Global Routing by New Approximation Algorithms for Multicommodity Flow

Christoph Albrecht

***Abstract*—We show how the new approximation algorithms by Garg and Könemann with extensions due to Fleischer for the multicommodity flow problem can be modified to solve the linear programming relaxation of the global routing problem. Implementation issues to improve the performance, such as a discussion of different functions for the dual variables and how to use the Newton method as an additional optimization step, are given. It is shown that not only the maximum relative congestion is minimized, but the congestion of the edges is distributed equally such that the solution is optimal in a well-defined sense: the vector of the relative congestion of the edges sorted in nonincreasing order is minimal by lexicographic order. This is an important step toward improving signal integrity by extra spacing between wires. Finally, we show how the weighted netlength can be minimized. Our computational results with recent IBM processor chips show that this approach can be used in practice even for large chips and that it is superior on difficult instances where ripup and reroute algorithms fail.**

***Index Terms*—Global routing, physical design, VLSI.**

I. Introduction

ROUTING is usually split up into two subtasks: global routing, which gives the approximate area for the Steiner tree for a net, and detailed routing, which assigns the actual tracks and vias to the nets. In this paper, we consider the problem of global routing only.

Many global routers are based on an initial route of all nets followed by a ripup and reroute procedure that tries to reduce the congestion of the edges by rerouting segments of nets on overloaded edges. For difficult instances, these algorithms may run forever and not come up with a solution, even though a solution exists.

Ting and Tien [1] present such an iterative improvement algorithm for global routing. Their work was later improved by Shragowith and Keel [2], who present an algorithm that terminates in some cases with a proof that a solution does not exist. Meixner and Lauther [3] consider several nets which have one terminal in common at the same time by a single commodity flow formulation.

Our algorithm is based on linear programming relaxation and randomized rounding. This approach is not new. It was proposed by Raghavan and Thompson [4], [5]. However, they do not state explicitly how to solve the linear programming relaxation. Solving the linear program directly is possible for very small instances only and a simple formulation as a multicommodity flow problem works only for nets with two terminals.

Early work on linear programming approaches has already been done by Aoshima and Kuh [6], Hu and Shing [7], who use the column generation technique, and Huang *et al.* [8]. Carden IV *et al.* [9], [24] use the approximation algorithm for multicommodity flow by Shahrokhi and Matula [10]. Their basic approach is similar to ours. However, we use a new and different approximation algorithm with various extensions with which we can solve much larger instances than in [9] and [24]. Finally, we show how the weighted netlength can be minimized, which is important for the approach to be used in practice.

Other global routers use a hierarchical decomposition approach, for example, Brouwer and Banerjee [11] and Dai and Kuh [12]. Cong and Preas [13] give a spanning forest formulation.

Our contribution is to show that the new combinatorial approximation algorithm by Garg and Könemann [14] for the maximum concurrent flow problem can be modified to solve the linear programming relaxation of the global routing problem and to show that an idea introduced by Fleischer [15], [25] to improve the theoretical worst case running time of the maximum multicommodity flow problem can be used to speed up the approximation algorithm in practice while the same upper bound on the worst case running time can be proved. We use a function for the dual variables that is different to the one used by Garg and Könemann and give a comparison of the results for both functions in practice. Further performance improvements are achieved by using the Newton method as an additional optimization step. We show that the algorithm computes a solution for which the congestion of all edges is minimized and distributed equally: the vector of the relative congestion of the edges sorted in nonincreasing order is minimal by lexicographic order. Finally, it is possible to minimize the total weighted netlength with respect to a given maximum relative congestion. Our computational results show for the first time that these approximation algorithms give good results in practice even for very large chips.

The approximation algorithm also provides a solution for the dual linear program. By linear programming duality, the value of any feasible solution for the dual linear program is a lower bound on the optimum maximum relative congestion and, therefore, gives a factor by which the capacities of the edges have to be multiplied at least in order to make the chip routable.

It is possible to add additional constraints for some nets. Nets that are timing-critical may be routed with minimum L_1 length only, perhaps even with respect to a given delay optimal topology, or a bound for the L_1 length of the net may be given.

Manuscript received July 24, 2000; revised November 13, 2000. This paper was recommended by Associate Editor D. Hill.

The author is with the Research Institute for Discrete Mathematics, University of Bonn, 53113 Bonn, Germany (e-mail: albrecht@or.uni-bonn.de).

Publisher Item Identifier S 0278-0070(01)03085-8.

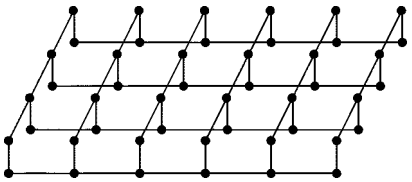


Fig. 1. Global routing graph with two layers and via edges.

The rest of this paper is organized as follows. In Section II, we introduce some notation and terminology, formulate a mixed integer program for global routing, give the linear programming relaxation, and derive a lower bound on the optimum maximum relative congestion by linear programming duality. In Section III, we describe the approximation algorithm for minimizing the maximum relative congestion and give the complete proof that it can achieve any desired approximation ratio. In Section IV, we compare different functions for the dual variables and compare the results achieved in practice. In Section V, we describe how the Newton method can be used to improve the results in practice. In Section VI, we show that not only does the approximation algorithm minimize the maximum relative congestion, but it also "works toward" a solution that is optimal by lexicographic order. In Section VII, we show how the total weighted netlength can be minimized and in Section VIII, we describe our computational results.

II. Problem Formulation

In global routing, an *undirected grid graph* $G = (V, E)$ is constructed: a two-dimensional grid is placed over the chip. For each tile, there is a vertex $v \in V$ and two vertices corresponding to adjacent tiles are connected by an edge.

It is also possible that the grid graph G consists of different layers such that via capacities as well as capacities for different layers can be considered.

Fig. 1 shows a global routing graph with two layers: one for wiring in x direction, the other for wiring in y direction, and via edges in between.

We denote by m the number of edges in G.

For global routing, only nets with pins in different tiles are considered. Let k be the total number of these nets and for each net i, $i = 1, \ldots, k$, let $N_i \subseteq V$ be the set of vertices for which there exists a pin of the net in the corresponding tile. We call the vertices of N_i the *terminals* of net i. In addition, for each net i and each edge e, a *width* $w_{i,e}$ is given. The width of a net may vary from edge to edge because the width of a wire of a net may be different for different layers.

For a given net i, let $\mathcal{T}_i$ be the set of all possible Steiner trees. This set might be restricted such that it contains only a subset of all possible Steiner trees, for example, for timing-critical nets, only the Steiner trees with minimum L_1 length. We assume that given any nonnegative lengths for the edges, the algorithm can query a subroutine to compute a Steiner tree $T \in \mathcal{T}_i$ of minimum length with respect to these lengths.

In practice, a heuristic that does not necessarily return the optimum Steiner tree is often good enough. However, to compute a lower bound on the optimum solution, a subroutine that computes a lower bound of the minimum length for all Steiner trees in $\mathcal{T}_i$ with respect to given nonnegative lengths on the edges is needed.

For each edge $e = \{u, v\}$, a *capacity* $c(e)$ is computed according to the number of free channels between the two tiles corresponding to u and v, taking into account the tanglement of the nets which have all pins either in u or in v.

Global routing asks for a Steiner tree T_i for each net i. Given these Steiner trees, the *relative congestion* of an edge e is defined as $\lambda(e) := \sum_{i:e \in T_i} w_{i,e}/c(e)$ and the maximum relative congestion is $\lambda := \max_{e \in E} \lambda(e)$.

As a first approach, we will consider the task to find a Steiner tree T_i for each net i such that the maximum relative congestion is minimized. Later, we will consider another version of the global routing problem: find Steiner trees such that the maximum relative congestion is at most one and the total weighted netlength is minimized.

With the notation above, the *global routing problem* can be formulated as a mixed integer program

$$\min \lambda$$

subject to

$$\begin{aligned} \sum_{i,T:e \in T \in \mathcal{T}_i} w_{i,e} x_{i,T} &\le \lambda c(e), \qquad \text{for } e \in E \\ \sum_{T \in \mathcal{T}_i} x_{i,T} &= 1, \qquad \text{for } i = 1, \ldots, k \\ x_{i,T} &\in \{0, 1\}, \qquad \text{for } i = 1, \ldots, k;\ T \in \mathcal{T}_i. \end{aligned} \tag{1}$$

Instead of solving the mixed integer program directly, we consider the linear programming relaxation by substituting the last constraint in (1) by

$$x_{i,T} \ge 0 \qquad \text{for } i = 1, \ldots, k;\ T \in \mathcal{T}_i.$$

We call the problem of finding an optimal solution to this linear program the *fractional global routing problem* and we denote the value of the optimum solution by λ^*. For any feasible solution of this linear program, the relative congestion of an edge e is given by $\lambda(e) := \sum_{i,T:e \in T \in \mathcal{T}_i} w_{i,e} x_{i,T}/c(e)$. We will sometimes write a solution $(x_{i,T})_{i=1,\ldots,k;T \in \mathcal{T}_i}$ for the fractional global routing problem simply as a vector x.

The dual of this linear program is given by

$$\max \sum_{i=1}^{k} z_i$$

subject to

$$\begin{aligned} \sum_{e \in E} c(e) y_e &= 1 \\ \sum_{e \in T} w_{i,e} y_e &\ge z_i, \qquad \text{for } i = 1, \ldots, k;\ T \in \mathcal{T}_i \\ y_e &\ge 0, \qquad \text{for } e \in E. \end{aligned} \tag{2}$$

By linear programming duality (a good overview can be found in [16]), any feasible solution of the dual linear program provides a lower bound on the optimum solution for the fractional global routing problem and for the optimum solutions equality holds.

```
(1)  Set y_e := 1/c(e) for all e ∈ E.
(2)  Set x_{i,T} := 0 for i = 1,...,k;  T ∈ T_i.
(3)  Set T_i := ∅ for i = 1,...,k.
(4)  While ( Σ_{e∈E} c(e)y_e < M )
(5)  begin
(6)    For i := 1 to k
(7)    begin
(8)      If T_i = ∅ or Σ_{e∈T_i} w_{i,e}y_e > (1+γε)z_i
(9)      begin
(10)       Find a minimal Steiner tree T_i ∈ T_i
             for net i with respect to length
             w_{i,e}y_e, e ∈ E.
(11)       Set z_i := Σ_{e∈T_i} w_{i,e}y_e.
(12)     end
(13)     Set x_{i,T_i} := x_{i,T_i} + 1.
(14)     Set y_e := y_e e^{ε w_{i,e}/c(e)} for all e ∈ T_i.
(15)   end
(16) end
```

Fig. 2. Approximation algorithm for fractional global routing.

According to the second inequality in (2), z_i has to be smaller than the minimum length of all Steiner trees $T \in \mathcal{T}_i$ with respect to the length $w_{i,e}y_e$ for edge e. As $\sum_{i=1}^{k} z_i$ is maximized, z_i can be substituted by this minimum value and by rescaling y_e such that the first inequality in (2) holds, we get the following theorem.

Theorem 1: Given any nonnegative values y_e for the edges $e \in E$, the expression

$$\frac{\sum_{i=1}^{k} \min_{T \in \mathcal{T}_i} \sum_{e \in T} w_{i,e} y_e}{\sum_{e \in E} c(e) y_e}$$

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.

III. Solving the Fractional Global Routing Problem

The approximation algorithm that solves the fractional global routing problem for any given approximation ratio $1 + \epsilon_0$ is shown in Fig. 2.

The variables are initialized in lines 1 to 3. The algorithm is called with the parameters γ, ϵ, and M. The proof of the theorem in this section will show which value to choose for these parameters in order to get the desired approximation ratio. The same holds for γ and ϵ.

The algorithm runs through several phases. A phase starts in line 5 and ends in line 16. In the first phase we have $T_i = \emptyset$ and, therefore, for each net i, a minimal Steiner tree $T_i \in \mathcal{T}_i$ with respect to lengths $w_{i,e}y_e$, $e \in E$ is computed (line 10). Variable T_i stores the Steiner tree found for net i and in variable z_i, the length of this Steiner tree at the time it was computed is stored (line 11).

During a phase for each net i, the variable x_{i,T_i} is increased by one (line 13). To achieve that for each net i, the variables $x_{i,T}, T \in \mathcal{T}_i$ sum up to one; all variables $x_{i,T}$ are divided by the total number of phases at the end of the algorithm. Finally, the dual variables y_e are increased for all edges used by the Steiner tree T_i (line 14); they are increased more if the net uses a greater fraction of the capacity of the edge.

In subsequent phases, a new minimal Steiner tree for net i is only computed if the length of the last Steiner tree found with respect to the updated edge lengths $w_{i,e}y_e$, $e \in E$ has increased by more than $(1+\gamma\epsilon)$, where γ is a parameter of the algorithm (line 8). This is a modification of the algorithm by Garg and Könemann [14] for the maximum concurrent multicommodity flow problem. This idea was used by Fleischer [15], [25] to reduce the theoretical worst case running time for the maximum multicommodity flow problem. Here we prove that it does not increase the worst case running time complexity and practical results show that much better results can be obtained in the same running time. This is a substantial improvement to previous implementations (for example [9], [17], [24]), which loop over all nets (commodities) and compute a Steiner tree (min cost flow) for each net.

We use an update rule for the dual variables y_e (line 14) that is different from the one used by Garg and Könemann. A comparison will be given in the next section.

Theorem 2: Let $\gamma \geq 0$ be a constant. If there exists a solution for the fractional global routing problem with maximum relative congestion at most one, the algorithm finds a $(1+\epsilon_0)$-approximation in

$$O\left(\frac{1}{\epsilon_0^2 \lambda^*} \ln m\right)$$

phases, if

$$\epsilon := \min\left\{\frac{1}{\gamma}, \frac{1}{\gamma}\left(\sqrt{1+\epsilon_0}-1\right), \frac{1}{4}\left(1-\left(\frac{1}{1+\epsilon_0}\right)^{1/6}\right)\right\}$$

and

$$M := \left(\frac{m}{1-\epsilon'}\right)^{1/\epsilon'} \qquad \text{with } \epsilon' := \epsilon(1+\epsilon)(1+\epsilon\gamma).$$

Moreover, the variables y_e, $e \in E$ provide at some time during the algorithm a $(1+\epsilon_0)$ approximation for the dual linear program.

The total number of phases of the algorithm depends on λ^*, but usually in the application of global routing, λ^* is not arbitrarily small, for example, we can assume $\lambda^* \geq 1/2$.

To prove this theorem, we basically follow the proof by Garg and Könemann [14], but we repeat it here in order to show that the modifications [a new Steiner tree has to be computed only if its length has increased by at least a certain amount (line 8)] do not increase the worst case running time complexity. The proof of the modified update rule for y_e is due to Fleischer and will appear in [25].

For the analysis of the algorithm, we need to assume that the set $\mathcal{T}_i$ is restricted to Steiner trees T with $w_{i,e} \leq c(e)$ for all $e \in T$. Since we are finally interested in an integral solution with maximum relative congestion at most one, this is no restriction.

Proof: Let t be the total number of phases executed by the algorithm. We will prove that if the algorithm had stopped one phase before the last one, namely, after $t-1$ phases, the solution would have had the desired approximation ratio.

Let $y_e^{(p,i)}$ be the value of variable y_e after net i has been considered in phase p and y_e has been increased in line 14 and let $y_e^{(p)} := y_e^{(p,k)}$ be the value at the end of phase p. At the beginning we have

$$\sum_{e\in E} c(e)y_e^{(0)} = \sum_{e\in E} c(e)\frac{1}{c(e)} = m. \tag{3}$$

When the dual variables y_e are increased in line 14 after a Steiner tree T_i has been found, the expression $\sum_{e\in E} c(e)y_e$ increases and we have

$$\begin{aligned}
&\sum_{e\in E} c(e)y_e^{(p,i)} \\
&\quad= \sum_{e\notin T_i} c(e)y_e^{(p,i-1)} + \sum_{e\in T_i} c(e)y_e^{(p,i-1)}e^{\epsilon(w_{i,e}/c(e))} \\
&\quad\le \sum_{e\notin T_i} c(e)y_e^{(p,i-1)} \\
&\qquad+ \sum_{e\in T_i} c(e)y_e^{(p,i-1)}\left(1+\epsilon\frac{w_{i,e}}{c(e)}+\epsilon^2\left(\frac{w_{i,e}}{c(e)}\right)^2\right).
\end{aligned}$$

For the last inequality, we used $e^x \le 1+x+x^2$ for $0\le x\le 1$. (By the assumption on the set $\mathcal{T}_i$, we have $w_{i,e}/c(e)\le 1$ for all $e\in T_i$ and we assume $\epsilon\le 1$.)

For the same reason we get

$$\begin{aligned}
&\sum_{e\in E} c(e)y_e^{(p,i)} \\
&\quad\le \sum_{e\notin T_i} c(e)y_e^{(p,i-1)} \\
&\qquad+ \sum_{e\in T_i} c(e)y_e^{(p,i-1)}\left(1+\epsilon(1+\epsilon)\frac{w_{i,e}}{c(e)}\right) \\
&\quad= \sum_{e\in E} c(e)y_e^{(p,i-1)} + \epsilon(1+\epsilon)\sum_{e\in T_i} w_{i,e}y_e^{(p,i-1)}.
\end{aligned}$$

In the following phases, a new Steiner tree for net i is only computed if the length of the last Steiner tree found has increased by more than $(1+\gamma\epsilon)$. Since y_e is increased only during the algorithm, we always have at the beginning of line 14

$$\sum_{e\in T_i} w_{i,e}y_e^{(p,i-1)} \le (1+\gamma\epsilon)z_i \le (1+\gamma\epsilon)\min_{T\in\mathcal{T}_i}\sum_{e\in T} w_{i,e}y_e^{(p)}$$

which means that at the end of phase p, we have

$$\begin{aligned}
\sum_{e\in E} c(e)y_e^{(p)} &\le \sum_{e\in E} c(e)y_e^{(p-1)} \\
&\quad+\epsilon(1+\epsilon)(1+\gamma\epsilon)\sum_{i=1}^{k}\min_{T\in\mathcal{T}_i}\sum_{e\in T} w_{i,e}y_e^{(p)}.
\end{aligned} \tag{4}$$

For brevity, we set $\epsilon' := \epsilon(1+\epsilon)(1+\gamma\epsilon)$. By linear programming duality (Theorem 1) the expression

$$\lambda_{lb}^{(p)} := \frac{\sum_{i=1}^{k}\min_{T\in\mathcal{T}_i}\sum_{e\in T} w_{i,e}y_e^{(p)}}{\sum_{e\in E} c(e)y_e^{(p)}}$$

is a lower bound on the maximum relative congestion, that is, $\lambda_{lb}^{(p)} \le \lambda^*$.

With this, (4) can be rewritten as

$$\sum_{e\in E} c(e)y_e^{(p)} \le \sum_{e\in E} c(e)y_e^{(p-1)} + \epsilon'\lambda_{lb}^{(p)}\sum_{e\in E} c(e)y_e^{(p)}$$

which can be transformed to

$$\sum_{e\in E} c(e)y_e^{(p)} \le \frac{1}{1-\epsilon'\lambda_{lb}^{(p)}}\sum_{e\in E} c(e)y_e^{(p-1)}.$$

If we set $\lambda_{lb} := \max_{p=1,\ldots,t}\lambda_{lb}^{(p)}$, we get with (3)

$$\begin{aligned}
\sum_{e\in E} c(e)y_e^{(p)} &\le \frac{m}{(1-\epsilon'\lambda_{lb})^p} \\
&= \frac{m}{(1-\epsilon'\lambda_{lb})}\left(1+\frac{\epsilon'\lambda_{lb}}{1-\epsilon'\lambda_{lb}}\right)^{p-1} \\
&\le \frac{m}{(1-\epsilon')}\left(1+\frac{\epsilon'\lambda_{lb}}{1-\epsilon'}\right)^{p-1} \\
&\le \frac{m}{(1-\epsilon')}e^{\epsilon'\lambda_{lb}(p-1)/1-\epsilon'}.
\end{aligned} \tag{5}$$

For the last inequality, $1+x\le e^x$ for $x\ge 0$ is used.

An upper bound on the relative congestion of an edge e can now be derived. Suppose edge e is used s times by some tree during the first $t-1$ phases and let the jth increment in the relative congestion of edge e be $a_j := w_{i,e}/c(e)$ for the appropriate i. After rescaling the variables $x_{i,T}$, the relative congestion of edge e is $\lambda(e) = \sum_{j=1}^{s} a_j/(t-1)$. Since $y_e^{(0)} = 1/c(e)$ and $y_e^{(t-1)} < M/c(e)$ (because the condition in line 4 still holds before the last phase is executed) and since

$$y_e^{(t-1)} = \frac{1}{c(e)}\prod_{j=1}^{s} e^{\epsilon a_j}$$

we get

$$\frac{1}{c(e)}\prod_{j=1}^{s} e^{\epsilon a_j} \le \frac{M}{c(e)}.$$

It follows that

$$\exp\left(\epsilon\sum_{j=1}^{s} a_j\right) \le M$$

and, hence, we get

$$\lambda(e) = \frac{\sum_{j=1}^{s} a_j}{t-1} \le \frac{\ln M}{\epsilon(t-1)}. \tag{6}$$

Since $\sum_{e\in E} c(e)y_e^{(t)} \geq M$, solving (5) with $p = t$ for λ_{lb} gives a lower bound on the optimum solution value

$$\lambda_{lb} \geq \frac{1-\epsilon'}{\epsilon'(t-1)} \ln\left(\frac{M}{m}(1-\epsilon')\right)$$

from which together with (6), we get an upper bound on the approximation ratio ρ

$$\frac{\max_{e\in E}\lambda(e)}{\lambda_{lb}} \leq \frac{\frac{\ln M}{\epsilon(t-1)}}{\frac{1-\epsilon'}{\epsilon'(t-1)}\ln\left(\frac{M}{m}(1-\epsilon')\right)} = \frac{\epsilon'}{(1-\epsilon')\epsilon}\frac{\ln M}{\ln\left(\frac{M}{m}(1-\epsilon')\right)}.$$

If M is now chosen to be

$$M := \left(\frac{m}{1-\epsilon'}\right)^{1/\epsilon'}$$

we get

$$\frac{\ln M}{\ln\left(\frac{M}{m}(1-\epsilon')\right)} = \frac{1}{1-\epsilon'}$$

such that

$$\rho \leq \frac{\epsilon'}{(1-\epsilon')^2\epsilon} = \frac{\epsilon(1+\epsilon)(1+\gamma\epsilon)}{(1-\epsilon(1+\epsilon)(1+\gamma\epsilon))^2\epsilon} = \frac{(1+\epsilon)(1+\gamma\epsilon)}{(1-\epsilon(1+\epsilon)(1+\gamma\epsilon))^2}.$$

If $\epsilon \leq 1/\gamma$, we have $1+\gamma\epsilon \leq 2$ and $1+\epsilon \leq 2$ (assuming $\gamma > 1$) and we get

$$\rho \leq \frac{(1+\epsilon)(1+\gamma\epsilon)}{(1-4\epsilon)^2} \leq (1+\gamma\epsilon)\frac{1}{(1-4\epsilon)^3} \qquad \left(\begin{matrix}\text{because}\\ 1+\epsilon \leq \frac{1}{1-\epsilon} \leq \frac{1}{1-4\epsilon}\end{matrix}\right).$$

If ϵ is chosen such that $1+\gamma\epsilon \leq \sqrt{1+\epsilon_0}$ and $1/(1-4\epsilon)^3 \leq \sqrt{1+\epsilon_0}$, we choose

$$\epsilon = \min\left(\frac{1}{\gamma}, \frac{1}{\gamma}\left(\sqrt{1+\epsilon_0}-1\right), \frac{1}{4}\left(1-\left(\frac{1}{1+\epsilon_0}\right)^{1/6}\right)\right)$$

and get $\rho \leq 1+\epsilon_0$. After all, we have $\epsilon = O(\epsilon_0)$ and also $\epsilon' = O(\epsilon_0)$.

Since $\max_{e\in E}\lambda(e) \geq \lambda^*$, we get from (6)

$$\lambda^* \leq \frac{\ln M}{\epsilon(t-1)}$$

which means that the maximum number of phases is bounded by

$$t \leq 1 + \frac{\ln M}{\lambda^*\epsilon} = 1 + \frac{1}{\epsilon\epsilon'\lambda^*}\ln\left(\frac{m}{1-\epsilon'}\right).$$

The last expression can be bounded by $O(1/\epsilon_0^2\lambda^* \ln m)$. ■

In the implementation, we do not have a variable $x_{i,T}$ for each possible Steiner tree (which would be impossible because of memory limitations), but only one variable for each Steiner tree generated by the subroutine in line 10 is needed. These Steiner trees are simply stored in a list.

Fig. 3 shows how the maximum relative congestion decreases during the algorithm for different values for γ for the test case quorina (see Section VIII for details). If $\gamma = 0$, we have the algorithm by Garg and Könemann. With larger values for γ, a better relative congestion is achieved in the same running time.

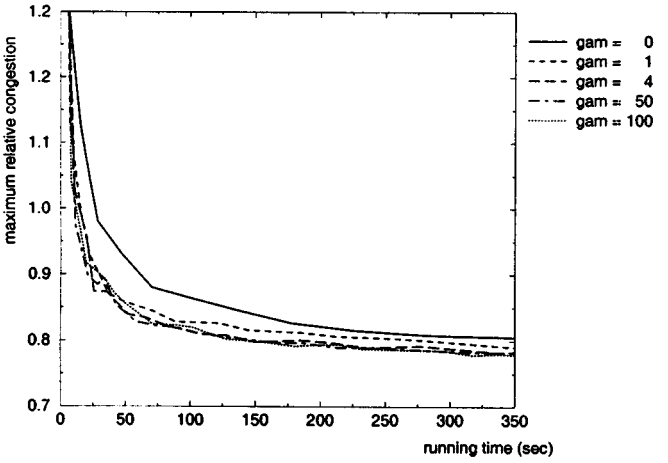


Fig. 3. Maximum relative congestion in terms of running time for the algorithm in Fig. 2 for different values of γ for the test case quorina ($\epsilon = 2.0$).

IV. Different Functions for Dual Variables

Garg and Könemann [14] use a different function for the dual variables. The same function was also used in an earlier version of this paper (see [18]). Here, the update rule for the dual variables y_e in line 14 of the algorithm in Fig. 2 was

$$\text{(14)} \qquad \text{Set } y_e := y_e\left(1+\epsilon\frac{w_{i,e}}{c(e)}\right) \qquad \text{for all } e \in T_i.$$

For small values of ϵ, we have $e^{\epsilon(w_{i,e}/c(e))} \approx (1+\epsilon(w_{i,e}/c(e)))$, but for larger values of ϵ, this approximation does not hold. In practice, much larger values for ϵ than the ones given by theory (Theorem 2) are necessary for the algorithm to make progress. If the fraction $w_{i,e}/c(e)$ is different during the algorithm with the function $y_e := 1/c(e)\cdot(1+\epsilon(w_{i,e}/c(e)))\cdot\cdots$, the maximum relative congestion diverges for larger values of ϵ. It is possible to construct small examples to see this phenomenon. The reader is asked to consider the result of the algorithm for both functions for the dual variables y_e for the following instance of the global routing problem: $G = (V, E)$ with $V = \{v_1, v_2\}$ and $E = \{e_1, e_2\}$ with $e_1 = e_2 = \{v_1, v_2\}$, $c(e_1) = 10$, $c(e_2) = 1$ and only one net, that is $k = 1$ with $N_1 = \{v_1, v_2\}$ and $w_{1,e} = 1$ for $e = e_1, e_2$.

In Figs. 4 and 5, we show the maximum relative congestion in terms of the running time for the two different functions for the dual variables for the testcase quorina. The advantage of the new function is obvious.

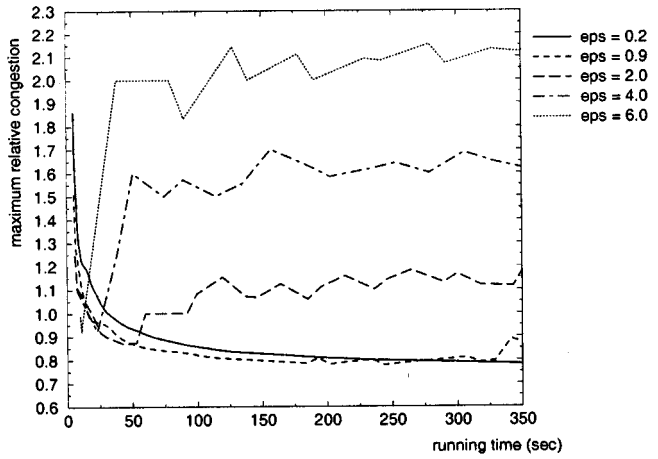


Fig. 4. Maximum relative congestion in terms of running time for different values of ϵ with the function $y_e := 1/c(e) \cdot (1 + \epsilon(w_{i,e}/c(e))) \cdot \ \cdots$ for the dual variables ($\gamma = 10$).

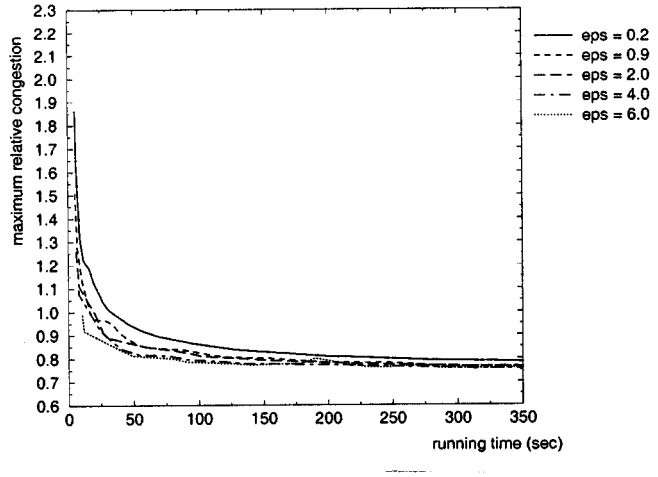


Fig. 5. Maximum relative congestion in terms of running time for different values of ϵ with the function $y_e := 1/c(e) \cdot e^{\epsilon(w_{i,e}/c(e))} \cdot \ \cdots$ for the dual variables ($\gamma = 10$).

V. Newton Method for an Additional Optimization Step

The function for the dual variables used by the algorithm in Fig. 2 is the same that is used by earlier approximation algorithms (for example Shahrokhi and Matula [10] and Plotkin *et al.* [19]). The function for y_e of these algorithms is $y_e := 1/c(e)e^{\alpha\lambda(e)}$ for given α and the idea is to minimize the potential $\Phi := \sum_{e\in E} c(e)y_e = \sum_{e\in E} e^{\alpha\lambda(e)}$ for fixed α (a nonlinear optimization problem subject to the linear constraints of the fractional global routing problem), to increase α afterwards, and to continue to minimize Φ. For the algorithm in Fig. 2, at the end of phase p we have $y_e := 1/c(e)e^{\epsilon p\lambda(e)}$. So, $\alpha := \epsilon p$ and α increases by ϵ with each phase.

We use the Newton method as an additional optimization step to minimize the potential Φ. Goldberg *et al.* [17] used the same technique for their implementation of an approximation algorithm for the minimum cost multicommodity flow problem.

By the Newton method, for each net i, a convex combination of the current fractional solution for net i and the new Steiner tree T_i is computed. For this, line 13 of the algorithm in Fig. 2 is exchanged as follows.

At the beginning of phase p, the expression $\sum_{T\in\mathcal{T}_i} x_{i,T}$ has value $p-1$. It has to be increased by one when net i is considered. If the current fractional solution for net i is not changed, we would get the solution

$$\hat{x}_{j,T} := x_{j,T}, \qquad \text{for } j \neq i,\ T \in \mathcal{T}_j$$

and

$$\hat{x}_{i,T} := \frac{p}{p-1} x_{i,T}, \qquad \text{for } T \in \mathcal{T}_i.$$

If only the new Steiner tree T_i found for net i was taken for the solution for net i, we would get

$$x^*_{j,T} := x_{j,T}, \qquad \text{for } j \neq i,\ T \in \mathcal{T}_j$$
$$x^*_{i,T_i} := p$$

and

$$x^*_{i,T} := 0, \qquad \text{for } T \in \mathcal{T}_i,\ T \neq T_i.$$

By the Newton method, the optimal step size $\sigma \in [0, 1]$ is computed such that the potential Φ is minimized for the new solution $x := (1-\sigma)\hat{x} + \sigma x^*$. For the algorithm without the Newton method, we always have $\sigma = 1/p$. This can be used as a starting solution for the Newton method.

With

$$\hat{f}(e) := \sum_{j,T:e\in T\in\mathcal{T}_j} \hat{x}_{j,T} w_{j,e}$$
$$\hat{f}_i(e) := \sum_{T\in\mathcal{T}_i} \hat{x}_{i,T}$$

and

$$f^*_i(e) := \sum_{T\in\mathcal{T}_i} x^*_{i,T}$$

the potential $\Phi := \sum_{e\in E} c(e)y_e$ can be expressed as a function of the step size σ

$$\Phi(\sigma) := \sum_{e\in E} e^{\epsilon(\hat{f}(e)+\sigma(f^*_i(e)-\hat{f}_i(e)))/c(e)}$$
$$= \sum_{e\in E} e^{\epsilon\hat{f}(e)/c(e)}\, e^{\epsilon\sigma(f^*_i(e)-\hat{f}_i(e))/c(e)}.$$

The function $\Phi(\sigma)$ is convex and if $f^*_i(e) = \hat{f}_i(e)$ for at least one edge $e \in E$, $\Phi(\sigma)$ is strictly convex. In this case, there exists a unique optimal step size $\sigma \in [0, 1]$, which minimizes the potential $\Phi(\sigma)$. The first and second derivative of $\Phi(\sigma)$ are easy to compute.

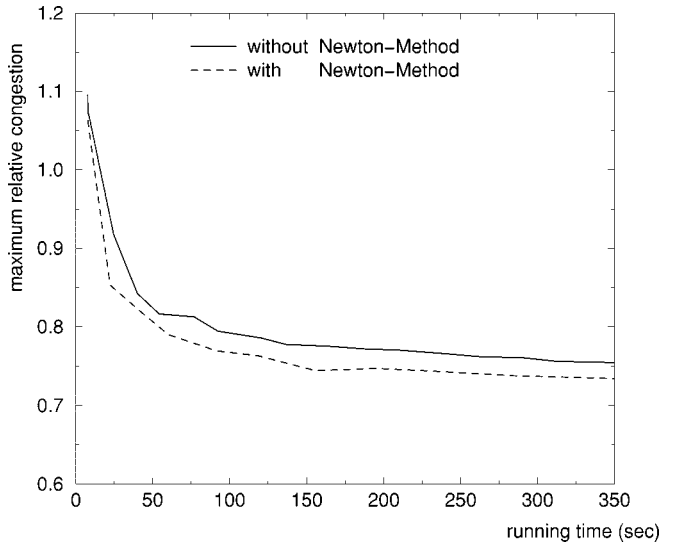


Fig. 6. Comparison of relative congestion in terms of running time with and without Newton method ($\gamma = 10$, $\epsilon = 4.0$).

After the new coefficients for the fractional solution of net i have been computed, the dual variables have to be recomputed also

$$y_e := \frac{1}{c(e)} \exp\left(\epsilon \left(\sum_{i,T:e\in T\in\mathcal{T}_i} x_{i,T}\right)/c(e)\right)$$

for all $e \in E$.

It is possible to implement the Newton method in such a way that to compute the step size σ for one net i, only those edges have to be considered that belong to one of the Steiner tree already computed for net i.

In practice, a better relative congestion of the edges could be achieved by the Newton method in the same running time. Fig. 6 shows the results with and without the Newton method for the test case quorina.

If the step size σ is greater than $1/p$, the relative congestion for some edges decreases and so does the value of some dual variables y_e. To derive (4), y_e increases only. It is an open problem to derive a polynomial bound on the number of phases under the assumption that the step size can computed optimally by the Newton method.

We have a gradient method with line search. The Newton method is a heuristic that gives better results in practice.

VI. Optimality of the Solution by Lexicographic Order

The approximation algorithms, which minimize the potential $\Phi = \sum_{e\in E} e^{\alpha\lambda(e)}$, does not only minimize the maximum relative congestion of the edges.

If the chip was split into different regions, which are either separated from each other by blockages or are far away from each other such that the edges in between the regions are not used, the algorithms would compute a solution such that the maximum relative congestion of each region is minimum. Here, we show that they "work toward" a solution for that the vector of the relative congestion of the edges sorted in nonincreasing order is minimal by lexicographic order.[1]

The following lemma shows the following. Suppose the vector x' is a solution for the fractional global routing problem such that the congestion of the edges is not optimal by lexicographic order and assume that for given α, the algorithm could compute a solution that minimizes the potential Φ, then there exists an α_0 such that for all $\alpha > \alpha_0$ the algorithm could not come up with the solution x'. In the lemmaf, $\lambda_1, \ldots, \lambda_m$ represents the congestion of the optimum solution and $\lambda'_1, \ldots, \lambda'_m$ of the solution x'.

Lemma 1: Let $\lambda_1 \geq \cdots \geq \lambda_m$, $\lambda'_1 \geq \cdots \geq \lambda'_m$, and $(\lambda_1, \ldots, \lambda_m) < (\lambda'_1, \ldots, \lambda'_m)$ by lexicographic order. Then, there exists α_0 such that for all $\alpha > \alpha_0$ the following holds:

$$\sum_{i=1}^{m} e^{\alpha\lambda_i} < \sum_{i=1}^{m} e^{\alpha\lambda'_i}.$$

Proof: Assume $\lambda_j = \lambda'_j$ for $j = 1, \ldots, k$ and $\lambda_{k+1} < \lambda'_{k+1}$. Then

$$\begin{aligned}\sum_{i=1}^{m} e^{\alpha\lambda'_i} - \sum_{i=1}^{m} e^{\alpha\lambda_i} &= \sum_{i=k+1}^{m} e^{\alpha\lambda'_i} - \sum_{i=k+1}^{m} e^{\alpha\lambda_i} \\ &> e^{\alpha\lambda'_{k+1}} - (m-k)e^{\alpha\lambda_{k+1}} \\ &= e^{\alpha\lambda'_{k+1}} - e^{\ln(m-k)+\alpha\lambda_{k+1}}.\end{aligned}$$

Choose $\alpha_0 := \ln(m-k)/(\lambda'_{k+1} - \lambda_{k+1})$. For $\alpha = \alpha_0$, the last expression is zero because then $\alpha\lambda'_{k+1} = \ln(m-k) + \alpha\lambda_{k+1}$ and for $\alpha > \alpha_0$, the last expression is greater than zero. ■

In practice, it is desirable not only to minimize the maximum relative congestion of the edges. If the congestion of the edges is smaller, extra spacing between the wires to ensure signal integrity is possible. The coupling capacitance is smaller and so is crosstalk. Finally, the running time of the local router decreases if the congestion is smaller.

Another advantage compared to ripup and reroute algorithms that minimize only the maximum relative congestion of the edges is that the congested areas can be detected easier from congestion maps (a picture of the chip where the congestion of the edges is displayed by different colors). This is important if a solution for the global routing problem does not exist. The following theorem shows that congestion maps for solutions that are optimal by lexicographic order look exactly the same.

Theorem 3: Let x and x' be two solutions for the fractional global routing problem such that the vector of the relative congestion of the edges sorted in nonincreasing order is minimal by lexicographic order.

Then, the relative congestion of each edge is the same for both solutions.

Proof: Suppose the relative congestion of the edges is different. Let us consider the solution $\overline{x} := 1/2(x + x')$. If $\lambda(e)$ is the relative congestion of edge e for x and $\lambda'(e)$ resp. for x', then $\overline{\lambda}(e) := 1/2(\lambda(e) + \lambda'(e))$ is the relative congestion of edge e for $\overline{x}$.

Choose $\hat{e} \in E$ such that $\overline{\lambda}(\hat{e})$ is maximal among all edges $e \in E$ with $\lambda(e) \neq \lambda'(e)$, and let us assume w. l. o. g. $\lambda(\hat{e}) > \lambda'(\hat{e})$. Then, $\lambda(e) = \overline{\lambda}(e)$ for all edges $e \in E$ with $\overline{\lambda}(e) > \overline{\lambda}(\hat{e})$, but for

[1] $(\lambda_1, \ldots, \lambda_m) < (\lambda'_1, \ldots, \lambda'_m)$ by lexicographic order means: there exists i with $1 \leq i \leq m$ such that $\lambda_j = \lambda'_j$ for $j < i$ and $\lambda_i < \lambda'_i$.

edge $\hat{e}$, we have $\lambda(\hat{e}) > \overline{\lambda}(\hat{e})$ such that the relative congestion of the edges for $\overline{x}$ is better by lexicographic order than for x. ■

VII. Minimizing the Total Netlength

Minimizing the maximum relative congestion is not the only objective in global routing. Here, we show how to modify the algorithm described in Section III such that the total weighted netlength is considered and minimized subject to the condition that the maximum relative congestion of the edges is at most 1. We follow the approach by Garg and Könemann [14] for the minimum cost multicommodity flow problem.

In addition to the capacity $c(e)$ for each edge $e = \{u, v\}$, the L_1 length $l(e)$ is given, that is, for an edge in x or y direction, the distance between the midpoints of the tiles corresponding to u and v. For each net i, a weight $g_i \in \mathbb{R}_+$ is given. Let L be a target for the total weighted netlength; then, the constraint

$$\sum_{i=1}^{k} g_i \sum_{T \in \mathcal{T}_i} \left(\sum_{e \in T} l(e) \right) x_{i,T} \leq \lambda L \tag{7}$$

is added to the linear program (1) of the fractional global routing problem. This constraint is very similar to the capacity constraints for the edges, the first constraint in (1), and the algorithm can be modified to treat this new constraint in the same way as the capacity constraints. To minimize the total weighted netlength, we want L to be as small as possible such that λ, the maximum relative congestion, is at most one. This is achieved by binary search over L. In practice, the netlength in the final solution is only slightly higher compared to the minimum netlength if each Steiner tree is as short as possible ignoring capacities. This gives a good estimate for L.

For the dual of the linear program an additional dual variable y_L for constraint (7) is needed. The dual linear program is given by

$$\max \sum_{i=1}^{k} z_i$$

subject to

$$\sum_{e \in E} c(e) y_e + L y_L = 1$$

$$\sum_{e \in T} (w_{i,e} y_e + g_i l(e) y_L) \geq z_i, \qquad \text{for } i = 1, \ldots, k;\ T \in \mathcal{T}_i$$

$$y_e \geq 0, \qquad \text{for } e \in E$$

$$y_L \geq 0.$$

Fig. 7 shows the modified algorithm. During the algorithm, minimal Steiner trees are computed with respect to the length $w_{i,e} y_e + g_i l(e) y_L$ for net i and edge $e \in E$ (line 10). The length of an edge e is a linear combination of $w_{i,e}$, measuring how much the edge is used, and the weighted L_1 length $g_i l(e)$ of the edge.

With the assumption that $g_i \sum_{e \in T_i} l(e) \leq L$ for each Steiner tree T_i found in line 10 (which usually holds for the global routing problem), the proof in Section III can be extended to show that a $(1 + \epsilon_0)$ approximation for the fractional global routing problem with constraint (7) is found by the algorithm in Fig. 7 in $O(1/(\epsilon_0^2 \lambda^*) \ln m)$ phases. If for some Steiner tree

(1) Set $y_e := \frac{1}{c(e)}$ for all $e \in E$ and $y_L := \frac{1}{L}$.
(2) Set $x_{i,T} := 0$ for $i = 1, ..., k$; $T \in \mathcal{T}_i$.
(3) Set $T_i := \emptyset$ for $i = 1, ..., k$.
(4) While $\left(\sum_{e \in E} c(e) y_e + L y_L < M \right)$
(5) **begin**
(6) For $i := 1$ to k
(7) **begin**
(8) If $T_i = \emptyset$ or $\sum_{e \in T_i} (w_{i,e} y_e + g_i l(e) y_L) > (1 + \gamma\epsilon) z_i$
(9) **begin**
(10) Find a minimal Steiner tree $T_i \in \mathcal{T}_i$ for net i with respect to lengths $(w_{i,e} y_e + g_i l(e) y_L)$, $e \in E$.
(11) Set $z_i := \sum_{e \in T_i} (w_{i,e} y_e + g_i l(e) y_L)$.
(12) **end**
(13) Set $x_{i,T_i} := x_{i,T_i} + 1$.
(14) Set $y_e := y_e \, e^{\epsilon \frac{w_{i,e}}{c(e)}}$ for all $e \in T_i$.
Set $y_L := y_L \, e^{\epsilon \frac{g_i \sum_{e \in T_i} l(e)}{L}}$.
(15) **end**
(16) **end**

Fig. 7. Approximation algorithm for fractional global routing regarding the total weighted netlength.

T_i, we have $g_i \sum_{e \in T_i} l(e) > L$, variable x_{i,T_i} in line 13 is increased only by $L/(g_i \sum_{e \in T_i} l(e))$, and another Steiner tree has to be found for the same net until the total increment of $\sum_{T \in \mathcal{T}_i} x_{i,T}$ is one (for details, see [14]).

To apply the Newton method as in Section V, we define

$$\lambda(r) := \frac{1}{L} \sum_{i=1}^{k} g_i \sum_{T \in \mathcal{T}_i} \left(\sum_{e \in T} l(e) \right) x_{i,T}$$

for constraint (7) and the potential $\Phi := \sum_{e \in E \cup \{r\}} e^{\alpha \lambda(e)}$.

Higher values for L relax constraint (7) and, as a result, the congestion of the edges is decreased further.

VIII. Computational Results

We have implemented the algorithm in C. All runs are on an IBM workstation RS/6000 model 270. The global routing algorithm has been used together with a local router in the routing package XRouter (see [20]–[22]) for several IBM processor chips.

For computing the Steiner trees, we use a subroutine that computes the optimum Steiner tree for nets with two or three terminals. For nets with more terminals, it is a heuristic: it computes the Steiner points of the minimal Steiner tree with respect to L_1 length (see [20]), then performs a Dijkstra search using the Steiner points as additional targets. However, these Steiner points do not necessarily have to belong to the Steiner tree, as antennas are removed in the end.

The condition in line 4 of the algorithm in Figs. 2 and 7 is not checked. Instead, the algorithm stops when a desired maximum relative congestion is achieved.

Table I shows some chips with their grid size, number of global routing nets (nets with pins in different tiles), average number of pins per net, and average capacity for the edges in

TABLE I
CHARACTERISTICS OF CHIPS FOR THE GLOBAL ROUTING PROBLEM

Chip	grid size	number of global routing nets	average number of pins per net	average edge capacity x	y	z
PRIZMA	256 × 252	506 884	3.11	124	119	29
630fpp	313 × 309	263 563	2.83	62	57	40
MBA00	200 × 204	696 377	2.75	99	113	26
WIND_ICP	235 × 286	591 006	3.13	65	79	46
quorina	112 × 99	46 368	2.73	61	61	45
m_3l	87 × 310	68 048	2.71	18	24	40

TABLE II
MINIMIZING THE MAXIMUM RELATIVE CONGESTION

Chip	phases	maximum relative congestion	lower bound	running time (sec)
PRIZMA	5	0.955	0.770	7 791
	10	0.945	0.859	23 016
630fpp	5	0.816	0.603	33 635
	10	0.780	0.592	41 510
MBA00	5	0.889	0.733	1 076
	10	0.862	0.762	3 672
	15	0.849	0.783	7 257
	20	0.843	0.787	11 741
WIND_ICP	5	0.927	0.718	2 509
	10	0.886	0.713	8 467
	15	0.861	0.720	15 871
	20	0.861	0.720	25 211
	25	0.853	0.714	35 194
quorina	5	0.775	0.633	74
	10	0.730	0.652	264
	15	0.724	0.658	521
	20	0.718	0.684	829
	25	0.715	0.688	1 200
m_3l	5	1.025	0.716	791
	10	0.999	0.571	2 421
	15	0.961	0.569	4 274
	20	0.949	0.537	6 231
	25	0.935	0.636	8 284

x-, y-, and z direction. PRIZMA is a chip used for telecommunication, 630 fpp is a followup of the Power3 microprocessor, and MBA00 is part of the S/390 processor chip set. All other chips are different ASICs.

All chips are placed flat, except for 630 fpp and m_3l, which are hierarchically designed and placed in blocks. The difference for the global routing problem is that these two chips have more nets which are longer. The chip m_3l has seven layers of metal for routing, chip 630fpp, PRIZMA, and MBA00 have six routing layers, and WIND_PCI has five layers of metal. The chip quorina has only three routing layers. The chip WIND_ICP is in technology SA-12E, the chip PRIZMA is SA-27 (for details see [23]).

In the current implementation, all routing layers in x direction are condensed into a single layer for global routing and the same is done for all routing layers in y direction. The graph for the global routing problem is the same as in Fig. 1.

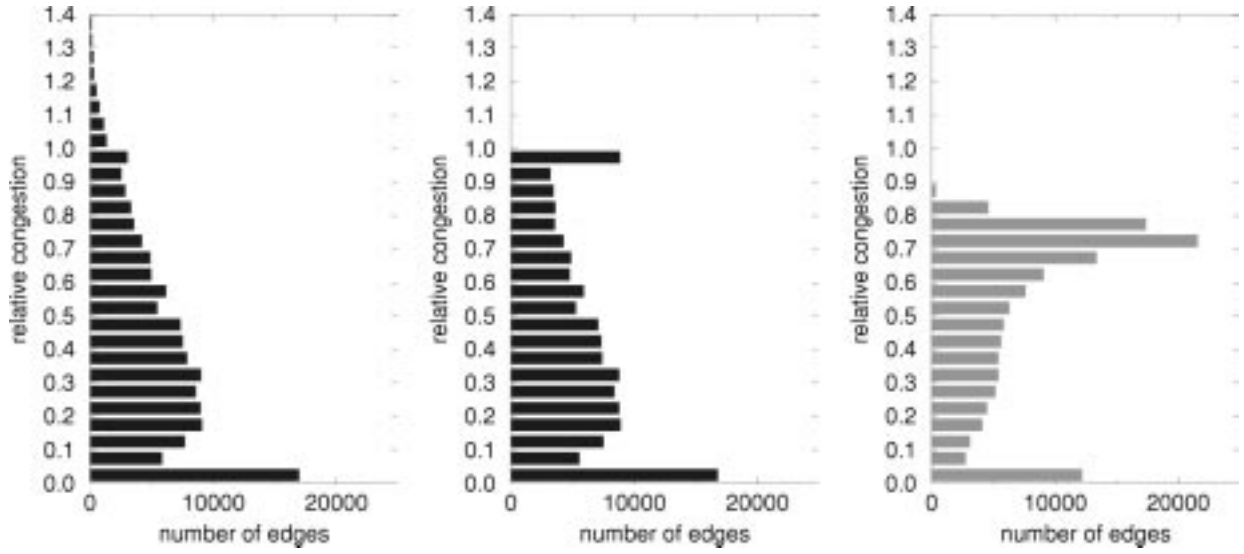
TABLE III
MINIMIZING CONGESTION AND TOTAL WEIGHTED NETLENGTH

Chip and target netlength (L)	phases	maximum relative congestion	total weighted netlength (meter)	running time (sec)
PRIZMA	5	0.956	378.5	5 264
	10	0.945	352.3	9 085
	15	0.942	344.6	12 335
	20	0.941	342.3	15 705
L = 335.4	25	0.946	341.0	18 904
630fpp	5	0.834	1046.2	6 065
	10	0.894	994.9	10 380
	15	0.894	991.4	12 159
	20	0.950	989.8	13 795
L = 980.1	25	0.951	989.6	15 341
MBA00	5	0.924	577.3	926
	10	0.932	544.6	1 943
	15	0.967	536.6	2 548
	20	0.978	534.2	3 151
L = 530.5	25	0.981	533.3	3 660
WIND_ICP	5	0.938	262.4	1 786
	10	0.917	243.6	3 290
	15	0.943	237.9	4 384
	20	0.948	236.6	5 519
L = 231.5	25	0.961	235.6	6 512
quorina	5	0.862	9.2	25
	10	0.915	8.8	44
	15	0.934	8.7	67
	20	0.959	8.6	86
L = 8.4	25	0.964	8.6	109
m_3l	5	1.027	55.1	420
	10	1.039	50.5	760
	15	1.010	48.5	1 080
	20	1.000	47.9	1 404
L = 45.8	25	1.010	47.4	1 714

Table II shows the running times of the approximation algorithm in Fig. 2 with the Newton method (Section V) for these chips and how the maximum relative congestion and the lower bound improve during the algorithm. The running time to compute the lower bound (compute a minimal Steiner tree for each net according to Theorem 1 at the end of a phase) is not included in the time given in the last column. The lower bound is based on the assumption that the subroutine computes the optimum Steiner tree, which is not the case for nets with more than three terminals.

Table III shows the results of the algorithm in Fig. 7, which also considers the netlength. In the fourth column, the weighted netlength of the fractional solution is shown. For the target netlength L, we always chose the minimum netlength (each Steiner tree routed as short as possible—ignoring capacities), which is found in the first column.

The computational results also show that the running time needed for a phase is smaller if the netlength is also considered. The reason is the following. The dual variables y_e increase more for highly congested edges and these edges build up barriers that have to be broken by the Dijkstra search for those nets that need to cross these barriers and, hence, more nodes have to be labeled. If the netlength is also considered, the length of an edge is the sum of the usage $w_{i,e}y_e$ and the weighted L_1 length $g_i l(e) y_L$ and the Dijkstra search can go straighter to the target if the terminals of the net lie in uncongested regions. For the same reason, the running time for a phase increases during the algorithm if the netlength is not considered.



(a) (b) (c)

Fig. 8. Congestion of edges for the chip MBA00. (a) Initial solution for the ripup and reroute algorithm. (b) Final solution of the ripup and reroute algorithm. (c) Fractional solution after ten phases of the approximation algorithm in Fig. 7, which also considers the netlength.

We compared the algorithm with the following ripup and reroute algorithm: In the first phase each net is routed with minimum L_1 length. The length of the overloaded edges is increased by a very small value, such that the total number of overloaded edges used is minimized as a second objective for the case that there are several routes of minimum L_1 length.

In a second phase, the congestion of overloaded edges is reduced by rerouting all segments crossing an edge with maximum overload and finally exchanging that segment for which the length increased by as little as possible. When segments are rerouted, edges with the maximum overload are either forbidden or the edges have high exponential cost as a function of their congestion.

For chip PRIZMA, this ripup and reroute algorithm failed and increased the netlength by more than 3% starting from the minimum netlength (each net routed separately), whereas the approximation algorithm found a solution with the total netlength being only 1.7% greater than the minimum netlength.

On easy instances, for example, chip 630 fpp, routing each net with minimum L_1 length gave almost a feasible solution. In this case, the running time of the approximation algorithm was much higher and there was almost no improvement in the quality of the solution.

Fig. 8 shows a comparison of the congestion of the edges for the ripup and reroute algorithm and of the approximation algorithm: (a) shows the congestion of the edges for the initial solution (each net routed as short as possible but overloaded edges are already avoided); (b) the final solution of the ripup and reroute algorithm; and (c) shows the fractional solution after 10 ten phases of the approximation algorithm in Fig. 7, which also considers the netlength. The number of edges, which are highly congested (for example, have a relative congestion between 0.85 and 1.0), is considerably smaller.

After solving the fractional global routing problem, randomized rounding is applied. For each net i, independently one Steiner tree is chosen at random and the probability that Steiner tree T is chosen is $x_{i,T}$. For details, see [4] and [5]. Randomized rounding is not at all critical during the whole process. We have not even implemented a derandomized version; we simply perform 100 randomized rounding steps and chose the solution for which the maximum relative congestion is minimum. The maximum relative congestion increased only on very few edges and by using the ripup and reroute algorithm, the congestion could be reduced to almost the value of the solution for the linear program in a few seconds.

For the chip MBA00, the distribution of the congestion of the edges after randomized rounding looked exactly the same as the distribution before randomized rounding. There was no difference in the Fig. 8(c).

We would also like to mention another advantage that we experienced by our comparison with the ripup and reroute algorithm. For each net, we computed the factor of how much longer the net is in the final solution compared to the minimum length possible. For the ripup and reroute algorithm, there were several nets with a high factor; for the approximation algorithm (the version which considers the netlength), the maximum factor and also the number of nets with a high factor was much smaller. When a ripup and reroute algorithm decreases the congestion of an edge, it has a very limited view and limited number of choices, so it may have to increase the length of the Steiner tree for a net substantially.

IX. Conclusion

We have presented an approximation algorithm for solving the linear programming relaxation of the global routing problem. If a solution for the fractional global routing problem with maximum relative congestion at most one does not exist, the global routing problem is not solvable. In this case, the algorithm provides a proof by the dual solution. This is one of the advantages over many other algorithms for global routing.

Our computational results show that the algorithm is superior on difficult instances where ripup and reroute algorithms fail.

ACKNOWLEDGMENT

The author would like to thank B. Korte for his continuous support, L. Fleischer for helpful discussions, and A. Rohe, K. Köhler, and J. Werber for joint development of XRouter. The author would also like to thank the IBM teams in Böblingen, Fishkill, and Austin for feedback, test cases, and joint development of XRouter.

REFERENCES

[1] B. S. Ting and B. N. Tien, "Routing techniques for gate array," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 301–312, Oct. 1983.

[2] E. Shragowith and S. Keel, "A global router based on a multicommodity flow model," *Integr. VLSI J.*, vol. 5, no. 2, pp. 3–16, 1987.

[3] G. Meixner and U. Lauther, "A new global router based on a flow model and linear assignment," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1990, pp. 44–47.

[4] P. Raghavan and C. D. Thompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[5] ——, "Multiterminal global routing: A deterministic approximation," *Algorithmica*, vol. 6, pp. 73–82, 1991.

[6] K. Aoshima and E. S. Kuh, "Multi-channel optimization in gate-array LSI layout," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1983, pp. 1005–1008.

[7] T. C. Hu and M. T. Shing, "A decomposition algorithm for circuit routing," in *VLSI Circuit Layout: Theory and Design*, T. C. Hu and E. S. Kuh, Eds. New York: IEEE Press, 1985, pp. 144–152.

[8] J. Huang, X.-L. Hong, C.-K. Cheng, and E. S. Kuh, "An efficient timing-driven global routing algorithm," in *Proc. 30th ACM/IEEE Design Automation Conf.*, June 1993, pp. 596–600.

[9] R. C. Carden IV, J. Li, and C.-K. Cheng, "A global router with a theoretical bound on the optimum solution.," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 208–216, Feb. 1996.

[10] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *J. Assoc. Comput. Mach.*, vol. 37, no. 2, pp. 318–334, Apr. 1990.

[11] R. J. Brouwer and P. Banerjee, "PHIGURE: A parallel hierarchical global router," in *Proc. 27th ACM/IEEE Design Automation Conf.*, June 1991, pp. 650–653.

[12] W.-M. Dai and E. S. Kuh, "Simultaneous floor planning and global routing for hierarchical building-block layout," *IEEE Trans. Computer-Aided Design*, vol. 6, pp. 828–837, May 1997.

[13] J. Cong and B. Preas, "A new algorithm for standard cell global routing," *Integr. VLSI J.*, vol. 14, no. 1, pp. 49–65, 1992.

[14] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *Proc. 39th Annu. Symp. Foundations of Computer Science*, Nov. 1998, pp. 300–309.

[15] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," in *Proc. 40th Annu. Symp. Foundations of Computer Science*, Oct. 1999, pp. 24–31.

[16] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Berlin, Germany: Springer-Verlag, 2000.

[17] A. V. Goldberg, J. D. Oldham, S. Plotkin, and C. Stein, "An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow," in *Proc. 6th Int. Integer Programming and Combinatorial Optimization Conf.*, June 1998, pp. 338–352.

[18] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," in *Proc. Int. Symp. Physical Design*, Apr. 2000, pp. 19–25.

[19] S. Plotkin, D. Shmoys, and É. Tardos, "Fast approximation algorithms for fractional packing and covering problems," *Math. Oper. Res.*, vol. 20, no. 2, pp. 257–301, May 1995.

[20] A. Hetzel, "Verdrahtungsprobleme im VLSI-Design: Spezielle Teilprobleme und ein sequentielles Lösungsverfahren," Ph.D. dissertation (in German), Univ. Bonn, Bonn, Germany, 1995.

[21] ——, "A sequential detailed router for huge grid graphs," in *Proc. Design, Automation, Test Eur.*, Feb. 1998, pp. 332–338.

[22] A. Rohe and M. Zachariasen, "Rectilinear group steiner trees and application in VLSI-design," Res. Inst. Discrete Mathematics, Univ. Bonn, Bonn, Germany, Tech. Rep. 00906, 2000.

[23] IBM Technol. [Online]. Available: http://www.chips.ibm.com/products/asics/products/stdcell.html

[24] R. C. Carden IV and C. K. Cheng, "A global router using an efficient approximate multicommunity multiterminal flow approximation," in *Proc. 28th ACM/IEEE Design Automation Conf.*, June 1991, pp. 316–321.

[25] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM J. Discrete Math.*, vol. 13, no. 4, pp. 505–520, 2000.

Christoph Albrecht received the Diploma and Ph.D. degree in mathematics from the University of Bonn, Bonn, Germany, in 1996 and 2001, respectively.

Since 1994, he has been with the Research Institute for Discrete Mathematics, the University of Bonn. His research interests include combinatorial optimization and the application of algorithms in VLSI design.