BonnRoute: Classic Routing Algorithms with Recent Advances

Jens Vygen

University of Bonn

BonnRoute: Classic Routing Algorithms with Recent Advances

Jens Vygen

University of Bonn

Thanks to the entire BonnRoute team, in particular:

Markus Ahrens, Daniel Blankenburg, Antonia Ellerbrock, Stephan Held, Stefan Hougardy, Niko Klewinghaus, Stefan Rabenstein, Christian Roth, Niklas Schlomberg, Armin Settels

BonnRoute: Classic Routing Algorithms with Recent Advances

Jens Vygen

University of Bonn

Thanks to the entire BonnRoute team, in particular:

Markus Ahrens, Daniel Blankenburg, Antonia Ellerbrock, Stephan Held, Stefan Hougardy, Niko Klewinghaus, Stefan Rabenstein, Christian Roth, Niklas Schlomberg, Armin Settels

---

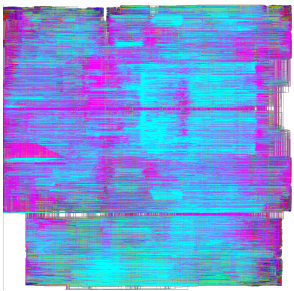
Note: There is an enormous amount of prior work in routing, but this talk contains only very few references

Routing: Challenge

- Very simple special cases are NP-hard and hard in practice (e.g., vertex-disjoint paths in planar grids)

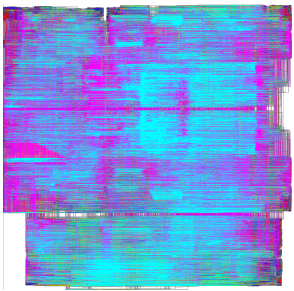
Routing: Challenge

- Very simple special cases are NP-hard and hard in practice
  (e.g., vertex-disjoint paths in planar grids)
- Instance sizes are huge
  (millions of disjoint Steiner trees in a graph with trillions of vertices)



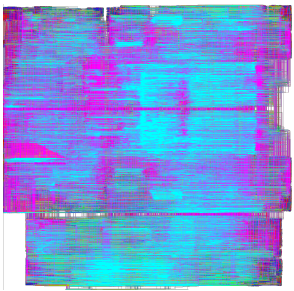
Routing: Challenge

- Very simple special cases are NP-hard and hard in practice
  (e.g., vertex-disjoint paths in planar grids)
- Instance sizes are huge
  (millions of disjoint Steiner trees in a graph with trillions of vertices)
- Design rules are very complicated
  (and post-routing correction can be difficult or impossible)



Routing: Challenge

- Very simple special cases are NP-hard and hard in practice
  (e.g., vertex-disjoint paths in planar grids)
- Instance sizes are huge
  (millions of disjoint Steiner trees in a graph with trillions of vertices)
- Design rules are very complicated
  (and post-routing correction can be difficult or impossible)
- Just finding any feasible solution is not good enough.
  Consider
  - timing constraints
  - power consumption
  - crosstalk
  - yield
  - leave space for late incremental changes



BonnRoute

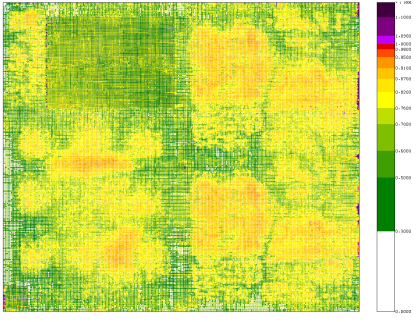
- Developed for more than thirty years
  (by the University of Bonn in cooperation with IBM)
- Used for the design of hundreds of most complex chips
  (including all IBM processors)
- Part of the BonnTools
- Driven by new mathematical theory and experience from practice

BonnRoute

- Developed for more than thirty years
  (by the University of Bonn in cooperation with IBM)
- Used for the design of hundreds of most complex chips
  (including all IBM processors)
- Part of the BonnTools
- Driven by new mathematical theory and experience from practice
- Global Routing: provably near-optimal fractional solution
- Detailed Routing: ultra-fast multi-label Dijkstra searches

Global routing in BonnRoute

- Carefully chosen resource sharing model (including timing constraints)
- Compute near-optimal fractional solution
- Apply randomized rounding to obtain a provably good integral solution
- Local postprocessing

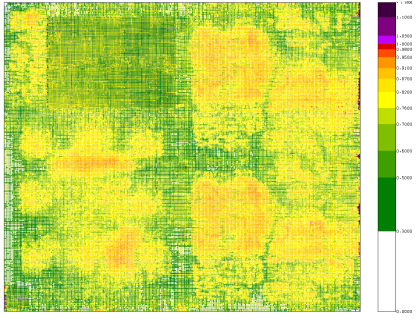


Global routing in BonnRoute

- Carefully chosen resource sharing model (including timing constraints)
- Compute near-optimal fractional solution
- Apply randomized rounding to obtain a provably good integral solution
- Local postprocessing

Additional features:

- Track assignment
- Multi-level global routing
- Incremental global routing
- Combination with buffering



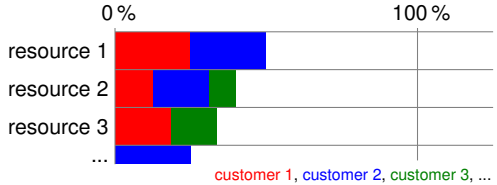
Computing a near-optimal fractional solution

Resource sharing model:

- d resources
- a set C of n customers
- a convex set $X_c \subseteq \mathbb{R}^d_{\geq 0}$ of solutions for each customer $c \in C$, given by an oracle that outputs $\arg\min_{x_c \in X_c} \langle x_c, y \rangle$ for given prices $y \in \mathbb{R}^d_{\geq 0}$

- nets are customers
- solutions are (convex combinations of) 3D Steiner trees
- resources are areas corresponding to edges of 3D global routing graph

Computing a near-optimal fractional solution



Resource sharing model:

- d resources
- a set C of n customers
- a convex set $X_c \subseteq \mathbb{R}^d_{\geq 0}$ of solutions for each customer $c \in C$, given by an oracle that outputs $\arg\min_{x_c \in X_c} \langle x_c, y \rangle$ for given prices $y \in \mathbb{R}^d_{\geq 0}$

Theorem (Müller, Radke, V. [2011], Blankenburg [2022])

There is a deterministic algorithm that computes a $(1+\varepsilon)$-approximate solution to $\min_{x_c \in X_c\,(c \in C)} \left|\left|\sum_{c \in C} x_c\right|\right|_\infty$ and makes $O(\varepsilon^{-2}(n+d)\log d)$ oracle calls.

- nets are customers
- solutions are (convex combinations of) 3D Steiner trees
- resources are areas corresponding to edges of 3D global routing graph

Resource sharing algorithm (simplified)

- Round-robin scheme: call each oracle once per phase
- Initially, unit prices for all resources
- Update prices multiplicatively when a resource is used

Resource sharing algorithm (simplified)

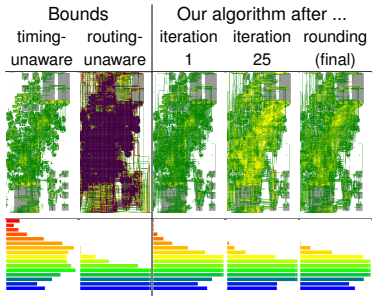
- Round-robin scheme: call each oracle once per phase
- Initially, unit prices for all resources
- Update prices multiplicatively when a resource is used

- Also works with approximate oracles. Most oracle calls avoidable
- Can be parallelized very well
- Very efficient in practice

Resource sharing algorithm (simplified)

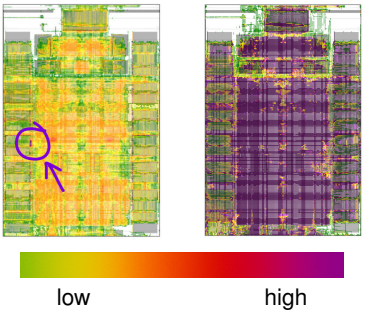
- Round-robin scheme: call each oracle once per phase
- Initially, unit prices for all resources
- Update prices multiplicatively when a resource is used

- Also works with approximate oracles. Most oracle calls avoidable
- Can be parallelized very well
- Very efficient in practice
- Quite flexible: resources for
  - area (Bihler, Blankenburg [2025])
  - power consumption
  - yield loss
  - time (Held, Müller, Rotter, Scheifele, Traub, V. [2018])



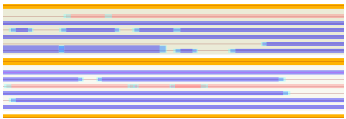
Global routing: future research directions

- Better oracle for RC-aware Steiner trees
- Better rounding
- Better correlation between global and detailed routing
- Scale between high accuracy and ultra-fast runtime
- Always take exact pin positions into account
- Minimize ordered norms (such as wACE) instead of ℓ_∞ norm (Ellerbrock, Blankenburg, Kesselheim, V. [2026])



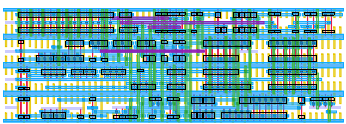
Detailed routing: three approaches

- Track assignment — fast
  (global view of a single layer,
  limited view of adjacent layers)



- Sequential path search — flexible
  (best connection for a single
  pair of components)

- Local mixed-integer programming — accurate
  (slow, but can resolve small
  very difficult sub-instances)

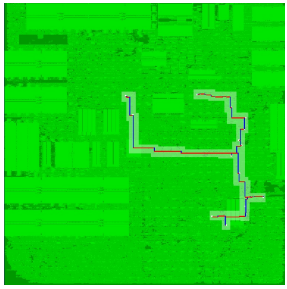


Detailed routing in BonnRoute

...is mainly based on sequential path search. Of course,

- it allows for rip-up and re-route
- it has very efficient data structures
- it is parallelized well

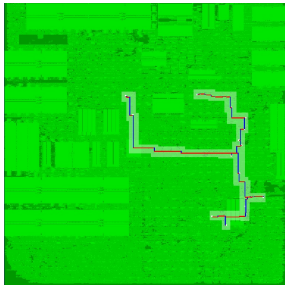
Detailed routing in BonnRoute



...is mainly based on sequential path search. Of course,

- it allows for rip-up and re-route
- it has very efficient data structures
- it is parallelized well
- it works in subgraphs suggested by global routing with certain edge weights
- it contains same-net error avoidance techniques (e.g., multi-label Dijkstra)
- it works with excellent distance estimates for goal-oriented path search (A*)

Detailed routing in BonnRoute



...is mainly based on sequential path search. Of course,

- it allows for rip-up and re-route
- it has very efficient data structures
- it is parallelized well
- it works in subgraphs suggested by global routing with certain edge weights
- it contains same-net error avoidance techniques (e.g., multi-label Dijkstra)
- it works with excellent distance estimates for goal-oriented path search (A*)

We sometimes use

- track assignment for long-distance nets (mostly on upper layers)
- mixed-integer programming to resolve the hardest subproblems

Goal-oriented path search

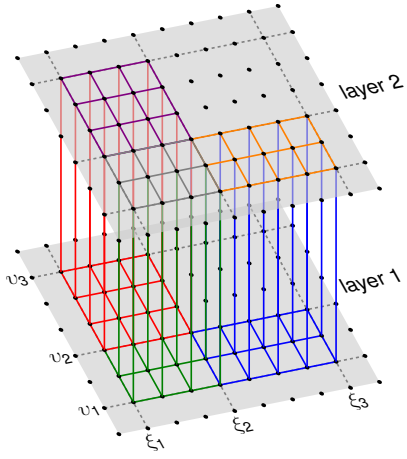
Run Dijkstra in G with reduced edge costs
$c_\pi(e) := c(e) - \pi(v) + \pi(w)$ for $e = (v, w)$,
where $\pi(v) := \mathrm{dist}_{(\overline{G},\overline{c})}(v, t) \leq \mathrm{dist}_{(G,c)}(v, t)$,

Goal-oriented path search

Run Dijkstra in G with reduced edge costs $c_\pi(e) := c(e) - \pi(v) + \pi(w)$ for $e = (v, w)$, where $\pi(v) := \mathrm{dist}_{(\overline{G},\overline{c})}(v, t) \leq \mathrm{dist}_{(G,c)}(v, t)$, and $\overline{G}$ and edge costs $\overline{c}$ depend only on:

- layer $(1, \ldots, \ell)$
- direction (horizontal/vertical/via)
- tile in a $p \times q$ grid

(e.g., higher cost for deviating from global routing or track assignment)

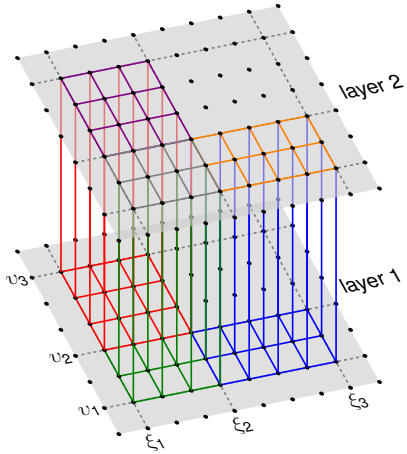


Goal-oriented path search

Run Dijkstra in G with reduced edge costs
$c_\pi(e) := c(e) - \pi(v) + \pi(w)$ for $e = (v, w)$,
where $\pi(v) := \mathrm{dist}_{(\overline{G},\overline{c})}(v, t) \leq \mathrm{dist}_{(G,c)}(v, t)$,
and $\overline{G}$ and edge costs $\overline{c}$ depend only on:

- layer $(1, \ldots, \ell)$
- direction (horizontal/vertical/via)
- tile in a $p \times q$ grid

(e.g., higher cost for deviating from global routing or track assignment)



Theorem (Ahrens, Henke, Rabenstein, V. [2022])

Let $\epsilon > 0$. We can query the distance to the nearest target in $O(\frac{1}{\epsilon} \log(p + q + \ell))$ time after $O(\frac{1}{\epsilon} pq(p + q)\ell^{2+\epsilon} \log(p + q + \ell))$ preprocessing time.

Detailed routing: future research directions

- Better interaction between track assignment and path search
- Preprocessing lower layers for pin access (in dense areas)
- Steiner tree search instead of successive path searches
- Timing-aware detailed routing
- SAT versus MIP
- Better space management