Provably good global routing by a new approximation algorithm for multicommodity flow

Christoph Albrecht
Research Institute for Discrete Mathematics, University of Bonn
Lennéstr. 2, 53113 Bonn, Germany
albrecht@or.uni-bonn.de

ABSTRACT

We show how the new approximation algorithms by Garg and Könemann for the multicommodity flow problem can be modified to solve the linear programming relaxation of the global routing problem. The algorithm presented here also provides a solution of the dual linear program, thus gives a lower bound on the optimum maximum relative congestion. 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 rip-up and reroute algorithms fail.

1. INTRODUCTION

Routing is usually split up into two subtasks: global routing, which gives the general 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 rip-up and reroute procedure which 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.
Our algorithm is based on a linear programming relaxation and randomized rounding. This approach is not new. It was proposed by Raghavan and Thompson [14], [15]. 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.
Work on linear programming approaches has already been done by Aoshima and Kuh [1] and Huang, Hong, Cheng and Kuh [10]. Some LP methods are based on network flow models, namely Shragowith and Keel [17], Meixner and Lauther [12] and Okamoto, Ishikasa and Fujita [13]. Carden IV and Cheng [3] use the approximation algorithm for multicommodity flow by Shahrokhi and Matula.

Other global routers use a hierarchical decomposition approach, for example Brouwer and Banerjee [2] and Dai and Kuh [5]. Cong and Preas [4] give a spanning forest formulation.
Our contribution is (1) to show that the new combinatorial approximation algorithm by Garg and Könemann [7] for the maximum concurrent flow problem can be modified to solve the linear programming relaxation of the global routing problem, and (2) to show that an idea introduced by Fleischer [6] 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 theoretical worst-case running time remains the same. Finally, (3) our computational results show for the first time that these approximation algorithms give good results in practice.
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 which 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.
Finally, it is possible to minimize the total netlength with respect to a given maximum relative congestion.
The rest of this paper is organized as follows: In Section 2 we introduce some notation and terminology, formulate a mixed integer program for global routing and give the linear programming relaxation. In Section 3 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 4 we show how the total netlength can be minimized, and in Section 5 we describe our computational results.

2. 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.

Figure 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. For each net i, $i = 1, ..., k$, 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 $\mathcal{T}_i = \{T_{i,1}, ..., T_{i,l_i}\}$ 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 which does not necessarily return the optimum Steiner tree is often good enough. However, to compute a lower bound on the optimum solution, a subroutine which 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 other versions of the global routing problem, for example to find Steiner trees such that the maximum relative congestion is at most 1 and the total netlength is minimized.

With this notation the *global routing problem* can be formulated as a mixed integer program:

$$\min \lambda$$

subject to

$$\begin{array}{rcll} \sum\limits_{i,j:e\in T_{i,j}} w_{i,e}x_{i,j} & \leq & \lambda c(e) & \text{for } e \in E \\ \sum\limits_{j=1}^{l_i} x_{i,j} & = & 1 & \text{for } i = 1, ..., k \\ x_{i,j} & \in & \{0,1\} & \text{for } i = 1, ..., k; \\ & & & \quad j = 1, ..., l_i \end{array} \qquad (1)$$

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

$$\begin{array}{rcll} x_{i,j} & \geq & 0 & \text{for } i = 1, ..., k; \\ & & & \quad j = 1, ..., l_i \end{array}$$

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 $\lambda(e) := \sum_{i,j:e\in T_{i,j}} w_{i,e}x_{i,j}/c(e)$.

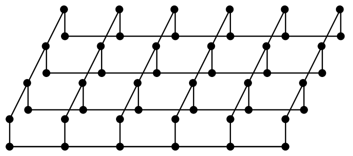
Figure 1: A global routing graph with two layers and via edges.

The dual of this linear program is given by

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

subject to

$$\begin{array}{rcll} \sum\limits_{e\in E} c(e)y_e & = & 1 & \\ \sum\limits_{e\in T_{i,j}} w_{i,e}y_e & \geq & z_i & \text{for } i = 1, ..., k; \\ & & & \quad j = 1, ..., l_i \\ y_e & \geq & 0 & \text{for } e \in E \end{array} \qquad (2)$$

By linear programming duality (a good overview can be found in [11]) 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.

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*

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

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. □

3. SOLVING THE FRACTIONAL GLOBAL ROUTING PROBLEM

Figure 2 shows the approximation algorithm which solves the fractional global routing problem for any given approximation ratio ρ. The implementation (for example the variables used) is different and will be discussed in Section 5.

The variables are initialized in lines 1 to 3. δ is a constant. The proof of the theorem in this section will show which value to choose for δ 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 $j_i = 0$ and therefore for each net i a minimal Steiner tree

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

Figure 2: Approximation algorithm for fractional global routing

$T_{i,j_i} \in \mathcal{T}_i$ with respect to lengths $w_{i,e} y_e$, $e \in E$ is computed (line 10). Variable j_i stores the index of 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,j_i} is increased by one. To achieve that for each net i the variables $x_{i,j}$, $j = 1, ..., l_i$, sum up to one, all variables $x_{i,j}$ are divided by the total number of phases at the end of the algorithm. Finally, in line 14 the dual variables y_e are increased for all edges used by the Steiner tree T_{i,j_i}; 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 [7] for the maximum concurrent multicommodity flow problem. This idea was used by Fleischer [6] to reduce the theoretical worst-case running time for the maximum multicommodity flow problem.
Here we show that it does not increase the worst-case running time complexity, and results in Section 5 show that it decreases the running time in practice.

THEOREM 2. *If there exists a solution for the fractional global routing problem with maximum relative congestion smaller than 1, the algorithm finds a ρ-approximation in at most*

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

phases, if $\epsilon' = 1 - \left(\frac{1}{\rho}\right)^{\frac{1}{3}}$ *and* $\epsilon = \sqrt{\frac{1}{\gamma}(\epsilon' - \frac{1}{4\gamma})} - \frac{1}{2\gamma}$. *Moreover, the variables y_e, $e \in E$, provide a ρ–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 $\lambda^* \geq \frac{1}{2}$.
To prove this theorem, we basically follow the proof by Garg and Könemann [7], 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 not increased by too much (line 8)) do not increase the worst-case running time complexity.

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 $y_e^{(p)}$ be the value of variable y_e after phase p. At the beginning we have

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

When the dual variables y_e are increased in line 14 after a Steiner tree T_{i,j_i} has been found, the expression $\sum_{e \in E} c(e) y_e$ increases by

$$\sum_{e \in T_{i,j_i}} c(e) y_e \epsilon \frac{w_{i,e}}{c(e)} = \epsilon \sum_{e \in T_{i,j_i}} w_{i,e} y_e$$

(here y_e are the old values before the increment in line 14). Whenever a minimal Steiner tree is found in line 10, its length is stored in variable z_i (line 11). 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:

$$\sum_{e \in T_{i,j_i}} w_{i,e} y_e \leq (1 + \gamma\epsilon) z_i \leq (1 + \gamma\epsilon) \min_{T \in \mathcal{T}_i} \sum_{e \in T} w_{i,e} y_e,$$

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

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

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

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

Using this inequality, inequality (4) can be rewritten

$$\sum_{e \in E} c(e) y_e^{(p)} \leq \sum_{e \in E} c(e) y_e^{(p-1)} + \epsilon' \lambda^* \sum_{e \in E} c(e) y_e^{(p)},$$

which can be transformed to

$$\sum_{e \in E} c(e) y_e^{(p)} \leq \frac{1}{1 - \epsilon' \lambda^*} \sum_{e \in E} c(e) y_e^{(p-1)}.$$

With equation (3) we get:

$$\begin{aligned}\sum_{e\in E} c(e)y_e^{(p)} &\le \frac{\delta m}{(1-\epsilon'\lambda^*)^p}\\ &= \frac{\delta m}{(1-\epsilon'\lambda^*)}\left(1+\frac{\epsilon'\lambda^*}{1-\epsilon'\lambda^*}\right)^{p-1}\\ &\le \frac{\delta m}{(1-\epsilon')}\left(1+\frac{\epsilon'\lambda^*}{1-\epsilon'}\right)^{p-1} \qquad (5)\\ &\le \frac{\delta m}{(1-\epsilon')}e^{\frac{\epsilon'\lambda^*(p-1)}{1-\epsilon'}}.\end{aligned}$$

For the last inequality the fact $1+x \le e^x$ 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, and let the jth increment in the relative congestion of edge e be $a_j := \frac{w_{i,e}}{c(e)}$ for the appropriate i. After rescaling the variables $x_{i,j}$, the relative congestion of edge e is $\lambda(e) = \sum_{j=1}^{s} a_j/(t-1)$. Since $y_e^{(0)} = \frac{\delta}{c(e)}$ and $y_e^{(t-1)} < \frac{1}{c(e)}$ (because the condition in line 4 still holds before the last phase is executed) and since

$$y_e^{(t-1)} = \frac{\delta}{c(e)}\prod_{j=1}^{s}(1+\epsilon' a_j),$$

we get

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

As $(1+\epsilon)^a \le 1+\epsilon a$ for $0 \le a \le 1$ (the expression $(1+\epsilon)^a$ is convex in a, $1+\epsilon a$ is linear, and equality holds for $a=0$ and $a=1$), it follows that

$$(1+\epsilon')^{\sum_{j=1}^{s} a_j} \le \frac{1}{\delta},$$

and hence we get

$$\lambda(e) = \frac{\sum_{j=1}^{s} a_j}{t-1} \le \frac{\ln_{1+\epsilon'}\left(\frac{1}{\delta}\right)}{t-1}. \qquad (6)$$

Since $\sum_{e\in E} c(e)y_e^{(t)} \ge 1$, solving inequality (5) with $p=t$ for λ gives a lower bound on the optimum solution value:

$$\lambda^* \ge \frac{1-\epsilon'}{\epsilon'(t-1)}\ln\left(\frac{1-\epsilon'}{\delta m}\right),$$

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

$$\begin{aligned}\frac{\max_{e\in E}\lambda(e)}{\lambda^*} &\le \frac{\frac{\ln_{(1+\epsilon')}\left(\frac{1}{\delta}\right)}{t-1}}{\frac{1-\epsilon'}{\epsilon'(t-1)}\ln\left(\frac{1-\epsilon'}{\delta m}\right)}\\ &= \frac{\epsilon'}{(1-\epsilon')\ln(1+\epsilon')}\frac{\ln\left(\frac{1}{\delta}\right)}{\ln\left(\frac{1-\epsilon'}{\delta m}\right)}.\end{aligned}$$

If δ is now chosen to be $\delta := \left(\frac{m}{1-\epsilon'}\right)^{-\frac{1}{\epsilon'}}$ we get

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

such that

$$\begin{aligned}\rho &\le \frac{\epsilon'}{(1-\epsilon')^2\ln(1+\epsilon')}\\ &\le \frac{\epsilon'}{(1-\epsilon')^2(\epsilon'-\frac{\epsilon'^2}{2})}\\ &\le \frac{1}{(1-\epsilon')^3}.\end{aligned}$$

Given a desired approximation ration ρ we choose ϵ' such that $\rho = \frac{1}{(1-\epsilon')^3}$ and ϵ such that $\epsilon' = \epsilon(1+\gamma\epsilon)$. Since $\max_{e\in E}\lambda(e) \ge \lambda^*$ we get from (6) that

$$\lambda^* \le \frac{\ln_{1+\epsilon'}\left(\frac{1}{\delta}\right)}{t-1},$$

which means that the maximum number of phases is bounded by:

$$\begin{aligned}t &\le 1+\frac{\ln_{1+\epsilon'}\left(\frac{1}{\delta}\right)}{\lambda^*}\\ &= 1+\frac{1}{\epsilon'\lambda^*}\ln_{1+\epsilon'}\left(\frac{m}{1-\epsilon'}\right).\end{aligned}$$

□

Interestingly, this algorithm does not only minimize the maximum relative congestion of the edges. If the algorithm were run on two completely separate regions of a chip, it would minimize the maximum relative congestion on each of these regions. In total the number of congested edges is much smaller and thus it helps to find the congested regions where the placement needs to be changed.

4. 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 3 such that the total netlength is considered and minimized subject to the condition that the maximum relative congestion of the edges is at most 1. We follow the idea by Garg and Könemann [7] 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. Let L be a target for the total netlength; then the constraint

$$\sum_{i=1}^{k}\sum_{j=1}^{l_i}\left(\sum_{e\in T_{i,j}} l(e)\right)x_{i,j} \le \lambda L \qquad (7)$$

is added to the linear program (1). 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 netlength, we want L to be as small as possible such that λ, the maximum relative congestion, is at most 1. This is achieved by binary search over L.

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

Figure 3: Approximation algorithm for fractional global routing regarding the total netlength.

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

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

subject to

$$\begin{array}{rcll} \sum_{e \in E} c(e) y_e + L y_L & = & 1 & \\ \sum_{e \in T_{i,j}} (w_{i,e} y_e + l(e) y_L) & \geq & z_i & \text{for } i = 1, ..., k; \\ & & & \quad j = 1, ..., l_i \\ y_e & \geq & 0 & \text{for } e \in E \\ y_L & \geq & 0 & \end{array}$$

Figure 3 shows the modified algorithm. The lines which have been modified have a bold number.

It is also possible to modify the algorithm such that the total weighted netlength is minimized, where a weight according to the timing criticality for each net is given. The additional modifications of the algorithm are straightforward.

5. COMPUTATIONAL RESULTS

We have implemented the algorithm in C. All runs are on an IBM RISC System/6000 Model 260. The global routing algorithm has been used together with a local router in the routing package XRouter (see [8], [9] and [16]) for several IBM processor chips.

Instead of having a variable $x_{i,j}$ for each possible Steiner tree (which would be impossible because of memory limitations), 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.

Chip	grid size	number of global routing nets	average number of pins per net	average edge capacity		
				x	y	z
chip1	256 x 252	506884	3.113	124	119	29
chip2	313 x 309	263563	2.834	62	57	40
chip3	87 x 310	68048	2.715	18	24	40

Table 1: Characteristics of chips.

For computing the Steiner trees we use a subroutine which computes the optimum Steiner tree for 2- and 3-terminal nets. 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 [8]), 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 is not checked. Instead, the algorithm stops when a desired maximum relative congestion is achieved. Since δ is relevant for the stopping condition only, it can simply be set to 1.

Table 1 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 x-, y- and z-direction. chip1 is a chip used for telecommunication, chip2 is a follow–up of the Power3 microprocessor, and chip3 is part of the S/390 processor chipset.

Table 2 shows the running times of the approximation algorithm for these chips and how the maximum relative congestion and the lower bound improve during the algorithm. Our experiments show that large numbers for ϵ reduce the running time: We chose ϵ between 0.6 and 2.0 and γ between 7 and 10.

Table 3 shows the running times for chip3 if γ is set to 0, which means that in each phase all nets are routed. The running time increases drastically, but the result does not improve much. With the values we chose for γ, in average only about 20% of all nets are routed in one phase.

We compared the algorithm with the following rip-up 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 chip1 this rip-up and reroute algorithm failed and increased the netlength by more than 14 % starting from the minimum netlength (each net routed separately), whereas the approximation algorithm found a solution with the total netlength being only 11% greater than the minimum netlength. Table 4 shows the running times for this chip for the case that L was set to the minimum netlength which was 530.75 m.

On easy instances, for example chip2 routing each net with

Chip	number of phases	maximum relative congestion	lower bound	running time (in seconds)
chip1	5	1.000	0.707	10336
	10	0.962	0.823	30313
	15	0.952	0.857	58606
	20	0.952	0.872	87458
	25	0.951	0.882	119951
chip2	5	1.238	0.368	2024
	10	0.943	0.519	5285
	15	0.886	0.585	9336
	20	0.851	0.619	13881
	25	0.827	0.641	19040
	30	0.833	0.657	24236
	35	0.857	0.669	28676
	40	0.875	0.679	35213
chip3	5	1.329	0.535	253
	10	1.122	0.737	568
	15	1.079	0.795	1001
	20	1.044	0.822	1623
	25	1.017	0.836	2587
	30	1.000	0.845	3388
	35	0.979	0.854	4228
	40	0.979	0.859	5634

Table 2: Running times: minimizing the maximum relative congestion.

Chip	number of phases	maximum relative congestion	lower bound	running time (in seconds)
chip3	5	1.236	0.543	668
	10	1.094	0.749	1906
	15	1.074	0.800	3732
	20	1.041	0.822	6374
	25	1.014	0.837	8392
	30	0.993	0.847	14063
	35	0.977	0.854	18421
	40	0.965	0.858	23076

Table 3: Running times with $\gamma = 0$.

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.

The computational results also show that the time needed for a phase increases as the algorithm proceeds. The reason is the following: The dual variables y_e increase more for highly congested edges and these edges build up barriers which have to be broken by the Dijkstra search for those nets which need to cross these barriers and hence more nodes have to be labeled.

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 $T_{i,j}$ is chosen is $x_{i,j}$. For details see [14], [15]. Figure 4 shows the relative congestion of the edges after they had been sorted with respect to increasing congestion for the solution of the fractional global routing problem and after randomized rounding for chip2. The maximum relative congestion increased only on very few edges, and by using the rip-up and reroute algorithm the congestion could be reduced to almost the value of the solution for the linear program in a few seconds.

Chip	number of phases	maximum relative congestion	total netlength (in meter)	running time (in seconds)
chip1	5	1.3008403	568.31	6587
	10	1.0696429	639.51	15466
	15	1.0210526	628.99	24747
	20	0.9938596	605.69	33226
	25	0.9834568	588.39	39272

Table 4: Running times: minimizing the total netlength.

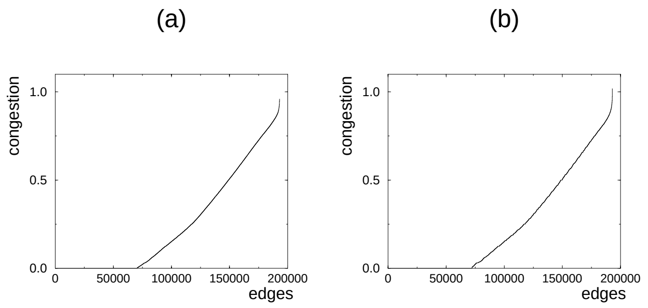

Figure 4: Congestion of edges for the solution of the fractional global routing problem (a) and after randomized rounding (b) for chip2.

6. CONCLUSIONS

We have presented an approximation algorithm for solving the linear programming relaxation of the global routing problem. If a chip cannot be routed, this algorithm will give a proof by the dual solution, which also gives a lower bound on the maximum relative congestion. 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 rip-up and reroute algorithms fail.

7. ACKNOWLEDGEMENTS

Many people contributed ideas for the global routing program and helped to implement it. The author would like to thank especially Karl Köhler and Jürgen Werber.

8. REFERENCES

[1] K. Aoshima and E. S. Kuh. Multi-Channel Optimization in Gate-Array LSI Layout. In *Proceedings ISCAS*, pages 1005–1008, 1983.

[2] R. J. Brouwer and P. Banerjee. PHIGURE: A Parallel Hierarchical Global Router. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 650–653, 1991.

[3] R. C. Carden IV and C.-K. Cheng. A Global Router Using an Efficient Approximate Multicommodity Multiterminal Flow Algorithm. In *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pages 316–321, 1991.

[4] J. Cong and B. Preas. A New Algorithm for Standard Cell Global Routing. *Integration, the VLSI journal*, 14:49–65, 1992.

[5] W.-M. Dai and E. S. Kuh. Simultaneous Floor Planning and Global Routing for Hierarchical Building-Block Layout. *IEEE Transactions on Computer-Aided Design*, 6(5):828–837, 1997.

[6] L. K. Fleischer. Approximating Fractional Multicommodity Flow Independent of the Number of Commodities. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 24–31, 1999.

[7] N. Garg and J. Könemann. Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[8] A. Hetzel. *Verdrahtungsprobleme im VLSI-Design: Spezielle Teilprobleme und ein sequentielles Lösungsverfahren.* Ph. D. Thesis (in German), University of Bonn, 1995.

[9] A. Hetzel. A Sequential Detailed Router for Huge Grid Graphs. In *Proceedings of the Conference "Design, Automation and Test in Europe"*, pages 332–338, 1996.

[10] J. Huang, X.-L. Hong, C.-K. Cheng, and E. S. Kuh. An Efficient Timing-Driven Global Routing Algorithm. In *Proceedings of the 30th ACM/IEEE Design Automation Conference*, pages 596–600, 1993.

[11] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer–Verlag, Berlin, 2000.

[12] G. Meixner and U. Lauther. A New Global Router Based on A Flow Model and Linear Assignment. In *Proceedings of the International Conference on Computer Aided Design*, pages 44–47, 1990.

[13] T. Okamoto, M. Ishikasa, and T. Fujita. A New Feed-Through Assignment Algorithm Based on A Flow Model. In *Proceedings of the International Conference on Computer Aided Design*, pages 775–778, 1993.

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

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

[16] A. Rohe and M. Zachariasen. Rectilinear Group Steiner Trees and Application in VLSI–Design. Technical Report, University of Bonn. 2000.

[17] E. Shragowith and S. Keel. A Global Router Based on a Multicommodity Flow Model. *Integration, the VLSI journal*, 5:3–16, 1987.