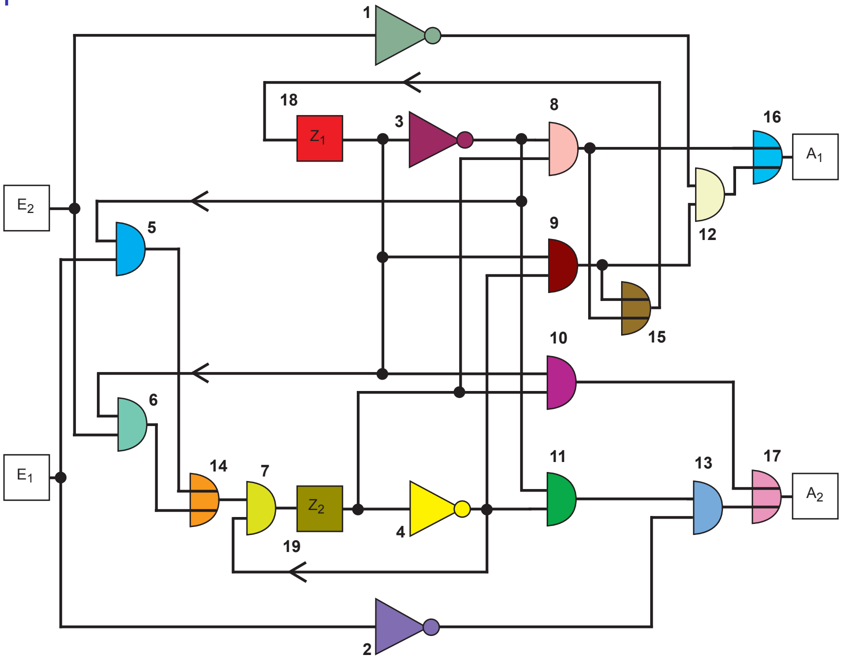
Combinatorial Optimization in VLSI Placement and Routing

Jens Vygen

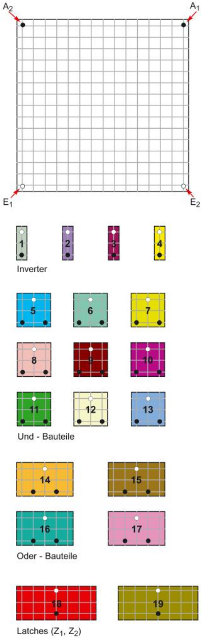
University of Bonn

Outline

Example

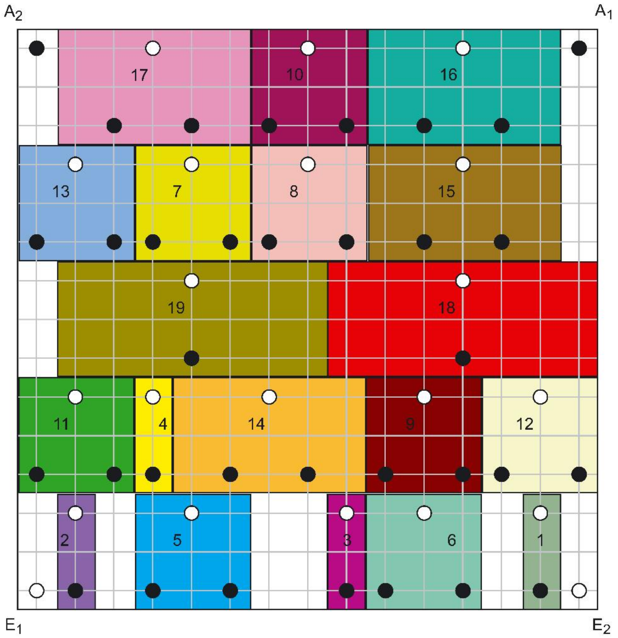


Example

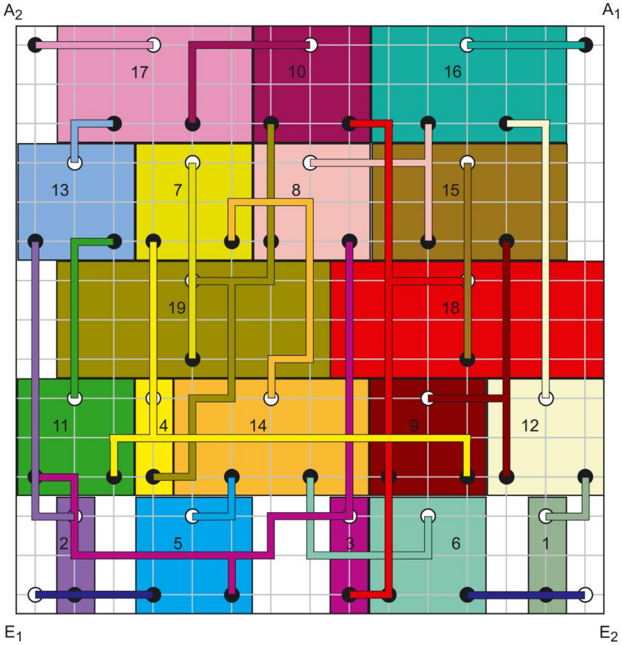


A_2
A_1
E_1
E_2
1
2
3
4
Inverter
5
6
7
8
9
10
11
12
13
Und - Bauteile
14
15
16
17
Oder - Bauteile
18
19
Latches (Z_1, Z_2)

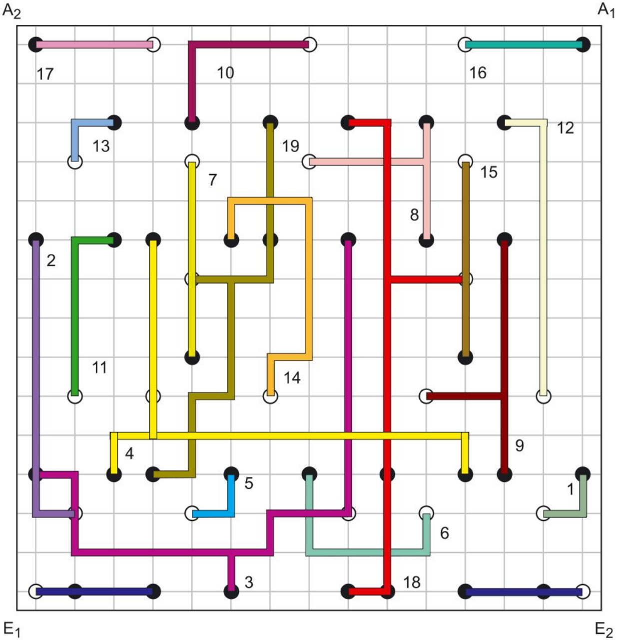
Example



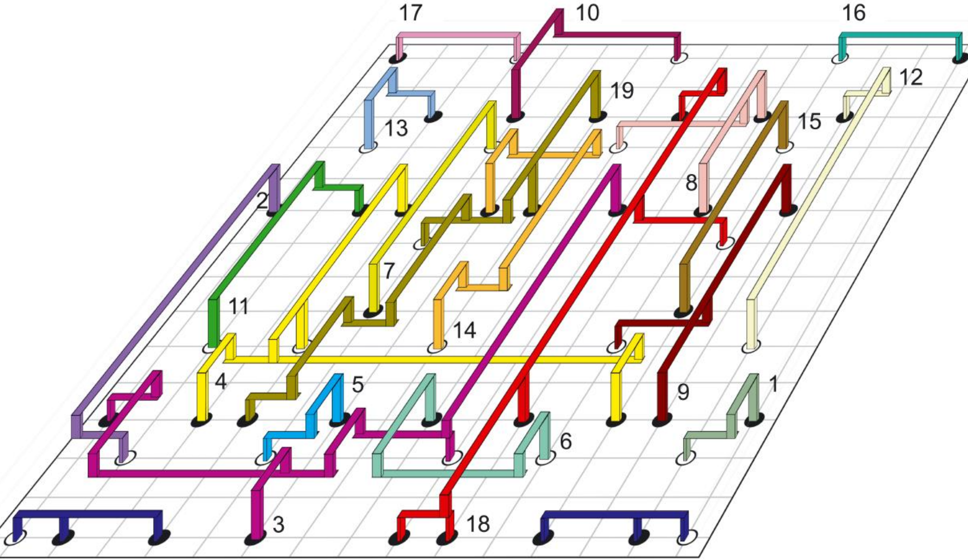
Example



Example



Example



VLSI design: overall task

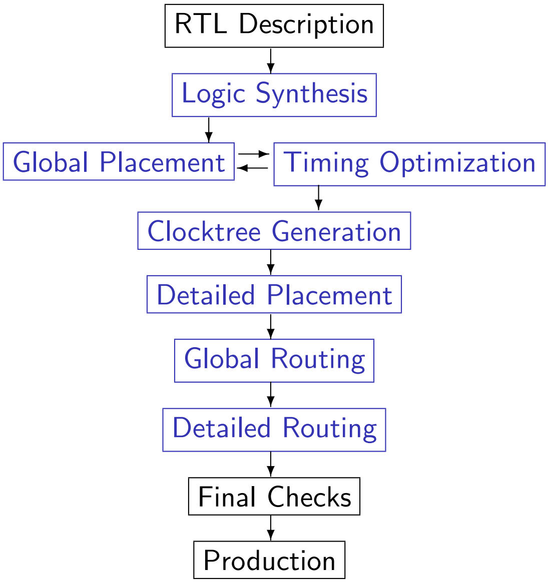
Given a netlist, with primary inputs, registers, primary outputs, complex logic cores, combinational logic,
and constraints for placement, routing, and timing, the task is to

- compute an equivalent netlist,
  where registers and parts of the combinational logic can be replaced if the Boolean function $\Phi : \{0,1\}^I \to \{0,1\}^O$ representing the netlist does not change. I contains the primary inputs and output pins of registers and cores, and O contains the primary outputs and input pins of registers and cores.
- place the components of this netlist
  without overlaps on the chip area,
- and route all nets
  i.e. find node-disjoint Steiner trees, each connecting the pins of a net, in a given 3-dimensional grid graph.

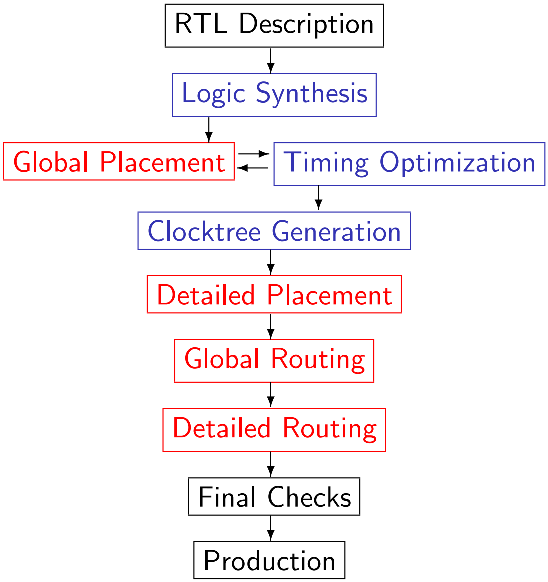
Combinatorial optimization in VLSI design automation

- shortest paths
- network design, in particular Steiner trees
- maximum flows, discrete time-cost tradeoff problems
- transportation and minimum cost flows
- multicommodity flows, disjoint paths and trees
- minimum mean cycles, parametric shortest paths
- facility location
- ... and others: minimum spanning trees, knapsack problem, bin packing, traveling salesman problem, Huffman codes, ...
- ... also used: advanced data structures, computational geometry, nonlinear programming, parallelization ...

Design flow


RTL Description
Logic Synthesis
Global Placement
Timing Optimization
Clocktree Generation
Detailed Placement
Global Routing
Detailed Routing
Final Checks
Production

Design flow


RTL Description
Logic Synthesis
Global Placement
Timing Optimization
Clocktree Generation
Detailed Placement
Global Routing
Detailed Routing
Final Checks
Production

Main objectives

- minimize cycle time / meet timing constraints
  (all signal arrival times within prescribed time intervals)
- minimize power consumption
  (depending on transistor sizes, length and widths of wires, coupling, leakage)
- minimize cost
  (area, number of masks, yield, design effort)

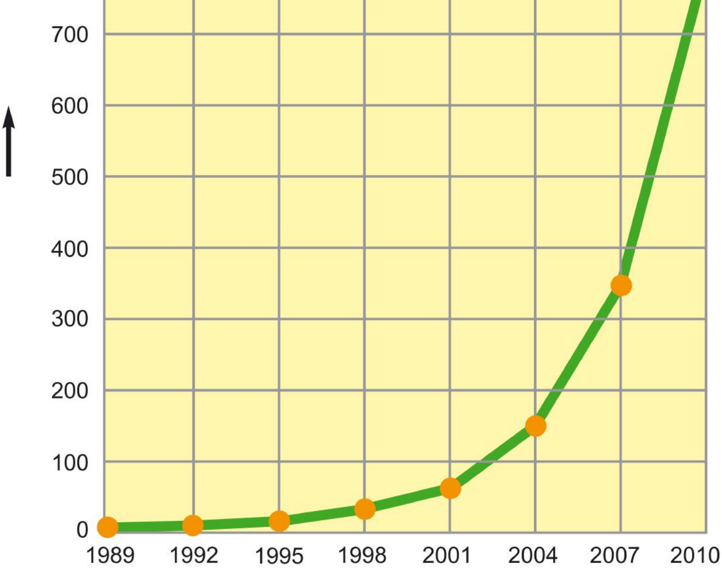
Main objectives in the design flow

Logic Synthesis	minimize area and depth
Global Placement	minimize wirelength
Timing Optimization	minimize delay and power

Now, assuming an optimum Steiner tree for each net, all signals must arrive in time.

Clocktree Generation	minimize power subject to timing
Detailed Placement	minimize changes
Global Routing	minimize power subject to timing
Detailed Routing	minimize changes

Moore's law



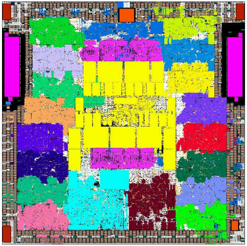
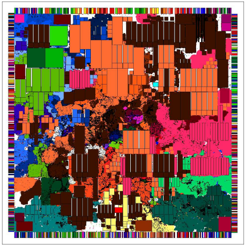
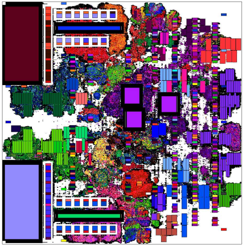
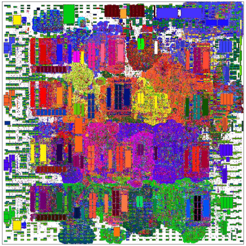
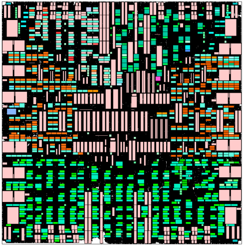
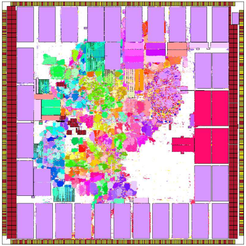
million transistors on a chip

The Bonn Tools

- are developped by the Research Institute for Discrete Mathematics at the University of Bonn,
- cover all major areas of layout and timing optimization,
- include libraries for combinatorial optimization, advanced, data structures, computational geometry, etc.,
- have more than one million lines of code in C and C++,
- are used by IBM and its customers for almost 20 years,
- are now also used by Magma Design Automation and its customers,
- have been used for the design of more than 1000 chips,
- including several complete microprocessor series,
- approximately hundred ASICs every year,
- and the most complex chips of major technology companies.

Outline in (Korte, Rautenbach and Vygen [2006])

Some recent chips

The Bonn group

currently consists of:

Christoph Bartoschek, Florian Berger, Ulrich Brenner, Alexander von Dambrowski, Laura Geisen, Michael Gester, Stephan Held, Günther Hutzl, Fritz Jahns, Johannes Klauser, Alexander Kleff, Bernhard Korte, Immo Krupke, Jens Maßberg, Andreas Menge, Dirk Müller, Karsten Muuss, Christian Panten, Sven Peyer, Dieter Rautenbach, Rüdiger Schmedding, Jan Schneider, Christian Schulte, Matthias Schwamborn, Markus Struzyna, Jens Vygen, Jürgen Werber

Thanks to all of them.

Thanks also to our cooperation partners at IBM and Magma

How to measure interconnect length?

Let N be a finite set of points in the plane. Define net models:

- $\textsc{Steiner}(N)$ is the length of an optimum rectilinear Steiner tree for N.
- $\textsc{BB}(N) := \max_{p \in N} x(p) - \min_{p \in N} x(p) + \max_{p \in N} y(p) - \min_{p \in N} y(p)$.
- $\textsc{MST}(N)$ is the length of a minimum spanning tree for N, where edge weights are rectilinear distances.
- $\textsc{Clique}(N) := \dfrac{1}{|N|-1} \displaystyle\sum_{p,p' \in N} (|x(p)-x(p')|+|y(p)-y(p')|)$.
- $\textsc{Star}(N) := \displaystyle\min_{(x',y') \in \mathbb{R}^2} \sum_{p \in N} (|x(p)-x'|+|y(p)-y'|)$.

Worst case ratios of various net models

Entry (r, c) is $\sup \frac{c(N)}{r(N)}$ over all point sets N with $|N| = n$.

	BB	STEINER	MST	CLIQUE	STAR
BB	1	1	1	1	1
STEI-NER	$\frac{n-1}{\lceil\sqrt{n}\rceil+\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil-2}$ $\cdots$ $\frac{\lceil\sqrt{n-2}\rceil}{2}+\frac{3}{4}$	1	1	$\begin{cases}\frac{9}{8} & (n=4)\\ 1 & (n\neq 4)\end{cases}$	1
MST	$\left\lfloor\frac{\sqrt{2n-1}+1}{2}\right\rfloor$ $\cdots$ $\frac{\sqrt{n}}{\sqrt{2}}+\frac{3}{2}$	$\frac{3}{2}$	1	$1+\Theta\left(\frac{1}{n}\right)$ $\cdots$ $\frac{3}{2}$	$\begin{cases}\frac{4}{3} & (n=3)\\ \frac{3}{2} & (n=4)\\ \frac{6}{5} & (n=5)\\ 1 & (n>5)\end{cases}$
CLIQUE	$\frac{\lceil\frac{n}{2}\rceil\lfloor\frac{n}{2}\rfloor}{n-1}$	$\frac{\lceil\frac{n}{2}\rceil\lfloor\frac{n}{2}\rfloor}{n-1}$	$\frac{\lceil\frac{n}{2}\rceil\lfloor\frac{n}{2}\rfloor}{n-1}$	1	1
STAR	$\lfloor\frac{n}{2}\rfloor$	$\lfloor\frac{n}{2}\rfloor$	$\lfloor\frac{n}{2}\rfloor$	$\frac{n-1}{\lceil\frac{n}{2}\rceil}$	1

(Hwang [1976], Brenner and Vygen [2001], Rautenbach [2004])

Net models in placement

- STEINER is best, but *NP*-hard to compute
- all others can be computed in $O(n)$ time (BB, STAR) or in $O(n \log n)$ time (MST, CLIQUE).
- in quadratic placement (see below), CLIQUE and STAR are used
- BB is often used as a simple measure. As most nets have few pins, this is not too bad.

Clique is the best topology-independent net model

Theorem

For $n \geq 2$, a connected graph G with $\{1, \dots, n\} \subseteq V(G)$, $c : E(G) \to \mathbb{R}_{>0}$, and $p : \{1, \dots, n\} \to \mathbb{R}^2$ let
$\mathcal{M}_{(G,c)}(p) :=$

$$\min\left\{ \sum_{e=\{v,w\}\in E(G)} c(e)||p(v)-p(w)||_1 \;\middle|\; p : V(G)\setminus\{1, \dots, n\} \to \mathbb{R}^2 \right\}.$$

Then the ratio of supremum and infimum of

$$\left\{ \mathcal{M}_{(G,c)}(p) \middle| p : \{1, \dots, n\} \to \mathbb{R}^2,\ \textsc{steiner}(\{p(1), \dots, p(n)\}) = 1 \right\}$$

is minimum for the complete graph K_n with unit weights.

(Brenner, Vygen [2001])

Placement: simplified problem formulation

Input:

- a rectangular chip area, and a set of rectangular blockages
- a finite set C of (rectangular) cells
- a finite set P of pins, and a partition $\mathcal{N}$ of P into nets
- a weight $w(N) > 0$ for each net N
- an assignment $\gamma : P \to C \cup \{\square\}$ of the pins to cells
  [pins p with $\gamma(p) = \square$ are fixed; we set $x(\square) := y(\square) := 0$]
- offsets $x(p), y(p) \in \mathbb{R}$ of each pin p

Task:

Find a position $(x(c), y(c)) \in \mathbb{R}^2$ of each cell c such that

- each cell is contained in the chip area,
- no cell overlaps with another cell or a blockage,

and the weighted netlength

$$\sum_{N \in \mathcal{N}} w(N)\, \mathrm{BB}\left(\{(x(\gamma(p)) + x(p), y(\gamma(p)) + y(p)) : p \in N\}\right)$$

is minimum.

Why minimize netlength?

- Netlength is a good estimate for power consumption
- Short nets have small delay (net weights for critical nets!)
- Nets have to be packed in routing, and long nets take more resources (but we must also avoid local congestion!)
- Experience shows: a good algorithm for the simplified placement problem can be extended to a good algorithm for real placement problems.
- Bounding box netlentgh is the main measure in benchmarks
- It's simple. But not easy...

Special case: Quadratic Assignment Problem (QAP)

Instance: A graph G. Weights $w : E(G) \to \mathbb{R}_+$. A set U with $|U| \geq |V(G)|$. Distances $d(\{u, v\}) \geq 0$ for all $u, v \in U$. Weights $c : V(G) \times U \to \mathbb{R}_+$.

Task: Find an injective mapping $f : V(G) \to U$ such that

$$\sum_{e=\{x,y\}\in E(G)} w(e)d(\{f(x), f(y)\}) + \sum_{x\in V(G), u\in U} c(x, u)d(\{f(x), u\})$$

is minimum.

Special case: Quadratic Assignment Problem (QAP)

Instance: A graph G. Weights $w : E(G) \to \mathbb{R}_+$. A set U with $|U| \geq |V(G)|$. Distances $d(\{u, v\}) \geq 0$ for all $u, v \in U$. Weights $c : V(G) \times U \to \mathbb{R}_+$.

Task: Find an injective mapping $f : V(G) \to U$ such that

$$\sum_{e=\{x,y\}\in E(G)} w(e)d(\{f(x), f(y)\}) + \sum_{x\in V(G), u\in U} c(x, u)d(\{f(x), u\})$$

is minimum.

Theorem

Unless $P = NP$ there is no constant-factor approximation algorithm for the special case of the QUADRATIC ASSIGNMENT PROBLEM *where $w(e) = 1$ for all $e \in E(G)$, c is identically zero, U is a finite subset of $\mathbb{Z}$ and $d(\{u, v\}) = |u - v|$ for all $u, v \in U$.*

(Queyranne [1986])

Proof of non-approximability

- BIN-PACKING is strongly *NP*-hard. More precisely, it is *NP*-hard to decide whether for given $n \in \mathbb{N}$ and $s_1, \dots, s_{4n}, B \in \{1, \dots, 10^{10} n^4\}$ there is a mapping $p : \{1, \dots, 4n\} \to \{1, \dots, n\}$ with $\sum_{i \in p^{-1}(j)} s_i \leq B$ for $j = 1, \dots, n$.

Proof of non-approximability

- $\textsc{Bin-Packing}$ is strongly *NP*-hard. More precisely, it is *NP*-hard to decide whether for given $n \in \mathbb{N}$ and $s_1, \dots, s_{4n}, B \in \{1, \dots, 10^{10} n^4\}$ there is a mapping $p : \{1, \dots, 4n\} \to \{1, \dots, n\}$ with $\sum_{i \in p^{-1}(j)} s_i \le B$ for $j = 1, \dots, n$.
- Define an instance of QAP by $V(G) = \{1, \dots, \sum_{i=1}^{n} s_i\}$; $E(G) := \{\{x, x+1\} : x \in V(G) \setminus \{\sum_{i=1}^{j} s_i : j = 1, \dots, n\}\}$; $U := \{(kn+1)Bj + z : z = 1, \dots, B,\ j = 1, \dots, n\}$.

Proof of non-approximability

- $\textsc{Bin-Packing}$ is strongly NP-hard. More precisely, it is NP-hard to decide whether for given $n \in \mathbb{N}$ and $s_1, \dots, s_{4n}, B \in \{1, \dots, 10^{10} n^4\}$ there is a mapping $p : \{1, \dots, 4n\} \to \{1, \dots, n\}$ with $\sum_{i \in p^{-1}(j)} s_i \leq B$ for $j = 1, \dots, n$.
- Define an instance of QAP by $V(G) = \{1, \dots, \sum_{i=1}^{n} s_i\}$; $E(G) := \{\{x, x+1\} : x \in V(G) \setminus \{\sum_{i=1}^{j} s_i : j = 1, \dots, n\}\}$; $U := \{(kn+1)Bj + z : z = 1, \dots, B,\ j = 1, \dots, n\}$.
- If there exists a mapping p as above, there is a placement $f : V(G) \to U$ defined by $f(\sum_{i=1}^{j-1} s_i + z) := (kn+1)Bp(j) + z$ for $j = 1, \dots, 4n$ and $z = 1, \dots, s_j$, such that $\sum_{e=\{x,y\} \in E(G)} |f(x) - f(y)| = |E(G)|$.

Proof of non-approximability

- $\textsc{Bin-Packing}$ is strongly NP-hard. More precisely, it is NP-hard to decide whether for given $n \in \mathbb{N}$ and $s_1, \dots, s_{4n}, B \in \{1, \dots, 10^{10} n^4\}$ there is a mapping $p : \{1, \dots, 4n\} \to \{1, \dots, n\}$ with $\sum_{i \in p^{-1}(j)} s_i \le B$ for $j = 1, \dots, n$.
- Define an instance of QAP by $V(G) = \{1, \dots, \sum_{i=1}^{n} s_i\}$; $E(G) := \{\{x, x+1\} : x \in V(G) \setminus \{\sum_{i=1}^{j} s_i : j = 1, \dots, n\}\}$; $U := \{(kn+1)Bj + z : z = 1, \dots, B,\ j = 1, \dots, n\}$.
- If there exists a mapping p as above, there is a placement $f : V(G) \to U$ defined by $f(\sum_{i=1}^{j-1} s_i + z) := (kn+1)Bp(j) + z$ for $j = 1, \dots, 4n$ and $z = 1, \dots, s_j$, such that $\sum_{e=\{x,y\} \in E(G)} |f(x) - f(y)| = |E(G)|$.
- Otherwise, for any injective mapping $f : V(G) \to U$ there is an edge $\{x, y\} \in E(G)$ with $|f(x) - f(y)| \ge (kn+1)B + 1 - \max_{i=1}^{4n} s_i \ge knB > k|E(G)|$.

Proof of non-approximability

- Bin-Packing is strongly NP-hard. More precisely, it is NP-hard to decide whether for given $n \in \mathbb{N}$ and $s_1, \dots, s_{4n}, B \in \{1, \dots, 10^{10} n^4\}$ there is a mapping $p : \{1, \dots, 4n\} \to \{1, \dots, n\}$ with $\sum_{i \in p^{-1}(j)} s_i \le B$ for $j = 1, \dots, n$.
- Define an instance of QAP by $V(G) = \{1, \dots, \sum_{i=1}^{n} s_i\}$; $E(G) := \{\{x, x+1\} : x \in V(G) \setminus \{\sum_{i=1}^{j} s_i : j = 1, \dots, n\}\}$; $U := \{(kn+1)Bj + z : z = 1, \dots, B,\ j = 1, \dots, n\}$.
- If there exists a mapping p as above, there is a placement $f : V(G) \to U$ defined by $f(\sum_{i=1}^{j-1} s_i + z) := (kn+1)Bp(j) + z$ for $j = 1, \dots, 4n$ and $z = 1, \dots, s_j$, such that $\sum_{e=\{x,y\} \in E(G)} |f(x) - f(y)| = |E(G)|$.
- Otherwise, for any injective mapping $f : V(G) \to U$ there is an edge $\{x, y\} \in E(G)$ with $|f(x) - f(y)| \ge (kn+1)B + 1 - \max_{i=1}^{4n} s_i \ge knB > k|E(G)|$.
- Hence a k-factor approximation algorithm for such instances of QAP can distinguish between these two cases. $\square$

Positive results

There are only few, and these are not very useful.

Special case: OPTIMUM LINEAR ARRANGEMENT PROBLEM
Given a graph G with $n := |V(G)|$, we ask for a bijection $f : V(G) \to \{1, \ldots, n\}$ minimizing $\sum_{\{x,y\} \in E(G)} |f(x) - f(y)|$.

- Even this problem is NP-hard. (Garey, Johnson [1976])
- There is an $O(\sqrt{\log n} \log \log n)$-factor approximation algorithm. (Charikar, Hajiaghayi, Karloff, Rao [2006])
- However, the problem is not known to be $MAXSNP$-hard!

Polylogarithmic approximation algorithm also for some two-dimensional problems
(Even, Naor, Rao, Schieber [2000], Even, Guha, Schieber [2003], Vempala [1998])

Placement approaches in practice

- simulated annealing: start with any placement and try to improve it
  (mostly used in the 80s)
- min-cut: successive bisection, with simple exchange heuristics
  (mostly used in the 90s)
- analytical placement: minimize either linear or quadratic netlength estimate, then work towards disjointness
  (dominant strategy today)

Here we discuss analytical placement only.

Minimizing weighted netlength

$$\min \sum_{N \in \mathcal{N}} w(N)(X_N + Y_N)$$

where

$$X_N := \max\{x(\gamma(p))+x(p) : p \in N\} - \min\{x(\gamma(p))+x(p) : p \in N\}$$

$$Y_N := \max\{y(\gamma(p))+y(p) : p \in N\} - \min\{y(\gamma(p))+y(p) : p \in N\}$$

Net weights $w(N)$ reflect timing criticality (slack, Lagrange multipliers).

Minimizing weighted netlength

Equivalent formulation:

$$\min \sum_{N \in \mathcal{N}} w(N)(r(N) - l(N) + t(N) - b(N))$$

subject to

$$l(N) \leq x(\gamma(p)) + x(p) \leq r(N) \qquad (p \in N \in \mathcal{N})$$
$$b(N) \leq y(\gamma(p)) + y(p) \leq t(N) \qquad (p \in N \in \mathcal{N})$$

Minimizing weighted netlength

Equivalent formulation:

$$\min \sum_{N \in \mathcal{N}} w(N)(r(N) - l(N) + t(N) - b(N))$$

subject to

$$l(N) \leq x(\gamma(p)) + x(p) \leq r(N) \qquad (p \in N \in \mathcal{N})$$
$$b(N) \leq y(\gamma(p)) + y(p) \leq t(N) \qquad (p \in N \in \mathcal{N})$$

This is the dual of a minimum cost flow problem.
(Cabot, Francis and Stary [1970])

Quadratic placement (QP)

$$\min \sum_{N\in\mathcal{N}} \frac{w(N)}{|N|-1} \sum_{p,q\in N} (X_{p,q} + Y_{p,q})$$

where

$$X_{p,q} := |x(\gamma(p)) + x(p) - x(\gamma(q)) - x(q)|^2$$

and

$$Y_{p,q} := |y(\gamma(p)) + y(p) - y(\gamma(q)) - y(q)|^2$$

The placement where this minimum is attained is unique if the netlist is connected. It is called quadratic placement.

Why using quadratic placement?

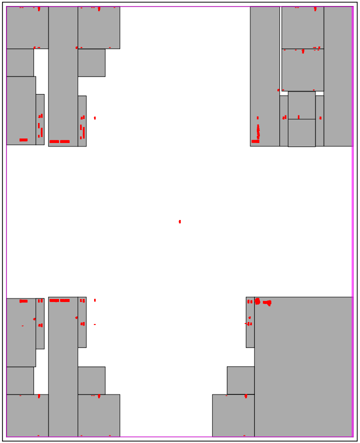
- QP can be solved very fast (conjugate gradient method)
- Delay along unbuffered wires grows quadratically with length
- QP gives a lot of information on relative positions
- QP is stable:

Theorem

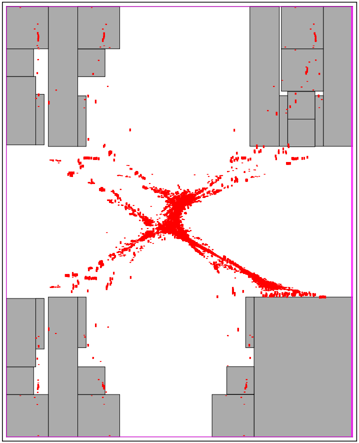
Small netlist changes imply small changes of QP solution.
In contrast, min-cut and local search are unstable.

(Vygen [2002])

Minimizing linear versus quadratic netlength



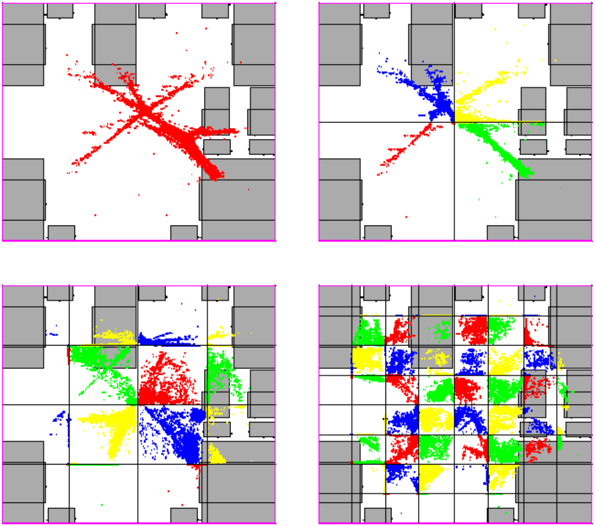
placement with minimum bounding box netlength



quadratic placement

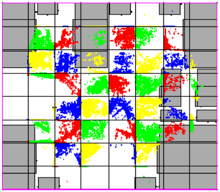
Global placement by successive partitioning

Remove overlaps by successive quadrisection.



Successively distribute the set C of cells to regions R.

Quadratic placement with an array of regions



$$\min \sum_{N\in\mathcal{N}} \frac{w(N)}{|N|-1} \sum_{p,q\in N} (X_{p,q} + Y_{p,q})$$

where $X_{p,q}$ is

- $|x(\gamma(p)) + x(p) - x(\gamma(q)) - x(q)|^2$ if $\gamma(p)$ and $\gamma(q)$ are cells assigned to regions in the same column.
- $|x(\gamma(p)) + x(p) - b|^2 + |x(\gamma(q)) + x(q) - a'|^2$ if $\gamma(p)$ is a cell assigned to a region with x-range $[a, b]$, $\gamma(q)$ is a cell assigned to a region with x-range $[a', b']$ and $b \le a'$.
- $|x(\gamma(p)) + x(p) - v|^2$ if $\gamma(p)$ is a cell assigned to a region with x-range $[a, b]$, q is fixed, and $v = \max\{a, \min\{b, x(q)\}\}$.
- 0 if p and q are both fixed.

$Y_{p,q}$ is defined analogously, but with respect to y-coordinates, and with rows playing the role of columns.
(Vygen [1997])

Replace large cliques by stars

The running time of the conjugate gradient method depends on

- the condition of the matrix (we apply incomplete Cholesky preconditioning)
- the number of variables (cells) and
- the number of connected pin pairs

Therefore, one should replace large cliques, i.e. sets of pins belonging to the same net and to cells in the same column (row) by a stars: introduce a new variable (representing the center of the star) and connect it to each of the pins.

With appropriate weights, this does not change the result.

A single partitioning step

Let C be a set of cells, each with a size,
and R a set of (sub)regions, each with a capacity.

Task: Find an assignment $f : C \to R$
meeting the capacity constraints

$$\sum_{c \in C: f(c)=r} size(c) \leq cap(r) \text{ for all } r \in R$$

such that the total movement

$$\sum_{c \in C} d(c, f(c))$$

is minimum.
Here d denotes, e.g., the ℓ_1-distance.

But partitioning is hard

Even the problem to decide whether a feasible assignment exists is *NP*-hard (it contains the PARTITION problem).

For the same reason it is *NP*-hard to decide whether a feasible placement exists.

However, for global placement this is only of theoretical interest.

Fractional relaxation: Hitchcock (transportation) problem

Find $g : C \times R \longrightarrow \mathbb{R}_+$
with

$$\sum_{r \in R} g(c, r) = size(c) \text{ for all } c \in C$$

and

$$\sum_{c \in C} g(c, r) \leq cap(r) \text{ for all } r \in R$$

such that

$$\sum_{c \in C} \sum_{r \in R} g(c, r) d(c, r)$$

is minimum.
Note: $|R| \ll |C|$.

Solving the fractional relaxation is sufficient

Proposition

From any optimum solution to the fractional relaxation we can obtain another one in $O(|C||R|^2)$ *time that is integral up to* $|R| - 1$ *cells.*

Solving the fractional relaxation is sufficient

Proposition

From any optimum solution to the fractional relaxation we can obtain another one in $O(|C||R|^2)$ time that is integral up to $|R| - 1$ cells.

Proof.

Define $V(G) := R$ and $E(G) := \{\{r, r'\} : c \in C, g(c, r) > 0, g(c, r') > 0, g(c, r'') = 0 \text{ for } r'' \in \{1, \ldots, \max\{r, r'\}\} \setminus \{r, r'\}\}$.

Solving the fractional relaxation is sufficient

Proposition

From any optimum solution to the fractional relaxation we can obtain another one in $O(|C||R|^2)$ time that is integral up to $|R| - 1$ cells.

Proof.

Define $V(G) := R$ and $E(G) := \{\{r, r'\} : c \in C, g(c, r) > 0, g(c, r') > 0, g(c, r'') = 0 \text{ for } r'' \in \{1, \ldots, \max\{r, r'\}\} \setminus \{r, r'\}\}$.

While G contains a cycle, consider g' and g'' that result from g by moving the same amount of flow around the cycle in each direction.

Solving the fractional relaxation is sufficient

Proposition

From any optimum solution to the fractional relaxation we can obtain another one in $O(|C||R|^2)$ time that is integral up to $|R|-1$ cells.

Proof.

Define $V(G) := R$ and $E(G) := \{\{r, r'\} : c \in C, g(c, r) > 0, g(c, r') > 0, g(c, r'') = 0 \text{ for } r'' \in \{1, \dots, \max\{r, r'\}\} \setminus \{r, r'\}\}$.

While G contains a cycle, consider g' and g'' that result from g by moving the same amount of flow around the cycle in each direction.

Both g' and g'' must be optimum solutions.
The number of cell fractions decreases. Iterate. □

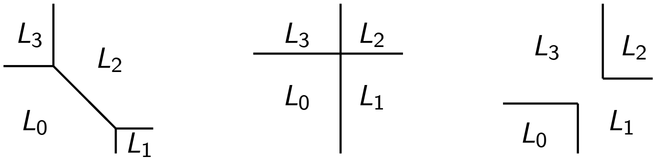
(Vygen [1996,2005])

Special case: quadrisection

Theorem

If R consists of the four quadrants and d is the weighted ℓ_1-distance, then

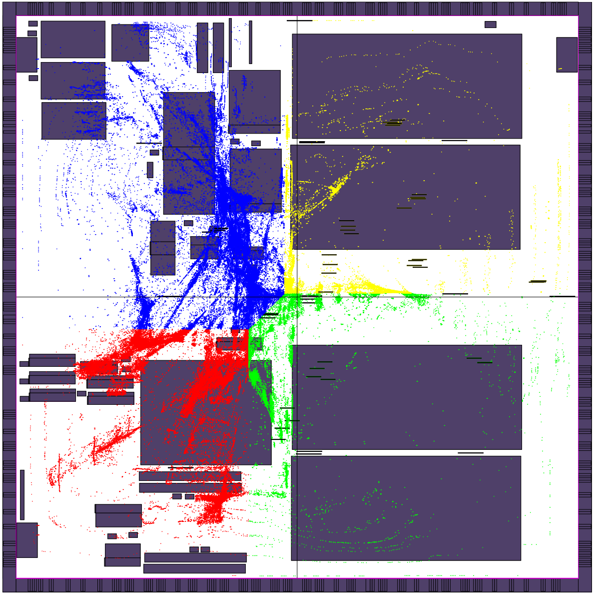
- *there is an optimal solution to the fractional relaxation that corresponds to an American map:*



- *such a solution can be found efficiently in $O(n)$ time.*

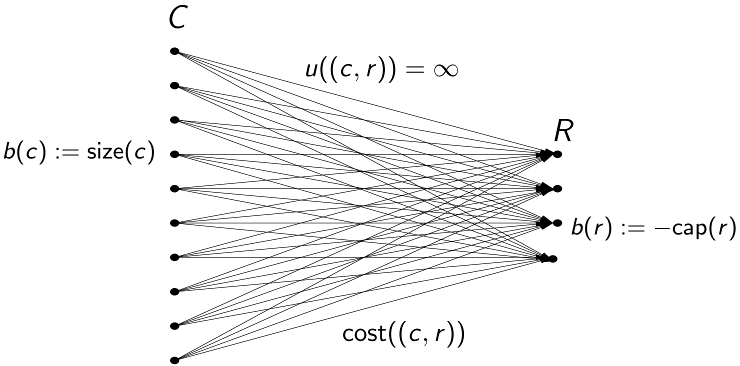
(Vygen [1996, 2005])

Quadrisection based on quadratic placement



General case: Hitchcock Problem

Let G be the digraph with $V(G) := C \dot\cup R$ and $E(G) := C \times R$.
Let $b(c) := \text{size}(c)$ for $c \in C$ and $b(r) := -\text{cap}(r)$ for $r \in R$.
Let $\text{cost}((c, r)) := \frac{d((c,r))}{\text{size}(c)}$ $(c \in C,\ r \in R)$



Task: Find an uncapacitated b-flow in G of minimum cost.

Algorithms for the Hitchcock problem

Let $n := |C|$ and $k := |R|$. We assume $n \geq k$.

- $O(n \log n(n \log n + kn))$: general transshipment algorithm (Orlin [1993])
- $O(n f(k))$ with exponential functions f, inefficient already for very small k: Dyer [1984], Zemel [1984], Tokuyama, Nakano [1991], Meggido, Tamir [1993], Matsui [1993]
- $O(nk^2 \log^2 n)$ Tokuyama, Nakano [1992, 1995]
- New algorithm: $O(nk^2(\log n + k \log k))$: Brenner [2005]

Residual graph

Given $f : E(G) \to \mathbb{R}_+$, we define the residual graph G_f as follows.

- $V(G_f) := V(G) \cup \{t\}$.
- $E(G_f)$ contains all arcs $e \in E(G)$, with $u_f(e) := \infty$.
- For each arc $(c, r) \in E(G)$ with $f((c, r)) > 0$, $E(G_f)$ contains the backward arc (r, c) with $u_f((r, c)) := f((c, r))$.
- For each $r \in R$ with $f(\delta^-(r)) + b(r) < 0$, $E(G_f)$ contains the arc (r, t) with $u_f((r, t)) := -b(r) - f(\delta^-(r))$.

Successive shortest path algorithm

Input: An instance (G, b, cost) of the HITCHCOCK PROBLEM.
Output: A minimum cost b-flow f in G.

① $f(e) := 0$ for $e \in E(G)$.

② Let $C = \{c_1, \ldots, c_n\}$.

③ For $i := 1$ to n:
  While $f(\delta^+(c_i)) < b(c_i)$:
    Find a shortest c_i-t-path P in G_f.
    $\gamma := \min \left\{ \min_{e \in E(P)} u_f(e), b(c_i) - f(\delta^+(c_i)) \right\}$.
    Augment f along P by γ.

Idea: Replace each phase (iteration of the outer loop) by one min-cost flow computation in a graph whose size depends on k only.

Lemma on almost integral solutions

Definition: Let f be a solution of the Hitchcock Problem.
For $c \in C$ let $\tau_f(c) := |\{r \in R : f((c, r)) > 0\}|$.
Let $F_f := \{c \in C : \tau_f(c) > 1\}$.

Lemma

Given an instance $(G, b, u, cost)$ of the Hitchcock Problem, *an optimum solution f, and the set F_f, we can transform f in $O(k \cdot \sum_{c \in F_f} \tau_f(c))$ time into an optimum solution g such that:*

- $|F_g| \leq k - 1$, *and*
- $\sum_{c \in F_g} \tau_g(c) \leq 2k - 2.$

General strategy

- Sort the cells such that $\text{size}(c_1) \geq \text{size}(c_2) \geq \cdots \geq \text{size}(c_n)$.
- We will show: In each phase we have to change the flow only on $O(k^2)$ arcs.

Notation

- Let f_{i-1} be the flow at the beginning of phase i.
- For $r \in R$ let $M_r^i := \{c \in C : f_{i-1}((c, r)) = \text{size}(c)\}$.
- If $M_r^i \neq \emptyset$ for $r \in R$, then choose for each $q \in R \setminus \{r\}$ an arbitrary $c_{r,q}^i \in M_r^i$ with

$$\text{cost}((c_{r,q}^i, q)) - \text{cost}((c_{r,q}^i, r)) =$$

$$\min \left\{ \text{cost}((c', q)) - \text{cost}((c', r)) : c' \in M_r^i \right\}.$$

The subgraph G_i

$$
\begin{aligned}
V(G_i) \; := \; & R \cup \{t\} \\
& \cup \{c_i\} \cup F_{f_{i-1}} \\
& \cup \{c^i_{r,q} : r, q \in R,\ r \neq q,\ M^i_r \neq \emptyset\}
\end{aligned}
$$

$$
\begin{aligned}
E(G_i) \; := \; & (R \times \{t\}) \\
& \cup (\{c_i\} \times R) \\
& \cup (F_{f_{i-1}} \times R) \\
& \cup \{(c^i_{r,q}, r) : r, q \in R,\ M^i_r \neq \emptyset\} \\
& \cup \{(c^i_{r,q}, q) : r, q \in R,\ M^i_r \neq \emptyset\}
\end{aligned}
$$

$\Rightarrow$ G_i has $O(k^2)$ vertices and $O(k^2)$ arcs.

Main lemma

In phase i we can choose augmenting paths such that there are no two subsequent arcs $(r, c), (c, q)$ with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Main lemma

In phase i we can choose augmenting paths such that there are no two subsequent arcs $(r, c), (c, q)$ with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Proof (Sketch).

Consider a sequence of shortest augmenting paths in $G_{f_{i-1}}$.
Consider the first path that contains arcs (r, c), (c, q) with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Main lemma

In phase i we can choose augmenting paths such that there are no two subsequent arcs $(r, c), (c, q)$ with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Proof (Sketch).

Consider a sequence of shortest augmenting paths in $G_{f_{i-1}}$.
Consider the first path that contains arcs (r, c), (c, q) with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Then $c \in M^i_p$ for some $p \in R$.

Main lemma

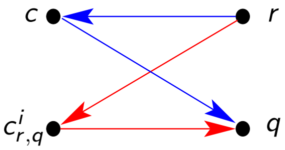
In phase i we can choose augmenting paths such that there are no two subsequent arcs $(r,c),(c,q)$ with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Proof (Sketch).

Consider a sequence of shortest augmenting paths in $G_{f_{i-1}}$.
Consider the first path that contains arcs (r,c), (c,q) with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Then $c \in M^i_p$ for some $p \in R$.

Case 1: $p = r$. Replace c by $c^i_{r,q}$ in the augmenting path.



cost of new subpath

$$\begin{aligned} &= \text{cost}((c^i_{r,q}, q) - \text{cost}((c^i_{r,q}, r)) \\ &\leq \text{cost}((c,q)) - \text{cost}((c,r)) \\ &= \text{cost of old subpath} \end{aligned}$$

Main lemma

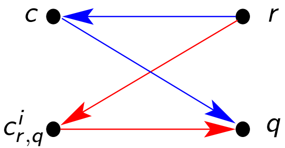
In phase i we can choose augmenting paths such that there are no two subsequent arcs $(r,c),(c,q)$ with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Proof (Sketch).

Consider a sequence of shortest augmenting paths in $G_{f_{i-1}}$.
Consider the first path that contains arcs (r,c), (c,q) with $c \in C \setminus (F_{f_{i-1}} \cup \{c^i_{r,q}, c^i_{q,r}\})$.

Then $c \in M^i_p$ for some $p \in R$.

Case 1: $p = r$. Replace c by $c^i_{r,q}$ in the augmenting path.



cost of new subpath

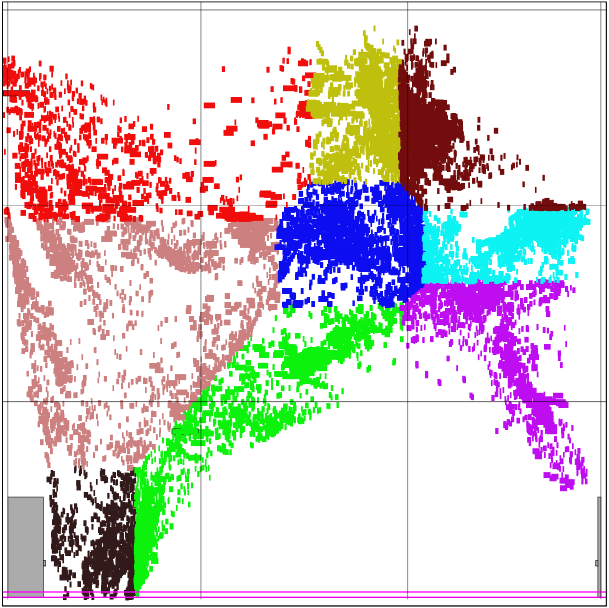
$$
\begin{aligned}
&= \mathrm{cost}((c^i_{r,q}, q) - \mathrm{cost}((c^i_{r,q}, r)) \\
&\leq \mathrm{cost}((c,q)) - \mathrm{cost}((c,r)) \\
&= \text{cost of old subpath}
\end{aligned}
$$

Case 2: $p \neq r$. Replace (c,q) by $(c,p,c^i_{p,q},q)$ in the augmenting path. □

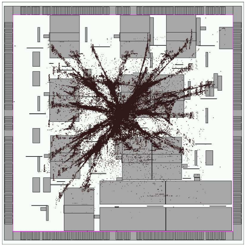
Main proof (sketch)

- After phase i, we adjust the flow f_i such that $|F_{f_i}| \leq k-1$ and $\sum_{c \in F_{f_i}} \tau_{f_i}(c) \leq 2k-2$.
- Two more modifications:
  - Replace each $c^i_{r,q}$ (with its two incident arcs) by one uncapacitated arc from r to q. $\Rightarrow$ Only $O(k)$ vertices remain.
  - Only $O(k)$ arcs (entering the elements of $F_{f_{i-1}}$) have finite capacity. Replace each of them equivalently by two uncapacitated arcs. $\Rightarrow$ All arcs are uncapacitated.
- Thus, if G_i is given, a phase can be computed in $O(k \log k(k^2 + k \log k)) = O(k^3 \log k)$ time (Orlin[1993])
- By storing the sets M^i_r in heaps, G_i can be computed from G_{i-1} in $O(k^2 \log n)$ time.
- In each phase: $O(k^2)$ insert and remove operations suffice.
- Time to adjust the flow after a phase: $O(k^3)$.
- Total running time: $O(nk^2(\log n + k \log k))$. $\square$

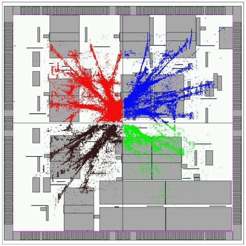
Multisection example



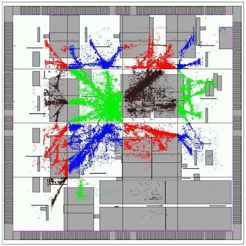
QP and quadrisection in BonnPlace



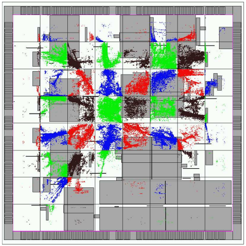
Level 0



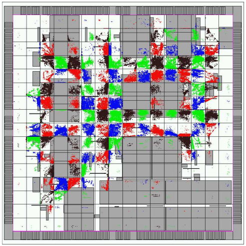
Level 1



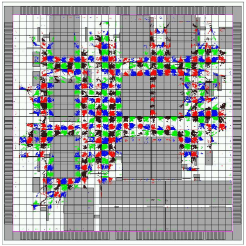
Level 2



Level 3



Level 4

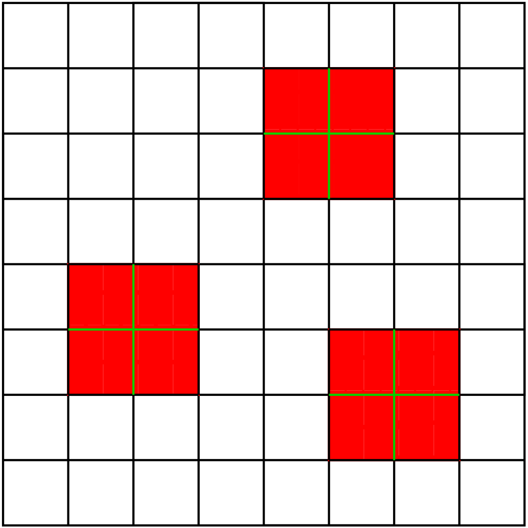


Level 5

Further major components of BonnPlace global placement

- Repartitioning
- Congestion-driven placement
- Macro placement

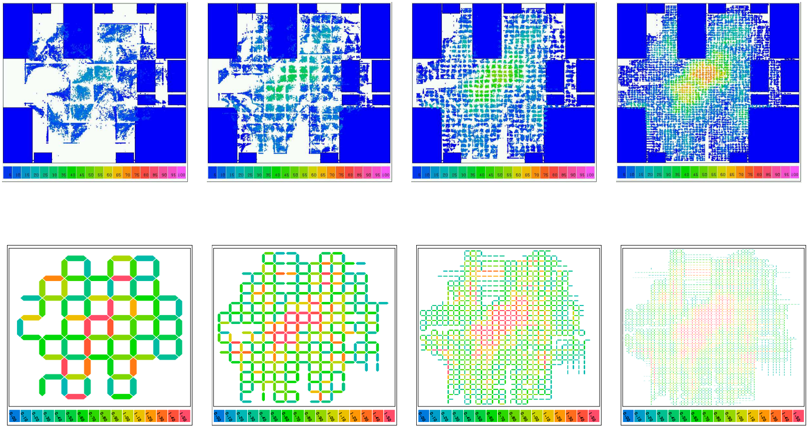
Repartitioning



for all 2×2 subarrays of regions do:

- free QP
- quadrisection
- QP w.r.t. new assignment
- check if new solution is better

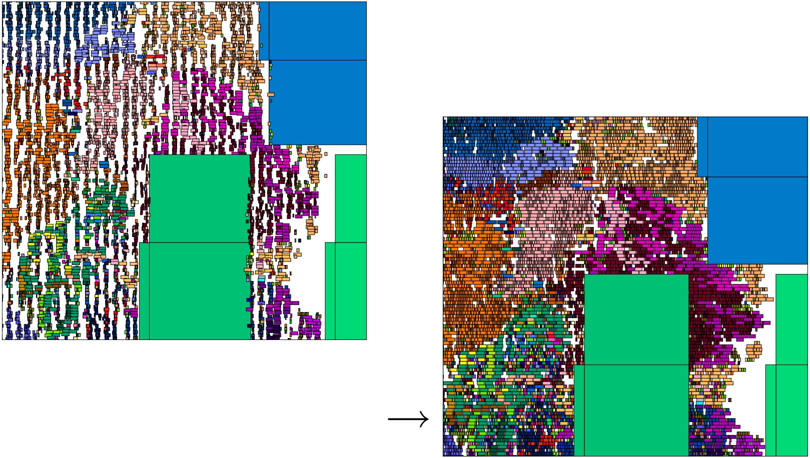
Congestion-driven placement



Macro placement

Detailed placement

After global placement we have an optimized but illegal placement.



We want to legalize it without changing it too much.

The legalization problem

Input:

- a rectangular chip area
- a set of rectangular blockages
- a set C of rectangular cells with unit height
- a width $w(c)$ and a position $(x(c), y(c)) \in \mathbb{R}^2$ of each cell $c \in C$.

Task:

Find new positions $(x'(c), y'(c)) \in \mathbb{Z}^2$ of the cells such that

- each cell is contained in the chip area,
- no two cells overlap,
- no cell overlaps with any blockage,

and $\sum_{c \in C} ((x(c) - x'(c))^2 + (y(c) - y'(c))^2)$ is minimum.

The problem is *NP*-hard.

Three-step approach:

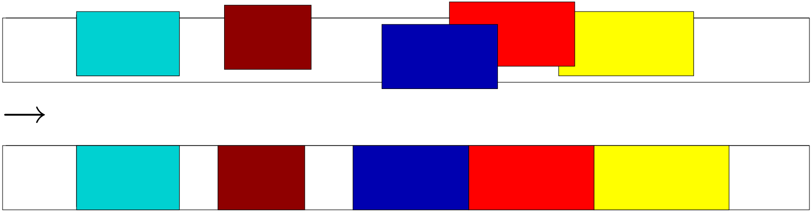
A zone is a maximal part of a row that is completely blocked or completely free.

Step 1: Make sure that no zone contains more cells than fit into it.

Step 2: Place the cells legally within their zones, keeping their horizontal order.

Step 3: Postoptimzation heuristics

Step 2: legalizing within zones



An optimal placement of n rectangles in a row in a given order can be found in

- $O(n \log n)$ time (for linear movement)
- $O(n)$ time (for quadratic movement)

(Garey, Tarjan, Wilfong [1988], Brenner, Vygen [2004])

Algorithm for a single zone

Idea:

- Consider cells from left to right.
- Let c be the leftmost unplaced cell.
- Place c at the leftmost optimal position.
- If c is not placed feasibly (i.e., to the right of its predecessor), merge it with its predecessor. The resulting cell is unplaced.
- Continue until all cells are placed.

Algorithm for a single zone

Input: $n \in \mathbb{N}$. Convex functions $f_1, \ldots, f_n : \mathbb{R} \to \mathbb{R}$.
Widths $w_1, \ldots, w_n > 0$ and bounds $x_{\min}, x_{\max} \in \mathbb{R}$ with $x_{\max} - x_{\min} \geq w_1 + \ldots + w_n$.

Output: $x_1, \ldots, x_n$ with $x_{\min} \leq x_1$, $x_i + w_i \leq x_{i+1}$ for $i = 1, \ldots, n-1$, $x_n \leq x_{\max}$, and $\sum_{i=1}^{n} f_i(x_i)$ minimum.

1. $x_0 := x_{\min}$.
   $W_0 := 0$, $W_i := w_i$ for $i = 1, \ldots, n$.
   Let $\mathcal{L}$ be the list consisting of $0, 1, \ldots, n$.
   $i := 1$.

Algorithm for a single zone (2)

② Let h be the predecessor of i in $\mathcal{L}$.
If $h = 0$ or
$x_h + W_h \leq \min\{x_{\max} - W_i, \max\{x : f_i(x) \text{ minimum}\}\}$
then go to ③ else go to ④.

③ $x_i := \max\{x_h + W_h, \min\{x_{\max} - W_i, \min\{x : f_i(x) \text{ minimum}\}\}\}$.
If there is a successor j of i in $\mathcal{L}$
then set $i := j$ and go to ② else go to ⑤.

④ Redefine f_h by $f_h : x \mapsto f_h(x) + f_i(x + W_h)$.
$W_h := W_h + W_i$.
Remove i from $\mathcal{L}$.
$i := h$
Go to ②.

⑤ For $i \in \{1, \ldots, n\} \setminus \mathcal{L}$ do: $x_i := x_h + \sum_{j=h}^{i-1} w_j$, where h is the maximum index smaller than i that belongs to $\mathcal{L}$.

Algorithm for a single zone: result

Theorem

This algorithm finds an optimum placement. If all f_i are quadratic, the algorithm can be implemented in linear time.

(Brenner, Vygen [2004], based on
Kahng, Tucker, Zelikovsky [1999])

Algorithm for a single zone: running time

Theorem

If all f_i are quadratic, the algorithm can be implemented in linear time.

Proof.

In each iteration, i increases by one or $|\mathcal{L}|$ and i decrease by one. As $1 \leq i \leq |\mathcal{L}| \leq n+1$, the total number of iterations is $\leq 2n$.

For quadratic functions $f_i : x \mapsto a_i x^2 + b_i x + \text{const}$, each iteration can be done in constant time as $\{x : f_i(x) \text{ minimum}\} = \{\frac{-b_i}{2a_i}\}$ and $f_h(x) + f_i(x + W_h) = (a_h + a_i)x^2 + (b_h + b_i + 2a_i W_h)x + \text{const.}$ □

Algorithm for a single zone: proof of optimality

Notation: $\rho_j := \max\{x : f_j(x) \text{ minimum}\}$.

Algorithm for a single zone: proof of optimality

Notation: $\rho_j := \max\{x : f_j(x) \text{ minimum}\}$.

Proof by induction. Consider one particular iteration with list $\mathcal{L}$ and index i. Let $\mathcal{L}' := \{j \in \mathcal{L} : j < i\}$, and suppose that

$$(*) \quad \min\{x_{\max} - W_j, \min\{x : f_j(x) \text{ minimum}\}\} \leq x_j \leq \max\{x_{\min}, \rho_j\}$$

for all $j \in \mathcal{L}'$. Let h be the maximal element of $\mathcal{L}'$.

Algorithm for a single zone: proof of optimality

Notation: $\rho_j := \max\{x : f_j(x) \text{ minimum}\}$.

Proof by induction. Consider one particular iteration with list $\mathcal{L}$ and index i. Let $\mathcal{L}' := \{j \in \mathcal{L} : j < i\}$, and suppose that

$$(*) \quad \min\{x_{\max} - W_j, \min\{x : f_j(x) \text{ minimum}\}\} \leq x_j \leq \max\{x_{\min}, \rho_j\}$$

for all $j \in \mathcal{L}'$. Let h be the maximal element of $\mathcal{L}'$.

If $h = 0$ or $x_h + W_h \leq \min\{x_{\max} - W_i, \rho_i\}$, then x_i is chosen in ③ such that $(*)$ holds also for $j = i$.

Algorithm for a single zone: proof of optimality

Notation: $\rho_j := \max\{x : f_j(x) \text{ minimum}\}$.

Proof by induction. Consider one particular iteration with list $\mathcal{L}$ and index i. Let $\mathcal{L}' := \{j \in \mathcal{L} : j < i\}$, and suppose that

$$(*) \quad \min\{x_{\max} - W_j, \min\{x : f_j(x) \text{ minimum}\}\} \leq x_j \leq \max\{x_{\min}, \rho_j\}$$

for all $j \in \mathcal{L}'$. Let h be the maximal element of $\mathcal{L}'$.

If $h = 0$ or $x_h + W_h \leq \min\{x_{\max} - W_i, \rho_i\}$, then x_i is chosen in ③ such that $(*)$ holds also for $j = i$.

Otherwise we claim that there is an optimum solution $(x_j^*)_{j \in \mathcal{L}' \cup \{i\}}$ of the subproblem defined by $(f_j, W_j)_{j \in \mathcal{L}' \cup \{i\}}$ where $x_h^* + W_h = x_i^*$. This justifies merging h and i in ④.

Algorithm for a single zone: proof of optimality

Notation: $\rho_j := \max\{x : f_j(x) \text{ minimum}\}$.

Proof by induction. Consider one particular iteration with list $\mathcal{L}$ and index i. Let $\mathcal{L}' := \{j \in \mathcal{L} : j < i\}$, and suppose that

$$(*) \quad \min\{x_{\max} - W_j, \min\{x : f_j(x) \text{ minimum}\}\} \leq x_j \leq \max\{x_{\min}, \rho_j\}$$

for all $j \in \mathcal{L}'$. Let h be the maximal element of $\mathcal{L}'$.

If $h = 0$ or $x_h + W_h \leq \min\{x_{\max} - W_i, \rho_i\}$, then x_i is chosen in ③ such that $(*)$ holds also for $j = i$.

Otherwise we claim that there is an optimum solution $(x_j^*)_{j \in \mathcal{L}' \cup \{i\}}$ of the subproblem defined by $(f_j, W_j)_{j \in \mathcal{L}' \cup \{i\}}$ where $x_h^* + W_h = x_i^*$. This justifies merging h and i in ④.

Let $(x_j^*)_{j \in \mathcal{L}' \cup \{i\}}$ be such an optimum solution. If $x_i^* - W_h \leq \rho_h$, then x_h^* can be set to $x_i^* - W_h$ without increasing $f_h(x_h^*)$.

Algorithm for a single zone: proof of optimality

Notation: $\rho_j := \max\{x : f_j(x) \text{ minimum}\}$.

Proof by induction. Consider one particular iteration with list $\mathcal{L}$ and index i. Let $\mathcal{L}' := \{j \in \mathcal{L} : j < i\}$, and suppose that

$$(*) \quad \min\{x_{\max} - W_j, \min\{x : f_j(x) \text{ minimum}\}\} \le x_j \le \max\{x_{\min}, \rho_j\}$$

for all $j \in \mathcal{L}'$. Let h be the maximal element of $\mathcal{L}'$.

If $h = 0$ or $x_h + W_h \le \min\{x_{\max} - W_i, \rho_i\}$, then x_i is chosen in ③ such that $(*)$ holds also for $j = i$.

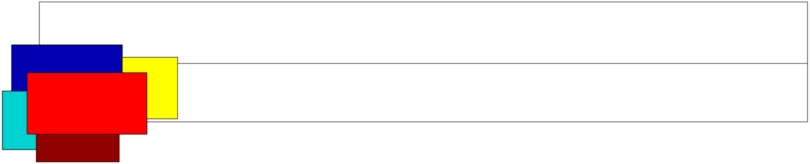
Otherwise we claim that there is an optimum solution $(x_j^*)_{j \in \mathcal{L}' \cup \{i\}}$ of the subproblem defined by $(f_j, W_j)_{j \in \mathcal{L}' \cup \{i\}}$ where $x_h^* + W_h = x_i^*$. This justifies merging h and i in ④.

Let $(x_j^*)_{j \in \mathcal{L}' \cup \{i\}}$ be such an optimum solution. If $x_i^* - W_h \le \rho_h$, then x_h^* can be set to $x_i^* - W_h$ without increasing $f_h(x_h^*)$.

So suppose that $x_i^* > x_h^* + W_h$ and $x_i^* > \rho_h + W_h$. Then $x_i^* > \max\{x_{\min}, \rho_h\} + W_h \ge x_h + W_h > \min\{x_{\max} - W_i, \rho_i\}$, a contradiction as decreasing x_i^* would reduce $f_i(x_i^*)$. □

Problem: zones can be very wide

Example:

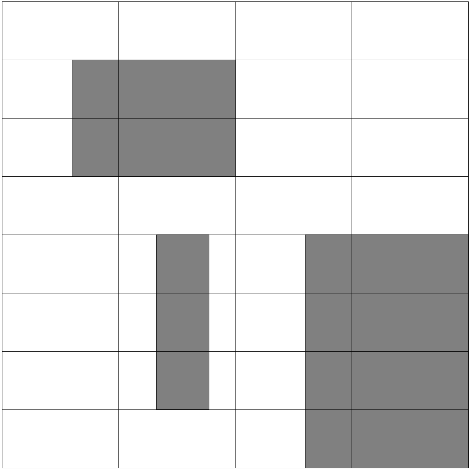


Even if all cells can be placed within the lower zone it is much better to move some of them to the upper zone.

Idea: partition into columns

Subdivide zones into regions.

Example:



An area with 22 zones and 44 regions.

Which cells should be moved where?

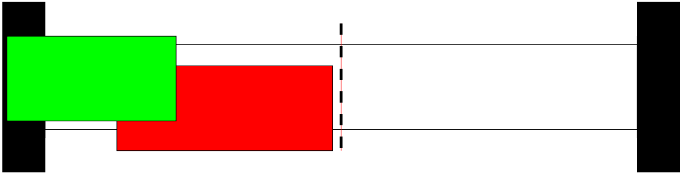
Idea:

Formulation as a minimum cost flow problem, where

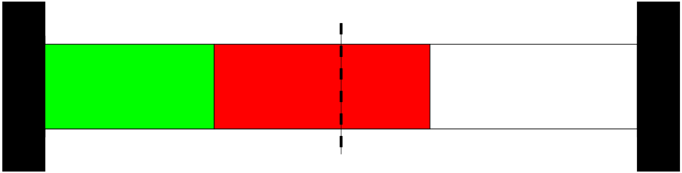
- the vertices are the regions,
- edges connect adjacent regions,
- regions with overload are sources, and
- regions with free capacity are sinks.

But this causes unnecessary movements

Example:



The left region would be a supply region although the two cells could be placed legally with their centers in this region:



Even for a legal placement it is often impossible to assign cells to regions such that no regions is overloaded!

Relaxing constraints

Ideas:

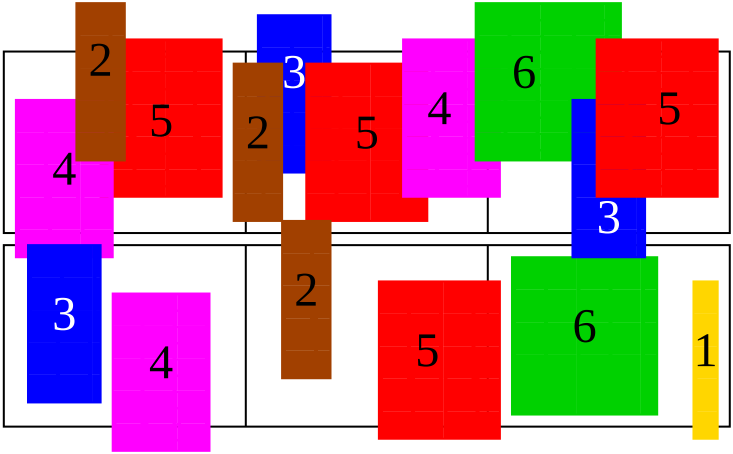
- Only require that at least half of a cell is placed within its region.
- Consider sequences of regions instead of single regions.

Notation:

An interval is a sequence of consecutive regions in the same zone.

- Let $\{A_1, \ldots, A_l\}$ be a set of regions that form a usable zone (ordered from left to right).
- Let $C^i = \{c^i_1, \ldots, c^i_{k_i}\}$ be the set of cells assigned to region A_i, ordered from left to right (for $i \in \{1, \ldots, l\}$).
- Let w denote (total) width.

Example



Supply intervals

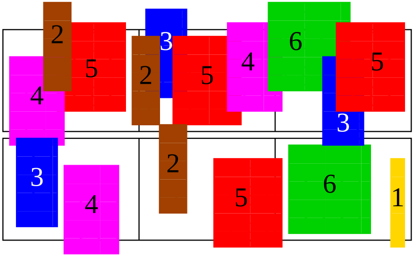
To compute the size of cells that have to be removed from an interval $A_{\mu,\nu}$ we define for $1 \leq \mu \leq \nu \leq l$:

$$s_{\mu,\nu} := \max\left\{0,\ \sum_{i=\mu}^{\nu}\Big(w(C^i) - w(A_i)\Big) - \frac{1}{2}\left(w(c_1^{\mu}) + w(c_{k_\nu}^{\nu})\right)\right\}.$$

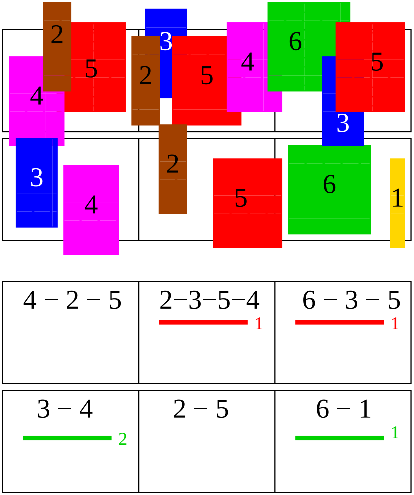
Using these numbers, we define recursively (for $1 \leq \mu \leq \nu \leq l$):

$$\operatorname{supp}(A_{\mu,\nu}) := \max\left\{0,\ s_{\mu,\nu} - \sum_{\substack{\mu \leq \mu' \leq \nu' \leq \nu \\ (\mu,\nu) \neq (\mu',\nu')}} \operatorname{supp}(A_{\mu',\nu'})\right\}.$$

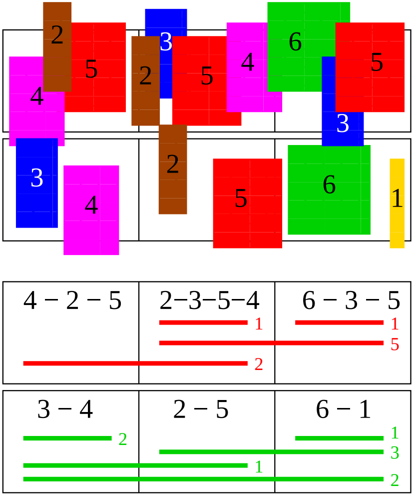
Example: initial placement



Example: supply and demand regions



Example: supply and demand intervals



Demand intervals

To compute the size of cells that can be moved into an interval $A_{\mu,\nu}$ we define for $1 \le \mu \le \nu \le l$:

$$t_{\mu,\nu} := \min\left\{0,\ \sum_{i=\mu}^{\nu}\Big(w(C^i) - w(A_i)\Big) + \frac{1}{2}\Big(w(c^{\mu-1}_{k_{\mu-1}}) + w(c^{\nu+1}_1)\Big)\right\}.$$

Using these numbers, we define recursively (for $1 \le \mu \le \nu \le l$):

$$\mathrm{dem}(A_{\mu,\nu}) := \min\left\{0,\ t_{\mu,\nu} - \sum_{\substack{\mu \le \mu' \le \nu' \le \nu \\ (\mu,\nu) \ne (\mu',\nu')}} \mathrm{dem}(A_{\mu',\nu'})\right\}.$$

Theorem

- No region can be both part of a demand interval and part of a supply interval.
- For $\mu < \kappa \leq \lambda < \nu$ with $\text{supp}(A_{\kappa,\lambda}) > 0$ we have $\text{supp}(A_{\mu,\nu}) = 0$.
- For $\mu < \kappa \leq \lambda < \nu$ with $\text{dem}(A_{\kappa,\lambda}) < 0$ we have $\text{dem}(A_{\mu,\nu}) = 0$.
- The number of supply and demand intervals is at most twice the number of regions.
- They can be computed in linear time.

(Brenner, Vygen [2004])

The minimum cost flow instance

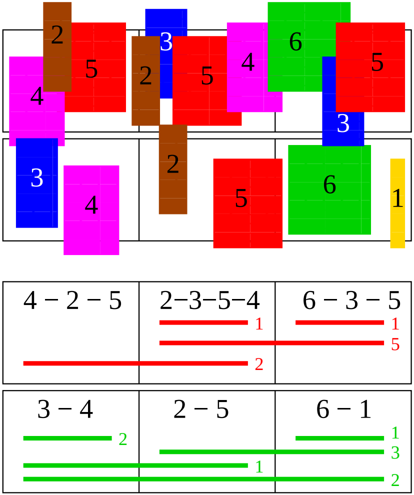
$$
\begin{aligned}
V(G) &:= \{\text{regions, supply intervals, demand intervals}, s, t\} \\
E(G) &:= \{(A, A') : A, A' \text{ adjacent regions}\} \\
&\quad \cup \{(A, A') : A \text{ supply interval}, A' \text{ maximal proper subset of } A\} \\
&\quad \cup \{(A, A') : A' \text{ demand interval}, A \text{ maximal proper subset of } A'\}
\end{aligned}
$$

- For two adjacent regions A and A', let $c(A, A')$ be the expected cost of moving a cell of width 1 from A to A'.
- All other arcs have zero cost. All arcs have infinite capacity.

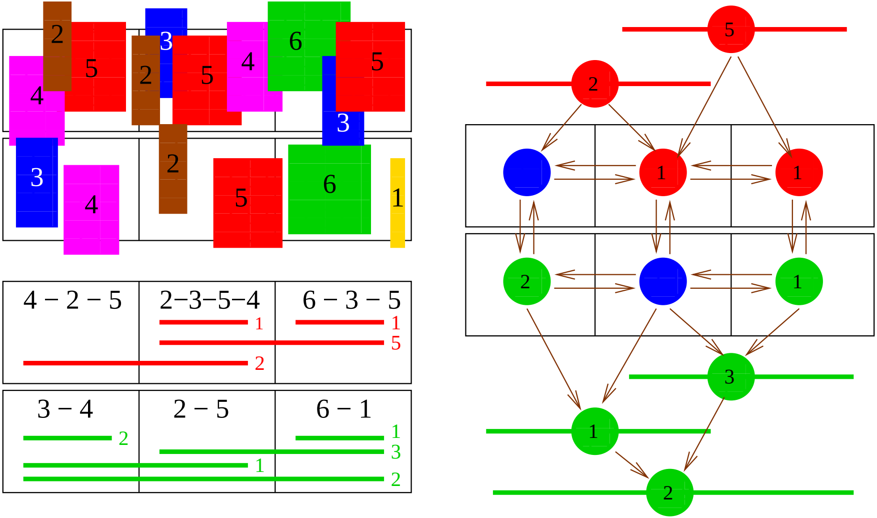
We look for a minimum cost flow f with $f(\delta^+(v)) - f(\delta^-(v)) \geq \text{supp(v)} + \text{dem(v)}$ for all $v \in V(G)$.

This can be done in $O(n^2 \log^2 n)$ time by standard min-cost flow algorithms (Orlin [1993], Vygen [2002])

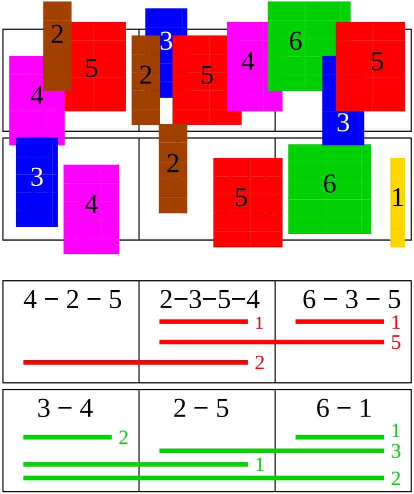
Example: supply and demand intervals

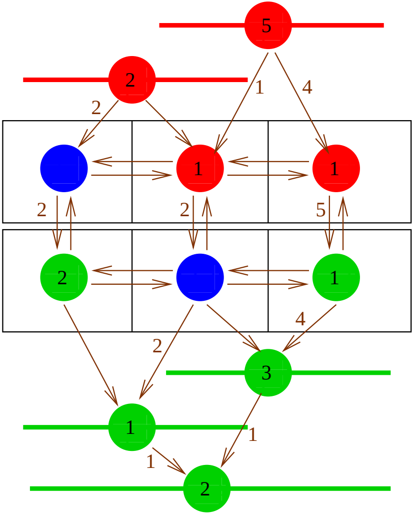


Example: minimum cost flow instance



Example: minimum cost flow





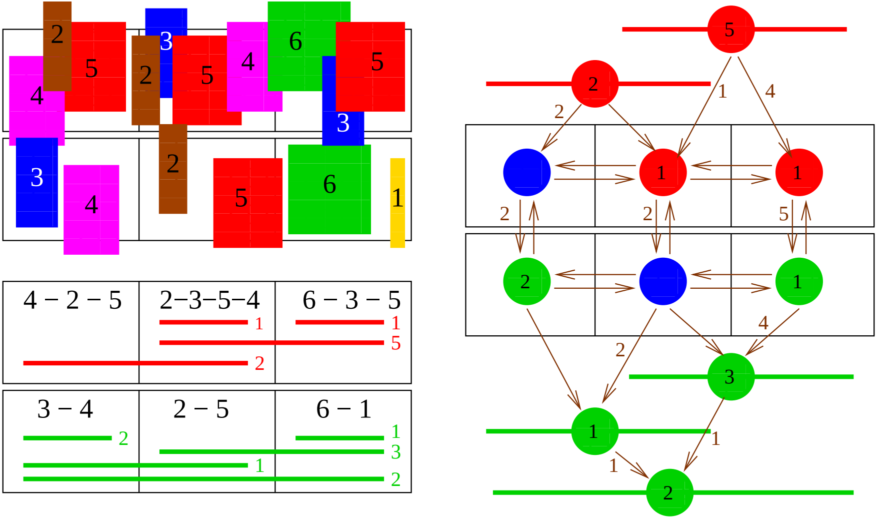
Realization of the flow

By realizing a flow f we mean moving cells of total size $f(A, A')$ from region A to region A' for each pair of neighbours (A, A').

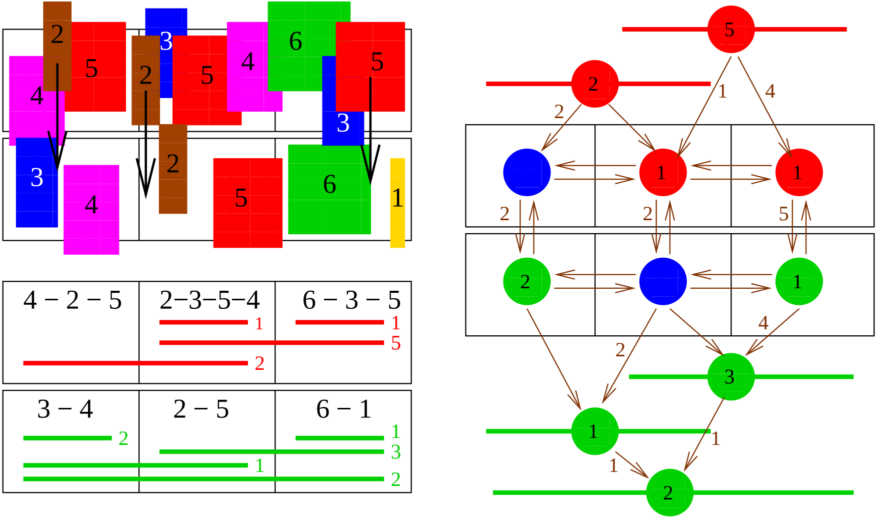
Theorem

- *Let f be a solution to the minimum cost flow instance. Then a realization of f that does not move any leftmost or rightmost cell of a region yields a feasible assignment of the cells.*
- *On non-trivial instances, we cannot decrease the supply- or increase the demand-values without losing this property.*

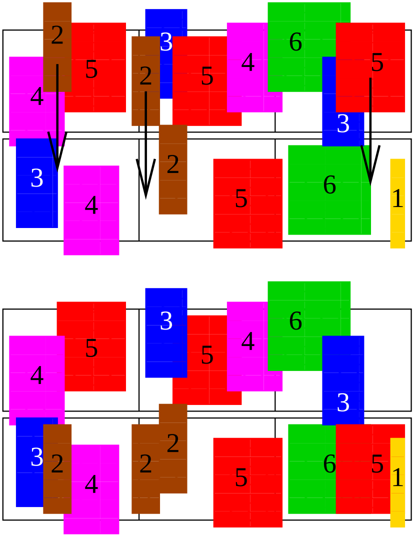
Example: minimum cost flow

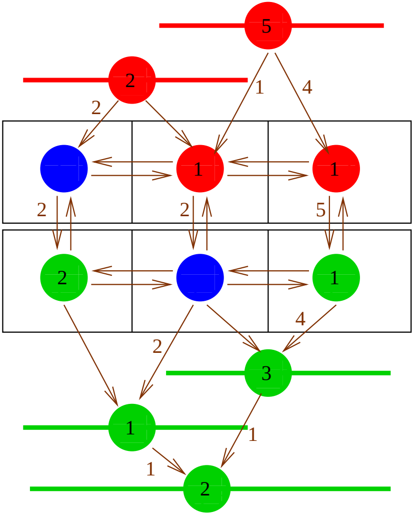


Example: realizing the flow

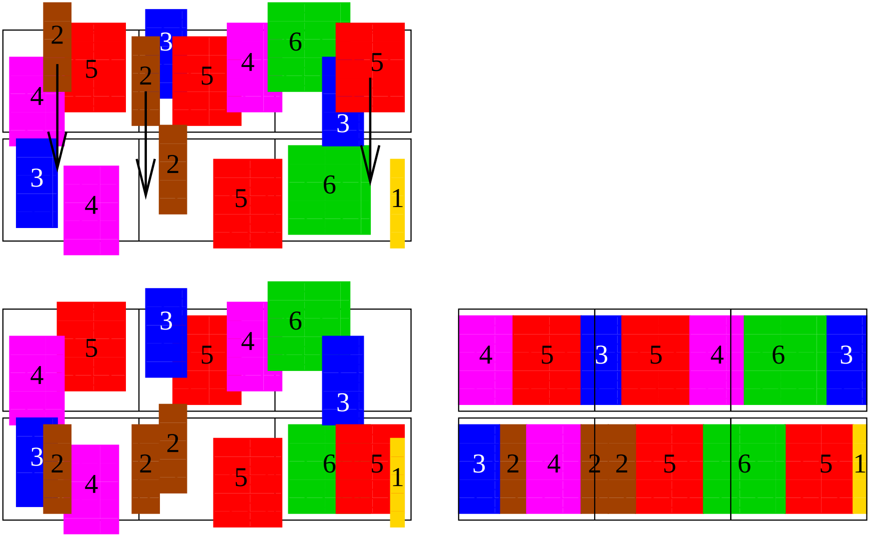


Example: realizing the flow





Example: legal placement



Realization of the flow

Exact realization is in general impossible. We consider approximate realizations:

Theorem

Moving cells between regions such that the total size of cells that leave $A_{\mu,\nu}$ minus the total size of cells that are moved into $A_{\mu,\nu}$ is at least

$$\sum_{i=\mu}^{\nu} \Big(w(C^i) - w(A_i) \Big) - \frac{1}{2} \left(w(c_1^{\mu}) + w(c_{k_\nu}^{\nu}) \right)$$

for each interval $A_{\mu,\nu}$ leads to an assignment of the cells to the regions for which there is a legal placement such that each cell is placed within the region it is assigned to or within a horizontally adjacent region.

Realization of the flow

- The arcs carrying flow form an acyclic subgraph. Consider the vertices in topological order w.r.t. this subgraph.
- The cells to be moved are chosen according to the solution of a Multi-Knapsack Problem (dynamic programming), trying to maintain feasibility.
- We cannot always find cells of appropriate total size
  $\Rightarrow$ There can still be overloads after the realization.

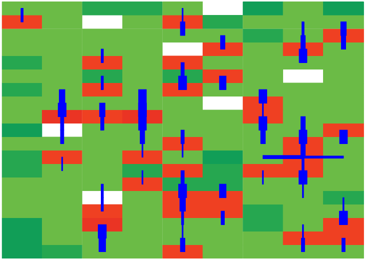
Overall algorithm

- Compute the min-cost flow instance.
- Find a minimum cost flow f.
- Realize f by moving cells along the flow edges.
- Repeat these steps as long as there are overloaded zones. (If necessary, increase column width, decrease demand values.)
- Step 2: Legalize the cells within their zones.
- Step 3: Postoptimization: each step consists of a legal sequence of moves

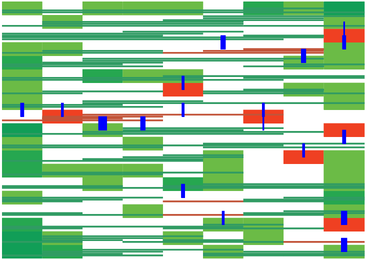
$$c_0 \to c_1 \to \cdots \to c_k \to \text{(place of } c_0 \text{ or free place)}$$

reducing total (squared) movement. Dynamic programming.

Detailed Placement: old and new approach



moving between regions
(old approach)



moving between intervals
(new approach)

rectangles = regions
horizontal lines = intervals
green = demand regions/intervals
red = supply regions/intervals
blue = edges with flow, width proportional to amount of flow

Lower bound: integer linear programming formulation

$$\text{minimize} \sum_{k=1}^{|C|} \sum_{i=1}^{W} \sum_{j=1}^{H} d_{i,j,k} \cdot x_{i,j,k}$$

subject to

$$x_{i,j,k} \in \{0,1\} \qquad \forall\, i = 1,\ldots,W,\ j = 1,\ldots,H,\ k = 1,\ldots,|C|$$

$$\sum_{i=1}^{W} \sum_{j=1}^{H} x_{i,j,k} = 1 \qquad \forall\, k = 1,\ldots,|C|$$

$$\sum_{k=1}^{|C|} \sum_{i'=i-w(c_k)+1}^{i} x_{i',j,k} \leq 1 \qquad \forall\, i = 2,\ldots,W,\ j = 1,\ldots,H$$

where $d_{i,j,k} := (x(c_k) - i)^2 + (y(c_k) - j)^2$

Lower bound: LP relaxation

Let $\delta > 0$ be a usually sufficient radius.

$$\text{minimize} \sum_{k=1}^{|C|} \left(\sum_{i=1}^{W} \sum_{j=1}^{H} d_{i,j,k} \cdot x_{i,j,k} + \delta \cdot x_{\delta,k} \right)$$

subject to

$$0 \leq x_{i,j,k} \leq 1 \qquad \forall\ i = 1, \ldots, W,\ j = 1, \ldots, H,\ k = 1, \ldots, |C|$$

$$\left(\sum_{i=1}^{W} \sum_{j=1}^{H} x_{i,j,k} \right) + x_{\delta,k} = 1 \qquad \forall\ k = 1, \ldots, |C|$$

$$\sum_{k=1}^{|C|} \sum_{i'=i-w(c_k)+1}^{i} x_{i',j,k} \leq 1 \qquad \forall\ i = 2, \ldots, W,\ j = 1, \ldots, H$$

$\Rightarrow$ We can skip all variables $x_{i,j,k}$ with $d_{i,j,k} \geq \delta$.

Integrality gap

- We do not know the integrality gap of this LP.
- However, a simple example shows that it is at least $\frac{6}{5}$ (for $\delta = \infty$).

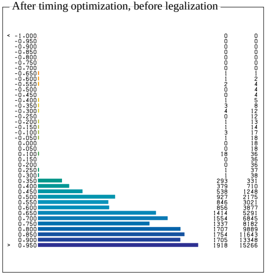
Detailed placement: experimental results

weighted average of squared Euclidean distances in μm:

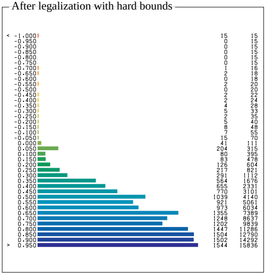
number of objects	old	new	difference (%)	lower bound	gap (%)
72 447	18.06	13.65	24.4	12.37	10.3
72 794	18.67	7.57	59.5	7.34	3.1
284 705	75.18	6.95	90.8	6.25	11.2
411 926	17.32	10.92	37.0	9.85	10.9
1 301 795	8.44	6.08	28.0	5.84	4.1
1 645 691	9.72	5.41	44.3	5.01	7.9
2 395 218	14.95	3.40	77.3	3.08	10.4

lower bound: LP relaxation, solved by CPLEX
HB = hard boundaries between regions (old approach)
SB = soft boundaries between regions (new approach)
maximum total runtime: 40 minutes, 8.5 GB memory

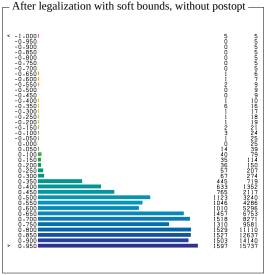
Timing experiments: legalization does not hurt



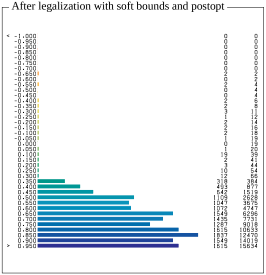
(a) before



(b) HB (old)



(c) SB (new)



(d) after postopt

VLSI routing: task

Instance:

- a number of routing planes
- a set of nets, where each net is a set of pins (terminals)
- a set of shapes for each pin, each of which is a rectangle in a routing plane
- a set of blockage shapes
- rules that tell when two shapes are connected and when they are separated
- timing constraints, information on power, crosstalk, yield, ...

Task:

Compute a feasible routing, i.e. a set of wire shapes for each net, connecting the pins, and separate from blockages and shapes of other nets

- such that all timing constraints are met
- and the (estimated) power consumption is minimized.

VLSI routing: simplified view

Find vertex-disjoint Steiner trees connecting given terminal sets in a 3-dimensional grid graph.

Order of magnitude: 10 million Steiner trees in a graph with 100 billion vertices!

$\longrightarrow$ Even linear-time algorithms are too slow!

Global and detailed routing

VLSI routing is usually performed in three phases:

- Global routing: Eliminates congestion and timing problems on a global level, performs global optimization, and determines corridors for each net to reduce search space in detailed routing
- Detailed routing: Actually constructs wires connecting each net within the corridors obtained from global routing, respecting all design rules necessary for the lithographic processes in fabrication
- Postoptimization: Improve the wiring by spreading and do some postprocessing for more robust manufacturing

Today's designs are huge: 100,000,000,000 vertices in detailed routing, 10,000,000 vertices in global routing. In fact even more, as the underlying grid is an abstraction that does not work anymore.

Key features of global and detailed routing

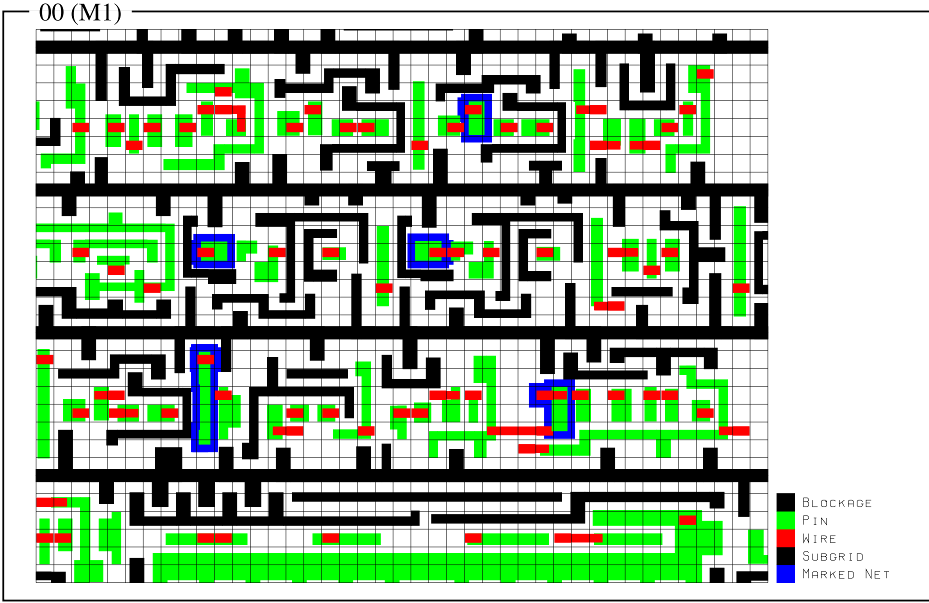
Global routing

- contract regions of approx. 100x100 points to a single vertex
- compute capacities of edges between adjacent regions
- pack Steiner trees with respect to these edge capacities
- do global optimization
- define a detailed routing area for each net according to its Steiner tree

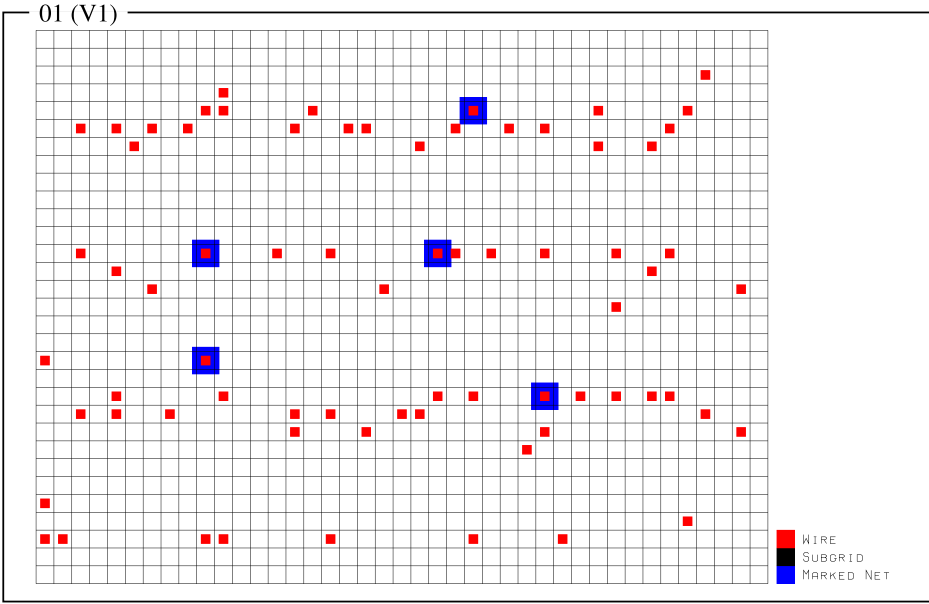
Detailed routing

- route nets sequentially, mainly by shortest path algorithms
- goal-oriented shortest path algorithms
- label intervals rather than single points
- restrict path search to small areas

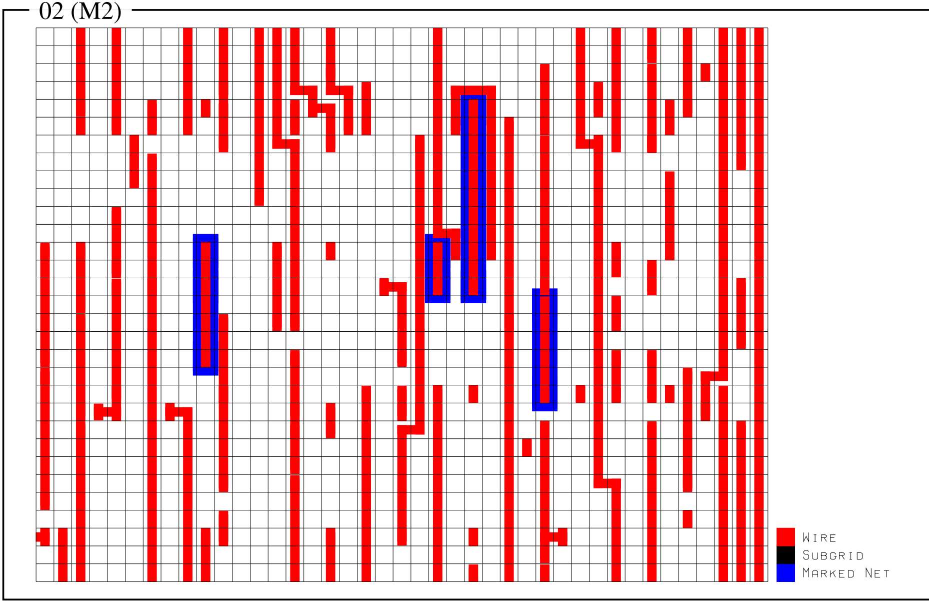
Detailed routing: example



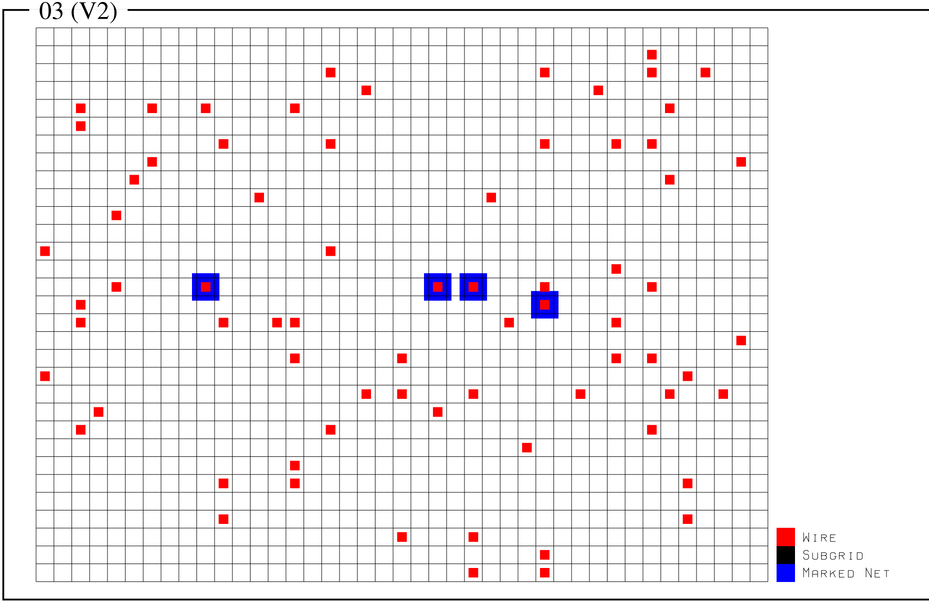
Detailed routing: example



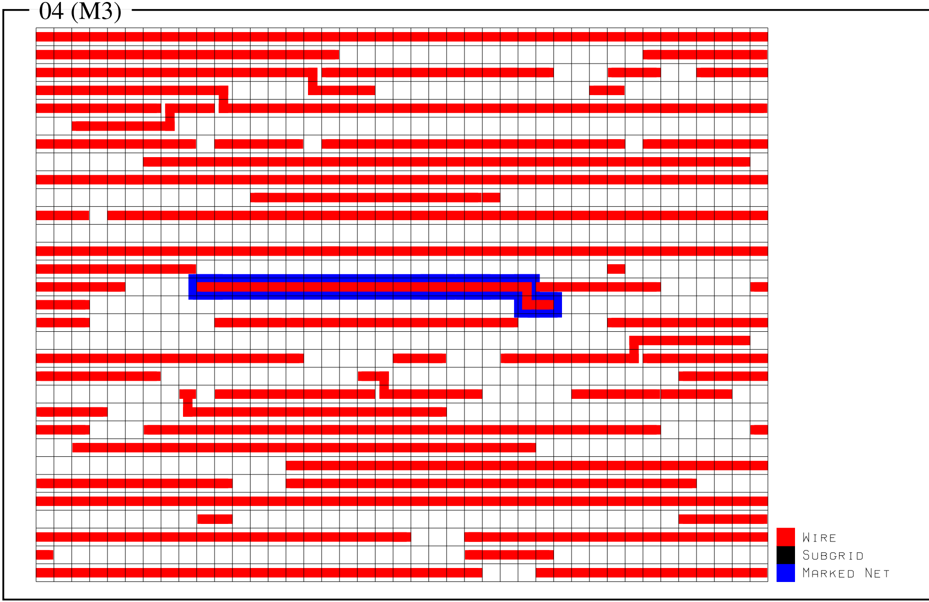
Detailed routing: example



Detailed routing: example

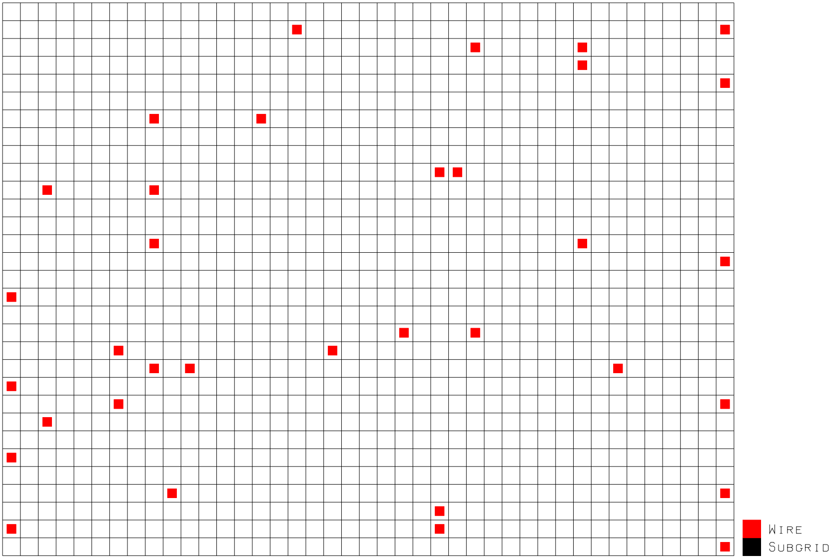


Detailed routing: example

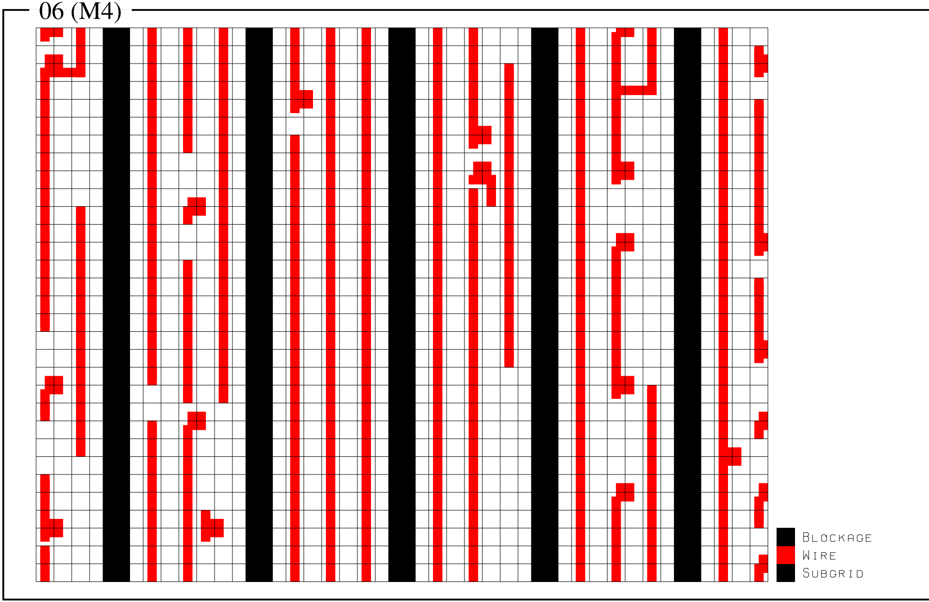


Detailed routing: example

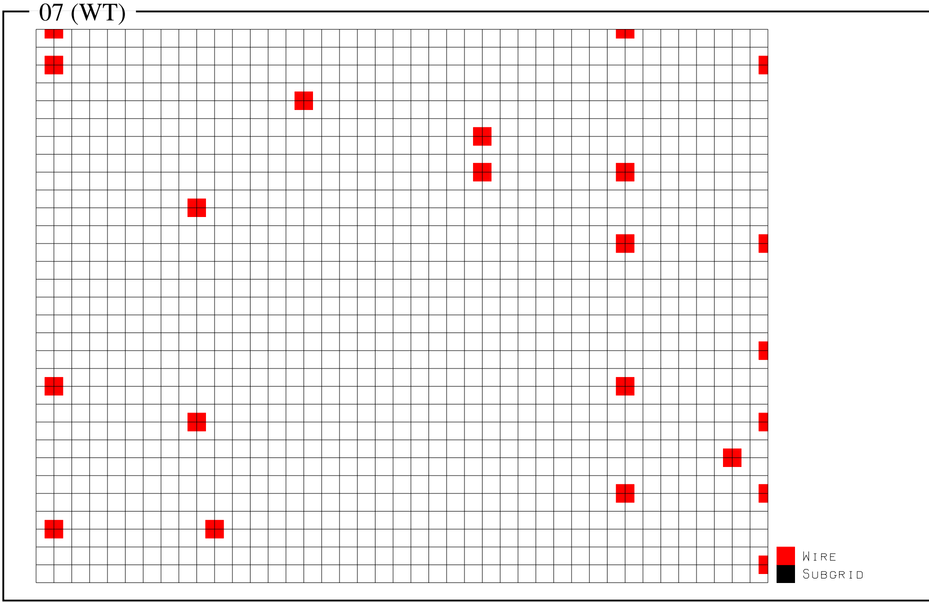
05 (V3)



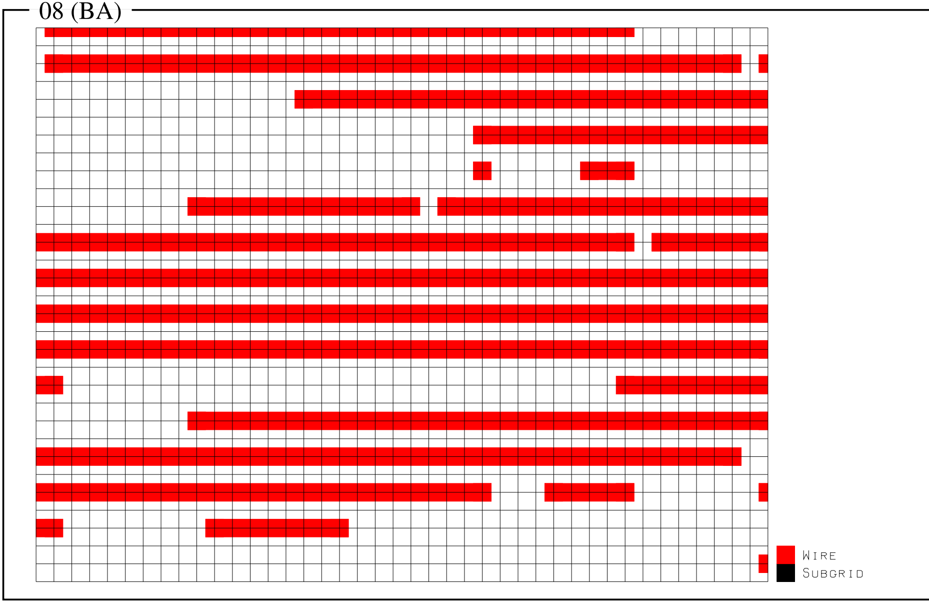
Detailed routing: example



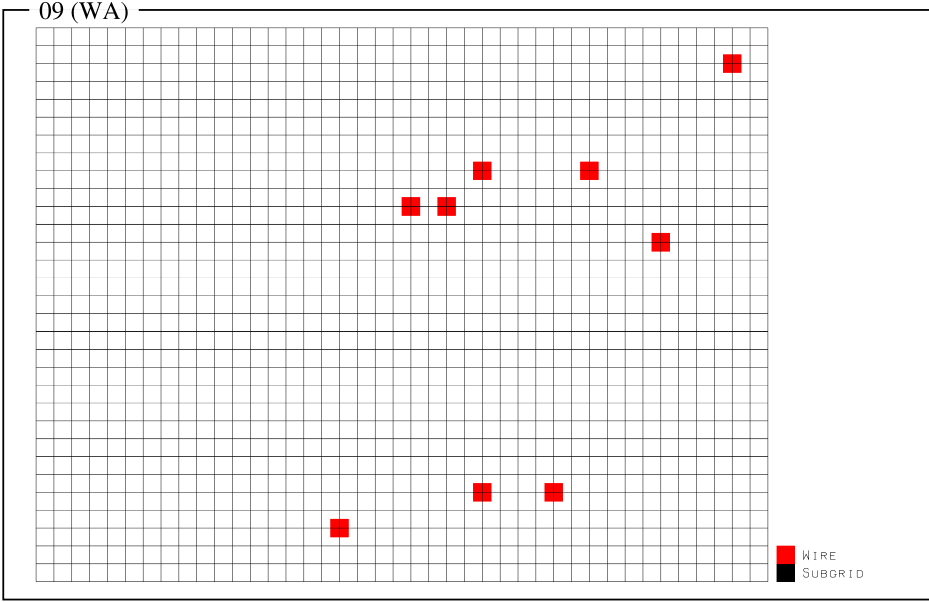
Detailed routing: example



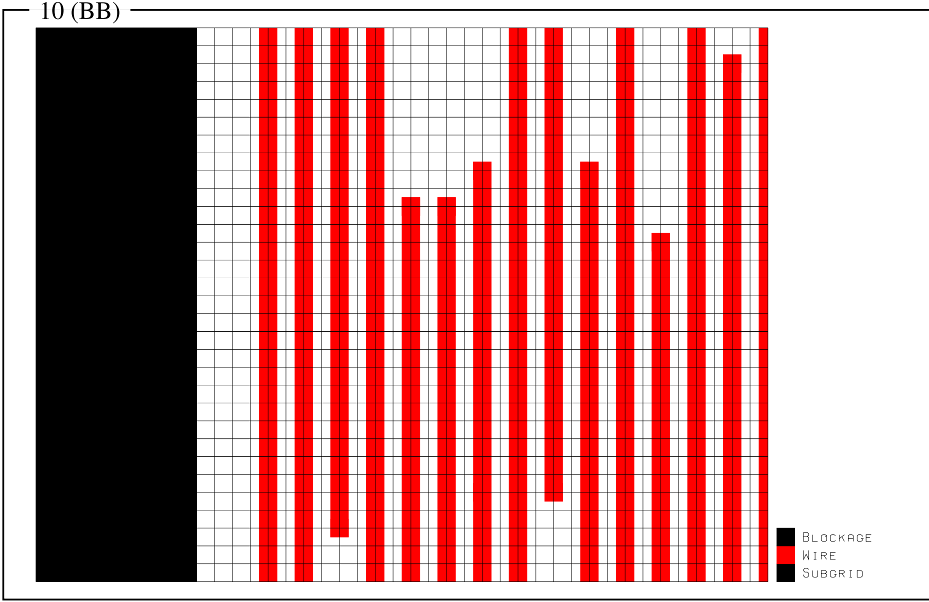
Detailed routing: example



Detailed routing: example



Detailed routing: example

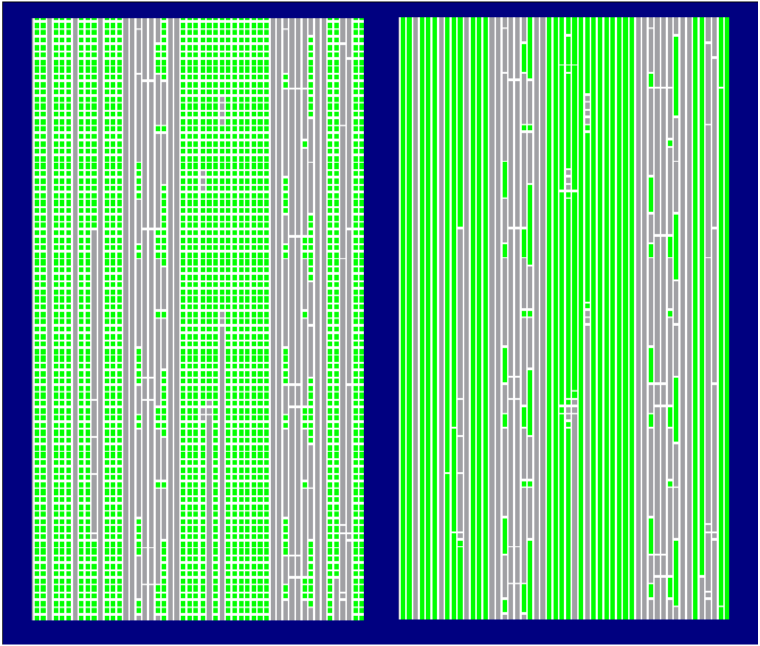


Key features of detailed routing

Detailed routing

- route nets sequentially, subnets by a variant of Dijkstra's algorithm
- restrict each path search to a relatively small area (computed by global routing)
- represent the routing area by a set of intervals (with constant properties)
- label intervals rather than single points
- goal-oriented path search

Detailed routing: intervals



Goal-oriented path search / future cost / feasible potentials

Given a digraph G with arc costs $c : E(G) \to \mathbb{R}_+$.

A function $\pi : V(G) \to \mathbb{R}$ is called a feasible potential if the reduced cost $c_\pi(e) := c(e) + \pi(v) - \pi(w)$ is nonnegative for each $e = (v, w) \in E(G)$.

Let $s, t \in V(G)$. We look for a shortest s-t-path w.r.t. c.

Observation: A shortest s-t-path w.r.t. c is a shortest s-t-path w.r.t. c_π, and vice versa.

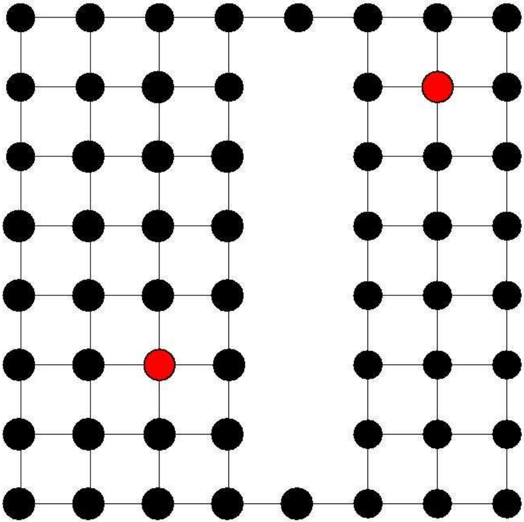
Suppose $\mathcal{L}(x)$ is a lower bound on the distance from x to t, and $\mathcal{L}(v) \leq c(e) + \mathcal{L}(w)$ for each $e = (v, w) \in E(G)$.
Then $\pi(x) := -\mathcal{L}(x)$ is a feasible potential.
$\mathcal{L}(x)$ is also called the future cost at x.

How to compute $\mathcal{L}$

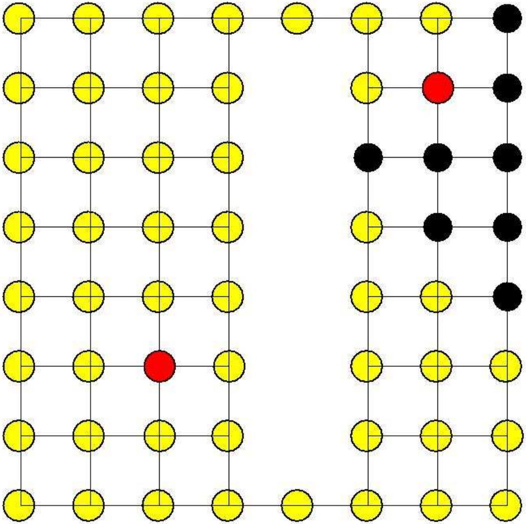
Set $\mathcal{L}(v)$ to the length of a shortest path from v to T in (G', c') where G' is a supergraph of G and $c'(e) \leq c(e)$ for all $e \in E(G)$.

Choose (G', c') such that $\mathcal{L}$ is a good lower bound which can be computed fast.

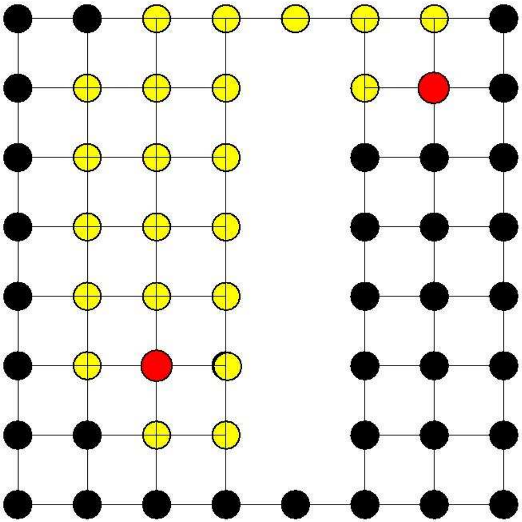
Future cost: example



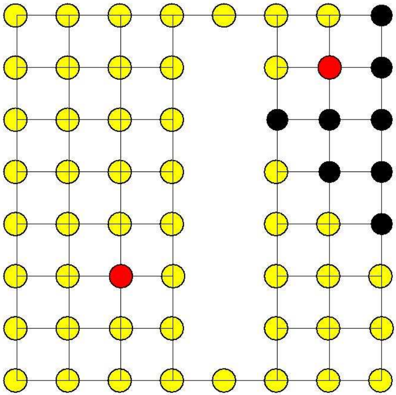
Dijkstra without future cost



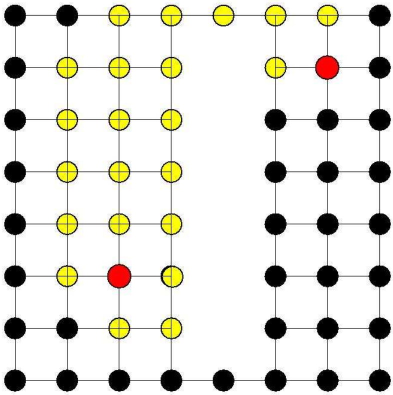
Dijkstra with future cost



Comparison with and without future cost

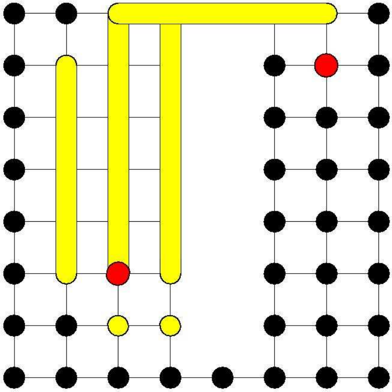


50 points labelled

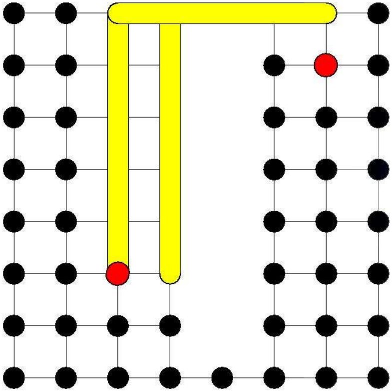


24 points labelled

Comparison with and without future cost



7 intervals labelled



4 intervals labelled

Dijkstra on intervals

- Goal-oriented Dijkstra, labeling intervals rather than single points
- Take the ℓ_1-distance as future cost
- Preprocessing: Voronoi diagram of targets

Theorem

This can be implemented with running time $O((d+1)I \log I)$, where d is the detour (actual length minus lower bound), and I is the number of intervals in the search space.

(Hetzel [1998])

Generalizing Dijkstra's algorithm

Given

- a digraph G with edge lengths $c : E(G) \to \mathbb{R}_+$
- a set $T \subseteq V(G)$
- sets $V_1, V_2, \ldots, V_l \subseteq V(G)$ and $1 \leq k \leq l$ such that $T = \bigcup_{i=1}^{k} V_i$ and $V(G) = \bigcup_{i=1}^{l} V_i$.

we want to determine

$$d(v) := \mathrm{dist}_{(G,c)}(v, T)$$

for all $v \in V(G)$.

Generalizing Dijkstra's algorithm

Given

- a digraph G with edge lengths $c : E(G) \to \mathbb{R}_+$
- a set $T \subseteq V(G)$
- sets $V_1, V_2, \ldots, V_l \subseteq V(G)$ and $1 \leq k \leq l$ such that $T = \bigcup_{i=1}^{k} V_i$ and $V(G) = \bigcup_{i=1}^{l} V_i$.

we want to determine

$$d(v) := \mathrm{dist}_{(G,c)}(v, T)$$

for all $v \in V(G)$.

We label the sets V_i instead of single vertices, by functions $d_i : V_i \to \mathbb{R}_+ \cup \{\infty\}$ with $d_i(u) \geq d(u)$ for all $u \in V_i$.

Generalizing Dijkstra's algorithm

Given

- a digraph G with edge lengths $c : E(G) \to \mathbb{R}_+$
- a set $T \subseteq V(G)$
- sets $V_1, V_2, \dots, V_l \subseteq V(G)$ and $1 \le k \le l$ such that $T = \bigcup_{i=1}^{k} V_i$ and $V(G) = \bigcup_{i=1}^{l} V_i$.

we want to determine

$$d(v) := \mathrm{dist}_{(G,c)}(v, T)$$

for all $v \in V(G)$.

We label the sets V_i instead of single vertices, by functions $d_i : V_i \to \mathbb{R}_+ \cup \{\infty\}$ with $d_i(u) \ge d(u)$ for all $u \in V_i$.

Initially, $d_i(u) := 0$ for $1 \le i \le k$ and $u \in V_i$, and $d_i(u) := \infty$ for $k < i \le l$ and $u \in V_i$. Then we repeatedly apply:

$\textsc{Update}(V_i, V_j)$:
Replace $d_j(u)$ by
$\min\{d_j(u), \min\{d_i(v) + \mathrm{dist}_{(G[V_i \cup V_j], c)}(u, v) : v \in V_i\}\}$
for all $u \in V_j$.

Generalizing Dijkstra's algorithm: optimality conditions

Theorem

Suppose that we have functions $d_1, d_2, \ldots, d_l$ with:

- $d_i(u) = 0$ *for all* $u \in V_i$ *and* $i = 1, \ldots, k$.
- $d_i(u) \geq d(u)$ *for all* $u \in V_i$ *and* $i = 1, \ldots, l$.
- *For each edge* $e = \{u, v\} \in E(G)$ *and each* $i \in \{1, \ldots, l\}$ *with* $u \in V_i$ *there exists a* $j \in \{1, \ldots, l\}$ *with* $v \in V_j$ *and* $d_j(v) \leq d_i(u) + c(e)$.

Then $d(v) = \min\{d_i(v) : i = 1, \ldots, l,\ v \in V_i\}$ *for all* $v \in V(G)$.

Generalizing Dijkstra's algorithm: optimality conditions

Theorem

Suppose that we have functions $d_1, d_2, \ldots, d_l$ *with:*

- $d_i(u) = 0$ *for all* $u \in V_i$ *and* $i = 1, \ldots, k$.
- $d_i(u) \geq d(u)$ *for all* $u \in V_i$ *and* $i = 1, \ldots, l$.
- *For each edge* $e = \{u, v\} \in E(G)$ *and each* $i \in \{1, \ldots, l\}$ *with* $u \in V_i$ *there exists a* $j \in \{1, \ldots, l\}$ *with* $v \in V_j$ *and* $d_j(v) \leq d_i(u) + c(e)$.

Then $d(v) = \min\{d_i(v) : i = 1, \ldots, l,\ v \in V_i\}$ *for all* $v \in V(G)$.

Proof.

Suppose that $d(v) < \min\{d_j(v) : j = 1, \ldots, l,\ v \in V_j\}$; choose v such that $d(v)$ is minimum; in case of ties the shortest v-T-path P shall have minimum number of edges. Let u be the neighbour of v on P.

Generalizing Dijkstra's algorithm: optimality conditions

Theorem

Suppose that we have functions $d_1, d_2, \ldots, d_l$ with:

- $d_i(u) = 0$ *for all* $u \in V_i$ *and* $i = 1, \ldots, k$.
- $d_i(u) \geq d(u)$ *for all* $u \in V_i$ *and* $i = 1, \ldots, l$.
- *For each edge* $e = \{u, v\} \in E(G)$ *and each* $i \in \{1, \ldots, l\}$ *with* $u \in V_i$ *there exists a* $j \in \{1, \ldots, l\}$ *with* $v \in V_j$ *and* $d_j(v) \leq d_i(u) + c(e)$.

Then $d(v) = \min\{d_i(v) : i = 1, \ldots, l,\ v \in V_i\}$ *for all* $v \in V(G)$.

Proof.

Suppose that $d(v) < \min\{d_j(v) : j = 1, \ldots, l,\ v \in V_j\}$; choose v such that $d(v)$ is minimum; in case of ties the shortest v-T-path P shall have minimum number of edges. Let u be the neighbour of v on P. By the choice of v, there exists an $i \in \{1, \ldots, l\}$ with $u \in V_i$ and $d_i(u) + c(\{u, v\}) = d(u) + c(\{u, v\}) = d(v) < \min\{d_j(v) : j = 1, \ldots, l,\ v \in V_j\}$. This is a contradiction. □

(Peyer, Rautenbach and Vygen [2006])

Generalized Dijkstra

Set $d_i(u) := 0$ for $1 \leq i \leq k$ and $u \in V_i$.
Set $d_i(u) := \infty$ for $k < i \leq l$ and $u \in V_i$.
Set $Q := \{1, \ldots, k\}$ and $\text{key}(i) := 0$ for $i = 1, \ldots, k$.
WHILE $Q \neq \emptyset$ DO:

  Choose $i \in Q$ with $\text{key}(i)$ minimum. Set $Q := Q \setminus \{i\}$.
  PROJECT(i).

PROJECT(i):

  Choose $J \subseteq \{1, \ldots, l\} \setminus \{i\}$ such that $\bigcup_{j \in \{i\} \cup J} V_j$ contains all neighbours of V_i.
  FOR $j \in J$:
    UPDATE(V_i, V_j).
    IF $d_j(v)$ changes for some $v \in V_j$,
    THEN let $\text{key}(j)$ be the minimum changed $d_j(v)$, $v \in V_j$, and set $Q := Q \cup \{j\}$.

Generalized Dijkstra: optimality

Theorem

This algorithm produces functions $d_1, d_2, \ldots, d_l$ satisfying the optimality conditions.

Generalized Dijkstra: optimality

Theorem

This algorithm produces functions $d_1, d_2, \ldots, d_l$ satisfying the optimality conditions.

Proof.

The statement is obvious for the first two conditions. Therefore, suppose, for a contradiction, that there exists an edge $e = \{u, v\} \in E(G)$ and an index $i \in \{1, \ldots, l\}$ such that $d_j(v) > d_i(u) + c(e)$ for all $j \in \{1, \ldots, l\}$ with $v \in V_j$.

Generalized Dijkstra: optimality

Theorem

This algorithm produces functions $d_1, d_2, \ldots, d_l$ satisfying the optimality conditions.

Proof.

The statement is obvious for the first two conditions. Therefore, suppose, for a contradiction, that there exists an edge $e = \{u, v\} \in E(G)$ and an index $i \in \{1, \ldots, l\}$ such that $d_j(v) > d_i(u) + c(e)$ for all $j \in \{1, \ldots, l\}$ with $v \in V_j$.

Then $v \notin V_i$. Since $d_i(u) < \infty$, we have $i \in Q$ at some moment.

Generalized Dijkstra: optimality

Theorem

This algorithm produces functions $d_1, d_2, \ldots, d_l$ satisfying the optimality conditions.

Proof.

The statement is obvious for the first two conditions. Therefore, suppose, for a contradiction, that there exists an edge $e = \{u, v\} \in E(G)$ and an index $i \in \{1, \ldots, l\}$ such that $d_j(v) > d_i(u) + c(e)$ for all $j \in \{1, \ldots, l\}$ with $v \in V_j$.

Then $v \notin V_i$. Since $d_i(u) < \infty$, we have $i \in Q$ at some moment.

Consider the last time that the algorithm executes $\textsc{Project}(i)$. Note that d_i does not change after this moment.

Generalized Dijkstra: optimality

Theorem

This algorithm produces functions $d_1, d_2, \ldots, d_l$ satisfying the optimality conditions.

Proof.

The statement is obvious for the first two conditions. Therefore, suppose, for a contradiction, that there exists an edge $e = \{u, v\} \in E(G)$ and an index $i \in \{1, \ldots, l\}$ such that $d_j(v) > d_i(u) + c(e)$ for all $j \in \{1, \ldots, l\}$ with $v \in V_j$.

Then $v \notin V_i$. Since $d_i(u) < \infty$, we have $i \in Q$ at some moment.

Consider the last time that the algorithm executes $\textsc{Project}(i)$. Note that d_i does not change after this moment.

As v is a neighbour of $u \in V_i$, there is some $j \in J$ with $v \in V_j$ and $\textsc{Update}(V_i, V_j)$ ensures

$$d_j(v) \leq d_i(u) + \mathrm{dist}_{(G[V_i \cup V_j], c)}(u, v) \leq d_i(u) + c(e).$$

As $d_j(v)$ never increases, this is a contradiction. □

(Peyer, Rautenbach and Vygen [2006])

Generalized Dijkstra: running time

- If we implement Q by a Fibonacci heap, the running time is is $O(n(\log l + p))$, where p is the time for one Project operation and n is the number of iterations.
- Since every $i \in \{1, \ldots, k\}$ enters Q exactly once and every $i \in \{k+1, \ldots, l\}$ enters Q at most $|V_i|$ times, we only have the bound $n \leq k + \sum_{i=k+1}^{l} |V_i|$ in general.
- If $V_1, \ldots, V_l$ is a partition of $V(G)$ into one-element sets, then this is the standard algorithm with running time $O(m + n \log n)$, where $n = |V(G)|$ and $m = |E(G)|$.
- Much faster for special graphs, in particular grid graphs
- Sorting $V_{k+1}, \ldots, V_l$ such that $c((u,v)) > 0$ for $(u,v) \in E(G) \cap ((V_i \times (V_j \setminus V_i)) \cup ((V_i \setminus V_j) \times V_j))$ and $i < j$ gives that each $i \in \{k+1, \ldots, l\}$ enters Q at most once for each key.

Modeling routing by grid graphs

Let G_0 be the infinite 3-dimensional grid graph, i.e. $V(G_0) = \mathbb{Z}^3$, and
$E(G_0) = \{\{(x,y,z),(x',y',z')\} : |x-x'|+|y-y'|+|z-z'| = 1\}$.
We assume that for each $z \in \mathbb{Z}$ there are three constants $c_{z,1}, c_{z,2}, c_z \in \mathbb{R}$ such that

$$\begin{aligned} c(\{(x,y,z),(x+1,y,z)\}) &= c_{z,1}, \\ c(\{(x,y,z),(x,y+1,z)\}) &= c_{z,2}, \text{ and} \\ c(\{(x,y,z),(x,y,z+1)\}) &= c_z \end{aligned}$$

for all $x, y \in \mathbb{Z}$, This reflects higher costs for vias and jogs and in access planes.
We look for shortest paths w.r.t. c in induced subgraphs of G_0.

Generalized Dijkstra on grids

Let G be an induced subgraph of the infinite 3-dimensional grid. Write $V(G)$ as the union of rectangles $V_1, \ldots, V_l$ such that each has $O(\log l)$ neighbours.
Assume that the number of different edge weights is constant.
Then:

- the number of iterations is $O(l)$
- the functions d_i can be stored in constant space
- an Update operation takes constant time
- the cardinality of the set J to be considered in the Project operation is $O(\log l)$

$\Rightarrow$ Running time of $O(l \log l)$

(Peyer, Rautenbach and Vygen [2006])

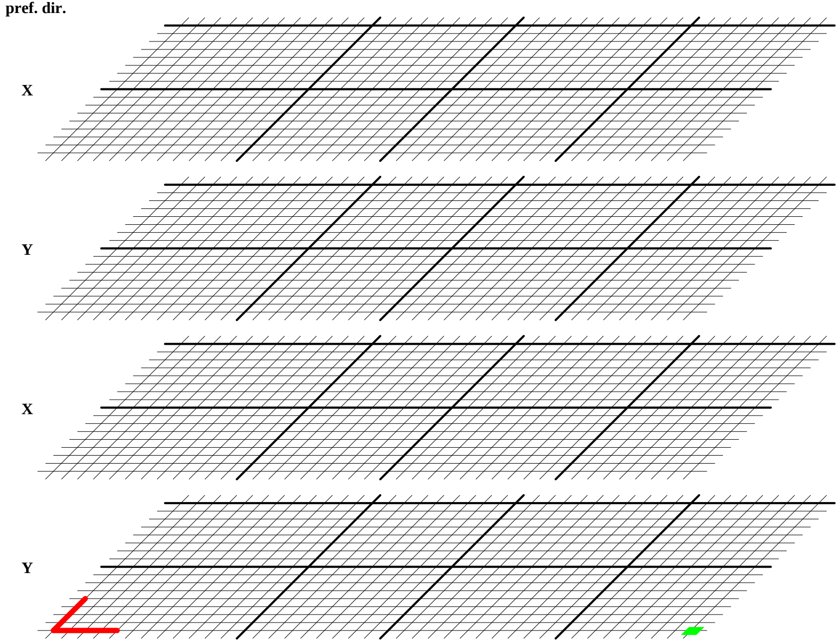
Generalized Dijkstra for accurate future costs

- Consider a supergraph G' of the graph G representing the routing area, such that G' can be decomposed into few rectangles (and in which distances are not much shorter).
- Apply Generalized Dijkstra to G', labeling these rectangles.
- As $d(v) = \mathrm{dist}_{(G',c)}(v, T) \leq \mathrm{dist}_{(G,c)}(v, T)$, the numbers $d(v)$ serve as future cost for shortest path computation in G.

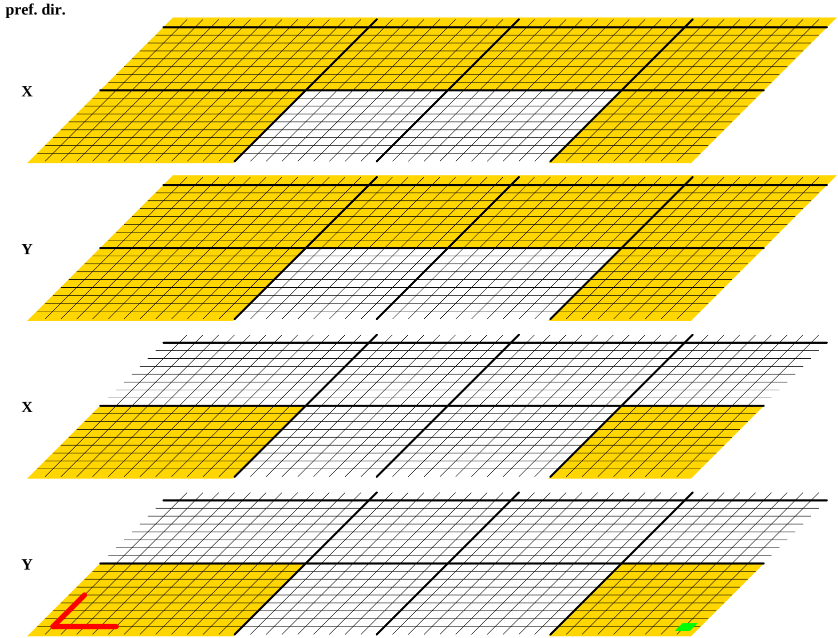
Example for accurate future costs

- four layers
- alternating preference directions
- we look for a path from a (green) source to a (red) target
- edge cost 1 in preference direction
- edge cost 4 in orthogonal direction
- edge cost 7 for vias

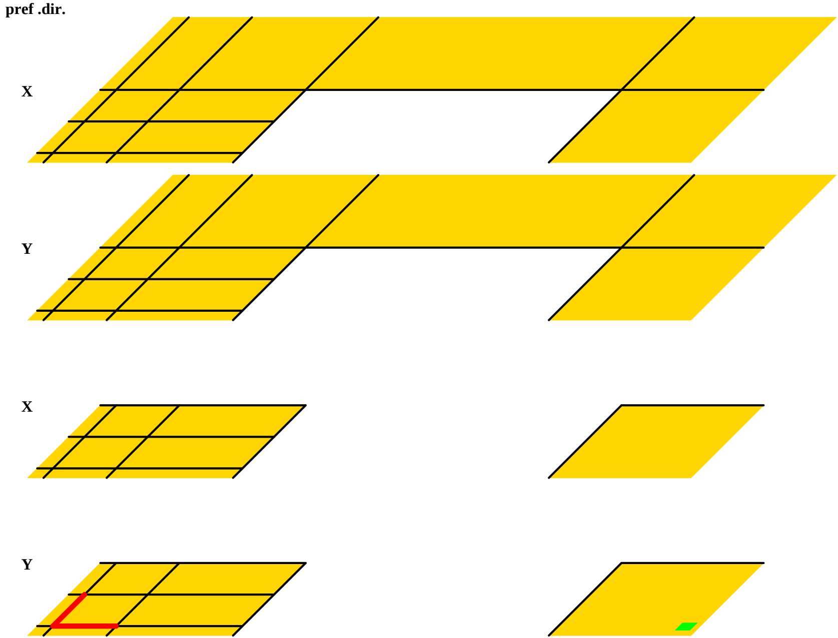
Example: local routing grid



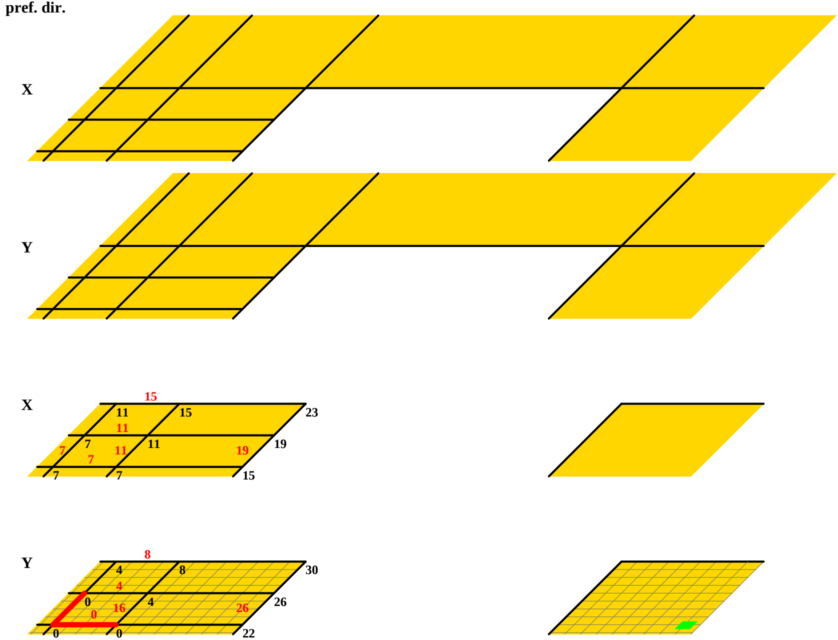
Example: global routing corrdidors



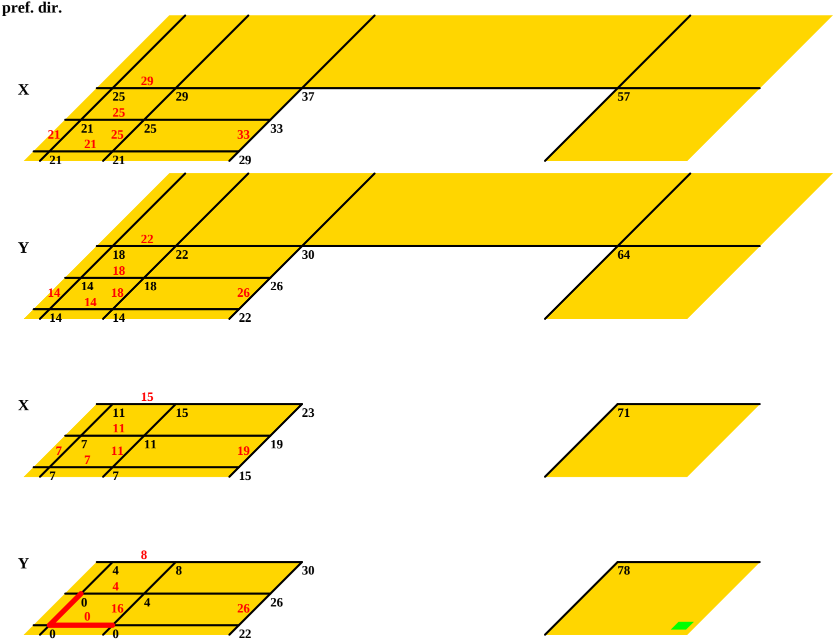
Example: Hanan grid



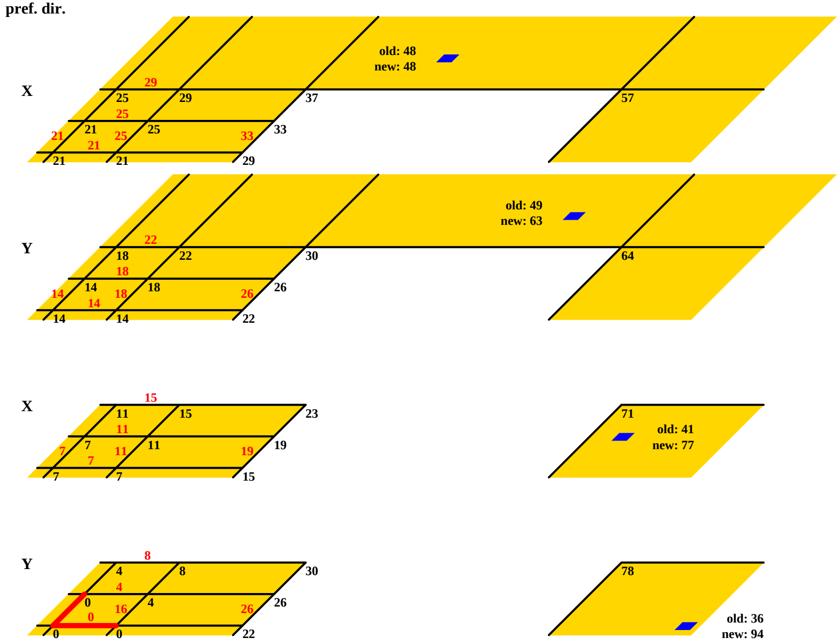
Example: Generalized Dijkstra



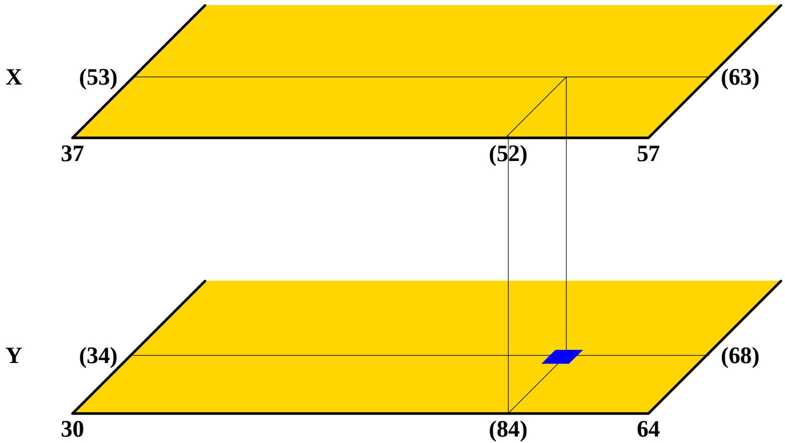
Example: Generalized Dijkstra



Example: old (ℓ_1-distance) versus new future cost



Example: how to compute the future cost



$$63 = \min\begin{pmatrix} 34 + 15 \times 4, \\ 68 + 5 \times 4, \\ 84 + 4 \times 1, \\ 53 + 15 \times 1 + 7, \\ 63 + 5 \times 1 + 7, \\ 52 + 4 \times 1 + 7 \end{pmatrix}$$

Global routing: simplified problem formulation

Instance:

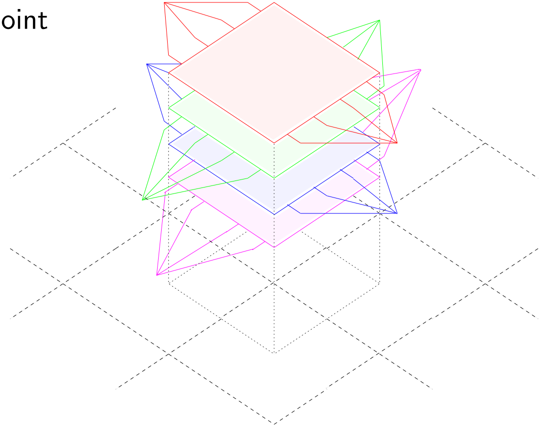
- a global routing (grid) graph with edge capacities
- a set of nets, each consisting of a set of vertices (terminals)

Task: find a Steiner tree for each net such that

- the edge capacities are respected,
- some objective function (e.g., netlength, yield, or power) is optimized,
- and the timing constraints are met.

Capacity estimation

- First route very short nets (within one region or two adjacent regions).
- Then consider each pair of adjacent regions. Assume that planes are mainly used in preferred wiring direction, alternatingly horizontal and vertical.
- Consider the following instance of the edge-disjoint paths problem:



Capacity estimation: fast augmenting path heuristic

- Apply a very fast multicommodity flow heuristic, exploiting the structure of the instances. (Müller [2002])
- Each augmenting path requires only $O(k)$ constant-time bit pattern operations, where k is the number of edges orthogonal to the preferred wiring direction.
- Heuristic finds a feasible integral multicommodity flow solution whose value is approx. 90% of the (weak) max-flow upper bound.
- Complete chip with 300 million paths in 15 minutes (Goldberg-Tarjan runs 1 month)

Global routing is hard

Restriction: EDGE-DISJOINT PATHS PROBLEM
Given a pair of graphs (G, H), find a family $(P_h)_{h \in E(H)}$ of edge-disjoint paths in G such that $P_h + h$ is a circuit for each $h \in E(H)$.
NP-complete even if

- G is a rectangle (Raghavan [1986])
- G is a rectangle, and we allow shortest paths only (Vygen [1994])
- G is a rectangle, and $G + H$ is Eulerian (Marx [2002])
- G is series-parallel (Nishizeki, Vygen, Zhou [2001])
- G is directed and planar, H consists of two sets of parallel arcs (Müller [2002])

Fractional relaxation: Multicommodity Flow Problem

Instance:

- an undirected graph G with capacities $u : E(G) \to \mathbb{Z}_+$ and lengths $l : E(G) \to \mathbb{R}$
- a family $\mathcal{N}$ of nets (terminal pairs) with demands $w : \mathcal{N} \to \mathbb{Z}_+$ and weights $c : \mathcal{N} \to \mathbb{Z}_+$

Task: Find a flow f_N for each N of value $w(N)$ such that

$$\sum_{N \in \mathcal{N}} w_N f_N(e) \leq u(e) \qquad \text{for } e \in E(G),$$

and

$$\sum_{N \in \mathcal{N}} c_N \sum_{e \in E(G)} l(e) f_N(e) \qquad \text{is minimum.}$$

In many applications: congestion costs — heavily used edges are more expensive

Examples: traffic flows, VLSI routing

Global routing: positive results

- There is a combinatorial fully polynomial approximation scheme for the MULTICOMMODITY FLOW PROBLEM (Sharokhi, Matula [1990], Leighton, Makedon, Plotkin, Stein, Tardos, Tragoudas [1991], Plotkin, Shmoys, Tardos [1991], Radzik [1995], Young [1995], Grigoriadis, Khachiyan [1996], Garg, Könemann [1998], Fleischer [2000], Karakostas [2002])
- If edges have sufficient capacity, randomized rounding can be applied to get an integral solution violating capacity constraints only slightly (Raghavan, Thompson [1987,1991], Raghavan [1988])
- This can be applied to Steiner trees instead of paths and works efficiently for large global routing instances (Albrecht [2001])

But this does not take timing constraints and global objectives (power consumption, yield) into account.

Timing constraints in routing

The delay on each path must not exceed its bound. A path can be viewed as a sequence of nets. The delay of a net depends on its electrical capacitance.

- first assume delay-optimal Steiner trees for all nets
- distribute slack optimally (Albrecht, Korte, Schietke, Vygen [2000], Held [2001]) to all nets for which sufficient slack is available. For these nets the slack defines a maximum tolerable capacitance
- call the remaining nets (with no or insufficient slack assigned) critical
- compute weights and a bound on the weighted sum of capacitances for each path containing a critical net

Main design objectives in routing

minimize power consumption

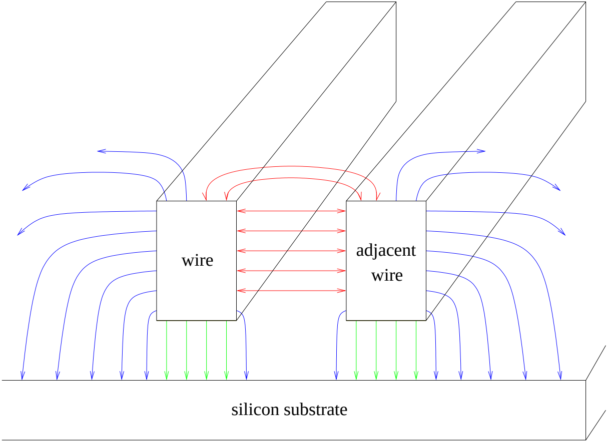
- active power consumption roughly proportional to the electrical capacitance, weighted by switching activity
- leakage power and capacitance of cells not influenced by routing.
- capacitance of nets depends on length, width, plane, and existence of neighbour wires

minimize cost

- minimize number of masks (number of routing planes), maximize yield (spreading), minimize design effort

Capacitance estimation

- area capacitance (parallel plate capacitor) – proportional to length times width
- fringing capacitance – proportional to length
- coupling capacitance – proportional to length if adjacent wire exists



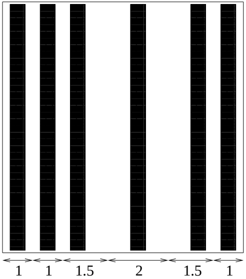
Modeling coupling capacitance

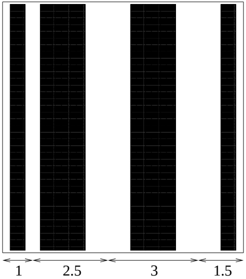
Assume linear dependence on distance to adjacent wire between the following bounds:

- minimum distance $\rightarrow$ coupling capacitance $\frac{1}{2}v(e)$
- minimum distance plus 1 $\rightarrow$ coupling capacitance 0

Example:

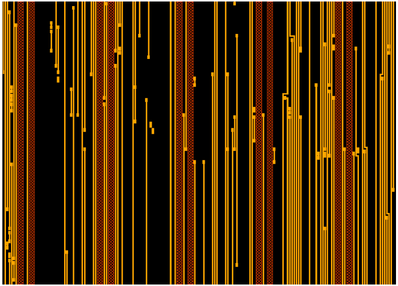
global routing edge e of capacity $u(e) = 8$, with two global routing solutions:



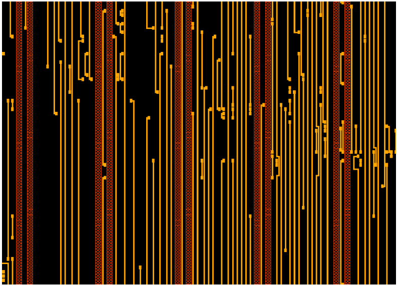


- Left: six unit width wires use 6–12 channels. Coupling capacitance $v(e)$ times $1, 1, \frac{1}{2}, 0, \frac{1}{2}, 1$
- Right: two unit width wires and two double width wires use 6–10 channels. Coupling capacitance $v(e)$ times $1, \frac{1}{2}, 0, \frac{1}{2}$

Example: result with and without spreading



minimizing netlength



maximizing yield

Global Routing Problem

Instance:

- An undirected graph G with edge capacities $u : E(G) \to \mathbb{R}_+$,
- a set $\mathcal{N}$ of nets and a set $\mathcal{Y}_N$ of feasible Steiner trees for each net N,
- wire widths $w : E(G) \times \mathcal{N} \to \mathbb{R}_+$,
  extra space $s : E(G) \times \mathcal{N} \to \mathbb{R}_+$,
- maximum capacitances $l : E(G) \times \mathcal{N} \to \mathbb{R}_+$ and
  coupling contributions $v : E(G) \times \mathcal{N} \to \mathbb{R}_+$.
- A family $\mathcal{M}$ of subsets of $\mathcal{N}$ with $\mathcal{N} \in \mathcal{M}$ with capacitance bounds $U : \mathcal{M} \to \mathbb{R}_+$ and weights $c(M, N) \in \mathbb{R}_+$ for $N \in M \in \mathcal{M}$.

Global Routing Problem

Task:

Find a Steiner tree $Y_N \in \mathcal{Y}_N$ and numbers $0 \leq y_{e,N} \leq 1$ for each $N \in \mathcal{N}$ and $e \in E(Y_N)$, such that

$$\sum_{N \in \mathcal{N}: e \in E(Y_N)} (w(e,N) + s(e,N) y_{e,N}) \leq u(e)$$

for each edge $e \in E(G)$,

$$\sum_{N \in M} c(M,N) \sum_{e \in E(Y_N)} (l(e,N) - v(e,N) y_{e,N}) \leq U(M)$$

for $M \in \mathcal{M}$, and such that

$$\sum_{N \in \mathcal{N}} c(\mathcal{N},N) \sum_{e \in E(Y_N)} (l(e,N) - v(e,N) y_{e,N})$$

is minimum.

LP relaxation of the Global Routing Problem

$\min \lambda$ subject to

$$\sum_{Y \in \mathcal{Y}_N} x_{N,Y} = 1 \qquad (N \in \mathcal{N})$$

$$\sum_{N \in M} c(M,N) \left(\sum_{Y \in \mathcal{Y}_N} \sum_{e \in E(Y)} l(e,N) x_{N,Y} - \sum_{e \in E(G)} v(e,N) y_{e,N} \right) \leq \lambda U(M) \qquad (M \in \mathcal{M})$$

$$\sum_{N \in \mathcal{N}} \left(\sum_{Y \in \mathcal{Y}_N : e \in E(Y)} w(e,N) x_{N,Y} + s(e,N) y_{e,N} \right) \leq \lambda u(e) \qquad (e \in E(G))$$

$$y_{e,N} \leq \sum_{Y \in \mathcal{Y}_N : e \in E(Y)} x_{N,Y} \qquad (e \in E(G), N \in \mathcal{N})$$

$$y_{e,N} \geq 0 \qquad (e \in E(G), N \in \mathcal{N})$$

$$x_{N,Y} \geq 0 \qquad (N \in \mathcal{N}, Y \in \mathcal{Y}_N)$$

The dual LP

$$\max \sum_{N \in \mathcal{N}} z_N \text{ subject to}$$

$$\sum_{e \in E(G)} u(e)\omega_e + \sum_{M \in \mathcal{M}} U(M)\mu_M = 1$$

$$z_N \leq \sum_{e \in E(Y)} \left(l(e,N) \sum_{M \in \mathcal{M} : N \in M} c(M,N)\mu_M + w(e,N)\omega_e - \chi_{e,N} \right) \qquad (N \in \mathcal{N}, Y \in \mathcal{Y}_N)$$

$$\chi_{e,N} \geq v(e,N) \sum_{M \in \mathcal{M} : N \in M} c(M,N)\mu_M - s(e,N)\omega_e \qquad (e \in E(G), N \in \mathcal{N})$$

$$\chi_{e,N} \geq 0 \qquad (e \in E(G), N \in \mathcal{N})$$

$$\omega_e \geq 0 \qquad (e \in E(G))$$

$$\mu_M \geq 0 \qquad (M \in \mathcal{M})$$

Edge costs

Let $\omega_e \in \mathbb{R}_+ (e \in E(G))$ and $\mu_M \in \mathbb{R}_+ (M \in \mathcal{M})$, and let us define edge costs

$$\begin{aligned}\psi_{N,e} \quad := \quad \min_{\delta\in\{0,1\}} \Bigg(& (l(e,N) \,-\, \delta v(e,N)) \sum_{M\in\mathcal{M}:N\in M} c(M,N)\mu_M \\ & + (w(e,N) + \delta s(e,N))\omega_e \Bigg).\end{aligned}$$

Then

$$\frac{\displaystyle\sum_{N\in\mathcal{N}} \min_{Y\in\mathcal{Y}_N} \sum_{e\in E(Y)} \psi_{N,e}}{\displaystyle\sum_{e\in E(G)} u(e)\omega_e + \sum_{M\in\mathcal{M}} U(M)\mu_M}$$

is a lower bound on the optimum LP value.

The fractional global routing algorithm

Input: An instance of the Global Routing Problem with $\mathcal{N} = \{1, \ldots, k\}$, $t \in \mathbb{N}$, $\epsilon \in \mathbb{R}_+$.
Output: Feasible solutions to the primal and dual LP.

Initialize:
Set $\omega_e := \frac{1}{u(e)}$ for $e \in E(G)$ and $\mu_M := \frac{1}{U(M)}$ for $M \in \mathcal{M}$.
Set $x_{i,Y} := 0$ for $i := 1, \ldots, k$, $Y \in \mathcal{Y}_i$.
Set $y_{e,i} := 0$ for $e \in E(G)$ and $i := 1, \ldots, k$.
Set $Y_i := \emptyset$ for $i := 1, \ldots, k$.

(Main Loop)

TakeAverage:
Set $x_{i,Y} := \frac{1}{t} x_{i,Y}$ for $i = 1, \ldots, k$ and $Y \in \mathcal{Y}_i$.
Set $y_{e,i} := \frac{1}{t} y_{e,i}$ for $e \in E(G)$ and $i = 1, \ldots, k$.

The fractional global routing algorithm: main loop

For $p := 1$ to t do:
- For $i := 1$ to k do:
  - Let $\psi_{i,e}$ be defined as above.
  - Let $Y_i \in \mathcal{Y}_i$ with $\sum_{e \in E(Y_i)} \psi_{i,e}$ minimum.
  - UpdateVariables:
  - Set $x_{i,Y_i} := x_{i,Y_i} + 1$.
  - For $e \in E(Y_i)$ do:
    - If $v(e,i) \sum_{M \in \mathcal{M}: i \in M} c(M,i)\mu_M < s(e,N)\omega_e$ then $\delta_e := 0$ else $\delta_e := 1$.
    - $y_{e,i} := y_{e,i} + \delta_e$.
    - $\omega_e := \omega_e e^{\epsilon \frac{w(i,e)+\delta_e s(e,i)}{u(e)}}$.
    - For $M \in \mathcal{M}$ with $i \in M$ do:
      - $\mu_M := \mu_M e^{\epsilon c(M,i) \frac{l(e,i)-\delta_e v(e,i)}{U(M)}}$.

Global routing algorithm: main theorem

This is a fully polynomial approximation scheme for the primal-dual pair of LPs

Enhanced global routing algorithm

- Compute new Steiner tree for net N only if previous one is longer than $(1+\epsilon_1)z_N$, where z_N is a continuously updated lower bound.
- If a new Steiner tree has to be computed, a $(1+\epsilon_2)$-optimal one suffices.

Theorem

Let λ^* be the optimum LP value and $t\epsilon\lambda^* > \log(m+|\mathcal{M}|)$. Then the algorithm computes feasible primal and dual solutions, whose values differ by at most a factor

$$\frac{\epsilon(1+\epsilon)(1+\epsilon_1)(1+\epsilon_2)}{\epsilon(1-\epsilon(1+\epsilon)(1+\epsilon_1)(1+\epsilon_2)\lambda^*)\left(1-\frac{\log(m+|\mathcal{M}|)}{t\epsilon\lambda^*}\right)}$$

By choosing $\epsilon, \epsilon_1, \epsilon_2, t$ appropriately, we get a $(1+\epsilon_0)$-optimal solution in $\frac{2\ln(m+|\mathcal{M}|)}{\epsilon_0^2}$ iterations, for any $\epsilon_0 > 0$.

(Vygen [2004])

The fractional global routing algorithm (enhanced)

For $p := 1$ to t do:

For $i := 1$ to k do:

Let $\psi_{i,e}$ be defined as above.

If $Y_i = \emptyset$ or $\sum_{e \in E(Y_i)} \psi_{i,e} > (1+\epsilon_1) z_i$ then:

Let $Y_i \in \mathcal{Y}_i$ with

$\sum_{e \in E(Y_i)} \psi_{i,e} \leq (1+\epsilon_2) \min_{Y \in \mathcal{Y}_i} \sum_{e \in E(Y)} \psi_{i,e}$.

Set $z_i := \sum_{e \in E(Y_i)} \psi_{i,e}$.

UpdateVariables

For $M \in \mathcal{M}$ and $j \in M$ do:

$z_j := z_j + (1+\epsilon_2) \mathcal{L}_j c(M,j) \left(\mu_M^{new} - \mu_M^{old} \right)$

Randomized rounding

Let (x, y, λ) be a fractional solution to the primal LP. Compute a rounded solution $(\hat{x}, \hat{y}, \hat{\lambda})$ as follows:

- choose $Y \in \mathcal{Y}_N$ as Y_N with probability $x_{N,Y}$ (independently for all $N \in \mathcal{N}$); then set $\hat{x}_{N,Y_N} := 1$ and $\hat{x}_{N,Y} := 0$ for $Y \in \mathcal{Y}_N \setminus \{Y_N\}$.
- Set $\hat{y}_{N,e} := \frac{y_{N,e}}{\sum_{Y \in \mathcal{Y}_N : e \in E(Y)} x_{N,Y}}$ if $e \in E(Y_N)$ and $\hat{y}_{N,e} := 0$ otherwise.
- Choose $\hat{\lambda}$ minimum possible such that $(\hat{x}, \hat{y}, \hat{\lambda})$ is a feasible solution to the primal LP.

Let $\Lambda \leq \frac{U(M)}{c(M,N)\sum_{e \in E(Y)}(l(e,N)+v(e,N))}$ for $N \in M \in \mathcal{M}$ and $Y \in \mathcal{Y}_N$, and $\Lambda \leq \frac{u(e)}{w(e,N)+s(e,N)}$ for $N \in \mathcal{N}$. Moreover, suppose that $|\mathcal{M}| + |E(G)| < e^{\lambda\Lambda}$.

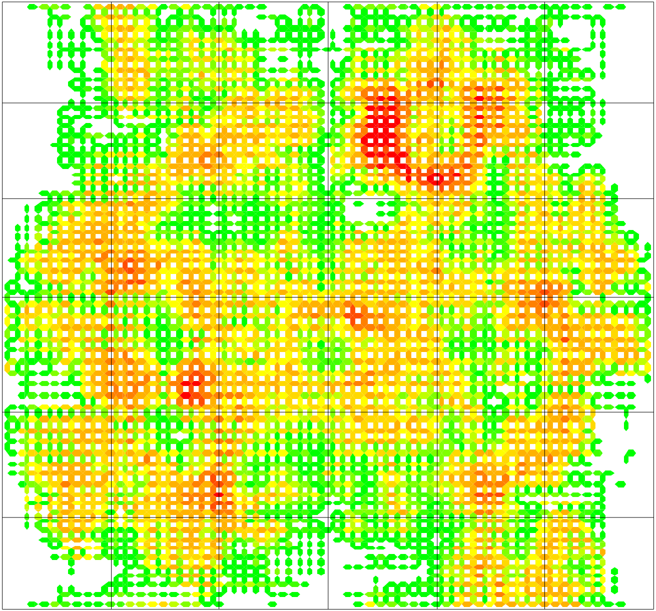
Then $\hat{\lambda} \leq \lambda \left(1 + (e-1)\sqrt{\frac{\ln(|\mathcal{M}|+|E(G)|}{\lambda\Lambda}}\right)$.

(Vygen [2004])

The global routing algorithm in practice

- In practice, results are much better than theoretical performance guarantees. Usually 10–20 iterations suffice.
- Only few upper bounds are violated; these are corrected easily by *ripup-and-reroute*.
- Detailed routing can realize the solution well, due to excellent capacity estimations.
- Small integrality gap and approximate dual solution implies that an infeasibility proof can be found for most infeasible instances.
- First global routing algorithm to take into account coupling, timing, and power consumption directly. Provably near-optimal.

Example: global routing congestion map



Connection to traffic flows

The global routing problem is equivalent to routing traffic flow

- with hard capacity bounds on edges (streets)
- without capacity bounds on vertices
- in a static setting (flow continuously repeated over time)
- with bounds on weighted sums of travel times
- and with the following transit time model: the transit time along an edge (latency) is constant up to $x\%$ congestion and grows linearly between $x\%$ and 100% congestion

Algorithm is equivalent to selfish routing but with taxes (exponential dependance on congestion)

Future cost in global routing

The edge costs

$$\psi_{N,e} \quad := \quad \min_{\delta \in \{0,1\}} \Bigg((l(e,N) - \delta v(e,N)) \sum_{M \in \mathcal{M} : N \in M} c(M,N)\mu_M + (w(e,N) + \delta s(e,N))\omega_e \Bigg)$$

consist of a geometrical length part and a congestion part.

The future cost considers geometrical length only (ℓ_1-distance).

A suitable weighting of the geometrical part can speed up the algorithm considerably.

Future cost: observations in practice

Electrical characteristics or defect sensitivities are encoded in the geometrical part of the edge costs.

Thus future cost quality can degrade with increasing differences of these values

- over different planes
- between a spreaded and an unspreaded wire on the same plane

and also with increasing congestion.

Example

Edge lengths for yield optimization in a recent technology:

- M5 – M7: 1.0 (1 channel extra space), 1.37 (no extra space)
- M1 – M4: 1.76 (1 channel extra space), 2.73 (no extra space)

Future cost and RC-delay

Let N be a two-terminal net, and e an edge on some path connecting these terminals.

The contribution of e to the RC-delay on N is

$$r_e \left(\frac{c_e}{2} + C_e \right),$$

where

- r_e is the resistance of the edge e,
- c_e is its capacitance, and
- C_e is the downstream capacitance "hanging behind" e on the path.

For approximating C_e, the future cost can be used.

Yield analysis: critical area

Consider faults caused by particles with size distribution

$$f(r) := \begin{cases} 0, r < r_0 \\ \frac{c}{r^3}, r \geq r_0 \end{cases}$$

for some $r_0 \in \mathbb{R}_+$ smaller than the smallest possible particle that can cause a fault, and c such that $\int_0^\infty f(r)\mathrm{d}r = 1$.

Then the critical area w.r.t. extra material faults on plane z is

$$\mathrm{CA}^z_{em} := \int_x \int_y \int_{t_{em}(x,y,z)}^\infty f(r)\mathrm{d}r\mathrm{d}y\mathrm{d}x,$$

where $t_{em}(x, y, z)$ is the smallest size of a particle that causes an extra material fault at location (x, y, z).

Yield analysis: expected number of faults

Weighted sum of critical areas is used to estimate the number of extra material faults per chip:

$$\mathsf{F}_{em} := \sum_z w_{em}^z \mathsf{CA}_{em}^z$$

Analogously define the number of miss material faults on wire planes, F_{wm}, and on via planes, F_{vm}.

Define the estimated total number of faults per chip as $\mathsf{F} := \mathsf{F}_{em} + \mathsf{F}_{wm} + \mathsf{F}_{vm}$.

The percentage of chips without a fault from one of the above classes is estimated by

$$e^{-\mathsf{F}}.$$

The complement $1 - e^{-\mathsf{F}}$ is called the wiring yield loss.

Experimental results: the testbed

Chip	Technology	Image Size (in 1000 channels)	# Nets (in 1000)
Edgar	Cu08	40 x 40	772
Hannelore	Cu08	36 x 33	140
Paul	Cu08	24 x 24	68
Monika	Cu11	35 x 35	1502
Ralf	Cu11	26 x 26	1349
Garry	Cu11	26 x 26	827
Heidi	Cu11	23 x 23	777
Elena	Cu11	19 x 19	421
Lotti	Cu11	14 x 14	132
Dieter	Cu11	19 x 19	58
Ingo	Cu11	19 x 19	58
Bill	Cu11	26 x 26	11
Roland	Cu11	16 x 16	11
Joachim	SA27E	14 x 14	288

(Müller [2006])

Experimental results: total running time (in seconds)

Chip	2D-GR	3D-GR, Netl. Opt.		3D-GR, Yield Opt.	
Edgar	63421	57096	(–10.0%)	91215	(+43.8%)
Hannelore	10847	12766	(+17.7%)	14552	(+34.2%)
Paul	4076	6019	(+47.7%)	5413	(+32.8%)
Monika	65064	62560	(–3.8%)	92995	(+42.9%)
Ralf	61473	55506	(–9.7%)	116221	(+89.1%)
Garry	48382	40399	(–16.5%)	70615	(+46.0%)
Heidi	31431	25936	(–17.5%)	45150	(+43.6%)
Elena	21197	20924	(–1.3%)	38327	(+80.8%)
Lotti	3978	5425	(+36.4%)	5895	(+48.2%)
Dieter	12705	11063	(–12.9%)	11152	(–12.2%)
Ingo	20733	11125	(–46.3%)	15661	(–24.5%)
Bill	4994	3924	(–21.4%)	5448	(+9.1%)
Roland	2528	3025	(+19.7%)	4200	(+66.1%)
Joachim	7432	9024	(+21.4%)	9526	(+28.2%)
Total	358591	325343	(–9.3%)	526819	(+46.9%)

Experimental results: wirelength

Chip	2D-GR	3D-GR, Netl. Opt.		3D-GR, Yield Opt.	
Edgar	211.656 m	212.022 m	(+0.2%)	214.162 m	(+1.2%)
Hannelore	30.110 m	30.239 m	(+0.4%)	31.006 m	(+3.0%)
Paul	9.888 m	9.903 m	(+0.2%)	9.999 m	(+1.1%)
Monika	263.936 m	264.123 m	(+0.1%)	273.793 m	(+3.7%)
Ralf	234.747 m	234.169 m	(–0.2%)	242.094 m	(+3.1%)
Garry	221.950 m	221.989 m	(+0.0%)	227.186 m	(+2.4%)
Heidi	150.775 m	150.863 m	(+0.1%)	153.837 m	(+2.0%)
Elena	92.234 m	92.226 m	(–0.0%)	94.511 m	(+2.5%)
Lotti	18.208 m	18.230 m	(+0.1%)	18.679 m	(+2.6%)
Dieter	13.226 m	13.329 m	(+0.8%)	13.574 m	(+2.6%)
Ingo	13.199 m	13.285 m	(+0.7%)	13.482 m	(+2.1%)
Bill	23.312 m	23.356 m	(+0.2%)	23.542 m	(+1.0%)
Roland	17.351 m	17.397 m	(+0.3%)	17.595 m	(+1.4%)
Joachim	62.250 m	62.432 m	(+0.3%)	63.721 m	(+2.4%)
Total	1363.675 m	1364.404 m	(+0.1%)	1398.024 m	(+2.5%)

Experimental results: number of vias

Chip	2D-GR	3D-GR, Netl. Opt.		3D-GR, Yield Opt.	
Edgar	6151607	6114859	(–0.6%)	8302895	(+35.0%)
Hannelore	795855	804856	(+1.1%)	1096198	(+37.7%)
Paul	474376	449112	(–5.3%)	606733	(+27.9%)
Monika	9335637	8916882	(–4.5%)	12409600	(+32.9%)
Ralf	10314838	9250179	(–10.3%)	12945468	(+25.5%)
Garry	6018048	5740090	(–4.6%)	8555230	(+42.2%)
Heidi	5030429	4790479	(–4.8%)	6821014	(+35.6%)
Elena	2738929	2689970	(–1.8%)	3486325	(+27.3%)
Lotti	669582	649336	(–3.0%)	797861	(+19.2%)
Dieter	426860	421537	(–1.2%)	537206	(+25.9%)
Ingo	441647	429608	(–2.7%)	586823	(+32.9%)
Bill	103812	101471	(–2.3%)	185742	(+78.9%)
Roland	95847	102976	(+7.4%)	191646	(+99.9%)
Joachim	1924130	1937133	(+0.7%)	2026975	(+5.3%)
Total	44594645	42470739	(–4.8%)	58623918	(+31.5%)

Experimental results: expected number of faults per chip

Chip	2D-GR	3D-GR, Netl. Opt.		3D-GR, Yield Opt.	
Edgar	0.09780	0.10493	(+7.3%)	0.08586	(–12.2%)
Hannelore	0.01396	0.01543	(+10.6%)	0.01027	(–26.4%)
Paul	0.00502	0.00568	(+13.2%)	0.00402	(–19.9%)
Monika	0.08744	0.09505	(+8.7%)	0.08055	(–7.9%)
Ralf	0.07832	0.08920	(+13.9%)	0.07361	(–6.0%)
Garry	0.07224	0.08017	(+11.0%)	0.06714	(–7.1%)
Heidi	0.05351	0.05804	(+8.5%)	0.04965	(–7.2%)
Elena	0.03167	0.03314	(+4.6%)	0.02966	(–6.3%)
Lotti	0.00658	0.00688	(+4.5%)	0.00575	(–12.6%)
Dieter	0.00482	0.00516	(+7.2%)	0.00416	(–13.6%)
Ingo	0.00457	0.00505	(+10.4%)	0.00392	(–14.2%)
Bill	0.00707	0.00833	(+17.8%)	0.00376	(–46.8%)
Roland	0.00563	0.00605	(+7.5%)	0.00396	(–29.7%)
Joachim	0.00432	0.00440	(+1.9%)	0.00431	(–0.1%)
Total	0.47336	0.51791	(+9.4%)	0.42703	(–9.8%)

The wiring yield loss is reduced by more than 10 % for most chips.

Conclusion

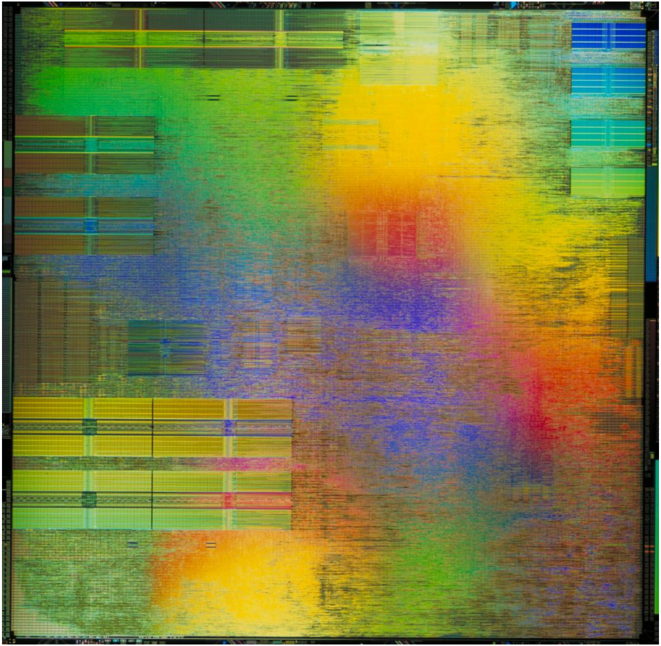
- VLSI design is probably the richest application area of combinatorial optimization
- Many classical and new combinatorial optimization problems are directly applied
- Rapidly developping technology poses constantly new problems
- Instances sizes pose challenges to algorithm design and implementation
- Placement and routing are studied for decades, but...
- ...there is still a lot to be done.

Conclusion

- VLSI design is probably the richest application area of combinatorial optimization
- Many classical and new combinatorial optimization problems are directly applied
- Rapidly developping technology poses constantly new problems
- Instances sizes pose challenges to algorithm design and implementation
- Placement and routing are studied for decades, but...
- ...there is still a lot to be done.

Better chips by better mathematics

Thank you!



Some references for further reading

- Brenner, U. [2005]: A faster polynomial algorithm for the unbalanced Hitchcock transportation problem. Report No. 05954, Research Institute for Discrete Mathematics, University of Bonn
- Brenner, U., and Struzyna, M. [2005]: Faster and Better Global Placement by a New Transportation Problem. Proceedings of the 42nd IEEE/ACM Design Automation Conference (2005), 591–596
- Brenner, U., and Vygen, J. [2001]: Worst-case ratios of networks in the rectilinear plane. Networks 38 (2001), 126–139
- Brenner, U., and Vygen, J. [2004]: Legalizing a Placement with Minimum Total Movement. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 23 (2004), 1597–1613
- Hetzel, A. [1998]: A sequential detailed router for huge grid graphs. Design, Automation and Test in Europe, Proceedings, IEEE 1998, 332–338
- Korte, B., Rautenbach, D., and Vygen, J. [2006]: BonnTools: Mathematical Innovation for Layout and Timing Closure of Systems on a Chip. Report No. 06962, Research Institute for Discrete Mathematics, University of Bonn, 2006

- Korte, B., and Vygen, J. [2006]: Combinatorial Optimization: Theory and Algorithms. Third edition. Springer, Berlin 2006
- Müller, D.: Optimizing yield in global routing. Proceedings of the IEEE International Conference on Computer-Aided Design (2006), to appear
- Peyer, S., Rautenbach, D., and Vygen, J.: Generalizing Dijkstra's algorithm for shortest paths in huge graphs. Manuscript 2006.
- Vygen, J. [1997]: Algorithms for large-scale flat placement. Proceedings of the 34th IEEE/ACM Design Automation Conference (1997), 746–751
- Vygen, J. [2002]: New theoretical results on quadratic placement. Integration, a VLSI Journal, to appear
- Vygen, J. [2004]: Near-optimum global routing with coupling, delay bounds, and power consumption. Proceedings of the 10th International IPCO Conference; Springer, Berlin 2004, pp. 308–324
- Vygen, J. [2005]: Geometric quadrisection in linear time, with application to VLSI placement. Discrete Optimization 2 (2005), 362–390