Provably good global buffering by multiterminal multicommodity flow approximation*

Feodor F. Dragan, Andrew B. Kahng§, Ion Măndoiu†, Sudhakar Muddu‡, and Alexander Zelikovsky¶

Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242
§UCSD CSE and ECE Departments, La Jolla, CA 92093-0114
†College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280
‡Sanera Systems, Inc., Santa Clara, CA
¶Department of Computer Science, Georgia State University, Atlanta, GA 30303
dragan@mcs.kent.edu, abk@cs.ucsd.edu, mandoiu@cc.gatech.edu, muddu@sanera.net, alexz@cs.gsu.edu

Abstract—To implement high-performance global interconnect without impacting the placement and performance of existing blocks, the use of buffer blocks is becoming increasingly popular in structured-custom and block-based ASIC methodologies. Recent works by Cong, Kong and Pan [5] and Tang and Wong [18] give algorithms to solve the *buffer block planning* problem. In this paper, we address the problem of how to perform buffering of global multiterminal nets given an existing buffer block plan. We give a provably good algorithm based on a recent approach of Garg and Könemann [8] and Fleischer [7] (see also Albrecht [1] and Dragan et al. [6]). Our method routes connections using available buffer blocks, such that required upper and lower bounds on buffer intervals—as well as wirelength upper bounds per connection—are satisfied. In addition, our algorithm allows more than one buffer to be inserted into any given connection and observes buffer parity constraints. Most importantly, and unlike previous works on the problem [5, 18, 6], we take into account multiterminal nets. Our algorithm outperforms existing algorithms for the problem [5, 6], which are based on 2-pin decompositions of the nets. The algorithm has been validated on top-level layouts extracted from a recent high-end microprocessor design.

I. Introduction

Process scaling leads to an increasingly dominant effect of interconnect on high-end chip performance. Each top-level global net must undergo repeater insertion (among other optimizations; see [4, 11, 14]) to maintain signal integrity and reasonable signal delay.[1] Estimates of the need for repeater insertion range up to $O(10^6)$ repeaters for top-level on-chip interconnect when we reach the 50nm technology node. These repeaters are large (anywhere from 40× to 200× minimum inverter size), affect global routing congestion, can entail non-standard cell height and special power routing requirements, and can act as noise sources. In a block- or reuse-based methodology, designers seek to isolate repeaters for global interconnect from individual block implementations.

For these reasons, a *buffer block* methodology has become increasingly popular in structured-custom and block-based ASIC methodologies. Two recent works by Cong, Kong and Pan [5] and Tang and Wong [18] give algorithms to solve the *buffer block planning* problem. Their buffer block planning formulation is roughly stated as follows: Given a placement of circuit blocks, and a set of two-pin connections with *feasible regions*[2] for buffer insertion, plan the location of *buffer blocks* within the available free space so as to route a maximum number of connections. In another recent development, Dragan et al. [6] give an algorithm for performing global buffered routing of two-pin nets under an existing buffer block plan.

In this paper, we address the problem of how to perform buffering of global *multiterminal* nets given an existing buffer block plan. (Hence, our work is compatible with and complements the methods in [5, 18, 6].) We give a provably good algorithm based on a recent approach of Garg and Könemann [8] and Fleischer [7]. Our method routes the nets using available buffer blocks, such that required upper and lower bounds on repeater intervals—as well as length upper bounds per connection—are satisfied.[3] In addition, our algorithm observes *repeater parity constraints*, i.e., it will choose to use an inverter or a buffer (= co-located pair of inverters) according to source and destination signal parity. The authors of [5, 18, 6] assumed that global nets have been already decomposed into two-pin connections; unlike these works our model takes into account *multiterminal nets*.

Informally, our problem is defined as follows.

Given:

- a planar region with rectangular obstacles;
- a set of nets in the region, each net has:
  - a single source and multiple sinks;
  - a non-negative importance (criticality) coefficient;
- each sink has:
  - a parity requirement, which specifies the required parity of the number of buffers (inverters) on the path connecting it to the source;
  - a timing-driven requirement, which specifies the maximum number of buffers on the path to the source;
- a set of buffer blocks, each with given capacity; and
- an interval $[L, U]$ specifying lower and upper bounds on the distance between buffers.

Global Routing via Buffer Blocks (GRBB) Problem: route a subset of the given nets, with maximum total importance, such that:

- the distance between the source of a route and its first repeater, between any two consecutive repeaters, respectively between the last repeater on a route and the route's sink, are all between L and U;
- the number of trees passing through any given buffer block does not exceed the block's capacity;
- the number of buffers on each source-sink path does not exceed the given upper bound and has the required parity; to meet the parity constraint two buffers of the same block can be used.

If possible, the optimum solution to the GRBB problem simultaneously routes all the nets. Otherwise, it maximizes the sum of the importance coefficients over routed nets. The importance coefficients can be used to model various practical objectives. For example, importance coefficients of 1 for each net correspond to maximizing the number of routed nets, and importance coefficients equal the number of sinks in the net correspond to maximizing the number of connected sinks.

The GRBB problem can be formulated as a generalized version of (vertex-capacitated) integer *multiterminal multicommodity flow* (MTMCF). The main contribution of this paper is an MTMCF based algorithm for the GRBB problem. Prior to our work, multicommodity flow (MCF) based heuristics have been applied to VLSI global routing [13, 17, 2, 9, 1]. As noted in [12], the applicability of these algorithms

*This work was partially supported by Cadence Design Systems, Inc., the MARCO Gigascale Silicon Research Center and NSF Grant CCR-9988331.

[1] Following the literature, we will use the terms *buffer* and *repeater* fairly interchangeably. When we need to be more precise: a repeater can be implemented as either an inverter or as a buffer (= two co-located inverters).

[2] In [18] only a single buffer per connection is allowed.

[3] For example, global repeater rules for a high-end microprocessor design in 0.25μm CMOS [10] require repeater intervals of at most 4500μm. The number of buffers needed for a given connection depends strongly on the length of the connection; as noted in [10], the repeater interval is not only required for delay reduction, but also for crosstalk noise immunity and edge slewtime control.

0-7803-6633-6/01/$10.00 ©2001 IEEE.

has often been limited to problem instances of relatively small size by the prohibitive cost of solving exactly the fractional relaxation. Following [1, 6], we avoid this limitation by using an approximate MTMCF algorithm based on results in [8, 7]. An important feature of our algorithm is that it allows for a smooth trade-off between running time and solution accuracy. Our experiments indicate that even MTMCF solutions with very low accuracy give good final solutions for the GRBB problem.

The most interesting feature of our algorithm is its ability to work with *multiterminal* nets. Previous work on the GRBB problem [5, 18, 6] has considered only the case of 2-pin nets. Experiments on top-level layouts extracted from a recent high-end microprocessor design validate our MTMCF algorithm, and indicate that it significantly outperforms existing algorithms for the problem [5, 6].

The rest of the paper is organized as follows. In Section 2, we reduce the Global Buffering Problem to a generalized version of integer multiterminal multicommodity flow. The fractional relaxation of this problem is a special case of *packing LP*, and can thus be approximated within any desired accuracy using the algorithm of Garg and Könemann [8]. In Section 3 we present a faster approximation algorithm, obtained by extending the ideas of Fleischer [7] to this special type of packing LPs. In Section 4 we describe the randomized rounding algorithm used to convert near-optimal fractional MTMCF solutions to near-optimal integral solutions. In Section 5 we describe several global buffering heuristics, some based on the MTMCF approach, and some based on less sophisticated greedy ideas. In Section 6 we give the results of an experimental comparison of these heuristics on test cases extracted from the top-level layout of a recent high-end microprocessor. Finally, we conclude in Section 7 with a list of open research directions.

II. Integer Program Formulation of GRBB

Given K *nets* $N_k = (s_k; t_k^1, \ldots, t_k^{q_k})$, $k = 1, \ldots, K$, and n *buffer blocks* $\{r_1, \ldots, r_n\}$, let $S = \{s_1, \ldots, s_K\}$, $T = \{t_1^1, \ldots, t_1^{q_1}, \ldots, t_K^1, \ldots, t_K^{q_K}\}$, and $R = \{r_1, \ldots, r_n\}$. Let also $a_k^i \in \{even, odd\}$, respectively l_k^i, be the *parity requirement*, respectively the prescribed upper bound, on the number of buffers on the path between source s_k and sink t_k^i.

Let p_{xy} be a rectilinear path connecting points x and y of a planar region and avoiding all rectangular obstacles given in the region. Denote by $d(x,y)$ the length of a shortest such path. Let $G = (V,E)$ be a graph with vertex set $V = S \cup T \cup R$. The edge set E contains all edges of type vv, $v \in R$ (such an edge is called a loop). Two different vertices x and y are adjacent (i.e., $xy \in E$) if and only if $L \le d(x,y) \le U$.

A path $p = (s_k, v_1, v_2, \ldots, v_l, t_k^i)$ in G between source s_k and sink t_k^i ($k = 1, \ldots, K$, $i = 1, \ldots, q^k$) is a *restricted* (s_k, t_k^i)*-path* if

- $v_i \in R$ for each $i = 1, \ldots, l$,
- the parity of l is a_k^i,
- $l \le l_k^i$,
- there can be some pairs of different indices $i, j \in \{1, \ldots, l\}$ such that $v_i = v_j$; in this case we must have $|i - j| = 1$.

A *feasible Steiner tree* for net N_k is a Steiner tree T_k in G connecting terminals $s_k, t_k^1, \ldots, t_k^{q_k}$ such that, for every $i = 1, \ldots, q_k$, the path of T_k connecting s_k to t_k^i is a restricted (s_k, t_k^i)-path as defined above.

Define capacities on all vertices of G by

$$c(v) := \begin{cases} 1, & \text{if } v \in S \cup T \\ \textit{capacity of buffer block } v, & \text{if } v \in R \end{cases}$$

Let $\mathcal{T}_k$ be the set of all feasible Steiner trees for net N_k, and let $\mathcal{T} = \bigcup_{k=1}^{K} \mathcal{T}_k$. For each $T \in \mathcal{T}_k$, $k = 1, \ldots, K$, define $g(T) := g_k$, where g_k is the importance of N_k.

The GRBB problem is then equivalent to the following integer linear program:

$$\begin{array}{ll} \textbf{maximize} & \sum_{T \in \mathcal{T}} g(T) f_T \\ \textbf{subject to} & \sum_{T \in \mathcal{T}} \pi_T(v) f_T \le c(v) \quad \forall v \in V \\ & f_T \in \{0,1\} \quad \forall T \in \mathcal{T}. \end{array}$$

where $f_T = 1$ if the tree T is used in the solution and $f_T = 0$ otherwise, and $\pi_T(v)$ is the number of occurrences of v in T, i.e.,

$$\pi_T(v) := \begin{cases} 0, \text{ if } v \notin T \\ 1, \text{ if } v \in T \text{ and loop } vv \text{ is not in } T \\ 2, \text{ if } v \in T \text{ and loop } vv \text{ is in } T \end{cases}$$

Our approach will be to solve the relaxation of the above integer program obtained by replacing the integrality constraint with $f_T \ge 0$ $\forall T \in \mathcal{T}$; we will then use randomized rounding to obtain an integer solution. We will refer to this relaxation as the *Multiterminal Multicommodity Flow Linear Program* (MTMCF LP).

Although the MTMCF LP is solvable in polynomial time (using, e.g., the ellipsoid algorithm), exact algorithms are highly impractical. On the other hand, the MTMCF LP is a special case of *packing LP*, and can thus be efficiently approximated within any desired accuracy using the recent combinatorial algorithm of Garg and Könemann [8]. In this paper we give a significantly faster approximation algorithm based on a speed-up idea due to Fleischer [7]. Fleischer's idea, originally proposed for approximating the maximum edge-capacitated MCF, has been recently extended [1, 6] to edge-capacitated *multiterminal* MCF and *vertex-capacitated* MCF, respectively. Here we take this approach further and show how to use it for efficient approximation of *vertex-capacitated multiterminal* multicommodity flow.

III. Approximation of Vertex-Capacitated MTMCF

Our MTMCF approximation algorithm simultaneously solves the MTMCF LP and its *dual LP*; the dual solution is used in proving the approximation guarantee of the algorithm. The dual of the MTMCF LP is:

$$\begin{array}{ll} \textbf{minimize} & \sum_{v \in V} w(v) c(v) \\ \textbf{subject to} & \frac{1}{g(T)} \sum_{v \in T} w(v) \ge 1 \quad \forall T \in \mathcal{T} \\ & w(v) \ge 0 \quad \forall v \in V. \end{array}$$

The dual LP can be viewed as an assignment of non-negative weights, $w(\cdot)$, to the vertices of G such that the weight of any tree $T \in \mathcal{T}$ is at least 1; the objective is to minimize the sum $\sum_{v \in V} w(v) c(v)$. Here, the *weight*, *weight*(T), *of the tree* T is the sum of the weights of vertices forming this tree (if the tree uses a loop vv then vertex v contributes twice to this sum) divided by the importance $g(T)$ of this tree.

Denote $D(w) = \sum_{v \in V} w(v) c(v)$ and let $\alpha(w)$ be the weight of a minimum weight tree from $\mathcal{T}$ (with respect to $w(\cdot)$). The dual problem is equivalent to finding a weight function $w : V \to \mathbf{R}^+$ such that $\beta = \frac{D(w)}{\alpha(w)}$ is minimized. In the following we will assume that $min\{g_k : k = 1, \ldots, K\} = 1$—this can be easily achieved by scaling—and will denote by Γ the maximum g_k. Our algorithm for MTMCF approximation is given in Fig. 1.

In the algorithm, $f_k(v)$ denotes how many times vertex v is visited by feasible Steiner trees used to connect net N_k, and f denotes the total number of feasible Steiner trees used by the algorithm. The algorithm associates a weight with each vertex, and in each iteration it uses a *minimum weight tree* $T \in \mathcal{T}_k$ to connect the pins of some net N_k. When tree T is selected, the algorithm multiplies the weight of every vertex in T by $1 + \frac{\varepsilon}{c(v)}$ for a fixed ε (if this tree uses a loop vv, then the weight of v is multiplied by $1 + \frac{2\varepsilon}{c(v)}$). Initially, every vertex v has weight δ for some constant δ. Thus, the more often is a vertex used, the larger its weight, which implies that often used vertices are less likely to be part of future minimum weight trees.

Input: Graph G with K nets $N_1,\ldots,N_K$, vertex capacities $c(v)$
Output: MTMCF value, f, and flows $f_k(v) \in [0,1]$, $k=1,\ldots,K$, $v \in V(G)$ satisfying capacity constraints

Set $f=0$
Set $w(v)=\delta$ for all $v \in V$
Set $f_k(v)=0$ for all $v \in V$ and $k=1,\ldots,K$
For $i=1$ to $log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}$ do
  For $k=1$ to K do
    Find a minimum weight tree T in $\mathcal{T}_k$.
    While $weight(T) < min\{1/\Gamma, \delta/\Gamma(1+2\varepsilon)^i\}$ do
      $f=f+1$;
      For all $v \in T$, if T uses a loop vv then set
        $f_k(v)=f_k(v)+2$ and $w(v)=w(v)(1+\frac{2\varepsilon}{c(v)})$; else set
        $f_k(v)=f_k(v)+1$ and $w(v)=w(v)(1+\frac{\varepsilon}{c(v)})$
      Find a minimum weight tree T in $\mathcal{T}_k$
    End while
  End for
End for
Output $\frac{f}{2\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}}$, and $\frac{f_k(v)}{2\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}}$ for each $v \in V$ and $k=1,\ldots,K$

Fig. 1. The fractional tree-routing algorithm

According to Garg and Könemann's approximation algorithm [8], each iteration must use a lightest (with respect to current weight function $w(\cdot)$) tree from $\mathcal{T}$ if the weight of this tree is less than $1/\Gamma$. The algorithm then stops after t iterations where t is the smallest number such that $\alpha(w)$, computed with respect to vertex weights $w(\cdot)$ of this iteration, is at least $1/\Gamma$. We extend an idea due to Fleischer [7] to our vertex-capacitated MTMCF problem to reduce the number of minimum weight tree computations during the algorithm. Instead of finding the lightest tree in $\mathcal{T}$ in each iteration (which essentially involves K shortest-path computations) we settle for some tree within a factor of $(1+2\varepsilon)$ of the lightest, and show that this still leads to a similar approximation guarantee.

Let $w_{i-1}(\cdot)$ be the weight function at the beginning of the ith iteration. We have $w_0(v)=\delta$ for each $v \in V$. For brevity denote $\alpha(w_i)$ and $D(w_i)$ by $\alpha(i)$ and $D(i)$, respectively. Following Fleischer, we cycle through the nets, sticking with a net until the lightest feasible Steiner tree for that net is above a $1+2\varepsilon$ factor times a lower bound estimate of the overall lightest tree. Let $\bar{\alpha}(i)$ be a lower bound on $\alpha(i)$. To start, we set $\bar{\alpha}(0)=\delta/\Gamma$. As long as there is some $T \in \mathcal{T}$ with $weight(T) \le min\{1/\Gamma, (1+2\varepsilon)\bar{\alpha}(i)\}$, we use tree T. When this no longer holds, we know that the weight of the lightest tree is at least $(1+2\varepsilon)\bar{\alpha}(i)$, and so we set $\bar{\alpha}(i+1)=(1+2\varepsilon)\bar{\alpha}(i)$. Thus, throughout the course of the algorithm, $\bar{\alpha}$ takes on values in the set $\{\delta/\Gamma(1+2\varepsilon)^i\}_{i\in\mathcal{N}}$. Since $\alpha(0) \ge \delta/\Gamma$ and $\alpha(t-1) < 1/\Gamma$, $\alpha(t) < (1+2\varepsilon)/\Gamma$. Thus, when we stop, $\bar{\alpha}(t)$ is between $1/\Gamma$ and $(1+2\varepsilon)/\Gamma$. Each increase of $\bar{\alpha}$ is by a $1+2\varepsilon$ factor, hence the number of increases of $\bar{\alpha}$ is $\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}$ (and the final value of i is $\lfloor \log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta} \rfloor$).

Between updates to $\bar{\alpha}$, the algorithm proceeds by considering each net one by one. As long as the lightest feasible Steiner tree T for net N_k has weight less than the minimum of $1+2\varepsilon$ times the current value of $\bar{\alpha}$ and $1/\Gamma$, this lightest tree T is used to connect the pins of the net N_k. When $min_{T\in\mathcal{T}_k} weight(T) \ge (1+2\varepsilon)\bar{\alpha}$, net N_{k+1} is considered. After all K nets are considered, $\bar{\alpha}$ is updated. A total of at most $K\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}$ minimum weight feasible Steiner tree computations are used to update α over the course of the algorithm.

Theorem 1 *The algorithm in Fig. 1 is a $(1+\omega)$-approximation algorithm for the MTMCF LP by choosing $\delta=(1+2\varepsilon)((1+2\varepsilon)L\Gamma)^{-\frac{1}{2\varepsilon}}$ and $\varepsilon < min\{.07, \frac{1+\omega-\Gamma)}{8\Gamma}\}$, where L is the number of vertices in the longest feasible Steiner tree of G connecting any net.*

Proof. Our proof is an adaptation of the proof of Garg and Könemann [8] (see also Fleischer [7]). First we show that the values $\frac{f_k(v)}{2\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}}$ $(v \in V, k=1,\ldots,K)$, computed by the algorithm, are feasible, i.e., $\frac{1}{2\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}}\sum_{k=1}^{K} f_k(v) \le c(v)$ and hence we do not exceed the capacity of any vertex v of G. Consider an arbitrary vertex v of G and let $M=\sum_{k=1}^{K} f_k(v)$ denotes how many times the vertex v was used by all feasible Steiner trees found by algorithm. For every two times that the vertex v was used by feasible Steiner trees, the weight of v increased by a factor of at least $(1+\frac{2\varepsilon}{c(v)})$. Since $w_0(v)=\delta$, it follows that $w_t(v) \ge \delta(1+\frac{2\varepsilon}{c(v)})^{\frac{M}{2}}$. Simplifying this expression, we get

$$w_t(v) \ge \delta(1+\frac{2\varepsilon}{c(v)})^{\frac{M}{2}} = \delta((1+\frac{2\varepsilon}{c(v)})^{c(v)})^{\frac{M}{2c(v)}} \ge \delta(1+2\varepsilon)^{\frac{M}{2c(v)}}.$$

The last time we increased the weight of v, it was on a feasible Steiner tree of weight less than $1/\Gamma$. Hence, the weight of v was less than 1. Since in each iteration we increase the vertex weight by factor of at most $(1+2\varepsilon)$, the final weight of v is at most $(1+2\varepsilon)$. Consequently,

$$\delta(1+2\varepsilon)^{\frac{M}{2c(v)}} \le w_t(v) \le 1+2\varepsilon, \text{ i.e., } M \le c(v)2\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}.$$

Now we show that the ratio of the values of the dual and the primal solutions, $\gamma=\frac{\beta}{f}2\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}$, is at most $(1+\omega)$.

For each iteration $i \ge 1$ we have

$$D(i)=\sum_{v\in V} w_i(v)c(v)=\sum_{v\in V} w_{i-1}(v)c(v)+\varepsilon\sum_{v\in T} w_{i-1}(v) \le$$

$$D(i-1)+\varepsilon(1+2\varepsilon)\Gamma\alpha(i-1).$$

Note that, if T used a loop vv, then v contributes to the sum $\sum_{v\in T} w_{i-1}(v)$ twice (since $w_i(v)=w_{i-1}(v)(1+\frac{2\varepsilon}{c(v)})$).

Then,

$$D(i)-D(0) \le \varepsilon(1+2\varepsilon)\Gamma\sum_{j=1}^{i}\alpha(j-1).$$

Consider the weight function $w_i(\cdot)-w_0(\cdot)$. We have $\alpha(w_i-w_0) \ge \alpha(w_i)-\delta L$, where L is the number of vertices in the longest feasible Steiner tree of G connecting any net.

Consequently, if $\alpha(w_i)-\delta L>0$, then

$$\beta \le \frac{D(w_i-w_0)}{\alpha(w_i-w_0)} \le \frac{D(i)-D(0)}{\alpha(i)-\delta L} \le \frac{\varepsilon(1+2\varepsilon)\Gamma\sum_{j=1}^{i}\alpha(j-1)}{\alpha(i)-\delta L}.$$

Thus, in any case (for the case $\alpha(w_i)-\delta L \le 0$, it is trivial) we have

$$\alpha(i) \le \delta L+\frac{\varepsilon(1+2\varepsilon)\Gamma}{\beta}\sum_{j=1}^{i}\alpha(j-1) \le$$

$$(1+\frac{\varepsilon(1+2\varepsilon)\Gamma}{\beta})^{i-1}(\delta L+\frac{\varepsilon(1+2\varepsilon)\Gamma}{\beta}\alpha(0)) \le$$

$$(1+\frac{\varepsilon(1+2\varepsilon)\Gamma}{\beta})^{i-1}(\delta L+\frac{\varepsilon(1+2\varepsilon)\Gamma}{\beta}\delta L) =$$

$$\delta L(1+\frac{\varepsilon(1+2\varepsilon)\Gamma}{\beta})^{i} \le \delta L e^{\frac{i\varepsilon(1+2\varepsilon)\Gamma}{\beta}}.$$

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

Since we stop at iteration t with $\alpha(t) \ge 1/\Gamma$, and $t = f$, we get

$$1/\Gamma \le \alpha(t) \le \delta L e^{\frac{t\varepsilon(1+2\varepsilon)\Gamma}{\beta}} = \delta L e^{\frac{f\varepsilon(1+2\varepsilon)\Gamma}{\beta}}.$$

Hence,

$$\frac{\beta}{f} \le \frac{\varepsilon(1+2\varepsilon)\Gamma}{\ln(\delta L\Gamma)^{-1}}.$$

Now, for the ratio γ we obtain

$$\gamma = \frac{\beta}{f} 2\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta} \le$$

$$\frac{2\varepsilon(1+2\varepsilon)\Gamma\log_{1+2\varepsilon}\frac{1+2\varepsilon}{\delta}}{\ln(\delta L\Gamma)^{-1}} = \frac{2\varepsilon(1+2\varepsilon)\Gamma\ln\frac{1+2\varepsilon}{\delta}}{\ln(1+2\varepsilon)\ln(\delta L\Gamma)^{-1}}.$$

Since we have chosen $\delta = (1+2\varepsilon)((1+2\varepsilon)L\Gamma)^{\frac{-1}{2\varepsilon}}$, we get $\frac{\ln\frac{1+2\varepsilon}{\delta}}{\ln(\delta L\Gamma)^{-1}} = \frac{1}{1-2\varepsilon}$ and hence,

$$\gamma \le \frac{2\varepsilon(1+2\varepsilon)\Gamma}{(1-2\varepsilon)\ln(1+2\varepsilon)} \le (1+2\varepsilon)(1-2\varepsilon)^{-2}\Gamma.$$

Here we used that $\ln(1+x) \ge x - x^2/2$ (by Taylor series expansion of $\ln(1+x)$).

Since $(1+2\varepsilon)(1-2\varepsilon)^{-2}$ is at most $(1+8\varepsilon)$, for $\varepsilon < .07$, and $(1+8\varepsilon)\Gamma$ should be no more than our approximation ratio $(1+\omega)$, we are done. □

In the algorithm in Fig. 1 we need to solve the following problem. Let G_k $(k=1,\ldots,K)$ be a subgraph of the graph G induced by vertices $\{s_k, t_k^1, \ldots, t_k^{q_k}\} \cup R$ (recall that each vertex $v \in R$ has a loop $vv \in E$). Let also each vertex v of G_k have a non-negative weight $w(v)$. Find a minimum weight tree T_k in G_k connecting s_k with $t_k^1, \ldots, t_k^{q_k}$ such that, for each $i = 1, \ldots, q_k$, the path of T_k between s_k and t_k^i passes through even (odd, depending on a_k^i) number of vertices, and that number of vertices does not exceed l_k^i. This path may contain a loop, in this case the weight of the vertex at which the loop is attached will contribute twice to the weight of the tree T_k.

Let $L_k = max\{l_k^1, \ldots, l_k^{q_k}\}$. We reduce this problem to the usual shortest directed rooted Steiner tree problem on an edge-weighted directed acyclic graph (dag) D_k with $V(D_k) = \{s_k\} \cup \{r_{i,j} \mid 1 \le i \le n, 1 \le j \le L_k\} \cup \{t_k^1, \ldots, t_k^{q_k}\}$ and $E(D_k) = E_1 \cup E_2 \cup E_3$, where

$$\begin{aligned} E_1 &= \{(s_k, r_{i,1}) \mid 1 \le i \le n, (s_k, r_i) \in E(G)\} \\ E_2 &= \{(r_{i,j}, r_{i',j+1}) \mid 1 \le i, i' \le n, 1 \le j < L_k, \\ &\quad (r_i, r_{i'}) \in E(G)\} \\ E_3 &= \{(r_{i,j}, t_k^h) \mid 1 \le i \le n, 1 \le h \le q_k, 1 \le j \le l_k^h, \\ &\quad j \equiv a_k^h (\mathrm{mod}\ 2), (r_i, t_k^h) \in E(G)\} \end{aligned}$$

If the cost of each arc (x,y) in D_k is given by $w(x)$, it is easy to see that finding the minimum weight tree in $\mathcal{T}_k$ reduces to finding a minimum cost directed rooted Steiner tree (DRST) in D_k. Generally, the *directed rooted Steiner tree problem* asks, for a given directed edge-weighted graph $H = (X, U)$, a specified root $r \in X$, and a set of terminals $Y \subset X$, to find the minimum cost arborescence rooted at r and spanning all the vertices in Y (in other words r should have a directed path to every vertex in Y). Unfortunately, the fact that D_k is acyclic does not help. There is a simple reduction for this problem from arbitrary directed graphs to acyclic graphs. As far as we know, the best result for the DRST problem is due to Charikar et al. [3] which says that an $O(\log^2 q_k)$-approximate solution can be found in quasi-polynomial time $O(n^{3\log q_k})$. Since this is very inefficient, we need to find some other ways to compute such trees. One approach, used by Albrecht [1] for edge-capacitated MTMCF approximation, is to compute (exactly or approximately) a DRST once, then use in each of the following iterations minimum directed *spanning* trees (with respect to the updated edge lengths) in the graph induced by $\{s_k, t_k^1, \ldots, t_k^{q_k}, p_1, \ldots, p_s\}$, where $p_1, \ldots, p_s$ are the Steiner points of the original DRST. To find a minimum spanning directed tree in directed acyclic graphs, one can use a very simple procedure: for each vertex choose a shortest incoming arc, then, after running this procedure, recursively delete all leaves that are not sinks of the net N_k.

Input: Multiterminal flows $f_k(e) \in [0,1]$, $k = 1, \ldots, K$, $e \in E(G)$
Output: Set of trees $T_k \in \mathcal{T}_k$

For each $k = 1, \ldots, K$, with probability f_k, do
  $T_k \leftarrow \{s_k\}$
  For each sink t_k^i in N_k do
    $P \leftarrow \emptyset$; $v \leftarrow t_k^i$
    While $v \notin T_k$ do
      Pick arc (u,v) with probability $f_k(u,v)/f_k(v)$
      $P \leftarrow P \cup \{(u,v)\}$; $v \leftarrow u$
    End while
    $T_k \leftarrow T_k \cup P$
  End for
End for

Fig. 2. Randomized MTMCF rounding algorithm

IV. Rounding the Fractional MTMCF

In the previous section we presented an algorithm for approximating the optimum multiterminal multicommodity flow (MTMCF) within any desired accuracy. The optimum MTMCF gives an upper-bound on the maximum number of routable nets (connections). In this section we show how to use the approximate MTMCF to route an almost optimal number of nets (resp. connections). Our construction is based on the randomized rounding technique of Raghavan and Thomson [16], in particular, on the random-walk based algorithm for rounding multicommodity flow [15] (see also [12]).

The MCF rounding algorithm in [15] chooses a set of source-sink pairs by including each pair (s,t) with a probability equal to the flow from s to t. Then, for each chosen pair, (s,t), the algorithm performs a random-walk from s to t, based on probabilities given by edge-flows. In our MTMCF rounding algorithm (see Figure 2), a net $N_k = (s_k; t_k^1, \ldots, t_k^{q_k})$ is also routed with probability equal to the net's total flow, $f_k = \sum_{T \in \mathcal{T}_k} f_T$. Since we need to construct a tree connecting all sinks t_k^i to the source s_k, we route the net by performing *backward* random walks from each sink until reaching either s_k or a vertex on a path already included in the tree. Thus, if the net has only one sink, our rounding algorithm becomes identical to the algorithm in [15], except for the direction of the random walk.

Ensuring that no vertex capacities are exceeded can be accomplished in two ways. Following [12], one way is to solve the MTMCF LP with capacities scaled down by a small factor that guarantees that the rounded solution will meet the *original* capacities with very high probability. A simpler approach, the so-called *greedy-deletion algorithm* [6], is to repeatedly drop routed nets that visit over-used vertices until feasibility is achieved. We implement a modification of the second approach: instead of dropping an entire tree, we drop only the sinks which use paths through over-used vertices.

Input: Graph G with K nets $N_1,\ldots,N_K$, vertex capacities $c(v)$
Output: Set of trees $T_k \in \mathcal{T}_k$

For each $k = 1,\ldots,K$, do
  $T_k \leftarrow \{s_k\}$
  For each sink t_k^i in N_k do
    Using a backward BFS search, find a shortest path P from t_k^i to T_k in G using only vertices v with $c(v) > 0$; if no such path exists let $P = \emptyset$
    $T_k \leftarrow T_k \cup P$
    For each vertex v in P, $c(v) \leftarrow c(v) - 1$
  End for
End for

Fig. 3. The multiterminal greedy (MTG) routing algorithm

V. Implemented Algorithms

In this section we describe the implemented algorithms for the Global Routing via Buffer Blocks problem.

Greedy Routing Algorithms

We have implemented 3 greedy algorithms for the GRBB problem. The first algorithm [5, 6] starts by decomposing each multiterminal net into 2-terminal nets. Then, the algorithm attempts to route the 2-terminal nets one by one, using for routing a shortest available path from the net's source to its sink, if such a path exists. We will refer to this algorithm as the *forward 2-terminal greedy* (F-2TG) algorithm.

The second greedy algorithm, referred to as the *multiterminal Greedy* (MTG) algorithm, routes multiterminal nets without splitting (Fig. 3). In this algorithm we also attempt to route the sinks of a net one by one. For a given net, the algorithm starts with a tree containing only the net's source, then iteratively adds shortest paths from each sink to the already constructed tree.

The third algorithm, the *backward 2-terminal greedy* (B-2TG), works as F-2TG, except for the fact that shortest paths are computed backward, from sinks toward sources and not from sources toward sinks. Notice that B-2TG becomes identical to MTG when applied to 2-terminal nets.

Flow Rounding Algorithms

We have implemented two flow rounding algorithms. The first is the MCF rounding algorithm of Dragan et al. [6], which we will refer to as 2TMCF. It starts by decomposing each multiterminal net into 2-terminal nets, and then casts the GRBB problem as an integer MCF problem.

The second algorithm is based on MTMCF rounding (Fig. 4). Our current implementation decomposes larger nets into 3-terminal nets before applying the MTMCF routing algorithm, we will refer to this implementation as 3TMCF. For 3-terminal nets we can find the optimum directed routed Steiner tree efficiently, and we do not need to resort to the approximations suggested at the end of Section III.

VI. Implementation Experience

All experiments were conducted on a SGI Origin 2000 with 16 195MHz MIPS R10000 processors—only one of which is actually used by the sequential implementations included in our comparison—and 4 G-Bytes of internal memory, running under IRIX 6.4 IP27. Timing was performed using low-level Unix interval timers, under similar load conditions for all experiments. All algorithms were coded in C and compiled using `gcc` version egcs-2.91.66 with `-O4` optimization.

Input: Graph G with K nets $N_1,\ldots,N_K$, vertex capacities $c(v)$
Output: Set of trees $T_k \in \mathcal{T}_k$

Find an approximate MTMCF using the algorithm in Fig. 1
Round the approximate MTMCF using the algorithm in Fig. 2
Use greedy deletion to find a feasible integer solution
Use the MTG algorithm in Fig. 3 on the unrouted nets to find a maximal routing

Fig. 4. The MTMCF routing algorithm

The three test cases used in our experiments were extracted from the next-generation microprocessor chip at SGI. We used an optimized floorplan of the circuit blocks and also optimized the location of the source/sink pin locations based on coarse timing budgets. We used $U = 4000\mu$m, and varied L between 500μm and 2000μm. Path-length upper-bounds were computed with the formula $l_k = \text{dist}(s_k,t_k)/1000$. In all test cases considered the number of nets was large (close to 5000), and the number of buffer blocks small (50), with relatively large capacity (400 buffers per block); such values are typical for this application [6].

Table I gives the number of routed sinks and the running time on the three instances by each of the algorithms included in our comparison. Figure VII plots the solution quality versus the CPU time (in seconds, excluding I/O and memory allocation) for each algorithm.

The first surprising thing to notice is that B-2TG gives noticeably better results than F-2TG, despite the fact that the two algorithms are nearly identical (they both add paths of the same length until some of the vertices use up the full capacity).[4] Perhaps not so surprising is the fact that the multiterminal greedy algorithm is better than both F-2TG and B-2TG. Notice that the running time of all three greedy algorithms is virtually the same, so MTG is the clear choice among them.

Our experiments clearly demonstrate the high quality of the solutions obtained by flow rounding methods. Significant improvement over the best of the greedy methods is possible even with a very small increase in running time, proof that even very coarse MCF/MTMCF approximations give helpful hints to the randomized rounding procedure. Since randomized rounding is very fast, faster in fact than any of the greedy algorithms, the MCF/MTMCF algorithms can be further improved by running randomized rounding with the same fractional flow a large number of times and taking the best of the rounded solutions; our current implementation does not exploit this idea.

Finally, our experiments show that even a limited use of multiterminal nets (decomposition into nets of size 3) gives improvements over the already very high-quality MCF algorithm of Dragan et al. [6]. In fact, the 3TMCF algorithm outperforms the MCF algorithm in [6] even when the same time budget is given to both algorithms.

VII. Conclusions and Future Directions

In this paper, we addressed the problem of how to perform buffering of global *multiterminal* nets given an existing buffer block plan. We gave a provably good algorithm based on a novel approach to MTMCF approximation inspired by recent results of Garg and Könemann [8] and Fleischer [7] on edge-capacitated MCF. Our MTMCF algorithm outperforms existing algorithms for the problem

[4] We presume that the advantage for computing backward shortest paths, as opposed to forward shortest paths, is that the former gives a set of paths that are better spread out in the vicinity of the source of a large net. If the sinks of such a net are grouped in a small number of clusters, which is typically the case in real designs, the forward greedy algorithm is likely to use a small number of neighbors of the source for all these paths, thus leading to the faster exhaustion of the available capacity in these vertices.

TABLE I
PERCENT OF SINKS CONNECTED (BOLDFACE) AND CPU TIME ON 3 INDUSTRIAL TEST CASES

Instance				GREEDY			2TMCF						3TMCF					
ID	Nets	Sinks	N/S	F-2TG	B-2TG	MTG	$\varepsilon = 0.64$	$\varepsilon = 0.32$	$\varepsilon = 0.16$	$\varepsilon = 0.08$	$\varepsilon = 0.04$	$\varepsilon = 0.02$	$\varepsilon = 0.64$	$\varepsilon = 0.32$	$\varepsilon = 0.16$	$\varepsilon = 0.08$	$\varepsilon = 0.04$	$\varepsilon = 0.02$
i1	4764	6038	2.27	**89.5**	**90.6**	**93.5**	**94.8**	**95.8**	**96.5**	**96.6**	**96.8**	**96.8**	**95.7**	**96.8**	**97.3**	**97.5**	**97.6**	**97.6**
				0.58	*0.54*	*0.53*	*2.84*	*12.13*	*39.50*	*139.83*	*600.89*	*2321.67*	*16.57*	*53.62*	*203.03*	*817.59*	*3166.03*	*12736.22*
i2	4925	6296	2.28	**89.9**	**91.6**	**93.6**	**96.2**	**97.1**	**97.4**	**97.5**	**97.6**	**97.6**	**97.0**	**98.0**	**98.4**	**98.5**	**98.6**	**98.4**
				0.84	*0.58*	*0.55*	*4.35*	*11.34*	*40.55*	*156.89*	*690.31*	*2604.34*	*19.50*	*64.13*	*242.17*	*942.34*	*3721.95*	*14854.06*
i3	4938	6321	2.28	**89.8**	**91.5**	**93.3**	**96.2**	**96.9**	**97.3**	**97.3**	**97.5**	**97.5**	**96.8**	**97.8**	**98.3**	**98.4**	**98.4**	**98.3**
				0.65	*0.59*	*0.54*	*3.37*	*11.08*	*40.84*	*163.32*	*730.95*	*2638.04*	*18.99*	*66.12*	*246.29*	*956.83*	*3813.42*	*15088.50*

[5, 6], and has been validated on top-level layouts extracted from a recent high-end microprocessor design.

Ongoing work is aimed at increasing the space of methodologies to which our new techniques apply. As presented here, our work is clearly targeted to very early global wireplanning activity. In other words, the application domain is pre-synthesis chip planning: prescribed repeater intervals are driven only by coarse estimates of Miller coupling factors, repeater sizing, and source impedance or sink capacitance. The presented formulation also does not address timing criticalities or budgets except via net weighting (prioritization); this is fortunately fairly common for initial wireplanning that breaks the "chicken-egg" problem of budgeting between-block and within-block paths in pre-synthesis RTL planning with aggressive global wire optimization.[5]

We are presently extending our approach in the following ways. (1) Handling routing congestion, e.g., by introducing capacitated "virtual" nodes in the flow graph. (2) Handling timing criticality and budgets is another goal; our ideas include better use of net ordering and weighting, and post-processing of the solution to eliminate unneeded repeaters. (It is also possible to attempt to introduce layer awareness, source and sink parasitic awareness, etc., but this risks losing the flavor of early feasibility checking with available buffer block plans.) Here, maintaining provable solution quality is a key issue. (3) Finally, since accurate treatment of multiterminal nets is the key motivation for our present work, we are implementing better heuristics for net decomposition into 2- and/or 3-terminal groups; we are also implementing optimal graph Steiner solutions for up to 4-terminal nets, to assess the associated quality-runtime tradeoffs.

[5] In other words, maximal repeater insertion allows maximum timing budgets for within-block timing paths, and this permits blocks to go through synthesis, place and route with more aggressive area targets. A strategy of uniform buffering of as many global nets as possible also helps control signal integrity and delay uncertainty problems.

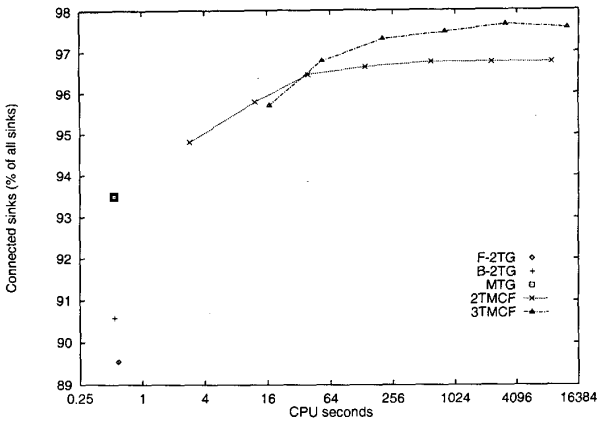

Fig. 5. Percent of sinks connected vs. time on test case i1

REFERENCES

[1] Ch. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow", *Proc. ISPD*, 2000.

[2] R. C. Carden and C.-K. Cheng, "A global router using an efficient approximate multicommodity multiterminal flow algorithm", *Proc. DAC*, 1991, pp. 316-321.

[3] M. Charikar, Ch. Chekuri, T. Cheung, Z. Dai, A. Goel, and S. Cheung, "Approximation algorithms for directed Steiner problems", *J. Algorithms*, 33 (1999), 73-91.

[4] J. Cong, L. He, C.-K. Koh and P. H. Madden, "Performance optimization of VLSI interconnect layout", *Integration* 21 (1996), pp. 1-94.

[5] J. Cong, T. Kong and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning", *Proc. ICCAD*, Nov. 1999, pp. 358-363.

[6] F. F. Dragan, A. B. Kahng, I. I. Măndoiu, S. Muddu and A. Zelikovsky, "Provably good global buffering using an available buffer block plan", *Proc. ICCAD'2000*.

[7] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities", *Proc. 40th Annual Symposium on Foundations of Computer Science*, Oct. 1999, pp. 24-31.

[8] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems", *Proc. 39th Annual Symposium on Foundations of Computer Science*, Nov. 1998, pp. 300-309.

[9] J. Huang, X.-L. Hong, C.-K. Cheng and E. S. Kuh, "An efficient timing-driven global routing algorithm", *Proc. DAC*, 1993, pp. 596-600.

[10] A. B. Kahng, S. Muddu, E. Sarto and R. Sharma, "Interconnect tuning strategies for high-performance ICs", *Proc. DATE*, Feb. 1998.

[11] J. Lillis, C. K. Cheng and T. T. Y. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model", *Proc. ICCAD*, 1995, pp. 138-143.

[12] R. Motwani, J. Naor, and P. Raghavan, "Randomized approximation algorithms in combinatorial optimization", In *Approximation algorithms for NP-hard problems* (Boston, MA, 1997), D. Hochbaum, Ed., PWS Publishing, pp. 144–191.

[13] A.P.-C. Ng, P. Raghavan, and C.D. Thomson, "Experimental results for a linear program global router". *Computers and Artificial Intelligence*, 6 (1987), pp. 229–242.

[14] T. Okamoto and J. Cong, "Buffered Steiner tree construction with wire sizing for interconnect layout optimization", *Proc. ICCAD*, 1996, pp. 44-49.

[15] P. Raghavan and C.D. Thomson, "Provably good routing in graphs: regular arrays", *Proc. 7th ACM Symp. on Theory of Computing* (1985), pp. 79–87.

[16] P. Raghavan and C.D. Thomson, "Randomized rounding", *Combinatorica*, 7 (1987), pp. 365–374.

[17] E. Shragowitz and S. Keel, "A global router based on a multicommodity flow model", *Integration* 5(1) (1987), pp. 3-16.

[18] X. Tang and D. F. Wong, "Planning buffer locations by network flows", *Proc. ISPD*, April 2000.