Global Routing with Timing Constraints

Stephan Held, Dirk Müller, Daniel Rotter, Rudolf Scheifele, Vera Traub, Jens Vygen

***Abstract*—We show how to incorporate global static timing constraints into global routing. Our approach is based on the min-max resource sharing model that proved successful for global routing in theory and practice. Static timing constraints are modeled by a linear number of additional resources and customers. The algorithm dynamically adjusts delay budgets and can, thus, trade off wiring congestion for delay.**

As a subroutine, the algorithm routes a single net. If this subroutine is near-optimal, we will find near-optimal solutions for the overall problem very efficiently. The approach works for many delay models; here we discuss a linear delay model (before buffering) and the Elmore delay model (after buffering).

We demonstrate the benefit of our timing-constrained global routing algorithm by experimental results on industrial chips.

***Index Terms*—global routing, timing constraints, RC delay, interconnect synthesis**

I. Introduction

Global routing is an essential part of any chip design flow, not only as a preparation for detailed routing, but also for the early detection of routing congestion problems, and as input to other algorithms, in particular those for buffering. The global routing greatly impacts the timing behavior of a chip. If the routing performs detours to avoid congestion, timing constraints might get violated. In fact, critical nets are competing for using congested routing resources.

This situation is shown in Figure 1. Assume that the paths $P_1 = (N_1, N_2, N_3)$ and $P_2 = (N_4, N_5, N_3)$ are equally critical. There are two congested areas shown in light red. Only one net fits into each of them. A timing-unaware router might create a detour for N_1 and N_2, letting P_1 fail its timing requirement. Pre-computed delay budgets would be unlikely to help, because they typically distribute the budgets evenly to N_1, N_2 and N_4, N_5 so that no net might be allowed to avoid the congested area. The only feasible solution might be to allow a detour of N_1 and N_5 or, alternatively, for N_4 and N_2, so that only one net per path contains a detour.

The situation is complicated further by modern metal stacks, where the wire widths and heights increase from layer to layer [2], [44] so that the speed of a signal greatly depends on the chosen layers and on the wire widths and spacings. To achieve the fastest possible signal delay, nets compete for using the limited routing resources on higher routing layers and for choosing wider width and spacing than the layer-specific routing pitch.

For this reason, many design flows contain a step called *layer assignment*, which assigns a wire type and a range of wiring planes to nets that are timing-critical [38], [19]. These end up as constraints for routing [30], [25]. Recently, a congestion-aware layer assignment was proposed [44]. It uses global routing as a black box for congestion estimation. However, it can only assign entire nets, including connections to uncritical sinks, where it would often be sufficient to assign parts of a net (near the source, or like a backbone). For such precise decisions, global timing constraints must be embedded into global routing algorithms, with an as good timing model as possible.

All authors are with the Research Institute for Discrete Mathematics, University of Bonn, Lennéstr. 2, 53113 Bonn, Germany (e-mail: {held,mueller,rotter,scheifele,traub,vygen}@or.uni-bonn.de)

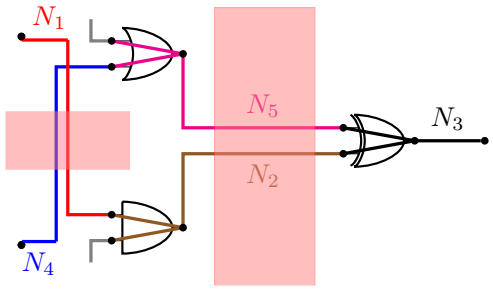


Fig. 1. Two timing-critical paths compete for space in a congested area (light red rectangles).

A. Timing-driven global routing algorithms

Modeling global routing as a multicommodity flow problem has a long tradition, starting with Shragowitz and Keel [41]. In a more general form, it was modeled as a min-max resource sharing problem by Müller, Radke, and Vygen [29]. As shown in [28], [29], [43], it is easy to integrate power consumption, coupling, or manufacturing yield.

There have been several approaches to consider timing within a multicommodity flow or resource sharing formulation. Net-based delay bounds were proposed by Huang et al. [17], rejecting Steiner trees that violate the delay bound. They were generalized to path-based delay bounds by Hong et al. [15], now discarding Steiner trees for a net that result in a delay violation of a path through that net. A fully polynomial time approximation scheme (FPTAS) for minimizing the total wire and buffer area with respect to buffer and wire space as well as net-based delay constraints for two-terminal nets was developed by Albrecht et al. [1]. Net-based and path-based delay bounds were considered by Vygen [43] differently from [17] and [15]. Here, delay bounds on paths are treated in the same way as routing space constraints. That is, delay violations are not banned but minimized simultaneously with routing congestion.

Other notable approaches outside the resource sharing framework were proposed by Hu and Sapatnekar [18], Yan and Lin [45], and Yan, Lee, and Chen [46]. They all start with timing-driven but congestion-unaware Steiner trees for all nets and differ in the way these trees are embedded. A recursive bisection of the chip area is performed in [18]. In each iteration, they embed the trees and try to minimize congestion, using the

flexibility of soft (diagonal) edges and by shifting Steiner points. This flexibility of embedding the tree topology was combined with a probabilistic algorithm in [45] and a multi-level routing algorithm in [46].

Recently, Samanta et al. [37] proposed to precompute a set of alternative timing-driven Steiner trees for each net. Then, they find near-optimal fractional solutions minimizing the total quadratic over-congestion, choosing convex combinations of the pre-processed alternatives.

However, all these approaches have serious drawbacks: The approaches proposed in [1], [17], [18], [37], [45], [46] require delay budgets for all nets as input. Computing budgets that guarantee that all timing constraints will be met *and* allow for a feasible routing solution is a challenging task. This can hardly be done outside a global router.

In [15], timing constraints are guaranteed at any stage, but this may prevent the router to remove routing congestion. This is because delay violations are never accepted although they could often be compensated by a better Steiner tree for another net on the same path.

The path-based formulation in [43] requires enumerating critical paths. Their number can be exponential in the size of the netlist. If we use only a small fraction of paths, a previously non-critical path can become critical and many iterations might be required.

B. Our contribution

To overcome the above limitations, we propose a new model based on the min-max resource sharing framework. It integrates all static timing constraints as a linear number of additional resources and customers into the resource sharing model for global routing. Our model works for RC-delay, linear delay, and other delay models in which a net determines the delay through its driving gate and the net itself. An implicit delay budgeting is done as part of the algorithm.

The algorithm requires a subroutine (called routing oracle) that routes a single net. Ideally, it minimizes the total cost of used resources, where for each resource we multiply its current price with the amount that we use. If the routing oracle is near-optimal, we prove strong convergence and near-optimality guarantees.

Our overall algorithm first computes a solution to the fractional relaxation of the global routing problem with a near-linear number of calls to the routing oracle. Finally, the fractional solution is rounded using the well-known randomized rounding technique [33].

In our implementation, we are concentrating on two delay models. First, we consider linear wire delays that are used in design steps before global buffering [4], [44]. In this case, we present a polynomial-time exact routing oracle for a bounded number of terminals. Our implementation contains speed-up tricks and runs fast also for nets with many terminals.

Second, we consider RC delays, which is appropriate if global routing is run after buffering. Here, we present an RC-aware path search for point-to-point connections and a topology algorithm for multi-sink nets. The RC-aware path search uses estimates for the Elmore delay through a wire which turn out to be quite precise in practice. The topology algorithm is a constant-factor approximation algorithm in a simplified two-dimensional setting without considering routing congestion – congestion costs and different layer characteristics are incorporated heuristically in our implementation.

In experiments on industrial microprocessor units, we demonstrate that our algorithm significantly improves signal delays and routing congestion compared to an industrial design flow.

In Section II, we recapitulate the min-max resource sharing model for global routing from [29], and show how to integrate timing constraints. Our algorithm requires two subroutines. The so-called arrival time oracle is the subject of Section III. Sections IV and V discuss the routing oracles and present our approaches to compute delay-aware Steiner trees in the two considered delay models. We conclude with a description of our overall algorithm in Section VI and experimental results in Section VII.

This paper is based on uniting the conference papers [14] and [39] but also contains several theoretical and practical improvements, simplifications, and new experimental results.

II. Resource sharing with timing constraints

A. Min-max resource sharing

In the min-max resource sharing problem, we are given finite sets $\mathcal{R}$ of *resources* and $\mathcal{C}$ of *customers*, a convex set B_c (the possible *solutions* for c), called *block*, for every $c \in \mathcal{C}$, and a convex function $\text{usg}_{c,r} \colon B_c \to \mathbb{R}_{\geq 0}$ for every $c \in \mathcal{C}$ and $r \in \mathcal{R}$, specifying the percentage that a specific solution consumes of resource r. The task is to find a solution for all customers approximately minimizing the maximum resource usage, i.e., to find $b(c) \in B_c$ for all $c \in \mathcal{C}$ approximately attaining

$$\lambda^* := \inf \left\{ \max_{r \in \mathcal{R}} \sum_{c \in \mathcal{C}} \text{usg}_{c,r}(b(c)) : b(c) \in B_c \, (c \in \mathcal{C}) \right\}. \quad (1)$$

In traditional global routing, $\mathcal{C}$ is the set $\mathcal{N}$ of nets, the block of a net contains all Steiner trees connecting the pins of this net, the resources are the edges of the global routing graph, and $\text{usg}_{c,r}$ tells which fraction of the available space a certain Steiner tree would consume at each edge. Then, $\max_{r \in \mathcal{R}} \sum_{c \in \mathcal{C}} \text{usg}_{c,r}(b(c))$ is the worst relative congestion of any edge, and λ^* is the minimum attainable worst relative congestion.

In the abstract model, the blocks are given implicitly by *oracle* functions that optimize linear functions over the blocks. In the traditional global routing example, these oracle functions are implemented by an algorithm for computing an approximately minimum-cost Steiner tree for a given terminal set in the global routing graph. In general, an oracle $f_c \colon \mathbb{R}^{\mathcal{R}}_{\geq 0} \to B_c$ computes for given prices $\text{price}_r \geq 0$ ($r \in \mathcal{R}$) an element of $b(c) \in B_c$ approximately minimizing $\sum_{r \in \mathcal{R}} \text{price}_r \text{usg}_{c,r}(b(c))$. Müller, Radke and Vygen [29] described a simple algorithm that yields:

Theorem 1 ([29]): One can solve the min-max resource sharing problem with approximation ratio $\sigma(1+\omega)$ for any $\omega > 0$ in $\mathcal{O}(\theta(|\mathcal{C}| + |\mathcal{R}|) \log |\mathcal{R}| (\log\log |\mathcal{R}| + \omega^{-2}))$ time. Here, $\sigma \geq 1$ is a constant bounding the approximation ratio of

the oracle and θ is the time for an oracle call. If $\frac{1}{2} \le \lambda^* \le 2$, the running time reduces to $\mathcal{O}(\theta(|\mathcal{C}| + |\mathcal{R}|)\omega^{-2} \log |\mathcal{R}|)$.

The algorithm proceeds essentially as follows. After an initialization of the prices (to one), every iteration does the following for each customer (in arbitrary order):

(a) call the oracle with the current prices, and
(b) update the prices of the resources used: they depend exponentially on the total use of a resource during the course of the algorithm. More precisely, whenever an amount of α_r is consumed of resource r, we set

$$\text{price}_r \leftarrow \text{price}_r \cdot e^{\gamma \cdot \alpha_r}, \tag{2}$$

where $\gamma > 0$ is a parameter (that can depend on ω).

The prices can be interpreted as Lagrange multipliers. After a certain number of iterations, the output for each customer is simply the (fractional) arithmetic mean of the computed solutions. Müller, Radke and Vygen [29] described the algorithm in detail and proposed important speed-up techniques.

The arithmetic mean of the computed solutions is an element of the block due to the convexity assumption, but it is not a useful global routing output. In the traditional global routing case, we get a convex combination of Steiner trees for each net, but we would like to output only one Steiner tree. Therefore we apply randomized rounding at the end (cf. [33], [29]).

B. Timing model

The nets are still customers (but no longer the only ones). For a net N we call the elements of B_N the *routing solutions* for N, and we say that $b = (b(N))_{N \in \mathcal{N}}$ is a *routing* if $b(N) \in B_N$ for all $N \in \mathcal{N}$.

Static timing constraints are modeled using an acyclic digraph D. The vertex set of D is defined as $V(D) := V_{\text{in}} \dot\cup V_{\text{out}} \dot\cup V_{\text{gate}}$, where V_{in} denotes the set of primary inputs and latch outputs, V_{out} the set of primary outputs and latch inputs, and V_{gate} the set of input pins of gates. The edges of D correspond to signal propagation, i.e. D contains an edge (u, v) if either $u \in V_{\text{in}}$ and v is a sink pin of the net driven by u or if $u \in V_{\text{gate}}$ and v is a sink pin of a net driven by an output pin of the circuit that has u as input pin. For every $v \in V_{\text{in}}$ an arrival time $\text{at}(v)$ and for every $v \in V_{\text{out}}$ a required arrival time $\text{rat}(v)$ is given as input. Timing constraints are then modeled by the usual inequalities

$$a(v) = \text{at}(v) \qquad \text{for all } v \in V_{\text{in}}, \tag{3}$$

$$a(v) = \text{rat}(v) \qquad \text{for all } v \in V_{\text{out}}, \tag{4}$$

$$a(u) + d_b(u, v) \le a(v) \qquad \text{for all } (u, v) \in E(D), \tag{5}$$

where $d_b(u, v)$ is the signal propagation delay along the edge $(u, v) \in E(D)$ that is determined by b. We assume that the signal propagation delay $d_b(u, v)$ from u to v only depends on the routing solution $b(\text{N}(v))$ of the net containing v. Examples are linear delays and RC-delays.

The numbers $a(v)$ will be fixed to $\text{at}(v)$ for $v \in V_{\text{in}}$ and variables for $v \in V_{\text{gate}} \cup V_{\text{out}}$. It would also be possible to fix the numbers $a(v)$ to $\text{rat}(v)$ for $v \in V_{\text{out}}$, but we will present a flexible timing relaxation approach that allows violating required arrival times at high costs. A routing solution $b = (b(N))_{N \in \mathcal{N}}$ meets all timing constraints if and only if there are numbers $a(v)$ for all $v \in V(D)$ such that (3)–(5) hold.

All our results remain valid if a net influences the delay of several edges. Formally, each function $b \mapsto d_b(u, v)$ must be positive, separable (i.e., $d_b(u, v) = \sum_{N \in \mathcal{N}} f(b(N))$ for some nonnegative functions f, which holds trivially if the delay depends only on $\text{N}(v)$), and convex. But the convexity is trivially given with the following specification of a block. For a net $N \in \mathcal{N}$, let $\mathcal{Y}_N$ denote the set of its Steiner trees (possibly with assigned space or wire types). We define the block for N as the convex set $B_N := \{\beta \in [0, 1]^{\mathcal{Y}_N} : \sum_{Y \in \mathcal{Y}_N} \beta_Y = 1\}$. Now, for an edge $(u, v) \in E(D)$ and a (fractional) solution $b(\text{N}(v)) = \beta \in B_{\text{N}(v)}$, we assign the delay as

$$d_b(u, v) := \sum_{Y \in \mathcal{Y}_{\text{N}(v)}} \beta_Y d_Y(u, v),$$

where $d_Y(u, v)$ is the delay imposed by the Steiner tree Y. This way, $b \mapsto d_b(u, v)$ is a linear (and thus convex) function. Also the consumption $\text{usg}_{N,r}(b(N))$ of a resource r is simply the convex combination of the consumptions by the Steiner trees. Note that the running time of the resource sharing algorithm does not depend on the dimension of B_N: we compute one Steiner tree per oracle call, and never store explicitly all the entries of $b(N)$, most of which will be zero.

There are other possibilities to define V_{gate}. One could simply choose all vertices of the timing graph. Our approach reduces the number of vertices (which will become customers in the resource sharing formulation) and reflects the typical stage of delay calculation through a gate and a net. In presence of macros with high fan-ins, we can use the timing graph directly to avoid a quadratic number of edges.

C. Lower and upper bounds on arrival times

We begin by computing an interval of reasonable arrival times for each vertex. To this end, let $d_{\text{lb}}(u, v)$ be a lower bound on the delay from u to v.

Let $a^{\rightarrow}_{\text{lb}}(v) := \text{at}(v)$ for $v \in V_{\text{in}}$ and

$$a^{\rightarrow}_{\text{lb}}(v) := \max\left\{a^{\rightarrow}_{\text{lb}}(u) + d_{\text{lb}}(u, v) : (u, v) \in \delta^-(v)\right\}$$

for $v \in V_{\text{gate}} \cup V_{\text{out}}$, where $\delta^-(v)$ denotes the set of edges in $E(D)$ entering v.

In practice, even with these lower bound delays we will often not meet the required arrival times. Therefore we compute an upper bound on the worst slack and allow violating any required arrival time by

$$\text{relax} := \max\{0,\ \max\{a^{\rightarrow}_{\text{lb}}(v) - \text{rat}(v) : v \in V_{\text{out}}\}\}.$$

Now, let $a^{\leftarrow}_{\text{lb}}(v) := \text{rat}(v) + \text{relax}$ for $v \in V_{\text{out}}$ and

$$a^{\leftarrow}_{\text{lb}}(v) := \min\left\{a^{\leftarrow}_{\text{lb}}(w) - d_{\text{lb}}(v, w) : (v, w) \in \delta^+(v)\right\}$$

for $v \in V_{\text{gate}} \cup V_{\text{in}}$, where $\delta^+(v)$ denotes the set of edges in $E(D)$ leaving v. By our choice of relax we have:

Proposition 2: $a^{\rightarrow}_{\text{lb}}(v) \le a^{\leftarrow}_{\text{lb}}(v)$ for all $v \in V(D)$. ∎

For uncritical paths, these intervals can be very large, which would lead to a slow convergence. Therefore we tighten them as follows. Let $d_{\text{ub}}(u, v)$ be a supposed upper bound on the delay from u to v; we will describe in Sections IV-D and V-D how we compute them. We assume $0 < d_{\text{lb}}(u, v) \le d_{\text{ub}}(u, v)$ for all $(u, v) \in E(D)$ for the rest of this paper.

Let $a^{\leftarrow}_{\text{ub}}(v) := \text{rat}(v)$ for $v \in V_{\text{out}}$ and

$$a^{\leftarrow}_{\text{ub}}(v) := \min\left\{a^{\leftarrow}_{\text{ub}}(w) - d_{\text{ub}}(v, w) : (v, w) \in \delta^+(v)\right\}$$

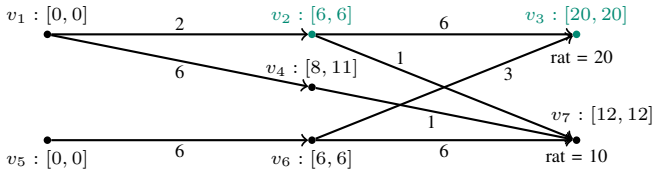


Fig. 2. Example of a timing graph D with arrival time intervals $[a_{\min}(v), a_{\max}(v)]$ shown at each $v \in V(D)$, assuming arrival time 0 at primary inputs v_1 and v_5, required arrival times at v_3 and v_7 as shown, lower bound delays $d_{\text{lb}}(u, v)$ as shown at each $(u, v) \in E(D)$, and $d_{\text{ub}} := 2d_{\text{lb}}$. In this example, we have relax $= 2$ because of the path from v_5 to v_7. The arrival time intervals of the black vertices are determined by (6). The green vertices are uncritical, hence their arrival time intervals are determined by (7).

for $v \in V_{\text{gate}} \cup V_{\text{in}}$.

Let $a^{\rightarrow}_{\text{ub}}(v) := \text{at}(v)$ for $v \in V_{\text{in}}$ and

$$a^{\rightarrow}_{\text{ub}}(v) \;:=\; \max\left\{a^{\rightarrow}_{\text{ub}}(u) + d_{\text{ub}}(u, v) : (u, v) \in \delta^-(v)\right\}$$

for $v \in V_{\text{gate}} \cup V_{\text{out}}$.

Choosing $a(v)$ smaller than $a^{\leftarrow}_{\text{ub}}(v)$ is pointless because $a^{\leftarrow}_{\text{ub}}(v)$ leaves enough time for all paths from v to V_{out}. In other words, $a^{\leftarrow}_{\text{ub}}(v)$ is a lower bound on the required arrival time at v, while $a^{\leftarrow}_{\text{lb}}(v)$ is an upper bound on the relaxed required arrival time at v.

Similarly, $a^{\rightarrow}_{\text{lb}}(v)$ and $a^{\rightarrow}_{\text{ub}}(v)$ are a lower and upper bound on the arrival time at v. We have:

Proposition 3: $a^{\rightarrow}_{\text{lb}}(v) \le a^{\rightarrow}_{\text{ub}}(v)$ and $a^{\leftarrow}_{\text{ub}}(v) \le a^{\leftarrow}_{\text{lb}}(v)$ for all vertices $v \in V(D)$. ∎

If $a^{\leftarrow}_{\text{ub}}(v) \le a^{\rightarrow}_{\text{ub}}(v)$, we define our final arrival time interval by

$$\begin{aligned} a_{\min}(v) &:= \max\{a^{\rightarrow}_{\text{lb}}(v), a^{\leftarrow}_{\text{ub}}(v)\}, \\ a_{\max}(v) &:= \min\{a^{\leftarrow}_{\text{lb}}(v), a^{\rightarrow}_{\text{ub}}(v)\}. \end{aligned} \tag{6}$$

Note that the interval $[a_{\min}(v), a_{\max}(v)]$ is nonempty due to Propositions 2 and 3.

However, $a^{\leftarrow}_{\text{ub}}(v)$ can be larger than $a^{\rightarrow}_{\text{ub}}(v)$. We call v *uncritical* if $a^{\leftarrow}_{\text{ub}}(v) \ge a^{\rightarrow}_{\text{ub}}(v)$. If v is uncritical, we fix the arrival time by setting

$$a_{\min}(v) := a_{\max}(v) := \tfrac{1}{2}\left(a^{\rightarrow}_{\text{ub}}(v) + a^{\leftarrow}_{\text{ub}}(v)\right). \tag{7}$$

As long as at least one endpoint of an edge is uncritical, inequality (5) will be satisfied:

Lemma 4: For every edge $(v, w) \in E(D)$ for which v or w is uncritical, we have $a_{\max}(v) + d_{\text{ub}}(v, w) \le a_{\min}(w)$.

PROOF. Case 1: $a^{\leftarrow}_{\text{ub}}(v) \ge a^{\rightarrow}_{\text{ub}}(v)$ and $a^{\leftarrow}_{\text{ub}}(w) < a^{\rightarrow}_{\text{ub}}(w)$. Then $a_{\max}(v) \le a^{\leftarrow}_{\text{ub}}(v) \le a^{\leftarrow}_{\text{ub}}(w) - d_{\text{ub}}(v, w) \le a_{\min}(w) - d_{\text{ub}}(v, w)$.

Case 2: $a^{\leftarrow}_{\text{ub}}(v) < a^{\rightarrow}_{\text{ub}}(v)$ and $a^{\leftarrow}_{\text{ub}}(w) \ge a^{\rightarrow}_{\text{ub}}(w)$. Then $a_{\max}(v) + d_{\text{ub}}(v, w) \le a^{\rightarrow}_{\text{ub}}(v) + d_{\text{ub}}(v, w) \le a^{\rightarrow}_{\text{ub}}(w) \le a^{\leftarrow}_{\text{ub}}(w) \le a_{\min}(w)$.

Case 3: $a^{\leftarrow}_{\text{ub}}(v) \ge a^{\rightarrow}_{\text{ub}}(v)$ and $a^{\leftarrow}_{\text{ub}}(w) \ge a^{\rightarrow}_{\text{ub}}(w)$. Then $a_{\max}(v) + d_{\text{ub}}(v, w) = \frac{1}{2}\left(a^{\rightarrow}_{\text{ub}}(v) + a^{\leftarrow}_{\text{ub}}(v)\right) + d_{\text{ub}}(v, w) \le \frac{1}{2}\left(a^{\leftarrow}_{\text{ub}}(w) + a^{\rightarrow}_{\text{ub}}(w)\right) = a_{\min}(w)$. ∎

Fig. 2 shows an example of a timing graph with arrival time bounds as defined in (6) and (7). The edges (v_1, v_2) and (v_6, v_3) (both Case 2), (v_2, v_7) (Case 1), and (v_2, v_3) (Case 3) have an uncritical endpoint.

D. Delay resources and arrival time customers

To model the timing constraints in the min-max resource sharing formulation of global routing, we add every edge $e \in E(D)$ as a new resource to $\mathcal{R}$, and each $v \in V(D)$ as a new customer to $\mathcal{C}$ whose purpose is to determine an arrival time $a(v) \in B_v$, where

$$B_v \;:=\; [a_{\min}(v), a_{\max}(v)]$$

for all $v \in V(D)$. An *arrival time solution* consists of numbers $a(v) \in B_v$ for all $v \in V(D)$. Our new set of customers is comprised of net and arrival time customers: $\mathcal{C} = \mathcal{N} \dot\cup V(D)$. It is of course not necessary to add customers for those vertices whose arrival time is already fixed, but it simplifies the notation.

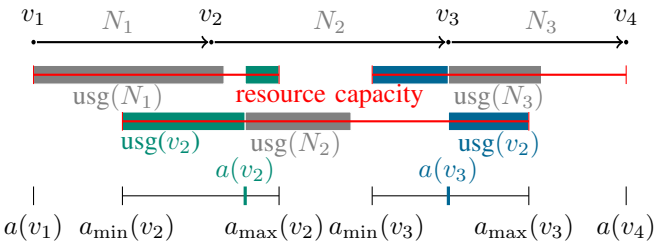


Fig. 3. Arrival time customers v_1, v_2, v_3, v_4 sharing delay resources (red) with the arcs N_1, N_2, and N_3.

The resource $e = (u, v) \in E(D)$ can be consumed only by the customers u, v, and $N := \text{N}(v)$: For $b(N) \in B_N$, $a(u) \in B_u$, and $a(v) \in B_v$ we define the usages of e as

$$\begin{aligned} \text{usg}_{u,e}(a(u)) &:= \frac{a(u) - a_{\min}(u)}{a_{\max}(v) - a_{\min}(u)}, \\ \text{usg}_{N,e}(b(N)) &:= \frac{d_b(u, v)}{a_{\max}(v) - a_{\min}(u)}, \\ \text{usg}_{v,e}(a(v)) &:= \frac{a_{\max}(v) - a(v)}{a_{\max}(v) - a_{\min}(u)}, \end{aligned} \tag{8}$$

while $\text{usg}_{N',e}$ and $\text{usg}_{v',e}$ are constantly zero for all $N' \in \mathcal{N} \setminus \{N\}$ and $v' \in V(D) \setminus \{u, v\}$. We call the denominator $a_{\max}(v) - a_{\min}(u)$ the *delay capacity* of e and require that it is positive, which is naturally fulfilled if $d_{\text{lb}}(u, v) > 0$.

Figure 3 illustrates (for a path $v_1 \to v_2 \to v_3 \to v_4$) how the delay resources are shared. The arrival time intervals are shown at the bottom in black. The edge (v_i, v_{i+1}) has delay capacity $a_{\max}(v_{i+1}) - a_{\min}(v_i)$, shown as red intervals ($i = 1, 2, 3$). The arrival times $a(v_1)$ and $a(v_4)$ are fixed, and possible values for the arrival times $a(v_2)$ and $a(v_3)$ are shown as green and blue vertical bars inside their interval. They imply the resource consumptions (numerators of the above usg-functions) indicated as green and blue horizontal bars. Net delays are shown by grey bars.

For each $v \in V_{\text{out}}$ we also add a *relaxation resource* $r_{\text{relax}}(v)$ to $\mathcal{R}$. Only the arrival time customer v consumes this resource, and its usage is

$$\text{usg}_{v, r_{\text{relax}}(v)}(a(v)) \;:=\; 1 + \omega \frac{a(v) - a_{\min}(v)}{a_{\max}(v) - a_{\min}(v)}$$

so that any time beyond $a_{\min}(v)$ will lead to a usage of more than 1, and the usage on this resource is $1 + \omega$ if we relax this by the full amount. Here $\omega > 0$ is the parameter from Theorem 1.

As all resource consumption functions are nonnegative and convex, we have indeed constructed an instance of the min-max resource sharing problem.

E. Timing properties of low-congestion solutions

Our model is justified by the following lemma.

Lemma 5: Let $(b(N))_{N\in\mathcal{N}}$ be a routing and $(a(v))_{v\in V(D)}$ an arrival time solution. Then (b,a) meets all timing constraints (3) – (5) if and only if $a^{\rightarrow}_{\text{lb}}(v) \le \text{rat}(v)$ and $\text{usg}_{v,r_{\text{relax}}(v)}(a(v)) = 1$ for all $v \in V_{\text{out}}$ and

$$\text{usg}_{u,e}(a(u)) + \text{usg}_{\text{N}(v),e}(b(\text{N}(v)) + \text{usg}_{v,e}(a(v)) \;\le\; 1$$

for all $e = (u,v) \in E(D)$.

PROOF. (3) is trivially fulfilled since arrival times are fixed for all $v \in V_{\text{in}}$. $a^{\rightarrow}_{\text{lb}}(v) \le \text{rat}(v)$ for all $v \in V_{\text{out}}$ is clearly necessary for meeting all required arrival times. If $a^{\rightarrow}_{\text{lb}}(v) \le \text{rat}(v)$ for $v \in V_{\text{out}}$ we have $a_{\min}(v) = \text{rat}(v)$ and hence $\text{usg}_{v,r_{\text{relax}}(v)}(a(v)) = 1$ is equivalent to $a(v) = \text{rat}(v)$, i.e. (4).

Let $e = (u,v) \in E(D)$. Consider the inequality

$$\begin{aligned} &\frac{a(u) - a_{\min}(u) + d_b(u,v) + a_{\max}(v) - a(v)}{a_{\max}(v) - a_{\min}(u)} \\ =\;& \text{usg}_{u,e}(a(u)) + \text{usg}_{\text{N}(v),e}(b(\text{N}(v))) + \text{usg}_{v,e}(a(v)) \\ \le\;& 1. \end{aligned}$$

Multiplying by $a_{\max}(v) - a_{\min}(u)$ (> 0) shows that this is equivalent to $a(u) + d_b(u,v) - a(v) \le 0$, i.e. (5). ∎

Therefore, we can find a feasible global routing solution that meets all timing constraints by solving (1) if such a solution exists.

Let $d_b(P) := \sum_{(u,v)\in E(P)} d_b(u,v)$ be the total delay along a path P. Then we also have:

Lemma 6: If no resource is used by more than $1+\omega$, then the worst slack

$$s(b) \;:=\; \min_{\substack{P \text{ path from} \\ x\in V_{\text{in}} \text{ to } z\in V_{\text{out}}}} \big(\text{rat}(z) - \text{at}(x) - d_b(P)\big), \tag{9}$$

is at least $-(\text{relax} + \omega H)$, where

$$H \;:=\; \max_{\substack{P \text{ path from} \\ V_{\text{in}} \text{ to } V_{\text{out}}}} d_{\text{ub}}(P) + \max_{\substack{P \text{ path from} \\ V_{\text{in}} \text{ to } V_{\text{out}}}} \big(d_{\text{ub}}(P) - d_{\text{lb}}(P)\big)|E(P)|.$$

PROOF. Let $\Delta^{\rightarrow}(v) := \max_{Q \text{ path to } v}(d_{\text{ub}}(Q) - d_{\text{lb}}(Q))$, $\Delta^{\leftarrow}(v) := \max_{Q \text{ path from } v}(d_{\text{ub}}(Q) - d_{\text{lb}}(Q))$, and $\Delta(v) = \Delta^{\rightarrow}(v) + \Delta^{\leftarrow}(v)$ for $v \in V(D)$. Note that $\Delta(v) \le \max_{\substack{P \text{ path from} \\ V_{\text{in}} \text{ to } V_{\text{out}}}} \big(d_{\text{ub}}(P) - d_{\text{lb}}(P)\big)$ for all $v \in V(D)$.

Let $E^{\text{uncritical}} := \{e = (v,w) \in E(D) : a_{\max}(v) + d_{\text{ub}}(v,w) \le a_{\min}(w)\}$. Note that we will never violate any resource $e \in E^{\text{uncritical}}$, so these resources could actually be removed without harm (as long as delays are really bounded by d_{ub}). By Lemma 4, $E^{\text{uncritical}}$ includes all edges for which at least one of the endpoints is uncritical (but may contain more edges).

For every edge $(v,w) \in E(D) \setminus E^{\text{uncritical}}$ we have

$$a^{\rightarrow}_{\text{ub}}(v) + d_{\text{ub}}(v,w) \ge a_{\max}(v) + d_{\text{ub}}(v,w) > a_{\min}(w) \ge a^{\rightarrow}_{\text{lb}}(w)$$

and hence

$$\begin{aligned} &a_{\max}(w) - a_{\min}(v) \\ \le\;& a^{\rightarrow}_{\text{ub}}(w) - a^{\rightarrow}_{\text{lb}}(v) \\ <\;& a^{\rightarrow}_{\text{ub}}(w) - a^{\rightarrow}_{\text{lb}}(w) + \big(a^{\rightarrow}_{\text{ub}}(v) + d_{\text{ub}}(v,w) - a^{\rightarrow}_{\text{lb}}(v)\big) \\ \le\;& \Delta^{\rightarrow}(w) + \Delta^{\rightarrow}(v) + d_{\text{ub}}(v,w). \end{aligned}$$

Analogously, for every $(v,w) \in E(D) \setminus E^{\text{uncritical}}$ we get

$$a_{\max}(w) - a_{\min}(v) < \Delta^{\leftarrow}(w) + \Delta^{\leftarrow}(v) + d_{\text{ub}}(v,w).$$

Summing the two inequalities and dividing by two yields

$$a_{\max}(w) - a_{\min}(v) \;\le\; \tfrac{1}{2}(\Delta(v) + \Delta(w)) + d_{\text{ub}}(v,w).$$

If we sum over all edges on a path P from $x \in V_{\text{in}}$ to $z \in V_{\text{out}}$, except the uncritical edges, the result follows because the total delay exceeds $\text{rat}(z) + \text{relax} - \text{at}(x)$ by at most

$$\omega \sum_{(v,w)\in E(P)\setminus E^{\text{uncritical}}} (a_{\max}(w) - a_{\min}(v)).$$

∎

III. Arrival time oracle

Let $v \in V(D)$ be fixed in this section, and let $u_1,\dots,u_k$ be the predecessors and $w_1,\dots,w_l$ the successors of v in D. Choosing a locally optimal arrival time $a(v) \in B_v$ is easy:

Lemma 7: Given $\text{price}_e \ge 0$ for $e \in E(D)$, we can compute an arrival time $a(v) \in [a_{\min}(v), a_{\max}(v)]$ that minimizes $\sum_{e\in E(D)} \text{price}_e \text{usg}_{v,e}(a(v))$ in time $\mathcal{O}(|\delta^+(v)| + |\delta^-(v)|)$.

PROOF. Choosing $a(v) = t$ consumes $\frac{a_{\max}(v)-t}{a_{\max}(v)-a_{\min}(u_{k'})}$ from the resource $(u_{k'}, v)$ ($k' = 1,\dots,k$) and $\frac{t-a_{\min}(v)}{a_{\max}(w_{l'})-a_{\min}(v)}$ from the resource $(v, w_{l'})$ ($l' = 1,\dots,l$). Thus, choosing $a(v) = t$ has cost

$$\begin{aligned} f(t) \;=\;& \sum_{l'=1}^{l} \text{price}_{(v,w_{l'})} \frac{t - a_{\min}(v)}{a_{\max}(w_{l'}) - a_{\min}(v)} \\ &+ \sum_{k'=1}^{k} \text{price}_{(u_{k'},v)} \frac{a_{\max}(v) - t}{a_{\max}(v) - a_{\min}(u_{k'})}. \end{aligned}$$

Since $t \mapsto f(t)$ is a linear function, the optimal choice for $a(v)$ is $a_{\min}(v)$ if $f(a_{\min}(v)) < f(a_{\max}(v))$ and $a_{\max}(v)$ if $f(a_{\min}(v)) > f(a_{\max}(v))$. If equality holds, any choice $a(v) \in B_v$ would be optimal. ∎

After updating the resource costs according to the chosen arrival time, the other end of the interval $[a_{\min}(v), a_{\max}(v)]$ can become optimal. The easiest way to stabilize the arrival time oracle is to apply the following algorithm.

Algorithm 1 Iterated arrival time oracle

```
for i = 1,...,n do
    Compute a_i(v) ∈ {a_min(v), a_max(v)} to minimize f(a_i(v))
    price_r ↤ price_r · e^{γ·n^{-1}·usg_{v,r}(a_i(v))} for r ∈ R
return a(v) ↤ (1/n) Σ_{i=1}^{n} a_i(v).
```

Using Algorithm 1 we still maintain the overall convergence guarantee as the proof given in [29] shows. In practice, we observe better results: the arrival times converge rather than bouncing between the interval bounds. We now show how to avoid running Algorithm 1 explicitly.

Lemma 8: For $n \to \infty$, the output of Algorithm 1 converges to $\min\{\max\{a_{\min}(v), t^*\}, a_{\max}(v)\}$, where t^* is the unique root of the function

$$\begin{aligned} g(t) \;=\;& \sum_{l'=1}^{l} \frac{\text{price}_{(v,w_{l'})}}{a_{\max}(w_{l'}) - a_{\min}(v)} \cdot e^{\gamma \cdot \frac{t - a_{\min}(v)}{a_{\max}(w_{l'}) - a_{\min}(v)}} \\ &- \sum_{k'=1}^{k} \frac{\text{price}_{(u_{k'},v)}}{a_{\max}(v) - a_{\min}(u_{k'})} \cdot e^{\gamma \cdot \frac{a_{\max}(v) - t}{a_{\max}(v) - a_{\min}(u_{k'})}}. \end{aligned}$$

PROOF. W.l.o.g. we assume that $a_{\min}(v) < a_{\max}(v)$. First note that g is strictly monotonically increasing. Let price_r^i be the price of resource $r \in \mathcal{R}$ and f_i the function f at the start of iteration i. For $l' = 1, \dots, l$ and $k' = 1, \dots, k$ we have

$$\text{price}^i_{(v,w_{l'})} = \text{price}^1_{(v,w_{l'})} \cdot e^{\frac{\gamma}{n} \cdot \sum_{i'=1}^{i-1} \frac{a_{i'}(v) - a_{\min}(v)}{a_{\max}(w_{l'}) - a_{\min}(v)}} \quad \text{and}$$

$$\text{price}^i_{(u_{k'},v)} = \text{price}^1_{(u_{k'},v)} \cdot e^{\frac{\gamma}{n} \cdot \sum_{i'=1}^{i-1} \frac{a_{\max}(v) - a_{i'}(v)}{a_{\max}(v) - a_{\min}(u_{k'})}}.$$

If $t^* \le a_{\min}(v)$, then $0 \le g(a_{\min}(v))$ and for $i \le n$:

$$\begin{aligned} f_i(a_{\min}(v)) &= \sum_{k'=1}^{k} \text{price}^i_{(u_{k'},v)} \frac{a_{\max}(v) - a_{\min}(v)}{a_{\max}(v) - a_{\min}(u_{k'})} \\ &< \sum_{k'=1}^{k} \text{price}^1_{(u_{k'},v)} \frac{a_{\max}(v) - a_{\min}(v)}{a_{\max}(v) - a_{\min}(u_{k'})} e^{\gamma \frac{a_{\max}(v) - a_{\min}(v)}{a_{\max}(v) - a_{\min}(u_{k'})}} \\ &\le \sum_{l'=1}^{l} \text{price}^1_{(v,w_{l'})} \frac{a_{\max}(v) - a_{\min}(v)}{a_{\max}(w_{l'}) - a_{\min}(v)} \le f_i(a_{\max}(v)). \end{aligned}$$

Analogously, we will always choose $a_i(v) = a_{\max}(v)$ if $t^* \ge a_{\max}(v)$.

If $a_{\min}(v) < t^* < a_{\max}(v)$ we show that for fixed n, Algorithm 1 selects solution $a_{\min}(v)$ at most q_n times, where

$$q_n = \left\lceil \frac{a_{\max}(v) - t^*}{a_{\max}(v) - a_{\min}(v)} \cdot n \right\rceil.$$

For any iteration $i \le n$ in which we have already selected the left interval border q_n times, it holds that

$$\begin{aligned} &\frac{f_i(a_{\min}(v))}{(a_{\max}(v) - a_{\min}(v))} \\ &= \sum_{k'=1}^{k} \frac{\text{price}^1_{(u_{k'},v)}}{a_{\max}(v) - a_{\min}(u_{k'})} e^{\frac{\gamma}{n} \cdot q_n \cdot \frac{a_{\max}(v) - a_{\min}(v)}{a_{\max}(v) - a_{\min}(u_{k'})}} \\ &\ge \sum_{k'=1}^{k} \frac{\text{price}^1_{(u_{k'},v)}}{a_{\max}(v) - a_{\min}(u_{k'})} e^{\gamma \frac{a_{\max}(v) - t^*}{a_{\max}(v) - a_{\min}(u_{k'})}} \\ &= \sum_{l'=1}^{l} \frac{\text{price}^1_{(v,w_{l'})}}{a_{\max}(w_{l'}) - a_{\min}(v)} e^{\gamma \frac{t^* - a_{\min}(v)}{a_{\max}(w_{l'}) - a_{\min}(v)}} \\ &> \sum_{l'=1}^{l} \frac{\text{price}^1_{(v,w_{l'})}}{a_{\max}(w_{l'}) - a_{\min}(v)} \cdot \\ &\qquad e^{\frac{\gamma}{n}\left((i-1) - n \cdot \frac{a_{\max}(v) - t^*}{a_{\max}(v) - a_{\min}(v)}\right) \cdot \frac{a_{\max}(v) - a_{\min}(v)}{a_{\max}(w_{l'}) - a_{\min}(v)}} \\ &\ge \sum_{l'=1}^{l} \frac{\text{price}^1_{(v,w_{l'})}}{a_{\max}(w_{l'}) - a_{\min}(v)} e^{\frac{\gamma}{n}((i-1) - q_n) \cdot \frac{a_{\max}(v) - a_{\min}(v)}{a_{\max}(w_{l'}) - a_{\min}(v)}} \\ &= \frac{f_i(a_{\max}(v))}{(a_{\max}(v) - a_{\min}(v))}. \end{aligned}$$

Hence, we would never choose $a_{\min}$ again. An analogous calculation shows that we select the right interval border $a_{\max}(v)$ at most $\overline{q}_n$ times with $\overline{q}_n := \left\lceil \frac{t^* - a_{\min}(v)}{a_{\max}(v) - a_{\min}(v)} \cdot n \right\rceil$. Since $n \le q_n + \overline{q}_n \le n + 1$,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} a_i(v) = \lim_{n\to\infty} \frac{1}{n}\big(q_n a_{\min}(v) + \overline{q}_n a_{\max}(v)\big) = t^*.$$

■

We approximate t^* with Newton's method, which has a global quadratic convergence rate in our case.

Theorem 9: We can approximate the limit of the output of Algorithm 1 for $n \to \infty$ up to accuracy $\delta > 0$ in running time $\mathcal{O}\big((k+l) \cdot \log\log \frac{a_{\max}(v) - a_{\min}(v)}{\delta}\big)$.

PROOF. We use Lemma 8. If $g(a_{\min}(v)) \ge 0$ or $g(a_{\max}(v)) \le 0$, we return $a_{\min}(v)$ or $a_{\max}(v)$, respectively. So assume that $a_{\min}(v) < t^* < a_{\max}(v)$.

Substituting $x(t) := \frac{t - a_{\min}(v)}{a_{\max}(v) - a_{\min}(v)}$, we can write $g = \overline{g} \circ x$, where $\overline{g}$ is of the form

$$\overline{g}(x) = \sum_{j=1}^{m} p_j e^{\gamma q_j x}$$

for $m = k + l$ and constants $p_i, q_i \in \mathbb{R}$ with $p_j q_j > 0$ and $|q_j| \le 1$ for $j = 1, \dots, m$.

Starting with an arbitary $x_0 \in [0,1]$, Newton's method iteratively sets $x_{i+1} := x_i - \frac{\overline{g}(x_i)}{\overline{g}'(x_i)}$. By Taylor's theorem,

$$\begin{aligned} 0 &= \overline{g}(x(t^*)) \\ &= \overline{g}(x_i) + \overline{g}'(x_i) \cdot (x(t^*) - x_i) + \overline{g}''(\xi) \frac{(x(t^*) - x_i)^2}{2} \end{aligned}$$

for some value $\xi \in [\min\{x(t^*), x_i\}, \max\{x(t^*), x_i\}]$, which implies

$$|x(t^*) - x_{i+1}| = \frac{|\overline{g}''(\xi)|}{2|\overline{g}'(x_i)|} \cdot |x(t^*) - x_i|^2.$$

Now,

$$\frac{|\overline{g}''(\xi)|}{|\overline{g}'(x_i)|} \le \gamma \frac{\sum_{j=1}^{m} |p_j q_j^2 e^{\gamma q_j \xi}|}{\sum_{j=1}^{m} |p_j q_j e^{\gamma q_j x_i}|} \le \gamma e^{\gamma |\xi - x_i|} \le \gamma e^{\gamma |x(t^*) - x_i|}$$

for all ξ, x_i and hence,

$$|x(t^*) - x_i| \le \frac{\gamma e^{\gamma |x(t^*) - x_{i-1}|}}{2} \cdot |x(t^*) - x_{i-1}|^2.$$

As starting point x_0 we choose a point with $|x_0 - x(t^*)| \le \frac{1}{2\gamma}$. This can be achieved by a constant number of iterations of binary search for fixed γ. Then, by induction, for all $i \ge 0$ we have $|x(t^*) - x_i| \le |x(t^*) - x_0| \le \frac{1}{2\gamma}$ for all $i \ge 0$ and thus,

$$|x(t^*) - x_i| \le \left(\frac{\gamma e^{\gamma |x(t^*) - x_0|}}{2}\right)^{2^i - 1} \cdot |x(t^*) - x_0|^{2^i} < \frac{1}{\gamma} \cdot \frac{1}{2^{2^i}},$$

which yields the assertion. ■

IV. LINEAR DELAY ROUTING ORACLE

While for the classical global routing problem it suffices to have a subroutine that approximately solves the standard minimum cost Steiner tree problem, we need to compute Steiner trees that take both congestion and timing into account. More precisely, we are given a net N with source s and set of sinks $N \setminus \{s\} \subseteq V_{\text{gate}} \cup V_{\text{out}}$, and resource prices $\text{price}_r \ge 0$ for all resources $r \in \mathcal{R}$. For a sink v we abbreviate

$$\text{price}_v := \sum_{e=(u,v) \in \delta_D^-(v)} \frac{\text{price}_e}{a_{\max}(v) - a_{\min}(u)}.$$

Then the task of our oracle is to find a Steiner tree $Y \in \mathcal{Y}_N$ for N in the (three-dimensional) global routing graph G and a wire type $t(f)$ for each $f \in E(Y)$ such that

$$\sum_{f \in E(Y)} \text{price}_f \cdot \text{usg}_{N,f}(Y,t) + \sum_{v \in N \setminus \{s\}} \text{price}_v \cdot d_{Y,t}(v), \quad (10)$$

is minimized. Here, $\text{usg}_{N,f}(Y,t)$ denotes the percentage of the space at f that wire type $t(f)$ consumes (if $f \in E(Y)$), and $d_{Y,t}(v)$ is the delay from u to v induced by the routing

solution (Y,t), for $e=(u,v)\in E(D)$ with $v\in N\setminus\{s\}$. Note that in both delay models that we consider, this is independent of the predecessor u.

This problem is *NP*-hard regardless of the delay model, because it contains the rectilinear Steiner tree problem [11]. In this section, we focus on a linear delay model that is often used in design phases before and in particular in preparation for global buffering [4], [44]. In Section V we will consider RC delay.

A. Linear delay model

In the linear delay model, $d_{Y,t}(v)$ consists of a constant gate delay and a linear delay through the Steiner tree as follows. Each vertex $v\in V(G)$ of the global routing graph is associated with a position $p(v)=(x(v),y(v),l(v))\in\mathbb{R}^3$. Each edge $e\in E(G)$ is either a via, i.e. connects two vertices with the same x- and y-coordinates on adjacent layers, or a wire within one layer, with a geometric length $\mathrm{len}(e)>0$. Let $\mathrm{d_w}(l,t)$ denote the wire delay per length of a wire of type t on layer l. We denote the delay to traverse a via of type t between layers l and l' by $\mathrm{d_v}(l,l',t)$. Then, we define the delay that the signal needs to traverse a path P within a Steiner tree as

$$\sum_{\substack{e=\{u,v\}\in E(P):\\ l(u)=l(v)}}\mathrm{len}(e)\cdot\mathrm{d_w}(l(u),t(e))\;+\sum_{\substack{e=\{u,v\}\in E(P):\\ l(u)\neq l(v)}}\mathrm{d_v}(l(u),l(v),t(e)).$$

Details on how to estimate the values $\mathrm{d_w}$ and $\mathrm{d_v}$ can be found in Bartoschek et al. [4]. The delay imposed by the capacitance of side branches could be modeled as well [4], [13].

Even for this simple linear delay model, Chuzhoy et al. [9] showed that there is no routing oracle with approximation ratio $o(\log\log|N|)$ unless every problem in *NP* can be solved in $\mathcal{O}(n^{\log\log\log n})$ time (where n is the instance size). Here, $|N|$ denotes the number of pins of net N. The best known approximation algorithm is due to Meyerson, Munagala and Plotkin [27] and has approximation ratio $\mathcal{O}(\log|N|)$. However, as we will see, the problem can be solved optimally in polynomial time for constant $|N|$.

We propose a fast approach that first computes a topology and, then, embeds it into the global routing graph.

B. Shallow light routing topologies

First we compute a routing topology using the bicriteria approximation algorithm of Held and Rotter [13], generalizing the work of Khuller et al. [24]. It trades off the partially opposite objectives of bounding total length and path delays.

Theorem 10: ([13]) Given a net N with source s, a (short) rectilinear Steiner tree Y_0 for N, $2\geq\epsilon>0$, a constant $\mathrm{d_w}>0$, and delay bounds *rat*(v) for $v\in N\setminus\{s\}$ such that $\mathrm{rat}(v)\geq\mathrm{d_w}\cdot\mathrm{dist}(s,v)$, one can compute in $\mathcal{O}(|N|\log|N|)$ time a rectilinear Steiner tree Y for N such that for all $v\in N\setminus\{s\}$:

$$\mathrm{d_w}\sum_{e\in E(P_Y(s,v))}\mathrm{len}(e)\;\leq\;(1+\epsilon)\cdot\mathrm{rat}(v)\qquad\text{and}$$

$$\sum_{e\in E(Y)}\mathrm{len}(e)\;<\;\left(2+\left\lceil\log\left(\tfrac{2}{\epsilon}\right)\right\rceil\right)\cdot\sum_{e\in E(Y_0)}\mathrm{len}(e),$$

where $\mathrm{dist}(s,v)$ denotes the ℓ_1-distance between s and v and $P_Y(s,v)$ denotes the path from s to v in Y.

We choose $\mathrm{rat}(v)=\delta+\min\{a(v)-a(u):(u,v)\in\delta_D^-(v)\}$ for all $v\in N\setminus\{s\}$, where $\delta\in\mathbb{R}$ is chosen as small as possible so that $\delta\geq 0$ and $\mathrm{rat}(v)\geq\mathrm{d_w}\cdot\mathrm{dist}(s,v)$ for all $v\in N\setminus\{s\}$. The initial tree Y_0 is computed by a standard geometric Steiner tree algorithm. We choose $\epsilon\in[0.1,0.2]$, depending on how timing-critical the net currently is.

For large instances, we start with a geometric clustering of all sink pins using the algorithm by Maßberg and Vygen [26]. Bicriteria Steiner trees are computed separately for all clusters and for the top-level tree. For each cluster we place a root at the projection of the source into the cluster's bounding box. This clustering and two-stage topology generation allows us to use the wire delay parameter $\mathrm{d_w}$ corresponding to the lowest major routing layer inside the clusters and to the highest layers for the top-level tree.

C. Embedding a routing topology

We use the routing topology only to extract the connectivity information. We now embed a topology into the three-dimensional routing graph G, ignoring the location of its Steiner vertices. This problem can be solved optimally.

Theorem 11: An optimum embedding of a given routing topology for a net N into a graph G can be found in $\mathcal{O}\left(|N|\cdot(|\mathcal{T}|\cdot|E(G)|+|V(G)|\log|V(G)|)\right)$ time, where $|N|$ is the number of pins and $|\mathcal{T}|$ is the number of wire types.

PROOF. Let Y be a routing topology for a net N with source s. Orient Y as an arborescence with root s. Using local transformations, we may assume that the leaves are exactly the sinks of the net, s has out-degree 1, and every Steiner vertex has out-degree 2. For $w\in V(Y)$ we denote the subtree of Y with root w by $Y(w)$.

We embed edges of Y into G in reverse topological order. While processing an edge $(v,w)\in E(Y)$ we compute labels $(u,\alpha_{(v,w)}(u))_{(v,w)}$ for all $u\in V(G)$ corresponding to embeddings of $v+(v,w)+Y(w)$. The cost of the embedding will be at most $\alpha_{(v,w)}(u)$ and the position of v will be u. We also specify timing prices for all Steiner points.

If w is a sink, we run Dijkstra's algorithm from w in the graph $G':=(V(G),\{(x,y)_t,(y,x)_t:\{x,y\}\in E(G),\ t\text{ wire type}\})$. The price for traversing an edge $(x,y)_t\in E(G')$ is

$$\mathrm{cost}_w((x,y)_t)\;:=\;\mathrm{price}_{\{x,y\}}\cdot\mathrm{width}(t)+\mathrm{price}(w)\cdot\mathrm{d}((x,y)_t),\tag{11}$$

where

$$\mathrm{d}((x,y)_t)=\begin{cases}\mathrm{d_v}(l(x),l(y),t) & :\{x,y\}\text{ is type-}t\text{-via}\\ \mathrm{len}(\{x,y\})\cdot\mathrm{d_w}(l(x),t) & :\{x,y\}\text{ is type-}t\text{-wire.}\end{cases}$$

The permanent labels after termination of Dijkstra's algorithm are exactly as desired.

If w is a Steiner point in Y, let $\delta_Y^+(w)=\{z_1,z_2\}$ and assume that for $u\in V(G)$ we have already computed labels $(u,\alpha_{(w,z_i)}(u))_{(w,z_i)}$ $(i=1,2)$ as desired. We set $\mathrm{price}(w):=\mathrm{price}(z_1)+\mathrm{price}(z_2)$ and run Dijkstra's algorithm on the graph arising from G' by adding a new node w' and an edge (w',u) for all $u\in V(G)$. A new edge (w',u) gets cost $\mathrm{cost}_w((w',u)):=\alpha_{(w,z_1)}(u)+\alpha_{(w,z_2)}(u)$, while the cost of edges in G' are defined as in (11). Vertex w' serves as start

node for the path search. Due to the choice of costs of the edges outgoing of w', any w'-u-path found by this algorithm naturally corresponds to an embedding of $v + (v, w) + Y(w)$, and the algorithm produces labels as desired.

Finally, let $(s, w) \in E(Y)$ be the unique edge leaving s in Y. We output the embedding corresponding to label $(s, \alpha_{(s,w)}(s))_{(s,w)}$.

The running time of the overall algorithm is dominated by the $|V(Y)| - 1 = \mathcal{O}(|N|)$ applications of Dijkstra's algorithm, each taking time $\mathcal{O}(|\mathcal{T}| \cdot |E(G)| + |V(G)| \log |V(G)|)$.

To prove correctness, we show that for each $(v, w) \in E(Y)$ and $u \in V(G)$, there is no embedding of $v + (v, w) + Y(w)$ in which v is positioned at u and that has cost smaller than $\alpha_{(v,w)}(u)$.

If w is a sink, this is clear by correctness of Dijkstra's algorithm. Otherwise, let A^* be an optimum embedding of $v + (v, w) + Y(w)$ in which v has position u. Let p be the position of w in A^*. If $\delta^+(w) = \{z_1, z_2\}$, A^* consists of embeddings A_i of $w + (w, z_i) + Y(z_i)$ for $i = 1, 2$ plus an $p - u$ path. By induction and construction, the cost of the edges between w' and u in the modified graph created for the application of Dijkstra's algorithm during processing of $(v, w) \in E(Y)$ does not exceed the sum of costs of A_1 and A_2. By correctness of Dijkstra's algorithm, $\alpha_{(v,w)}(u)$ does not exceed the cost of A^*. ∎

Corollary 12: If $|N|$ is bounded by a constant, there is a linear delay routing oracle that finds an optimum solution in polynomial time.

PROOF. Enumerate and embed all $(2|N|-4)!/(2^{|N|-2}(|N|-2)!)$ topologies for N. ∎

One can also enumerate topologies during the embedding as in [16], but even then, this approach is too slow in practice.

In order to speed up the practical running time of the Algorithm of Theorem 11, we label only a subset of $V(G)$. Let (w, z_1), and (w, z_2) be the two outgoing edges of a Steiner node $w \in V(Y)$ and let (v, w) be the unique edge entering w in Y. We perform the two path searches that create labels of the form $(u, \alpha_{(w,z_1)}(u))_{(w,z_1)}$ and $(u, \alpha_{(w,z_2)}(u))_{(w,z_2)}$ simultaneously. When we have reached the first common point $q \in V(G)$ with both path searches, we store the sum ξ of the costs for both labels. In addition, we estimate the "future" cost ψ of embedding the edge entering w. We stop as soon as labels get larger than $\xi + \psi$. When we embed (v, w), we add edge (w', u) to G' if and only if u has been reached by both path searches before.

As an estimate for ψ one could use 0 or (better) start a path search from q to a grid point whose x, y coordinates equals the point of v in the pre-computed topology. In the second case, the embedding algorithm is optimum for up to three terminals.

When we embed the edge leaving s, we can stop as soon as we have reached s.

D. Lower and upper bounds

To define a lower bound $d_{\text{lb}}(v, w)$ on the delay between two pins v and w we compute the delay on a straight v-w-path on an optimum layer with optimum wire type. If v and w are located on layers $l(v)$, respectively $l(w)$, and their geometric distance in x/y direction is $\text{dist}(v, w)$, this delay is equal to

$$\min_{\substack{l \text{ layer},\\ t \text{ wire type}}} \left(\sum_{l'=\min\{l(v),l\}}^{\max\{l(v),l\}-1} d_v(l', l'+1, t) \right) + \text{dist}(v, w) \cdot d_w(l, t) + \left(\sum_{l'=\min\{l(w),l\}}^{\max\{l(w),l\}-1} d_v(l', l'+1, t) \right).$$

This optimum straight path delay plus the delay through the gate if $v \notin V_{\text{in}}$ yields $d_{\text{lb}}(v, w)$.

For the definition of $d_{\text{ub}}(v, w)$ we again start by computing the delay along a straight path. Instead of selecting optimum layer and wire type, we select the lowest available layer and the wire type with smallest width and spacing. We multiply this straight path delay with a *detour factor* that accounts for both

1. detours caused by the choice of the embedded topology, and
2. detours caused by the embedding of edges of that topology, and add a small constant.

The ratio between the ℓ_1-length of the path from the source s of $\text{N}(w)$ to w in the initial short Steiner tree Y_0 that is given as input to the algorithm of Held and Rotter [13] (cf. Theorem 10), and the geometric ℓ_1-distance between s an t is an upper bound on the detour due to the embedded topology. Bounding the detour caused by the embedding is more difficult. In our experiments, we chose an additional factor of 1.5. Statistics show that 99.8% of all computed Steiner trees stayed within that upper bound.

V. RC TREE ROUTING ORACLE

In this section we are going to present the RC tree routing oracle, which is intended to be used on an already buffered and well-optimized netlist. We will use the same problem formulation as in Section IV, so we again minimize (10), but now use the Elmore delay model instead [10], [36]. Moreover, in contrast to Section IV, we will not assign wire types to individual edges in our routing tree, but assume that every net has a fixed wire type to be used for the whole tree given in the input. Assigning wire types would be an enhancement of the RC tree oracle; this should be subject of future research. Due to the fixed wire type, we can w.l.o.g. assume $\text{usg}_{N,f}(Y, t) = 1$ for all edges f (by scaling the prices appropriately).

A. The Elmore delay model

We assume that for every layer l of the three-dimensional global routing graph we are given constants $\text{res}_{\text{wire}}(l)$ and $\text{cap}_{\text{wire}}(l)$ denoting the resistance and capacitance of a unit-length wire on l, and $\text{res}_{\text{via}}(l)$ and $\text{cap}_{\text{via}}(l)$ denoting the resistance and capacitance of a via from layer l to $l + 1$. These constants can be net-dependent due to differences in wire types, but since we are only showing how to compute a solution for one given net in this section without doing a wire type assignment, we will not make this distinction. Given such constants we can derive a resistance $\text{res}(e)$ and a capacitance $\text{cap}(e)$ for every edge e of the global routing graph. For the

rest of this section we will assume $\text{cap}_{\text{via}}(l) = 0$ for all layers l – this is an estimate that is commonly used in practice and it simplifies some of our results.

In order to define Elmore delay in our setting we will use the convention that any Steiner tree is an arborescence rooted at the source pin for the rest of this section. Given such a Steiner tree Y for a net with source s and set of sinks $N \setminus \{s\}$, a driver resistance $\text{res}(s) \geq 0$ and sink pin capacitances $\text{cap}\colon N \setminus \{s\} \to \mathbb{R}_{\geq 0}$, we can now define the Elmore delay $d_Y(v)$ between s and $v \in N \setminus \{s\}$ as

$$d_Y(v) := \text{res}(s) \cdot C_Y(s) + \sum_{e=(x,y) \in E(P_Y(s,v))} \text{res}(e) \left(\frac{\text{cap}(e)}{2} + C_Y(y) \right),$$

where we again denote the *subtree rooted at* $y \in V(Y)$ by $Y(y)$ in order to define the *downstream capacitance* $C_Y(y) := \sum_{e \in E(Y(y))} \text{cap}(e) + \sum_{t \in T \cap V(Y(y))} \text{cap}(t)$, and we let $P_Y(s,v)$ be the s-v path in Y.

Note that our definition of Elmore delay also includes the term $\text{res}(s) \cdot C_Y(s)$, which is an estimate for the gate delay if s is the output pin of a gate, or for the delay induced by the capacitance of Y if s is a primary input. The delay along an edge $e = (u,v) \in E(D)$ then is the Elmore delay between s and v, where s is the source pin of the net containing v.

As in (10), our routing oracle now has to minimize the weighted sum of Elmore delays and congestion costs. We call this problem the *Congestion-Aware Minimum Elmore Delay Steiner Tree Problem*.

The problem of constructing Steiner trees minimizing Elmore delay for multi-sink nets has been dealt with extensively in the literature. Boese et al. [7] proved the existence of an optimum solution on the Hanan grid when the weighted sum of source-sink delays is minimized. This allows them to solve the problem in exponential time. However, they give an example in [6] that the existence of an optimum solution on the Hanan grid is generally not given when the maximum source-sink delay is minimized. Kadodi [21] and Peyer [31] show how to minimize the maximum source-sink delay for instances with at most three sinks in constant time. Moreover, various heuristics have been implemented and evaluated in practice [7], [6], [42], [5], [32]; see also the book of Kahng and Robins [22] for an overview. However, in a more recent work, the first constant-factor approximation algorithm for constructing Steiner trees minimizing Elmore delay has been developed [40].

Our routing oracle will consist of two parts: We first show how to define edge costs for a path search algorithm such that a shortest path with respect to these costs approximately minimizes the sum of Elmore delay and congestion costs. This allows us to find approximately optimum solutions for two-pin connections, and afterwards we will deal with the construction of Steiner trees for multi-sink nets.

B. The RC routing oracle for two-terminal nets

We start with the description of our path search costs. To simplify notation, we will assume $\text{price}_v = 1$ for the sink v; this can be achieved by appropriate scaling. We will need the following definitions: For $x, y \in V(G)$ we let $\text{dist}(x,y)$ denote the rectilinear distance between x and y, and $\text{res}_{\text{lb}}(x,y)$ be a lower bound for the resistance of a wire connecting x and y. Moreover, we denote by cap_{min} and cap_{max} the minimum and maximum wire capacitance per unit length over all layers. Given a two-pin net N with source s and sink v, we can define costs rc-cost(e) for $e = (x,y) \in E(G)$ as follows:

$$\begin{aligned}
\text{cost}_{\text{cong}}(e) &:= \text{price}_e, \\
\text{cost}_{\text{src}}(e) &:= \text{res}(s) \cdot \text{cap}(e), \\
\text{cost}_{\text{wire}}(e) &:= \text{res}(e) \left(\frac{\text{cap}(e)}{2} + \text{cap}_{\text{min}} \cdot \text{dist}(y,v) + \text{cap}(v) \right), \\
\text{cost}_{\text{corr}}(e) &:= \text{res}_{\text{lb}}(s,x) \big(\text{cap}(e) - \text{cap}_{\text{min}} \cdot \text{dist}(x,y) \big), \\
\text{rc-cost}(e) &:= \text{cost}_{\text{cong}}(e) + \text{cost}_{\text{src}}(e) + \text{cost}_{\text{wire}}(e) \\
&\quad + \text{cost}_{\text{corr}}(e).
\end{aligned}$$

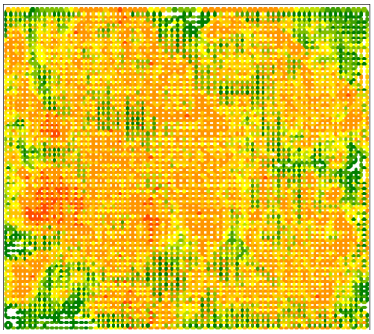
To give an intuition for these costs: $\text{cost}_{\text{cong}}(e)$ denotes the congestion cost of e and $\text{cost}_{\text{src}}(e)$ the driver delay induced by the capacitance of e. $\text{cost}_{\text{wire}}(e)$ is a lower bound estimate for the wire delay along e. $\text{cost}_{\text{corr}}(e)$ is a correction term: Since we always use the minimum wire capacitance per unit length for calculating $\text{cost}_{\text{wire}}$, we make a correction here if we use an edge on a layer with a higher unit length capacitance. We note that the theoretical bounds that we are going to present also hold if res_{lb} is set to zero. In this respect, $\text{cost}_{\text{corr}}$ could be omitted, but we still use it because it is beneficial in practice.

Since $\text{cost}_{\text{wire}}$ and $\text{cost}_{\text{corr}}$ contain estimates for the downstream capacitance and the upstream resistance of a wire, one can see that it is expensive to put high-resistance wires far away from the sink pin and high-capacitance wires far away from the source-pin. This motivates the router to use the upper layers when close to the source pin and taper down to the lower layers as it gets closer to the sink pin, since on modern metal stacks the resistance drops significantly when going to the upper layers while the capacitance stays roughly the same. As congestion is considered natively during the path search, this allows for a very efficient usage of routing resources.

In order to analyze the performance guarantee of our new path search costs, we extend the definition of rc-cost and price by setting $\text{rc-cost}(Y) := \sum_{e \in E(Y)} \text{rc-cost}(e)$ and $\text{price}(Y) := \sum_{e \in E(Y)} \text{price}_e + \sum_{v \in N \setminus \{s\}} \text{price}_v \cdot d_Y(v)$ for a Steiner tree Y for a net N with source s. Moreover, given a Steiner tree Y, we let $dist_Y(x,y)$ for $x, y \in V(Y)$ denote the length of the x-y-path in Y. We can now formulate the following theorem, using $\frac{0}{0} := 1$ here and for the rest of this section:

Theorem 13 ([39]): Consider a two-pin instance of the *Congestion-Aware Minimum Elmore Delay Steiner Tree Problem* with source s and sink v, and let Y be a shortest s-v-path with respect to rc-cost. Then we have $\text{price}(Y) \leq \alpha \cdot \frac{\text{cap}_{\text{max}}}{\text{cap}_{\text{min}}} \cdot OPT$, where $\alpha = \max_{y \in V(Y)} \frac{\text{dist}_Y(y,v)}{\text{dist}(y,v)}$ and OPT denotes the minimum achievable price.

For a proof the reader is referred to [39]. This bound turns out to be quite strong in practice: The ratio $\frac{\text{cap}_{\text{max}}}{\text{cap}_{\text{min}}}$ is technology-dependent, and on our current 14nm microprocessor test cases it is very close to 1 for most wire types. More precisely, this ratio is 1.06 for a minimum width wire (which will be used to route the majority of nets) if the lowest eight wiring layers are taken into consideration. α on the other hand depends on the router, but as our experiments show, it is also quite close



Detour %	% Paths
0	98.3
0-10	0.02
10-20	0.15
20-30	0.25
30-50	0.38
50-100	0.44
> 100	0.48

Fig. 4. Congestion map and detour histogram of unit US. The congestion target was set to 95%, which corresponds to dark orange edges.

to 1 for most paths found by our router: We extracted the final routing for every chip in our routing testbed and compared the routed length of every path to the ℓ_1-distance of its endpoints. Most detours were made on US (cf. Section VII), but even there 98.3% of all paths were shortest paths with respect to ℓ_1 distance, and the average detour made was only 1.4%. This data is shown in Figure 4.

C. Multi-terminal nets

For multi-sink nets we will use an extension of the algorithm presented in [40] in our implementation, as the original work does not incorporate congestion costs and different layer characteristics, i.e. it assumes a uniform wire resistance and capacitance per unit length and does not include via resistances. In addition to the usual instance specification it also gets a parameter $\epsilon > 0$ as input that trades off the potentially conflicting objectives of minimizing driver and wire delay: While minimizing driver delay requires the tree to be as short as possible, wire delay is minimized by a star-like topology. We also assume that we are given a short Steiner tree Y_0 as input, which can be constructed by a minimum Steiner tree algorithm. Letting cap_{wire} denote the uniform wire capacitance per unit length, the algorithm constructs a tree Y as follows:

Algorithm 2 RC tree topology algorithm

1: $Y \leftarrow\!\shortmid Y_0 \leftarrow\!\shortmid$ initial short Steiner tree rooted at s.
2: **for** all edges (x, y) of Y_0 in reverse topological order **do**
3: $\quad \text{bound}(y) \leftarrow\!\shortmid \frac{\epsilon}{2} \min\left\{\text{dist}(s, z) : z \in V(Y(y)) \cup \{x\}\right\}$
4: $\quad$ **if** $\frac{C_Y(y)}{\text{cap}_{\text{wire}}} + \text{dist}(x, y) \geq \text{bound}(y)$ **then**
5: $\quad\quad$ delete (x, y) and reconnect $Y(y)$ to s by a shortest path.
6: **return** Y

The algorithm then outputs a tree such that the total tree capacitance increases by at most a factor of $(1 + \frac{2}{\epsilon})$ compared to the initial tree and the wire delay is at most a factor of $(1 + \epsilon)^2$ larger than a lower bound for the wire delay in any tree. By choosing the right value of ϵ the algorithm achieves an approximation ratio of 3.39 assuming that the input tree is a shortest Steiner tree (see e.g. [3], [34], [8] for the construction of nearly shortest Steiner trees) and 4.31 assuming the 1.5 approximation ratio of a minimum spanning tree [20]. For more details on the algorithm, the reader is referred to [40].

Our implementation incorporates congestion costs and layer characteristics in the following way: Instead of computing an initial tree that only minimizes wiring length, our initial tree minimizes the weighted sum of congestion costs and driver delay. More precisely, we first run a minimum Steiner tree algorithm where the cost of an edge $e \in E(G)$ is given by $\text{cost}_{\text{cong}}(e)+\text{cost}_{\text{src}}(e)$. We then run Algorithm 2 for every value of $\epsilon \in [0.25, 25]$ that is a multiple of 0.25 without running a path search, but rather estimate the cost of the resulting solution by assuming that the newly found paths are shortest paths on the lowest allowed wiring layers for the given net. If we find a solution with lower estimated costs than the initial short Steiner tree Y_0, we run Dijkstra's algorithm using the aforementioned rc-cost in order to reconnect the components computed by the algorithm to the source pin.

Here we must point out a small subtlety: The algorithm requires all paths connecting the components to the source pin to be disjoint, but this is in general neither possible nor desired. We therefore run every path search independently and obtain a union of paths, which may contain loops and parallel segments. These are removed by running a shortest-path tree algorithm starting at the source at the very end. It might also be beneficial to use thicker wires for the parts of the tree driving multiple components, but this has not been implemented yet.

D. Lower and upper bounds

We will now shortly describe how we define d_{ub} when using the Elmore delay model: Let $(u, v) \in E(D)$ and s be the source pin of $\text{N}(v)$. We compute an approximately shortest Steiner tree Y for $\text{N}(v)$, and we will assume that all wires are located on the lowest allowed wiring layers for $\text{N}(v)$ using the default wire type of the net. We compute wire resistances and capacitances for Y as described in Section V-A and multiply them by some parameter $\zeta \geq 1$ in order to obtain the ζ-scaled Elmore delay $d_Y^\zeta(s, v)$ from s to v in Y. We set $d_{\text{ub}}(u, v) := d_Y^\zeta(s, v) + \eta$, where η is a second parameter. In our experiments, we set ζ to 1.5 and η to 0.25 picoseconds.

The reasoning here is as follows. Multiplying all wire RC values by ζ is roughly the same as multiplying all edge lengths by ζ, so $d_Y^\zeta(s, v)$ corresponds to the Elmore delay from s to v in Y where we plan for a detour of factor ζ on every edge in Y (and a few additional vias). η is a constant that is added to give some margin for very short nets. We expect our router to make bigger detours only in very few cases, so d_{ub} should indeed be a reasonable upper bound in practice. Our experiments clearly support this claim (see Figure 4).

For defining a lower bound $d_{\text{lb}}(u, v)$ for $(u, v) \in E(D)$, we let again s be the source pin of $\text{N}(v)$ and Y be an approximately shortest Steiner tree for $\text{N}(v)$. Moreover, let res_{min} and cap_{min} denote the minimum wire resistance and capacitance per unit length over all allowed routing layers for $\text{N}(v)$. We define

$$d_{\text{lb}}(u, v) := \text{res}(s)\left(\text{cap}_{\text{min}} \sum_{e=(x,y)\in E(Y)} \text{dist}(x, y) + \sum_{v' \in N(v)\setminus\{s\}} \text{cap}(v')\right) + \text{dist}(s, v) \cdot \text{res}_{\text{min}} \cdot \left(\tfrac{1}{2}\text{cap}_{\text{min}} \cdot \text{dist}(s, v) + \text{cap}(v)\right).$$

This is a lower bound if Y is a shortest Steiner tree for $N(v)$.

Algorithm 3 Timing-constrained global routing algorithm

1: **for** $p = 1, \ldots, p_{\max}$ **do**
2: **for** each net N **do**
3: $X \leftarrow 0$
4: **while** $X < 1$ **do**
5: Call routing oracle for N to obtain a solution Y
6: $\xi \leftarrow \min\left\{1 - X, \frac{1}{\max_{r \in \mathcal{R}} \text{usg}_{N,r}(Y)}\right\}$
7: $X \leftarrow X + \xi$
8: $\text{price}_r \leftarrow \text{price}_r \cdot e^{\gamma \cdot \xi \cdot \text{usg}_{N,r}(Y)}$ for $r \in \mathcal{R}$
9: **for** $i = 1, \ldots, n$ **do**
10: **for** each arrival time customer v **do**
11: $a(v) \leftarrow \text{COMPUTEAT}(v)$
12: $\text{price}_r \leftarrow \text{price}_r \cdot e^{\gamma \cdot n^{-1} \cdot \text{usg}_{v,r}(a(v))}$ for $r \in \mathcal{R}$
13: Iterated randomized rounding
14: Rip-up and re-route

VI. Overall Algorithm

The overall global routing algorithm is summarized in Algorithm 3. The resource sharing part consists of $p_{\max}$ iterations. At the beginning of each iteration, all nets are rerouted with the routing oracle (lines 2–8). Once a net N is routed, the costs of consumed resources are updated according to (2). Multiple iterations of the while loop are needed only if $\text{usg}_{N,r}(Y) > 1$ for some $r \in \mathcal{R}$. This cannot happen often, as shown in [29], hence the running time of Theorem 1.

Then, in lines 9–12, the arrival time oracle is called n times for all $v \in V(D)$. The proof in [29] allows to interleave the calls for different $v \in V(D)$, which we exploit to improve convergence given the interdependency of arrival time costs at neighboring vertices in the timing graph. When implementing the subroutine COMPUTEAT() in line 11 by line 2 of Algorithm 1, we obtain a good fractional solution and, thus, delay budgeting. Again after each call, the costs are updated.

Theorem 14: Let $\omega > 0$. Given a routing oracle with approximation ratio σ and $p_{\max} = \mathcal{O}(\omega^{-2} \log |\mathcal{R}|)$, lines 1–12 of Algorithm 3 compute a (fractional) solution that minimizes the maximum resource usage up to a factor $\sigma(1+\omega)$, assuming that this minimum is between $\frac{1}{2}$ and 2.

For nets with a bounded number of pins, we can obtain $\sigma = 1$ in the linear delay model. Then, if there is a global routing that satisfies all routing and timing constraints, the algorithm computes a fractional solution such that no edge is overloaded by more than a factor $1 + \omega$ and the worst slack is at least $-\text{relax} - \omega H$, where H is the constant from Lemma 6.

PROOF. This follows from Theorem 1, Lemma 5, Lemma 6, Lemma 7, and Corollary 12. ∎

The assumption that the minimum $\lambda^\star$ in (1) is within $[\frac{1}{2}, 2]$ is usually satisfied in practice, but one could also do without this assumption, using binary search [29]. We remark that the routing oracle as well as the arrival time oracle can be called for many customers in parallel.

In practice, we implement the subroutine COMPUTEAT() in line 11 by Newton's method as described in the proof of Theorem 9, using 3 iterations in each call, and reduce n in the outer loop in line 9 from 100 to 15. This gives comparable overall results in significantly less runtime.

When the resource sharing part finishes, we generate eight routing solutions. For each solution we pick for each net independently one of the computed Steiner trees randomly. We then select the solution minimizing the maximum resource usage. Finally, we try to eliminate remaining congestion or delay violations by traditional rip-up and re-route.

VII. Experimental Results

We integrated our new global routing algorithm with static timing constraints (Algorithm 3) with oracles for the linear delay model (Section IV) and the RC delay model (Section V) into BonnRoute, an existing timing-unaware resource sharing implementation for 3D global routing [12], that is the "golden" global routing algorithm in the IBM microprocessor design flow. We chose $p_{\max} = 25$ and $\gamma = 5$ in Algorithm 3.

Experiments were carried out on 7 microprocessor units in 22 and 14 nm technology. The experiments were conducted under Linux on an Intel Xeon E5-2667v2 server running at 3.30 GHz. All listed running times refer to parallel program executions using the available 16 cores. The running times reflect the wall time for running either the reference run or the global router without loading the data into the design environment.

All reported slack numbers were computed with IBM EinsTimer, which is also the timing engine for final sign-off. Depending on the experiments we either used it with a linear delay model on unbuffered netlists or with the RICE delay model [35] on highly optimized buffered netlists.

An overview of the results can be found in Tables I and II. They are arranged as follows. Besides unit names the first column shows the number of nets followed by the cycle time. We report *worst slack* (WS), *total negative endpoint slack* (TNS), *total routing overload* (OL), *wiring length* (WL), the *number of vias* (Vias), and *running times* (Wall time) for multiple experiments. The experiments are described in Sections VII-A and VII-B.

We compare our new algorithm with bounds, with an industrial flow, and with preliminary algorithms from [14] (ICCAD 2015) and [39] (ICCAD 2016). In [14] and [39] we had larger arrival time intervals (equivalent to $d_{\text{ub}} \equiv \infty$) and the simple arrival time oracle from Lemma 7.

For the rows "Bounds", we computed WS and TNS as described in Sections IV-D and V-D (congestion-unaware). Note that these are not always lower delay bounds in the RC delay case. The OL is computed with BonnRoute global router [12] as it is run in the design flow minimizing wire length and vias. For WL and Vias, we computed short 3D Steiner trees (timing- and congestion-unaware).

A. Results for linear delays

For linear delays, the input netlists are unbuffered and the result of the global placement optimization phase in the IBM flow. All delays in these experiments were computed using the linear delay model from Section IV-A. The results are shown in Table I.

The lines "CATALYST" refer to the layer assignment algorithm [44], which is part of the IBM microprocessor design flow and computes a congestion-aware layer assignment. This

Unit (#nets, cycle time)	Experiment	WS [ps]	TNS [ns]	OL	WL [m]	Vias [K]	Wall time [h:mm:ss]
US	"Bounds"	-57	-5	0	1.99	866	-
(141 238,	CATALYST [44]	-72	-39	0	2.06	1 429	0:09:36
240 ps)	ICCAD 2015 [14]	-58	-11	0	2.09	1 516	0:05:53
	Fractional	-58	-9	0	2.10	1 572	0:01:17
	Algorithm 3	-58	-8	0	2.08	1 487	0:04:56
UF	"Bounds"	-44	-6	0	5.04	942	-
(156 800,	CATALYST [44]	-91	-35	2	5.11	1 923	0:13:59
264 ps)	ICCAD 2015 [14]	-44	-11	0	5.25	2 304	0:27:39
	Fractional	-44	-6	0	5.36	2 304	0:03:12
	Algorithm 3	-44	-6	0	5.17	2 007	0:08:16
UP	"Bounds"	-5	-0.0	0	9.12	1 311	-
(305 689,	CATALYST [44]	-33	0	4	9.28	1 977	0:09:01
790 ps)	ICCAD 2015 [14]	-33	0	0	9.28	2 500	0:10:24
	Fractional	-39	-0.2	0	9.27	2 227	0:01:39
	Algorithm 3	-33	-0.1	0	9.27	2 192	0:08:59
UL	"Bounds"	-237	-32	0	8.54	2 152	-
(361 684,	CATALYST [44]	-279	-82	7	8.59	3 282	0:18:50
184 ps)	ICCAD 2015 [14]	-237	-41	164	8.86	3 580	0:18:45
	Fractional	-237	-39	173	8.84	3 443	0:03:46
	Algorithm 3	-237	-37	158	8.86	3 443	0:13:05
UV	"Bounds"	-38	-6	0	13.04	1 818	-
(415 592,	CATALYST [44]	-68	-27	0	13.24	3 373	0:10:55
208 ps)	ICCAD 2015 [14]	-43	-17	0	13.33	5 065	0:23:47
	Fractional	-43	-17	0	13.36	5 027	0:02:59
	Algorithm 3	-43	-11	0	13.28	3 870	0:12:48
UN	"Bounds"	-156	-38	554	10.66	2 170	-
(478 217,	CATALYST [44]	-174	-394	1 665	11.10	3 433	0:13:36
208 ps)	ICCAD 2015 [14]	-178	-151	1 350	11.25	4 196	0:19:37
	Fractional	-179	-147	1 577	11.27	3 838	0:04:16
	Algorithm 3	-177	-118	1 386	11.25	4 027	0:17:50
UI	"Bounds"	-80	-1 380	0	36.37	7 579	-
(1 257 242,	CATALYST [44]	-129	-1 829	6	36.77	12 266	1:05:06
184 ps)	ICCAD 2015 [14]	-81	-1 479	0	37.12	15 429	2:04:57
	Fractional	-81	-1 442	0	36.97	12 217	0:15:40
	Algorithm 3	-80	-1 442	0	37.07	13 406	0:53:04

TABLE I
RESULTS ON INDUSTRIAL MICROPROCESSOR UNITS (UNBUFFERED NETLISTS) USING THE LINEAR DELAY ROUTING ORACLE.

Unit (#nets, cycle time)	Experiment	WS [ps]	TNS [ns]	OL	WL [m]	Vias [k]	Wall time [h:mm:ss]
US	"Bounds"	-130	-383	0	2.05	1134	-
(174 436,	BonnRoute [12]	-165	-476	0	2.14	1514	0:03:04
240 ps)	ICCAD 2016 [39]	-138	-404	0	2.16	1581	0:07:01
	Algorithm 3	-132	-403	0	2.16	1579	0:06:40
UF	"Bounds"	-118	-152	0	4.82	1786	-
(254 208,	BonnRoute [12]	-125	-291	0	4.91	2303	0:03:01
264 ps)	ICCAD 2016 [39]	-118	-196	0	4.92	2527	0:13:22
	Algorithm 3	-118	-192	0	4.93	2519	0:12:14
UP	"Bounds"	-54	-3	0	9.29	1812	-
(376 664,	BonnRoute [12]	-83	-8	0	9.49	2309	0:02:32
790 ps)	ICCAD 2016 [39]	-53	-3	0	9.51	2864	0:15:00
	Algorithm 3	-53	-3	0	9.49	2517	0:09:21
UL	"Bounds"	-243	-60	52	8.88	2817	-
(482 340,	BonnRoute [12]	-243	-85	52	8.96	3785	0:03:20
184 ps)	ICCAD 2016 [39]	-243	-64	52	8.97	4255	0:24:37
	Algorithm 3	-243	-63	52	8.96	4093	0:17:12
UV	"Bounds"	-66	-158	0	13.41	2914	-
(584 336,	BonnRoute [12]	-97	-516	0	13.69	3616	0:04:05
208 ps)	ICCAD 2016 [39]	-71	-235	0	13.64	4336	0:21:26
	Algorithm 3	-69	-232	0	13.63	4295	0:19:16
UN	"Bounds"	-308	-622	695	11.24	3601	-
(632 226,	BonnRoute [12]	-531	-1312	695	11.66	4233	0:04:33
208 ps)	ICCAD 2016 [39]	-312	-865	732	11.69	4512	0:24:20
	Algorithm 3	-312	-772	750	11.70	4525	0:22:48
UI	"Bounds"	-79	-215	0	36.63	10681	-
(1 681 671,	BonnRoute [12]	-169	-1347	0	36.84	14525	0:15:24
184 ps)	ICCAD 2016 [39]	-91	-552	0	36.94	16506	2:19:53
	Algorithm 3	-96	-520	0	37.00	16305	2:04:54
	Algorithm 3 fast	-97	-630	4	36.89	15174	1:02:46

TABLE II
RESULTS ON INDUSTRIAL MICROPROCESSOR UNITS (BUFFERED NETLISTS) USING THE RC TREE ROUTING ORACLE.

is passed as a constraint to the timing-unaware global router. In addition to layer assignments, the reference run pre-assigns netlength bounds to critical nets and bounds on the source-to-sink distances of high fanout nets. Finally, it refines the layer-assignment for improving worst slacks using incremental global routing.

The rows "Fractional" and "Algorithm 3" show our algorithm, where "Fractional" refers to the fractional solution after the resource sharing phase (lines 1–12), and "Algorithm 3" to the final results after rip-up and re-route. In Algorithm 3, we ignore all layer assignments and length bounds.

On most instances our new algorithm could improve the worst slack and the total negative slacks compared to the CATALYST approach or almost reach the slack bounds. On US, UF, UL, UV, and UI the slack improvements are significant. Note that the larger WS deviation of 28 ps from the bound on UP can be explained by the larger cycle time and path delays compared to the other units. The relative deviation is less than 5% and comparable. On UN, Algorithm 3 achieves a better TNS and OL than CATALYST [44], but at the cost of a slightly smaller WS. All other reported routing overloads are small and exhibit inaccuracies of the global routing model at macro borders. Typically, they do not impact detailed routability. Algorithm 3 embeds timing-aware routing topologies, leading to slightly larger wire length. Its flexibility to use higher routing layers only partially results in more vias.

Due to faster convergence of the arrival times our new algorithm yields slightly better solutions in significantly shorter time compared to [14]. The comparison between fractional and final solutions shows that we can recover or even improve the quality of the fractional solution after rounding and rip-up and re-route.

The results suggest that on unbuffered instances layer assignment approaches have a limited power to fulfill timing constraints. First, assignments of large nets to high layers are often inhibited as they create congestion by also forcing connections to uncritical sinks to the limited routing space on the assigned layers. Second, the timing-unaware global routing may choose unfavorable topologies. The predefined bounds on netlengths or source-to-sink-distances in the reference run cannot be chosen sufficiently tight if routability should be preserved. In contrast, the delay prices and the implicit delay bounds computed throughout our algorithm ensure that the eventually critical connections are short and use fast layers.

B. Results for RC delays

For RC delays, the input data is taken from the IBM design flow after placement and global timing optimization, including buffering, gate sizing, and layer assignment, just before the routing phase. We could ignore the layer assignment, but we currently obtain even better results when using this information. The industrial flow generates netlength bounds on individual timing critical nets as additional constraints. We used these bounds in the reference run BonnRoute [12] in the timing-unaware standard mode, but neglected them in Algorithm 3.

The results are shown in Table II. Although the timing-constrained run is internally using the Elmore delay model, the timing numbers shown in Table II are using RICE extractions [35]. Compared to the reference run, the timing metrics are

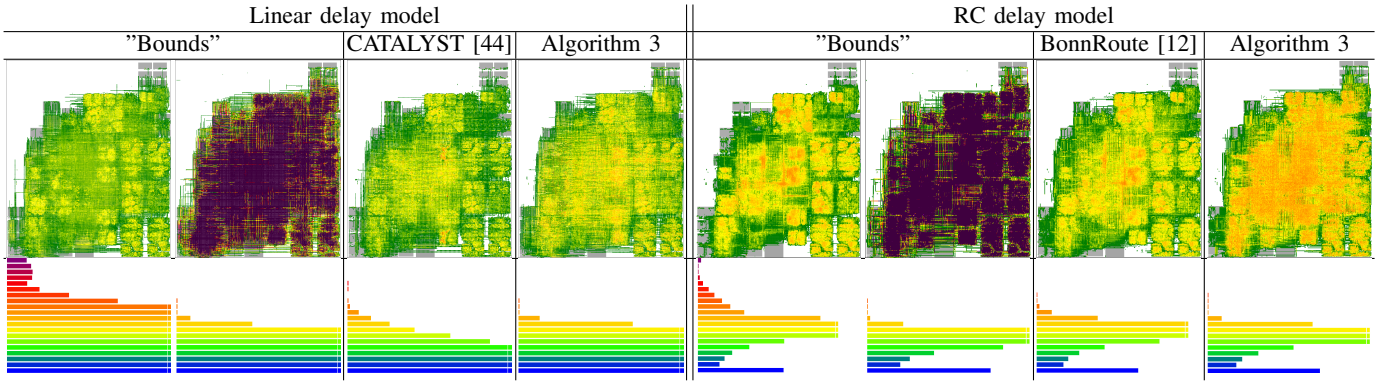


Fig. 5. Congestion plots and slack histograms for several algorithms on instance UI with linear and RC delay model.

clearly improved on every design while the overload hardly increases.

The only drawback of the new timing-constrained global routing algorithm is the increase in running time. Compared to [39], we could already reduce it. Further speed-up is possible by refining the implementation or by using more CPU cores. Alternatively, the effort for ripout & reroute (which we have not parallelized yet) can be reduced as shown in the line "Algorithm 3 fast" for the unit UI. After all, the running time of our new global routing algorithm is still small compared to the time spent in the subsequent steps such as detailed routing and final clean-up.

C. Visualized results

A visual comparison of Algorithm 3 with the timing-unaware and congestion-unaware global routing runs as well as the reference runs can be found in Figure 5 for the largest unit UI. In Figure 6, we show a typical progression of the fractional solution after several iterations of the resource sharing algorithm on UI. In both figures, the top rows show congestion plots and the bottom rows show slack histograms. Colors range from blue (uncritical) to violet (critical). The congestion plots show the maximum congestion over all routing layers in the three-dimensional global routing graph and the color red marks the transition from routable to unroutable. The slack histograms show the slack distribution of all gates, where each gate is represented by its worst slack. Here, yellow represents slack zero.

Figure 5 shows that, in constrast to the reference runs, Algorithm 3 can almost achieve the (estimated) best possible slack distribution. The congestion plots show that this is achieved by a more effective use of global routing resources.

In Figure 6, we see some routing congestion and timing violations after the first iteration. During the course of the algorithm both routing and timing violations are improving simultaneously and the worst violations are well balanced among all resources.

VIII. Conclusion

Overall, our integration of static timing constraints into the resource sharing algorithm for global routing improves the quality for both the linear delay and the RC delay model compared to approaches from an existing industrial design environment. Theorem 14 establishes a sound theoretical basis for our model. The running time overhead is still acceptable and could be reduced further. Summarizing, this work demonstrates how to efficiently integrate static timing constraints into global routing for a better quality of results.

References

[1] Albrecht, C., Kahng, A.B., Mandoiu, I., Zelikovsky, A.: Floorplan evaluation with timing-driven global wireplanning, pin assignment and buffer/wire sizing. ASP-DAC, 2002, 580–587.

[2] Ao, J., Dong, S., Chen, S., Goto, S.: Delay-driven layer assignment in global routing under multi-tier interconnect structure. ISPD, 2013, 101–107.

[3] Arora, S.: Polynomial time approximation schemes for Euclidean, traveling salesman and other geometric problems Journal of the ACM 45, 1998, 753–782.

[4] Bartoschek, C., Held, S., Rautenbach, D., Vygen, J.: Efficient generation of short and fast repeater tree topologies. ISPD, 2006, 120–127.

[5] Boese, K.D., Kahng, A.B., Robins, G.: High-performance routing trees with identified critical sinks. DAC, 1993, 182–187.

[6] Boese, K.D., Kahng, A.B., McCoy, B.A., Robins, G.: Rectilinear Steiner trees with minimum Elmore delay. DAC, 1994, 381–386.

[7] Boese, K.D., Kahng, A.B., McCoy, B.A., Robins, G.: Near-optimal critical sink routing tree constructions. IEEE TCAD 14, 1995, 1417–1436.

[8] Brazil, M., Zachariasen, M.: Optimal Interconnection Trees in the Plane. Springer, Berlin, 2015.

[9] Chuzhoy, J., Gupta, A., Naor, J. S., Sinha, A.: On the approximability of some network design problems. ACM Transactions on Algorithms 4, 2008, Article 23.

[10] Elmore, W.: The transient response of damped linear networks with particular regard to wideband amplifiers. Journal of Applied Physics 19, 1948, 55–63.

[11] Garey, M. R., Johnson, D. S.: The rectilinear Steiner tree problem is NP-complete. SIAM Journal on Applied Mathematics 32, 1977, 826–834.

[12] Gester, M., Müller, D., Nieberg, T., Panten, C., Schulte, C., Vygen, J.: BonnRoute: Algorithms and data structures for fast and good VLSI routing. ACM TODAES 18, 2013, Article 32.

[13] Held, S., Rotter, D.: Shallow-light Steiner arborescences with vertex delays. IPCO, 2013, 229–241.

[14] Held, S., Müller, D., Rotter, D., Traub, V., Vygen, J.: Global routing with inherent static timing constraints. ICCAD, 2015, 102–109.

[15] Hong, X., Xue, T., Huang, J., Cheng, C.-K., Kuh, E.S.: TIGER: an efficient timing-driven global router for gate array and standard cell layout design. IEEE TCAD 16, 1997, 1323–1331.

[16] Hougardy, S., Silvanus, J., Vygen, J.: Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm. arXiv:1406.0492. Mathematical Programming Computation 9, 2017, to appear.

[17] Huang, J., Hong, X.-L., Cheng, C.-K., Kuh, E.S.: An efficient timing-driven global routing algorithm. DAC, 1993, 596–600.

[18] Hu, J., Sapatnekar, S.S.: A timing-constrained simultaneous global routing algorithm. IEEE TCAD 21, 2002, 1025–1036.

[19] Hu, S., Li, Z., Alpert, C.J.: A fully polynomial-time approximation scheme for timing-constrained minimum cost layer assignment. IEEE Transactions on Circuits and Systems II: Express Briefs 56, 2009, 580–584.

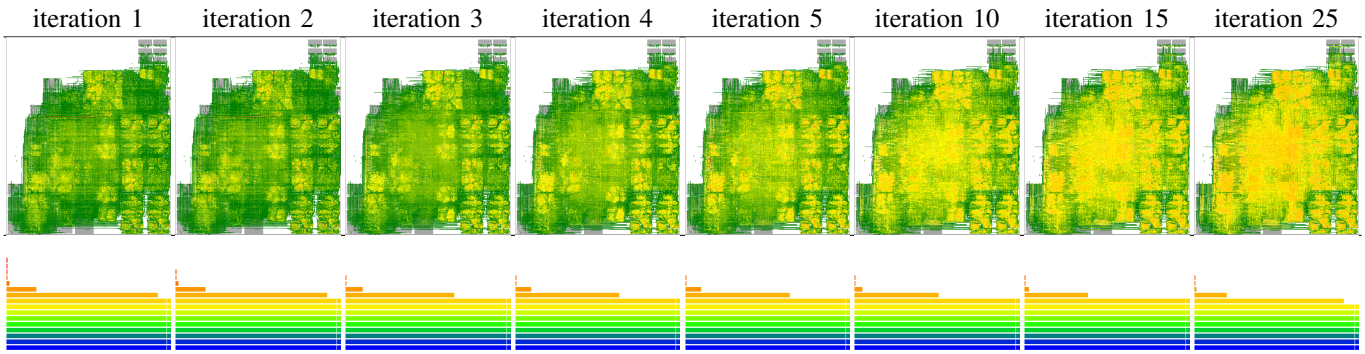


Fig. 6. Congestion plots and slack histograms after several stages of the resource sharing phase of Algorithm 3 with the linear delay model.

[20] Hwang, F.: On Steiner minimal trees with rectilinear distance. SIAM Journal on Applied Mathematics 30, 1976, 104–114.

[21] Kadodi, T.: Steiner routing based on Elmore delay model for minimizing maximum propagation delay. Master's Thesis, Japan Advanced Institute of Science and Technology, 1999.

[22] Kahng, A.B., Robins, G.: On Optimal Interconnections for VLSI. Kluwer Academic Publishers, Boston, 1995.

[23] Karp, R.: Reducibility among combinatorial problems. Complexity of Computer Computations, R. Miller and J. Thatcher, eds., Plenum Press, New York, 1972, 85–103.

[24] Khuller, S., Raghavachari, B., Young, N.: Balancing minimum spanning trees and shortest-path trees. Algorithmica 14, 1995, 305–321.

[25] Lee, T.-H., Chang, Y.-J., Wang, T.-C.: An enhanced global router with consideration of general layer directives. ISPD, 2011, 53–60.

[26] Maßberg, J., Vygen, J.: Approximation algorithms for a facility location problem with service capacities. ACM Transactions of Algorithms 4, 2008, Article 50.

[27] Meyerson, A., Munagala, K., Plotkin, S.: Cost-distance: two metric network design. FOCS, 2000, 624–630.

[28] Müller, D.: Optimizing yield in global routing. ICCAD, 2006, 480–486.

[29] Müller, D., Radke, K., Vygen, J.: Faster min-max resource sharing in theory and practice. Mathematical Programming Computation 3, 2011, 1–35.

[30] Moffitt, M.D., Sze, C.N.: Wire synthesizable global routing for timing closure. ASP-DAC, 2011, 545–550.

[31] Peyer, S.: Elmore-Delay-optimale Steinerbäume im VLSI-Design. Diploma's Thesis (in german), Research Institute for Discrete Mathematics, University of Bonn, 2000.

[32] Peyer, S., Zachariasen, M., Jørgensen, D.G.: Delay-related secondary objectives for rectilinear steiner minimum trees. Discrete Applied Mathematics 136, 2004, 271–298.

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

[34] Rao, S.B., Smith, W.D.: Approximating geometrical graphs via "spanners" and "banyans". STOC, 1998, 540–550.

[35] Ratzlaff, C.L., Pillage, L.T.: RICE: rapid interconnect circuit evaluation using AWE. IEEE TCAD 13, 1994, 763–776.

[36] Rubinstein, J., Penfield, P., Horowitz, M.A.: Signal delay in RC tree networks. IEEE TCAD 2, 1983, 202–211.

[37] Samanta, R., Erzin, A.I., Raha, S., Shamardin, Y.V., Takhonov, I.I., Zalyubovskiy, V.V.: A provably tight delay-driven concurrently congestion mitigating global routing algorithm. Applied Mathematics and Computation 255, 2015, 92–104.

[38] Saxena, P., Liu, C.L.: Optimization of the maximum delay of global interconnects during layer assignment. IEEE TCAD 20, 2001, 503–515.

[39] Scheifele, R.: RC-aware global routing. ICCAD, 2016, Article 21.

[40] Scheifele, R.: Steiner trees with bounded RC-delay. Algorithmica, pre-published online, 2016, 1–24.

[41] Shragowitz, E., Keel, S.: A global router based on a multicommodity flow model. Integration, the VLSI Journal 5, 1987, 3–16.

[42] Vittal, A., Marek-Sadowska, M.: Minimal delay interconnect design using alphabetic trees. DAC, 1994, 392–396.

[43] Vygen, J.: Near-optimum global routing with coupling, delay bounds, and power consumption. IPCO, 2004, 308–324.

[44] Wei, Y., Li, Z., Sze, C.C.N., Hu, S., Alpert, C.J., Sapatnekar S.S.: CATALYST: planning layer directives for effective design closure. DATE, 2013, 1873–1878.

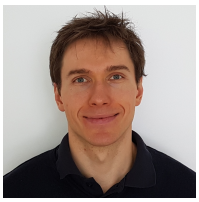
[45] Yan, J.-T., Lin, S.-H.: Timing-constrained congestion-driven global routing. ASP-DAC, 2004, 683–686.

[46] Yan, J.-T., Lee, C.-F., Chen, Y.-H.: Multilevel timing- constrained full-chip routing in hierarchical quad-grid model. ISCAS, 2006, 5439–5442.

Stephan Held received Diploma and Ph.D. degrees in mathematics from the University of Bonn. From 2009–2010 he was post-doc at the Georgia Institute of Technology. From 2010–2013 he was assistant professor and since 2013 he is an associate professor at the Research Institute for Discrete Mathematics, University of Bonn. His research interests include combinatorial optimzation and chip design.

Dirk Müller received the Diploma and Ph.D. degree in computer science in 2003 and 2009, respectively, from the University of Bonn, Germany. He is currently working as a postdoc researcher on electronic design automation, with a focus on combinatorial optimization applied to routing, at the Research Institute for Discrete Mathematics, University of Bonn.

Daniel Rotter received his Bachelor and Master of Science degrees in mathematics at the University of Bonn in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree in mathematics from the Research Institute for Discrete Mathematics, University of Bonn. His current research interest includes combinatorial optimization and VLSI design with focus on global buffering.



Rudolf Scheifele is a PhD student at the University of Bonn, Germany, where he received his Bachelor and Master of Science degrees in mathematics in 2011 and 2013, respectively. He now continues his PhD studies in mathematics, which includes working on BonnRoute. His research interest focuses mainly on global routing, in particular mathematical optimization problems that arise in this area.

Vera Traub is a graduate student at the University of Bonn, Germany, where she received her Bachelor of Science degree in mathematics in 2015. Her research interest focuses on global routing and the travelling salesman problem.

Jens Vygen (PhD Bonn 1997) is professor of discrete mathematics of the University of Bonn since 2003. He is also founding member and principal investigator of the Hausdorff Center for Mathematics. Vygen is managing two cooperations with industry, including the BonnTools development. He is author of two textbooks, many papers, and he has served as editor of seven scientific journals. His research interests include combinatorial optimization and chip design.