg" shows the average values when relevant.

Table 4: Results for the WIR model using the ISPD98 global routing benchmark suite

Circuit	Max $d(\cdot)$	Edges OV(%)	Total OV	OV/ edge	Solver TM(s)
ibm01	30	879 (10.6 %)	3374	3.8	1
ibm02	58	1204 (11.6 %)	7227	6.0	0
ibm03	50	863 (8.3 %)	3849	4.5	0
ibm04	43	1437 (11.5 %)	6992	4.9	1
ibm05	72	154 (0.9 %)	837	5.4	0
ibm06	71	1567 (9.5 %)	7970	5.1	1
ibm07	60	1542 (6.2 %)	8595	5.6	1
ibm08	55	2181 (8.8 %)	9932	4.6	1
ibm09	54	3071 (9.3 %)	12792	4.2	1
ibm10	64	2098 (6.3 %)	12884	6.1	3
Avg	-	- (8.3 %)	-	5.0	-

The amount of overflow is quite high with this model having an average overflow of 5.0 for each overflowing edge and a maximum total overflow near 13000. The wirelength model does not consider reducing congestion at all so these values were anticipated to be high.

To interpret the congestion maps, an edge demand histogram is shown how many edges have a specific amount of demand. The histogram for the WIR model for ibm06 is given in Figure 5. The vertical axis in this histogram is the percentage of edges with a specific demand. The vertical line drawn on the histogram is the edge capacity for either horizontal or vertical edges. Any bars to the right of the vertical line represent overflowing edges. The histogram mimics a normal distribution which may not come as a surprise after observing the congestion map. Since the capacity constraints are relaxed, there is no "bunching up" to the left

of the capacity line that would be seen in a legal solution as overflowing edges gradually use demand of neighboring edges to remove overflow. Rather, there is a very natural decay from peak to tail.

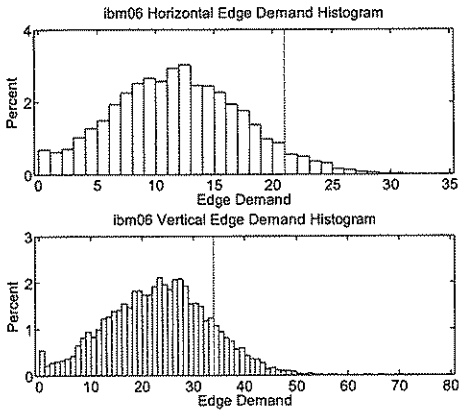


Figure 5: An edge demand histogram for ibm06 using the WIR model.

4.1.4 *Routing Supply Integer Relaxation Congestion Maps*

A new model is formulated in this section by making use of routing supply in the objective function. This is a way to account for edge capacities without the use of constraints to allow the integer relaxation structure.

The routing supply integer relaxation model (RSIR) is as follows

$$\begin{array}{lll} \text{(RSIR)} & \text{minimize} & -(\mathbf{rs})'\mathbf{x} \\ & \text{subject to} & \mathbf{Ux} = \mathbf{1} \\ & & \mathbf{0} \leq \mathbf{x}, \end{array} \tag{9}$$

where $\mathbf{rs}$ is a vector of routing supplies for the trees.

The results for the RSIR model are tabulated in Table 5. The amount of overflow moderate with about 6.5% of the edges overflowing and each of them exceeding their capacity by 3.9 on average. The time taken to solve the problem is very short and unvarying with the different problem sizes as was the case for the WIR model.

Table 5: Results for the RSIR model using the ISPD98 global routing benchmark suite

Circuit	Max $d(\cdot)$	Edges OV (%)	Total OV	OV/ edge	Solver time (s)
ibm01	31	764 (9.2 %)	2416	3.2	1
ibm02	53	1083 (10.4 %)	5162	4.8	0
ibm03	53	727 (7.0 %)	3250	4.5	0
ibm04	54	1097 (8.8 %)	5290	4.8	1
ibm05	64	10 (0.1 %)	16	1.6	1
ibm06	55	1273 (7.7 %)	4946	3.9	1
ibm07	59	1036 (4.2 %)	4994	4.8	1
ibm08	54	1483 (6.0 %)	5146	3.5	1
ibm09	51	2203 (6.7 %)	6995	3.2	2
ibm10	57	1511 (4.6 %)	7210	4.8	1
Avg	-	- (6.5 %)	-	3.9	-

The edge demand histogram for ibm06 produced by the RSIR model is given in Figure 6. The histogram shows relatively quickly decaying right tails past the capacity line for both horizontal and vertical edges. The slope descending to the left from the peaks is less steep than the slope to the right, particularly in the vertical edge case. This is a signature of the approach to a legal solution as all of the bins to the right of the capacity line get perturbed into the bins just to the left of the capacity line. This results from the shifting of routes going through overflowing edges into edges that have available capacity. When the solution is legal the slope to the right is a straight line going directly down.

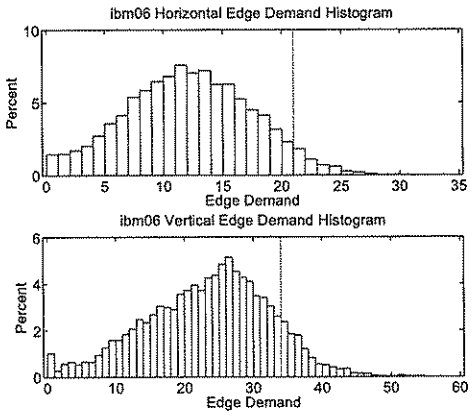


Figure 6: An edge demand histogram for ibm06 using the RSIR model.

4.1.5 *A Comparison Between the Proposed Models*

The WIR map is effective at locating congestion "hot spots" while the RSIR produces a map that may be more useful for fast global routing. This is because the RSIR map more closely resembles a legal solution. With this information a router is better able to predict how much detouring routes will need from a minimum length starting point to obtain a legal solution.

Table 6 contains a comparison between the WIR model in (8) and the RSIR model in (9). The tabulated values represent a percentage change between the RSIR values and the WIR values using the following equation

$$\Delta_{\%} = \left(\frac{(\text{RSIR value}) - (\text{WIR value})}{\text{WIR value}} \right) \times 100\%$$

where $\Delta_{\%}$ is the percentage change. Except for three maximum demand values, the RSIR model has lower congestion properties than the WIR model. The RSIR model has nearly one third less overflowing edges and 41% less total overflow on average. Solver times are virtually the same for both models and excluded from the table.

To visualize the differences between the two histogams, a difference histogram is shown.

The distinction between the two methods is clear in the edge demand difference histogram shown in Figure 7. This

Table 6: Percentage comparison between the WIR and RSIR model using the ISPD98 global routing benchmark suite

Circuit	Max $d(\cdot)$	Edges OV	Total OV	OV/ edge
ibm01	3.3%	-13.1%	-28.4%	-16.8%
ibm02	-8.6%	-10.0%	-28.6%	-20.6%
ibm03	6.0%	-15.8%	-15.6%	-0.7%
ibm04	25.6%	-23.7%	-24.3%	-1.6%
ibm05	-11.1%	-93.5%	-98.1%	-70.4%
ibm06	-22.5%	-18.8%	-37.9%	-23.8%
ibm07	-1.7%	-32.8%	-41.9%	-13.9%
ibm08	-1.8%	-32.0%	-48.2%	-24.6%
ibm09	-5.6%	-28.3%	-45.3%	-24.4%
ibm10	-10.9%	-28.0%	-44.0%	-21.8%
Avg	-2.7%	-29.6%	-41.2%	-21.8%

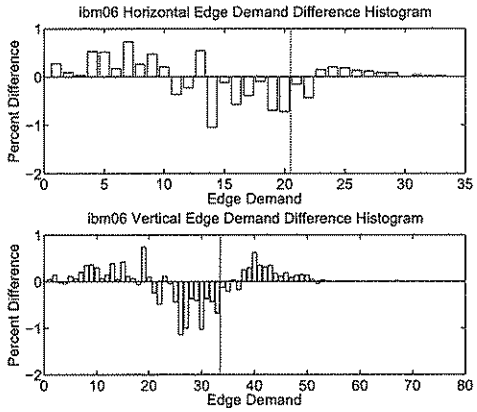


Figure 7: An edge demand difference histogram for ibm06 between the WIR and RSIR model.

diagram is produced by subtracting the RSIR percentage from the WIR percentage on a bin-by-bin basis. Any bin with a value greater than zero means that the WIR model has more edges with that particular demand value. A bin with a negative value implies that the RSIR model has a greater amount of edges with that demand. The large amount of bins with negative percentage difference in the center is showing that there are much more edges with demands near their capacity in the RSIR model than in the WIR model. Since overflow minimization is an objective of the RSIR model, trees with minimum wirelength that are chosen in the WIR model are exchange with alternate trees that have higher wirelength but create less congestion. This accounts for the positive percentage for bins with high demand. The positive percentage for bins with low demand is because the RSIR model seeks out the routes that pass through low demand edges, increasing the demand for them.

For global routing the RSIR model is of more utility because it better approximates a legal solution. The WIR model can provide a better indication of the most highly congested regions where circuit hot-spots are likely to arise. This may be of use in thermal-driven routing where maximum temperature during operation is to be minimized. However, if the WIR model is used, congestion levels may be high enough to make the circuit seem unroutable when it actually is routable. Both methods are able to produce congestion maps quickly, on the order of seconds. The time spent generating trees used for the congestion map does not increase the total runtime, since those trees need to be generated even if the map was not produced. The amount of time is never more than 5% and makes up a smaller percentage as the circuit size increases.

5. CONCLUSIONS

6. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] ISPD2007 Global Routing Contest, 2007. http://www.sigda.org/ispd2007/rcontest/, date visited: June 24, 2008.

[2] ISPD2008 Global Routing Contest, 2008. http://www.ispd.cc/contests/ispd08rc.html, date visited: June 24, 2008.

[3] C. Albrecht. Global routing by new approximation algorithms for multicommodity flow. *IEEE Trans. on CAD*, 20(5):622–632, 2001.

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

[5] M. Burstein and R. Pelavin. Hierarchical wire routing. *IEEE Trans. on CAD*, 2:223–234, 1983.

[6] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen, and B. Han. Full-chip routing considering double-via insertion. *IEEE Trans. on CAD*, 27(5):844–857, 2008.

[7] A. Chiang, L. Behjat, L. Rakai, and J. Li. An effective congestion-based integer programming model for VLSI global routing. In *Proc. CCECE*, pages 931–936, 2008.

[8] M. Cho, K. Lu, K. Yuan, and D. Z. Pan. BoxRouter 2.0: Architecture and implementation of a hybrid and robust global router. In *Proc. ICCAD*, pages 503–508, 2007.

[9] M. Cho and D. Z. Pan. BoxRouter: A new global router based on box expansion and progressive ILP. In *Proc. DAC*, pages 373–378, San Francisco, CA, USA, 2006.

[10] C. Chu and Y.-C. Wong. FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design. *IEEE Trans. on CAD*, 27(1):70–83, 2008.

[11] J. Cong and P. H. Madden. Performance driven routing with multiple sources. *IEEE Trans. on CAD*, 16:410–419, 1997.

[12] G. Cramer. Introduction á l'analyse des lignes courbes algbriques. *A Genève : Chez les Freres Cramer & Cl. Philibert*, pages 656–659, 1750.

[13] J.-R. Gao, P.-C. Wu, and T.-C. Wang. A new global router for modern designs. In *Proc. ASPDAC*, pages 232–237, 2008.

[14] L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics: a foundation for computer science – 2nd edition.* Addison Wesley, 1994.

[15] R. T. Hadsell and P. H. Madden. Improved global routing through congestion estimation. In *Proceedings of the 40th conference on Design automation*, pages 28–31, New York, NY, USA, 2003. ACM.

[16] W. L. Hare, M. J. J. Liu, and T. Terlaky. Efficient preprocessing for vlsi optimization problems. *Computational Optimization and Applications*, 2008.

[17] I. Heller and C. Tompkins. An extension of a theorem of dantzig's. *Linear Inequalities and Related Systems*, pages 247–254, 1956.

[18] A. Hoffman and J. Kruskal. Integral boundary points of convex polyhedra. *Linear Inequalities and Related Systems*, pages 223–246, 1956.

[19] J. Hu, J. A. Roy, and I. L. Markov. Sidewinder - a scalable ILP-based router. In *Proc. SLIP*, pages 73–80, Newcastle, United Kingdom, 2008.

[20] ILOG. CPLEX: High-performance software for mathematical programming and optimization. http://www.ilog.com/products/cplex/, date visited: July 20, 2008.

[21] A. B. Kahng and X. Xu. Accurate pseudo-constructive wirelength and congestion estimation. In *Proc. SLIP*, pages 61–68, 2003.

[22] C.-W. Lin, S.-Y. Chen, C.-F. Li, Y.-W. Chang, and C.-L. Yang. Obstacle-avoiding rectilinear steiner tree construction based on spanning graphs. *IEEE Trans. on CAD*, 27(4):643–653, 2008.

[23] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng. Estimating routing congestion using probabilistic analysis. *IEEE Trans. on CAD*, 21(1):32–41, 2002.

[24] L. McMurchie and C. Ebeling. Pathfinder: A negotiation-based performance-driven router for fpgas. In *Proc. FPGA*, pages 111–117, 1995.

[25] M. D. Moffitt, J. A. Roy, and I. L. Markov. The coming age of (academic) global routing. In *Proc. ISPD*, pages 148–155, 2008.

[26] M. Pan and C. Chu. Fastroute: a step to integrate global routing into placement. In *Proc. ICCAD*, pages 464–471, 2006.

[27] M. Pan and C. Chu. Fastroute 2.0: A high-quality and efficient global router. In *Proc. ASPDAC*, pages 250–255, 2007.

[28] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. In *Proc. ICCAD*, pages 496–502, 2007.

[29] Z. Shen, C. Chu, and Y.-M. Li. Efficient rectilinear steiner tree construction with rectilinear blockages. In *Proc. ICCAD*, pages 38–44, San Jose, CA, USA, 2005.

[30] T. Taghavi, F. Dabiri, A. Nahapetian, and M. Sarrafzadeh. Tutorial on congestion prediction. In *Proc. SLIP*, pages 15–24, 2007.

[31] J. Westra, C. Bartels, and P. Groeneveld. Probabilistic congestion prediction. In *Proc. ISPD*, pages 204–209, Phoenix, Arizona, USA, 2004.

[32] J. Westra and P. Groeneveld. Is probabilistic congestion estimation worthwhile? In *Proc. SLIP*, pages 99–106, 2005.

[33] Z. Yang, S. Areibi, and A. Vannelli. A comparison of ILP based global routing models for vlsi asic design. In *Proc. MWSCAS*, pages 1141–1144, 2007.