An analytical placement tool for the Coriolis toolchain

Gabriel Gouvine

Laboratoire d'informatique de Paris 6
Telecom Paristech

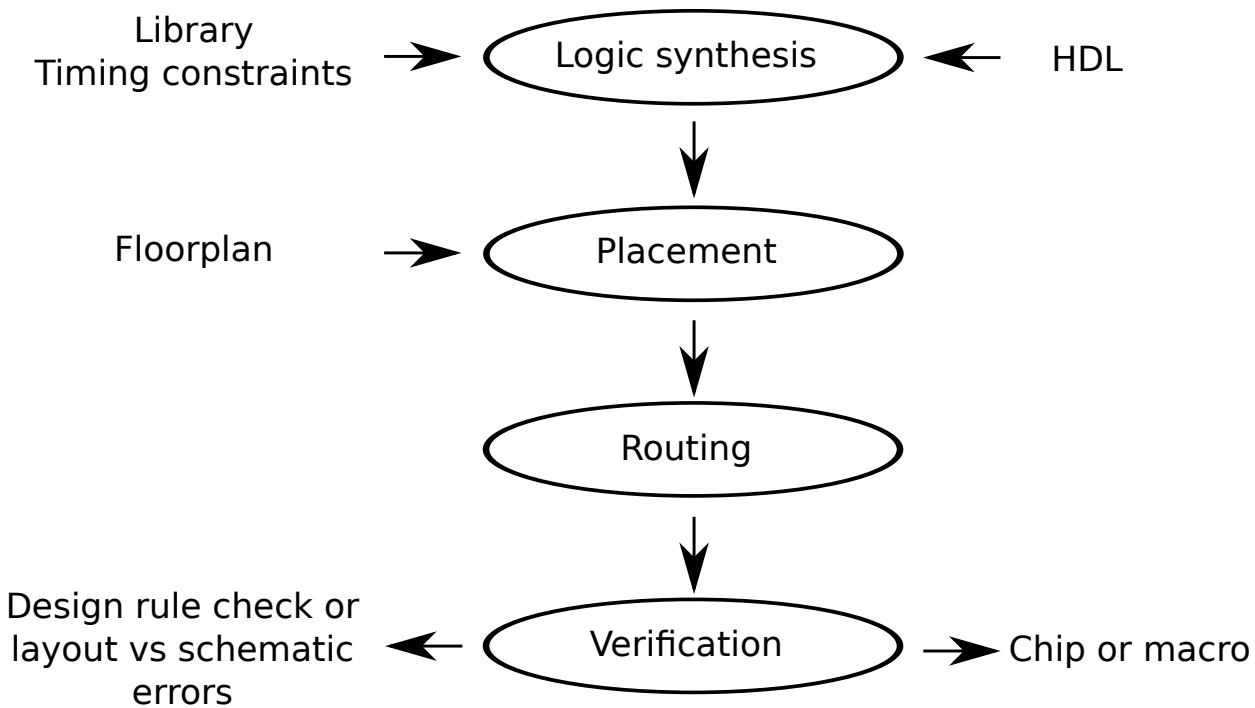
June 22, 2015

Advisors:

Jean-Paul Chaput; Yves Matthieu

The Coriolis toolchain

The digital synthesis flow



Remember Alliance?

Remember Alliance?

An complete, open toolchain developped in the lab

Remember Alliance?

An complete, open toolchain developped in the lab
Coriolis: Physical design only

Hurricane: a powerful database

Hurricane: a powerful database

- Hierarchical

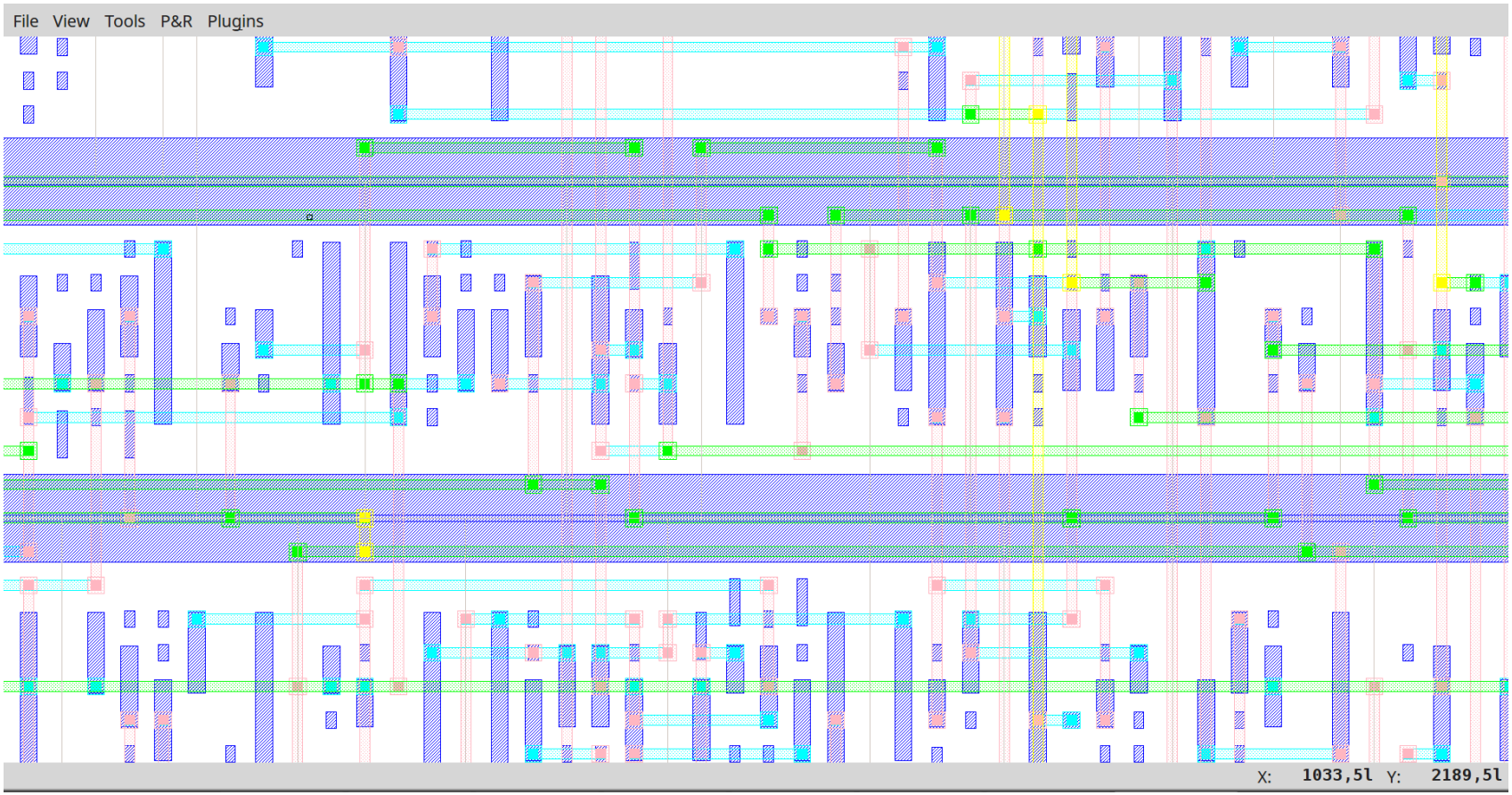
Hurricane: a powerful database

- Hierarchical
- Powerful access methods

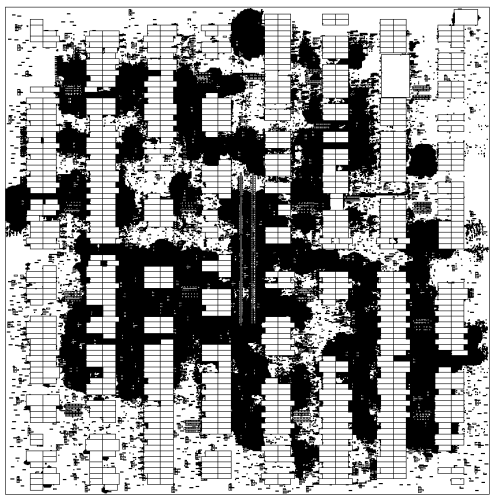
Hurricane: a powerful database

- Hierarchical
- Powerful access methods
- Frontends/Backends

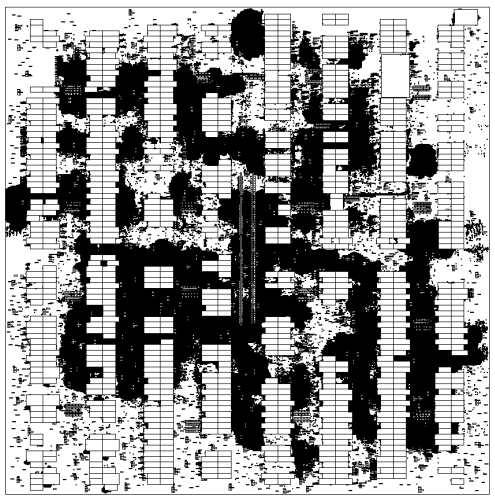
Kite: an efficient router



The need for placement tools



The need for placement tools



- Several thousands – millions – of cells
- Partly determines the circuit's quality

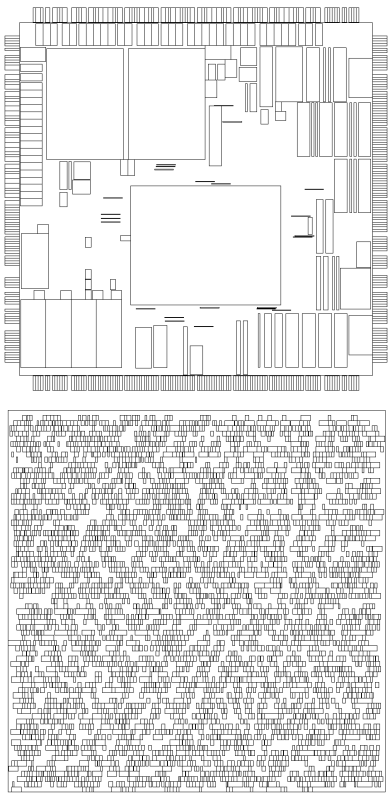
What is a good placer?

- Various optimization objectives
  - Wirelength
  - Routability
  - Timing
- Lots of open benchmarks [3]

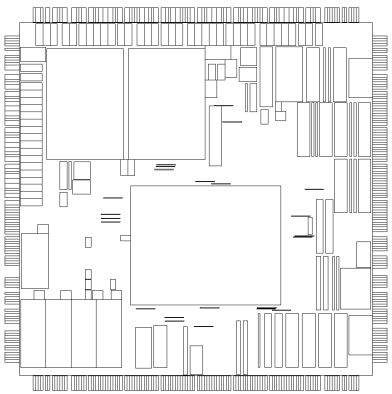
What is a good placer?

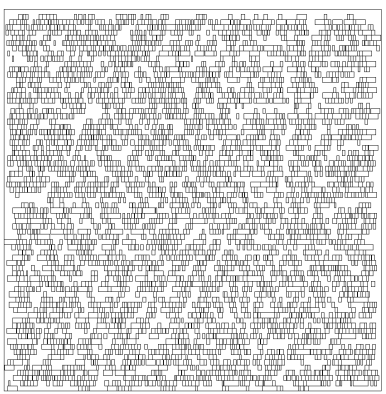
- Various optimization objectives
  - Wirelength
  - Routability
  - Timing
- Lots of open benchmarks [3]
- In practice, we want everything:
  - Error-free routing
  - Fast placement
  - Fast and low-power circuit

Two types of placement problems



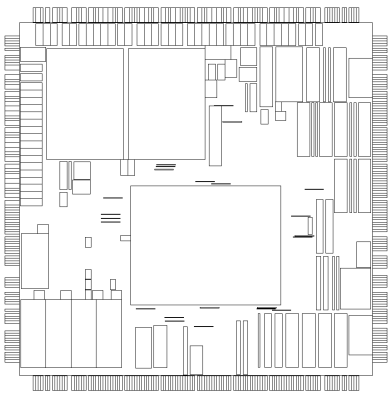
Two types of placement problems

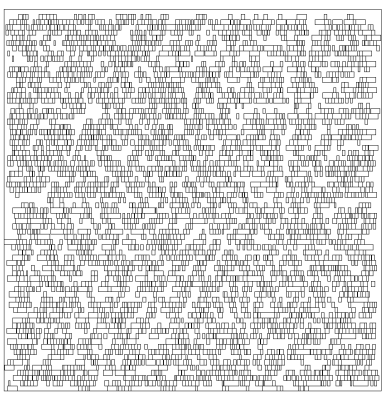




- Big modules $\rightarrow$ Floorplanning
  - Irregular rectangles
  - Few modules

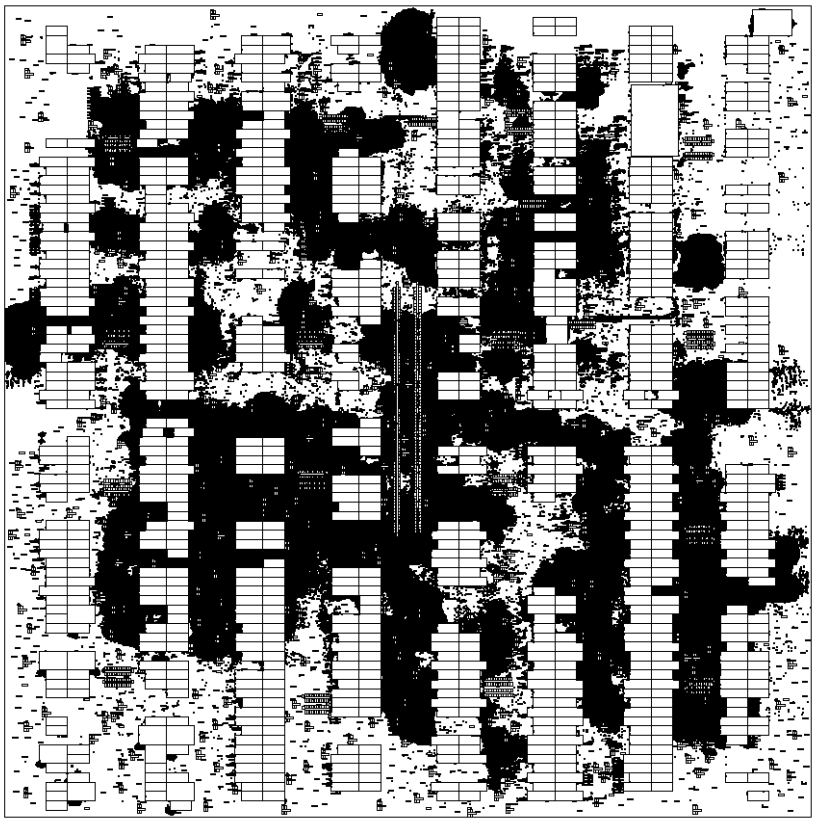
Two types of placement problems





- Big modules $\rightarrow$ Floorplanning
  - Irregular rectangles
  - Few modules
- Standard cell placement
  - Uniform height $\Rightarrow$ placed in rows
  - 10^6 cells or more

The real worlds: mixed-size placement



In practice, circuits have both macros and standard cells

Before the technical stuff...

Obviously....

Before the technical stuff...

Obviously.... It is NP-complete.

Before the technical stuff...

Obviously.... It is NP-complete.

- Bin-packing
- Multiple-machine scheduling
- Graph partitioning
- Steiner trees

Algorithms

Simulated annealing and metaheuristics

Simulated annealing

- Accepts bad moves depending on a temperature
- *Theoretically* converges

Simulated annealing and metaheuristics

Simulated annealing

- Accepts bad moves depending on a temperature
- *Theoretically* converges
- But not in *reasonable* time

Simulated annealing and metaheuristics

Simulated annealing

- Accepts bad moves depending on a temperature
- *Theoretically* converges
- But not in *reasonable* time

Made it quite far thanks to Timberwolf [6]

Three-step placement

Use multiple steps to obtain a good placement

Three-step placement

Use multiple steps to obtain a good placement

- A global placement step
  - Approximate positions
  - Overlaps are possible
  - Only density constraints

Three-step placement

Use multiple steps to obtain a good placement

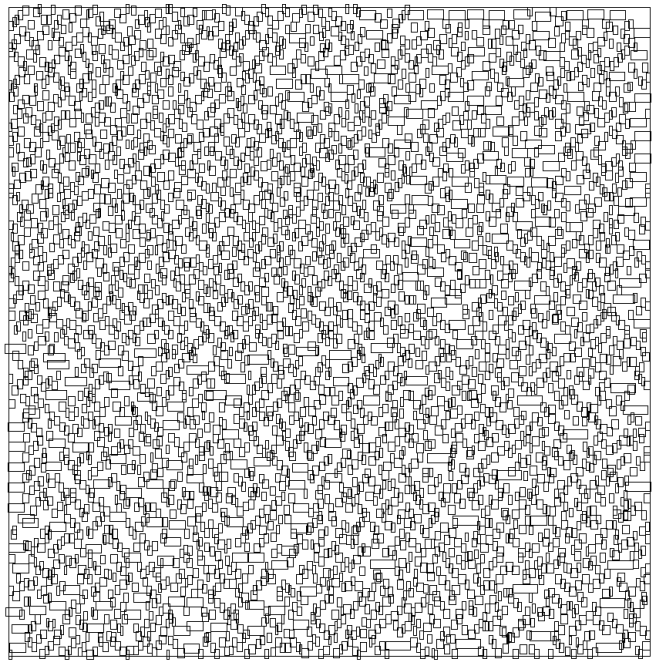
- A global placement step
  - Approximate positions
  - Overlaps are possible
  - Only density constraints
- A legalization step
  - Obtain an overlap-free placement

Three-step placement

Use multiple steps to obtain a good placement

- A global placement step
  - Approximate positions
  - Overlaps are possible
  - Only density constraints
- A legalization step
  - Obtain an overlap-free placement
- A detailed placement step
  - Remove local suboptimalities

Visually



Global placement result

Final placement

Global placement is everything

Global placement is the *most performance-critical* part

Global placement is everything

Global placement is the *most performance-critical* part

- There is only so much you can do with detailed placement

Detailed placement is limited

Legalization

Detailed placement is limited

Legalization

- Simple algorithms
- Needs to succeed, but not critical for quality

Detailed placement is limited

Legalization

- Simple algorithms
- Needs to succeed, but not critical for quality

Detailed placement

Detailed placement is limited

Legalization

- Simple algorithms
- Needs to succeed, but not critical for quality

Detailed placement

- Simple concepts

Detailed placement is limited

Legalization

- Simple algorithms
- Needs to succeed, but not critical for quality

Detailed placement

- Simple concepts
  - Swaps
  - Reordering
  - Moves

Detailed placement is limited

Legalization

- Simple algorithms
- Needs to succeed, but not critical for quality

Detailed placement

- Simple concepts
  - Swaps
  - Reordering
  - Moves
- Small search space

Global placement algorithms

Two approaches:

Global placement algorithms

Two approaches:

- Discrete view: the circuit is a graph
  $\Rightarrow$ Partitioning

Global placement algorithms

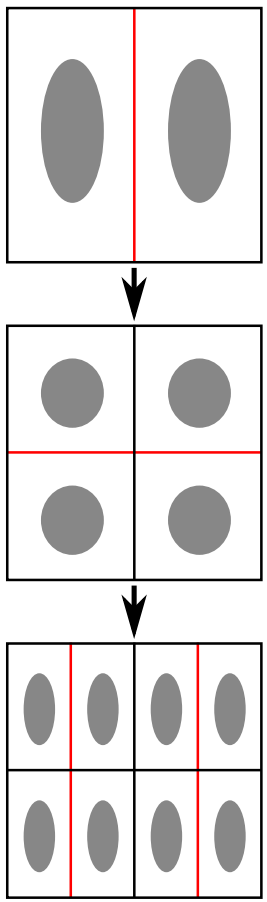
Two approaches:

- Discrete view: the circuit is a graph
  $\Rightarrow$ Partitioning
- Continuous view: it is a vector of positions
  $\Rightarrow$ Analytical placement

Partitioning placers

Basic idea

- Partition the circuit
- Cut as few wires as possible
- Allocate the parts to placement regions



Some nice achievements with Capo [5]

Lots of variations and improvements

- Partitioning algorithm (Fiduccia-Mattheyses)
- Cut costs and fixed terminals
- Free-space handling

Analytical placers

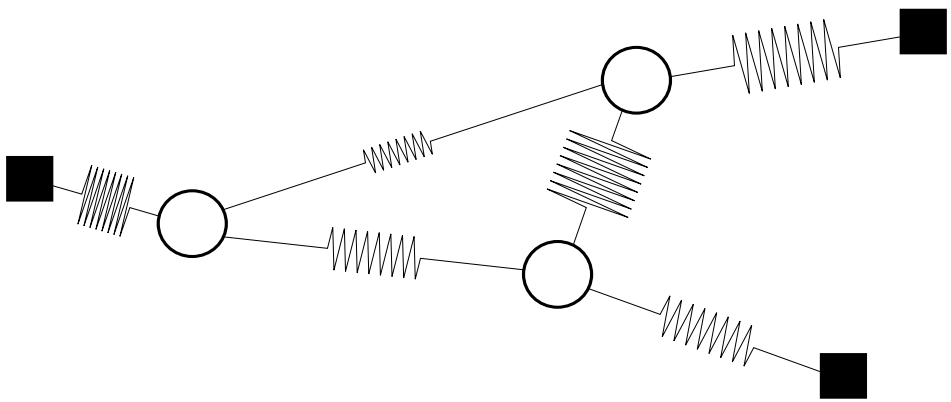
Basic idea

- A convex cost function
- A *huge* vector of positions
- Optimize

Most current placers [3]

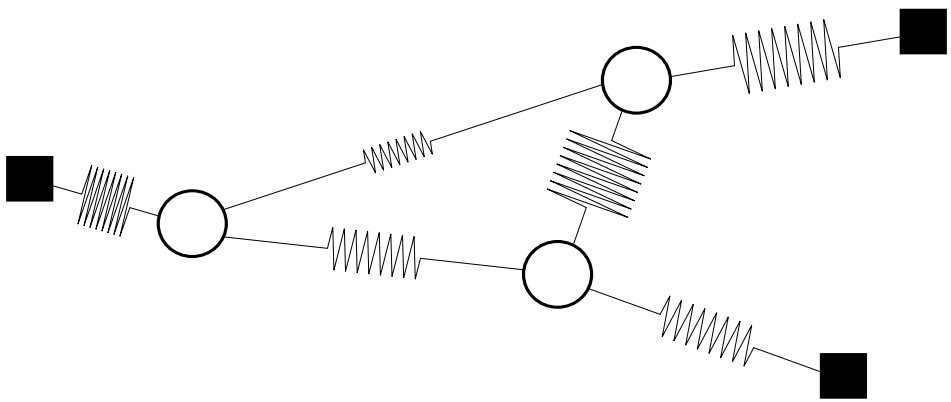
Optimization methods: quadratic placers

Model wires as springs [1]



Optimization methods: quadratic placers

Model wires as springs [1]



Easy to solve: sparse symmetric linear system

Optimization methods: nonlinear placers

Closer model of the cost function

Optimization methods: nonlinear placers

Closer model of the cost function

Only first-order methods:

- Conjugate gradient
  - Needs line-search
  - Slower than the linear one

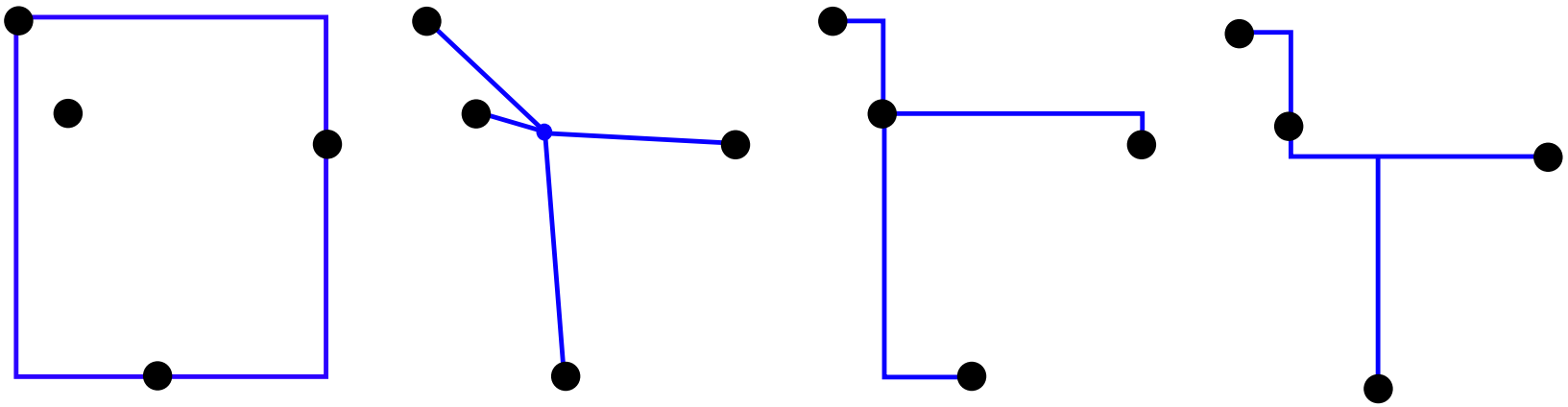
Optimization methods: nonlinear placers

Closer model of the cost function

Only first-order methods:

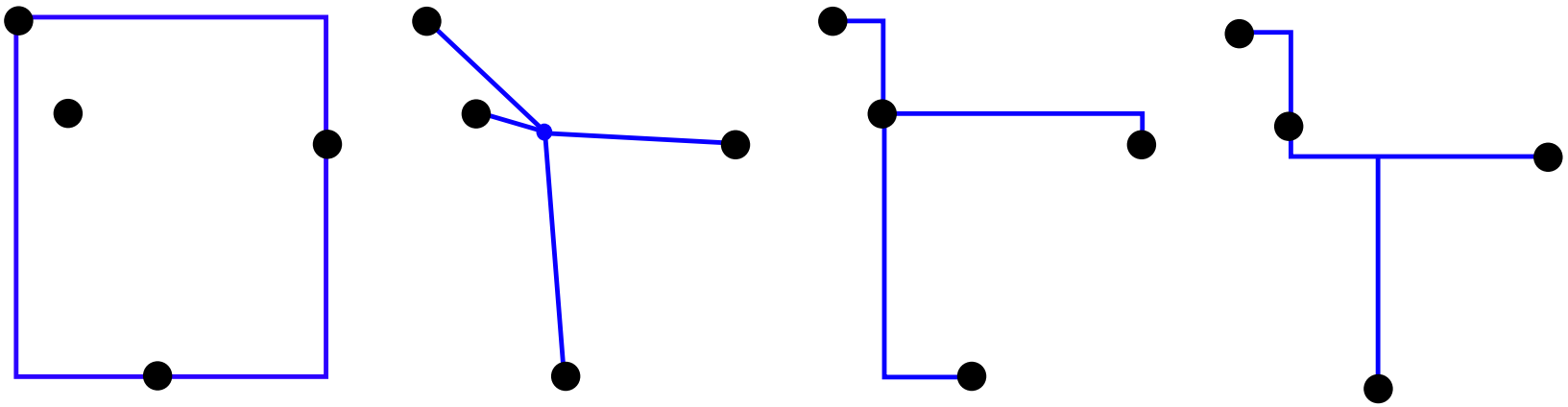
- Conjugate gradient
  - Needs line-search
  - Slower than the linear one
- Nesterov
  - Constant steps
  - Recent comeback of nonlinear methods [3]

Usual cost functions



Wirelength models

Usual cost functions



Wirelength models

$$HPWL = \sum_{c \in \{x,y\}} \sum_{n \in nets} \left(\max_{pin \in n} c_{pin} - \min_{pin \in n} c_{pin} \right)$$

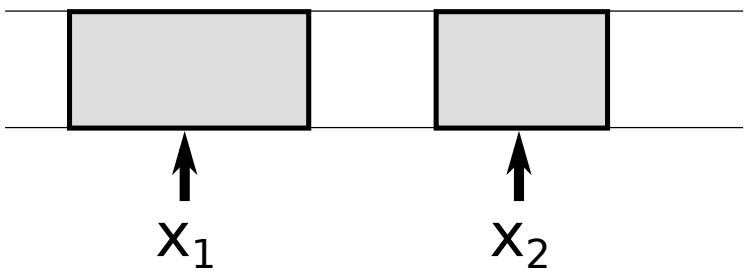
Beware!

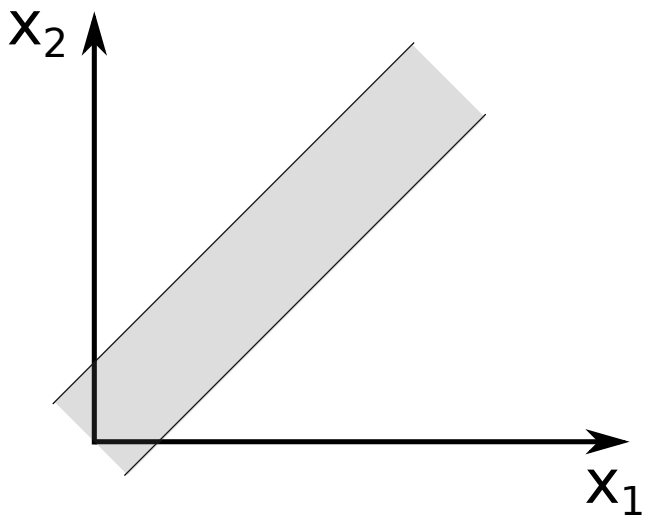
Beware!

Non-convex solution space!

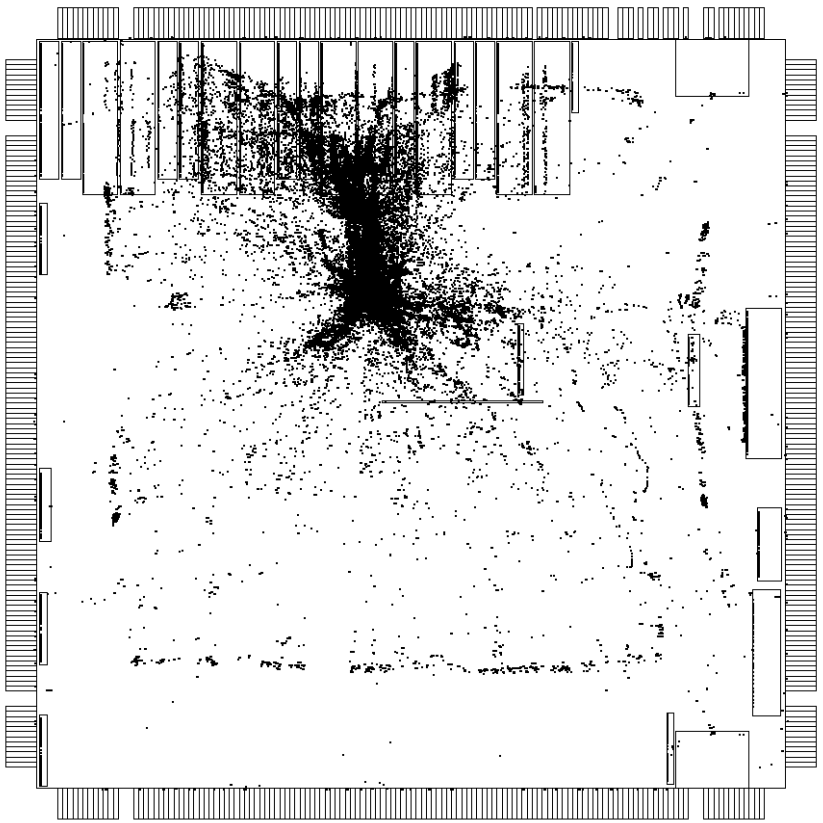
Beware!

Non-convex solution space!





Problem



Lots of overlaps

Handling density constraints

Handling density constraints

Back to partitioning

Handling density constraints

Back to partitioning $\Leftarrow$ based on the positions

Handling density constraints

Back to partitioning $\Leftarrow$ based on the positions

Force-directed placers $\Leftarrow$ add a penalty function

Handling density constraints

Back to partitioning $\Leftarrow$ based on the positions

Force-directed placers $\Leftarrow$ add a penalty function

- Local penalty

Handling density constraints

Back to partitioning $\Leftarrow$ based on the positions

Force-directed placers $\Leftarrow$ add a penalty function

- Local penalty
- Global potential

Handling density constraints

Back to partitioning $\Leftarrow$ based on the positions

Force-directed placers $\Leftarrow$ add a penalty function

- Local penalty
- Global potential
- ”Projection” on the solution space

Handling density constraints

Back to partitioning $\Leftarrow$ based on the positions

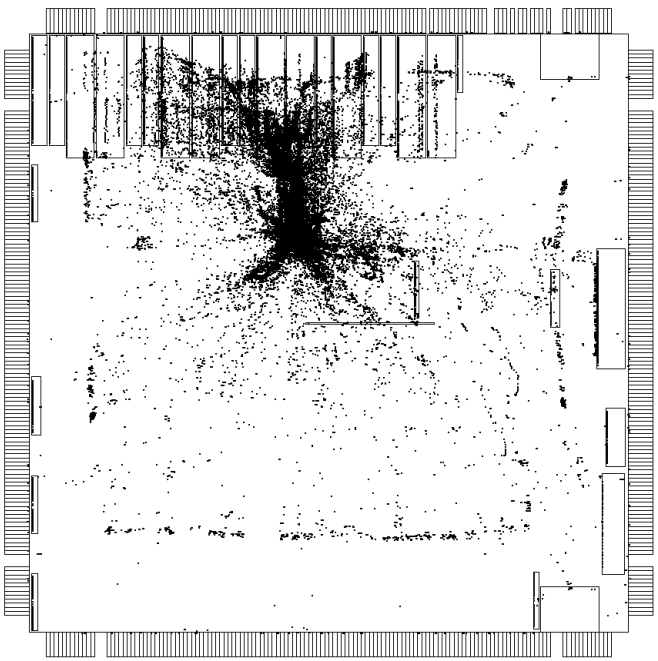
Force-directed placers $\Leftarrow$ add a penalty function

- Local penalty
- Global potential
- ”Projection” on the solution space

Repeat

Obtaining a feasible solution: lookahead legalization

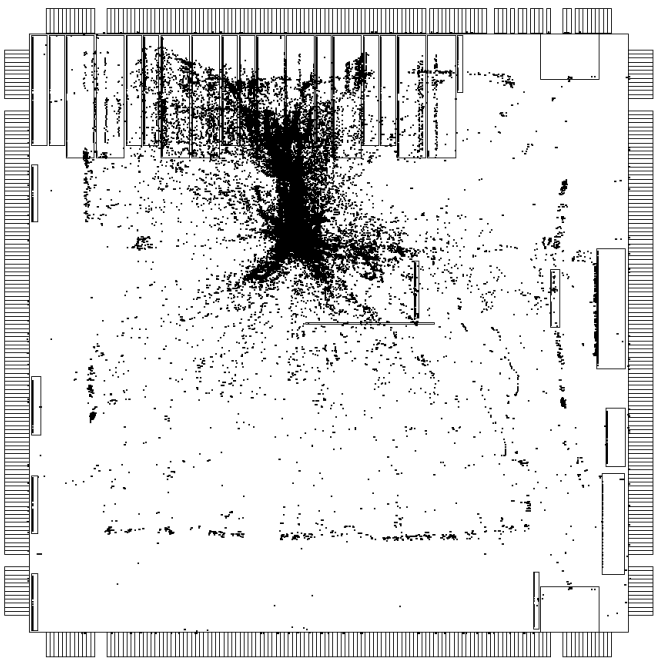
Remove density overflow, minimize the displacement



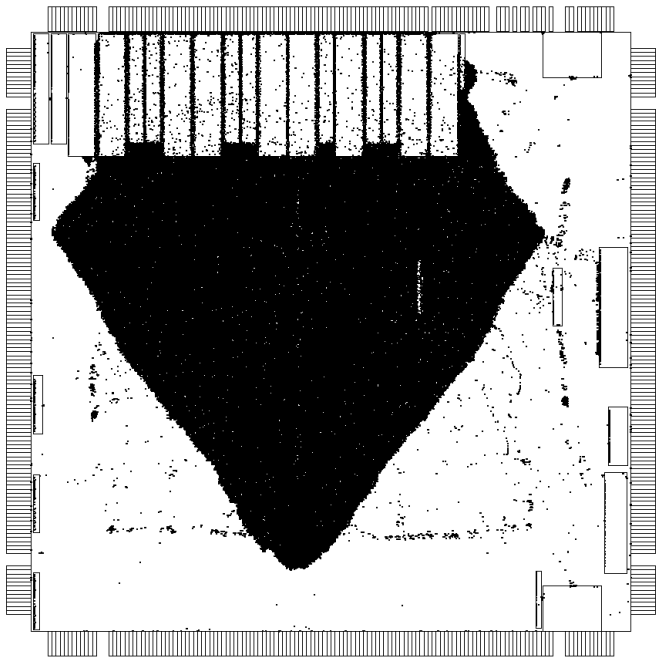
Optimization result

Obtaining a feasible solution: lookahead legalization

Remove density overflow, minimize the displacement



Optimization result



Projection

Lots of variations and improvements

- Penalty function
- Cost function
- Legalization algorithm

Lots of variations and improvements

- Penalty function
- Cost function
- Legalization algorithm
- Additional refinements

Benchmark winners

The best placers are analytical. No other rule

Benchmark winners

The best placers are analytical. No other rule

Placer	Optimization	Density handling	Refinement
ePlace	Nonlinear	FD (potential)	No
Maple	Quadratic	FD (legalizing)	Yes
Polar	Quadratic	FD (legalizing)	Yes
BonnPlace	Quadratic	Partitioning	No?

Benchmark winners

The best placers are analytical. No other rule

Placer	Optimization	Density handling	Refinement
ePlace	Nonlinear	FD (potential)	No
Maple	Quadratic	FD (legalizing)	Yes
Polar	Quadratic	FD (legalizing)	Yes
BonnPlace	Quadratic	Partitioning	No?

- Rather force-directed

Benchmark winners

The best placers are analytical. No other rule

Placer	Optimization	Density handling	Refinement
ePlace	Nonlinear	FD (potential)	No
Maple	Quadratic	FD (legalizing)	Yes
Polar	Quadratic	FD (legalizing)	Yes
BonnPlace	Quadratic	Partitioning	No?

- Rather force-directed
- + various algorithms

Benchmark winners

The best placers are analytical. No other rule

Placer	Optimization	Density handling	Refinement
ePlace	Nonlinear	FD (potential)	No
Maple	Quadratic	FD (legalizing)	Yes
Polar	Quadratic	FD (legalizing)	Yes
BonnPlace	Quadratic	Partitioning	No?

- Rather force-directed
- + various algorithms
- + **trial-and-error**

Coloquinte is a force-directed quadratic placer

Coloquinte is a force-directed quadratic placer

- Analytical $\Rightarrow$ Flexible cost function

Coloquinte is a force-directed quadratic placer

- Analytical $\Rightarrow$ Flexible cost function
- Quadratic $\Rightarrow$ best results (until recently)

Coloquinte is a force-directed quadratic placer

- Analytical $\Rightarrow$ Flexible cost function
- Quadratic $\Rightarrow$ best results (until recently)
- Force-directed $\Rightarrow$ best results

Coloquinte is a force-directed quadratic placer

- Analytical $\Rightarrow$ Flexible cost function
- Quadratic $\Rightarrow$ best results (until recently)
- Force-directed $\Rightarrow$ best results
- Lookahead legalization $\Rightarrow$ Local density control

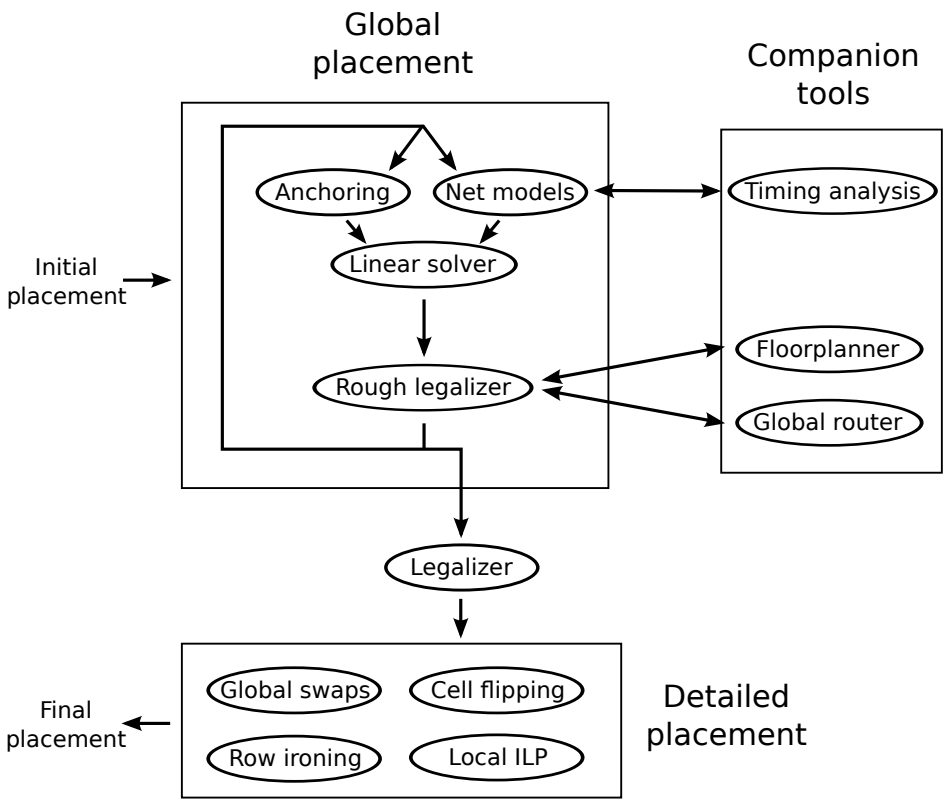
Coloquinte is a force-directed quadratic placer

- Analytical $\Rightarrow$ Flexible cost function
- Quadratic $\Rightarrow$ best results (until recently)
- Force-directed $\Rightarrow$ best results
- Lookahead legalization $\Rightarrow$ Local density control
- **But**

Coloquinte is a force-directed quadratic placer

- Analytical $\Rightarrow$ Flexible cost function
- Quadratic $\Rightarrow$ best results (until recently)
- Force-directed $\Rightarrow$ best results
- Lookahead legalization $\Rightarrow$ Local density control
- **But** no refinement

Coloquinte's structure



What is new in Coloquinte?

What is new in Coloquinte?

- Focus on good lookahead legalization

What is new in Coloquinte?

- Focus on good lookahead legalization
- Algorithms for detailed placement

What is new in Coloquinte?

- Focus on good lookahead legalization
- Algorithms for detailed placement
- Penalty scheduling

What is new in Coloquinte?

- Focus on good lookahead legalization
- Algorithms for detailed placement
- Penalty scheduling
- Net models

The need for extensive benchmarking

The need for extensive benchmarking

Small change $\Rightarrow$ random quality variations

The need for extensive benchmarking

Small change $\Rightarrow$ random quality variations

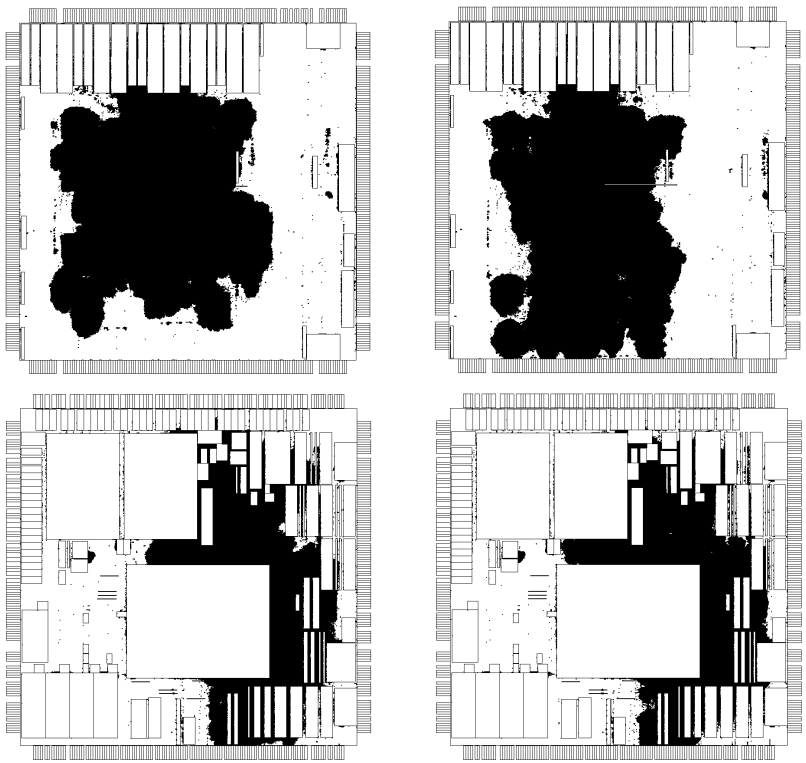
Experienced with:

- Penalty type
- Penalty weight

The need for extensive benchmarking

Bigblue1: WL +0.8%

Adaptec2: WL -2.1%



A simple change: linear vs quadratic penalty

Can we suppress these variations?

Can we suppress these variations?

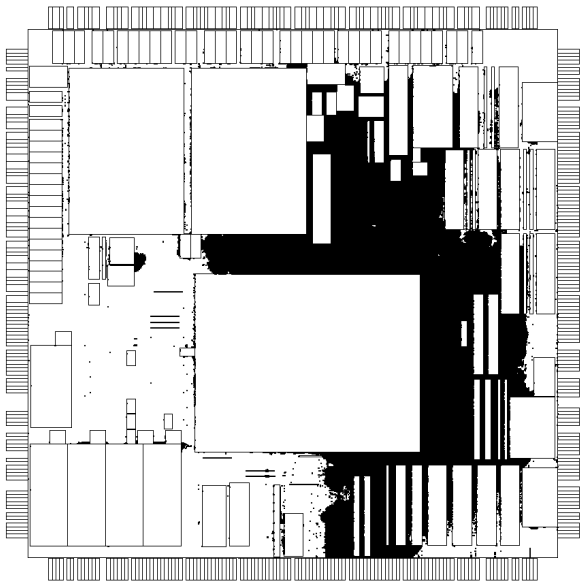
No!

Can we suppress these variations?

No!

Definitely not

$\Rightarrow$ measure again and again

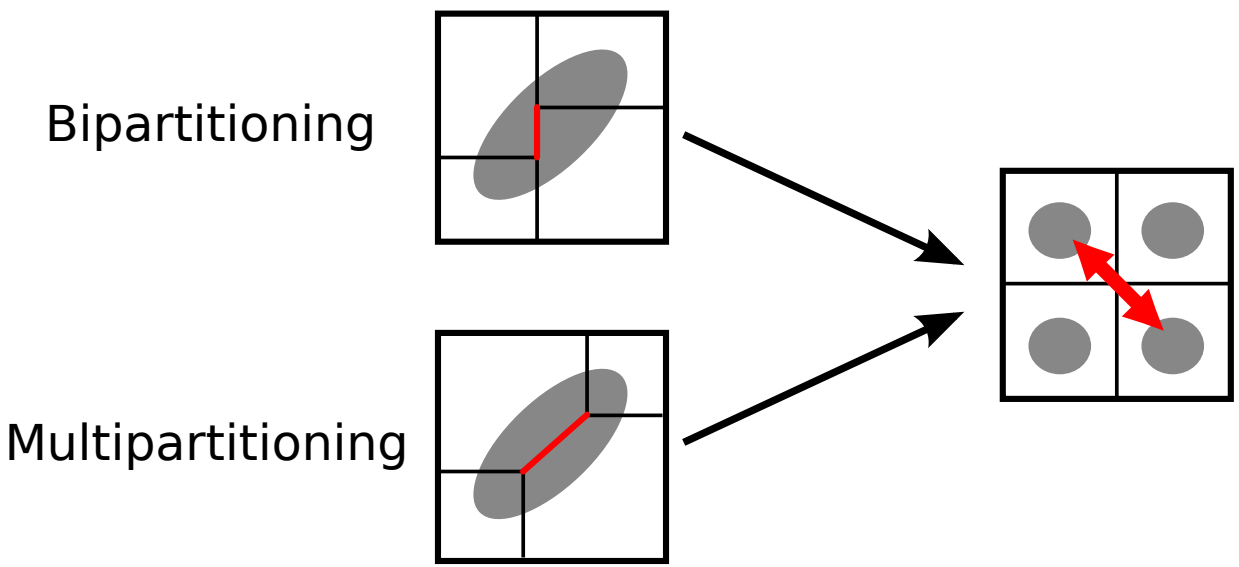


Lookahead legalization

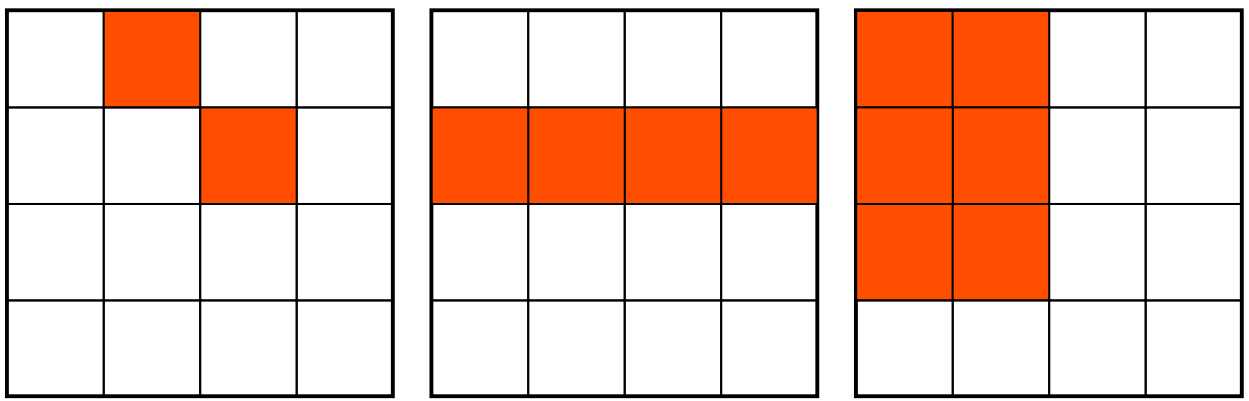
- Introduced by SimPL [2]

Lookahead legalization

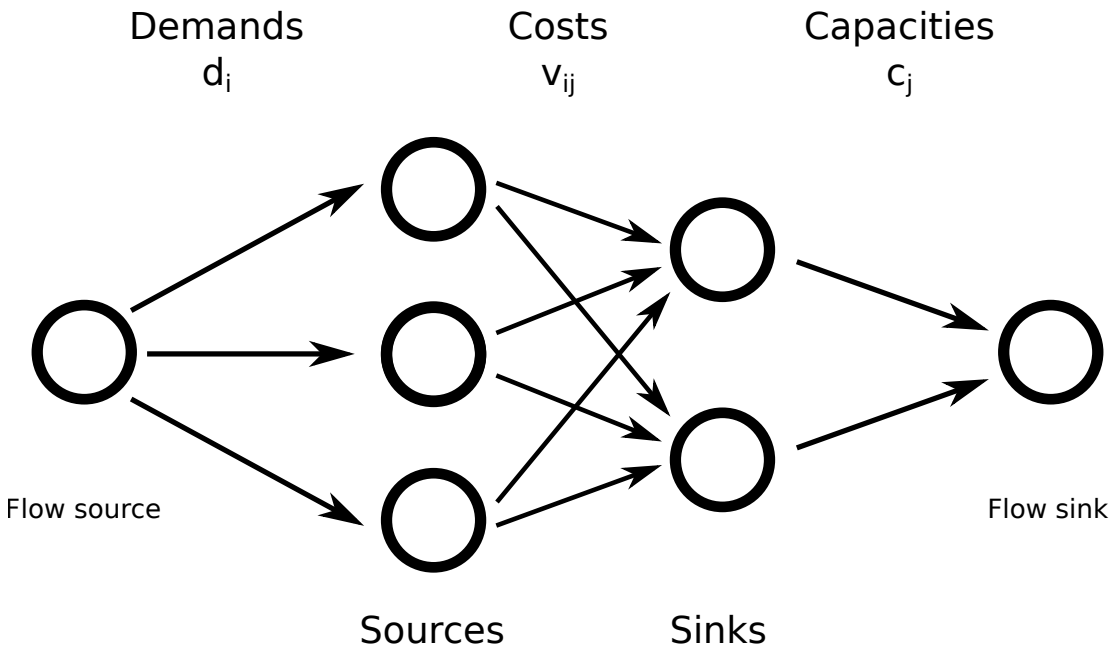
- Introduced by SimPL [2]
- Always based on partitioning



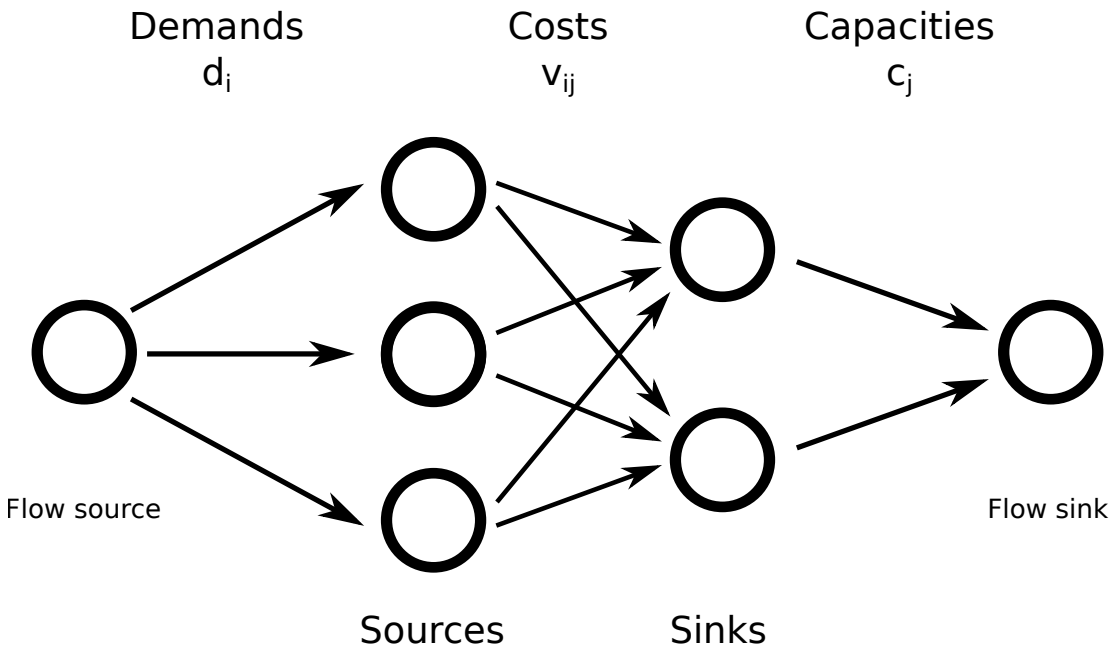
In Coloquinte: local optimizations



A transportation problem



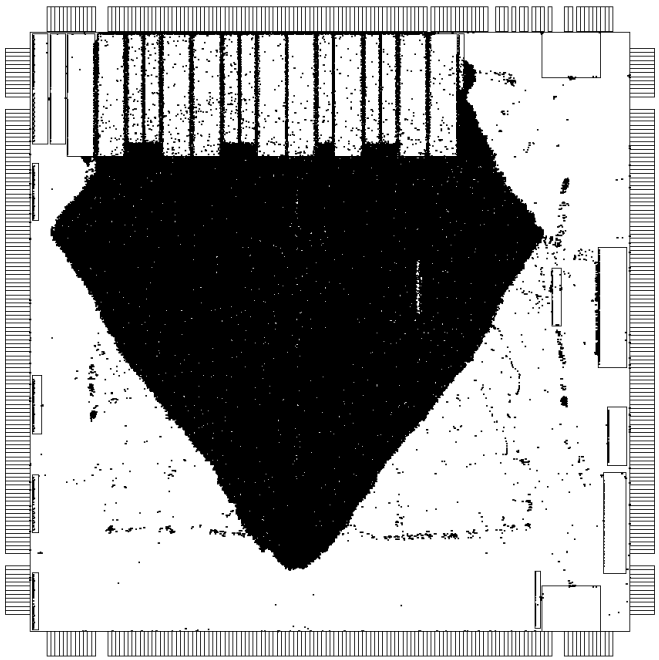
A transportation problem



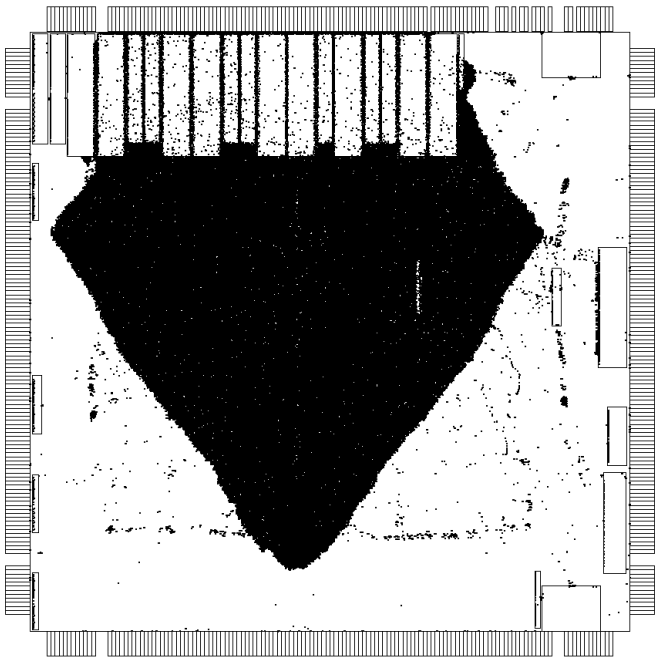
$\Omega(n^3)$ algorithms

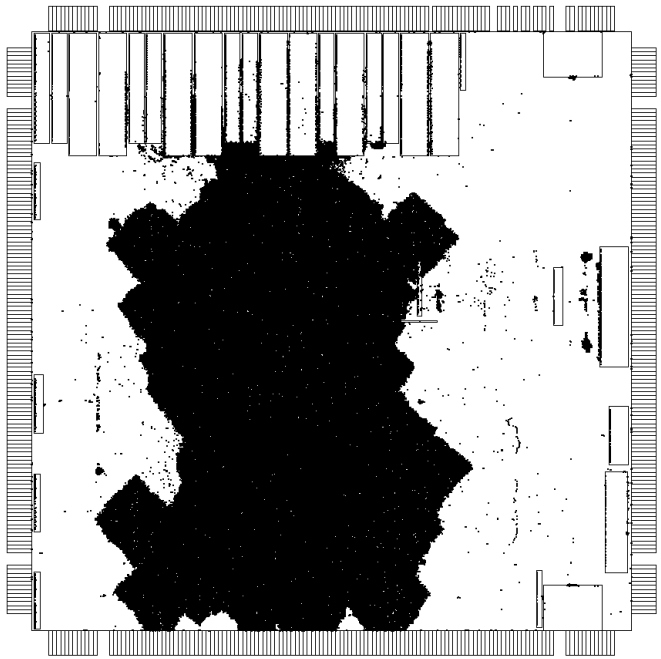
Visual results

Visual results



Visual results



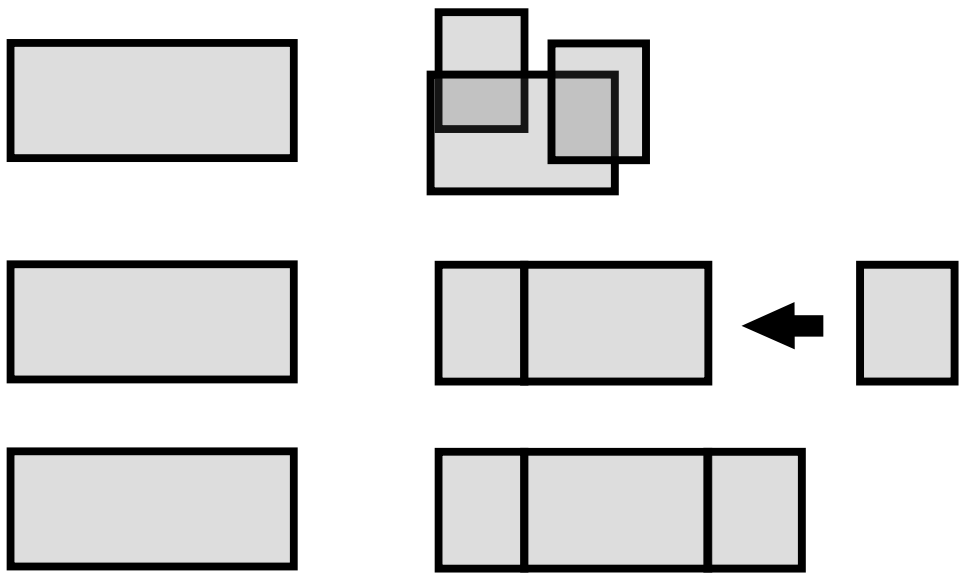


True legalization

Usually greedy

True legalization

Usually greedy

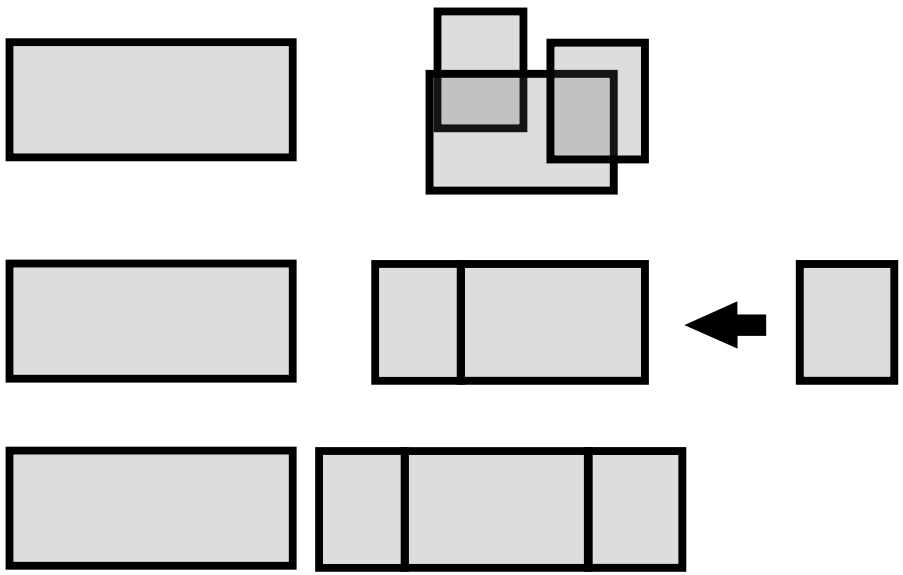


True legalization in Coloquinte

Push the previous cells [7]

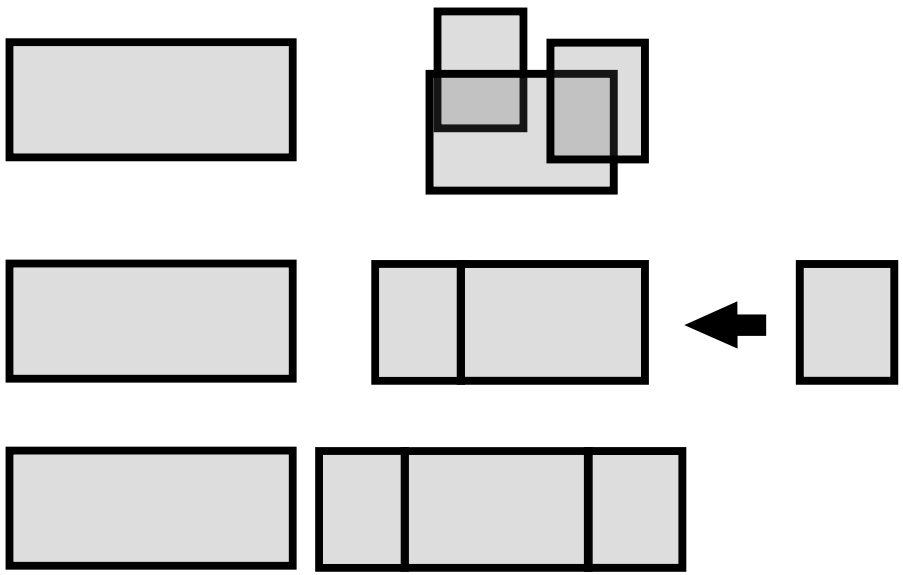
True legalization in Coloquinte

Push the previous cells [7]



True legalization in Coloquinte

Push the previous cells [7]



Even better: after a rough legalization

Placement $\approx$ scheduling

$$\text{Machines} \Leftrightarrow \text{Rows}$$
$$\text{Tasks} \Leftrightarrow \text{Uniform-height cells}$$
$$? \Leftrightarrow \text{Wirelength}$$

Our cost function is complicated

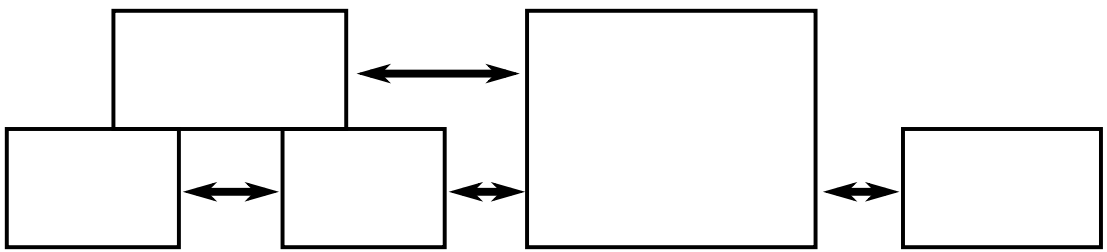
$\Rightarrow$ The cells are not independent

Our cost function is complicated

$\Rightarrow$ The cells are not independent

At fixed scheduling-order, dual of a *Minimum-cost flow problem*

Minimize $\sum \alpha_i x_i$
Subject to $x_{i_k} - x_{j_k} \leq d_k$



A simplified problem

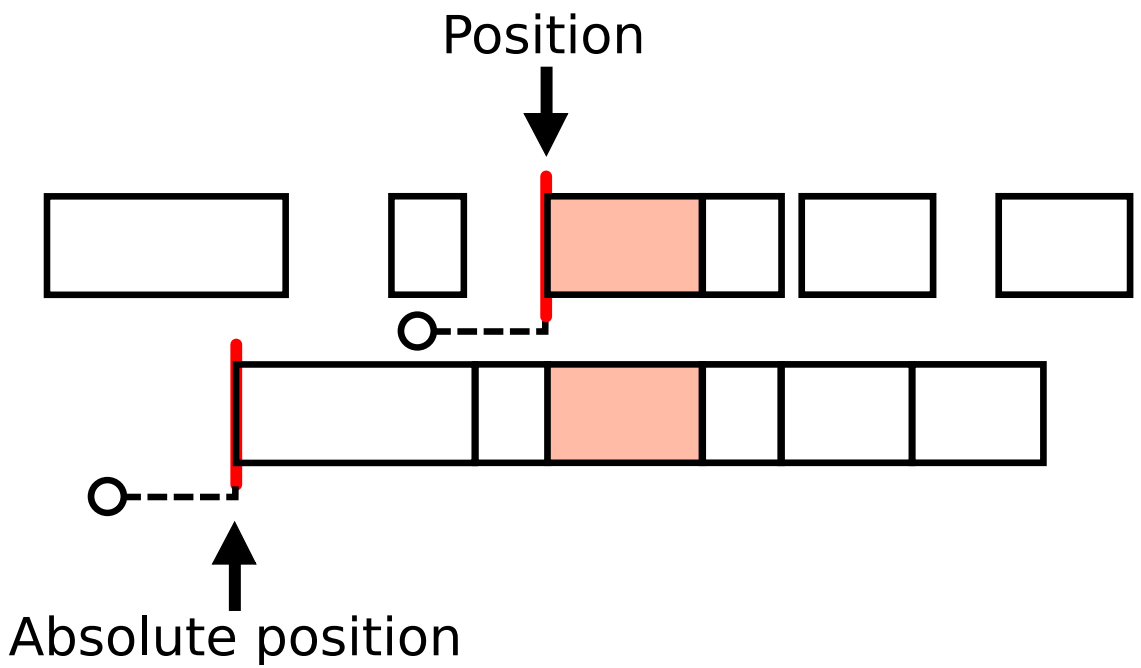
Single row with fixed ordering
=
Single machine with fixed ordering

The cascading descent algorithm

Greedy algorithm with a single heap

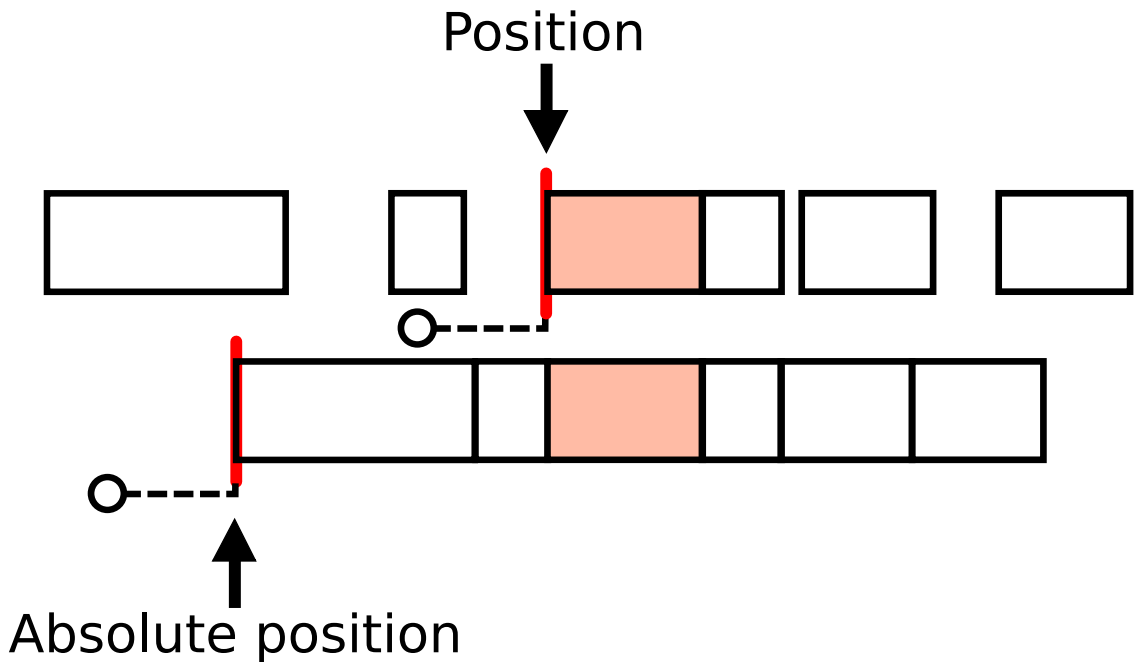
The cascading descent algorithm

Greedy algorithm with a single heap



The cascading descent algorithm

Greedy algorithm with a single heap



$O(n \log n)$ [4]

First use for placement

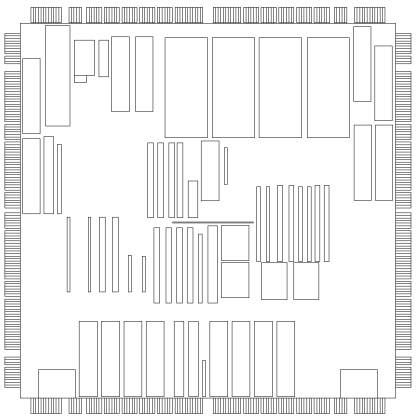
Everywhere in Coloquinte:

First use for placement

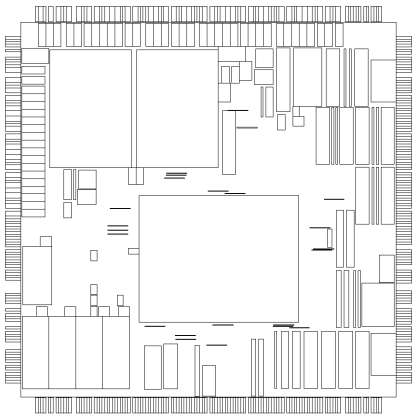
Everywhere in Coloquinte:

- Lookahead legalizer
- True legalizer
- Detailed placement reordering
- Whitespace allocation in detailed placement

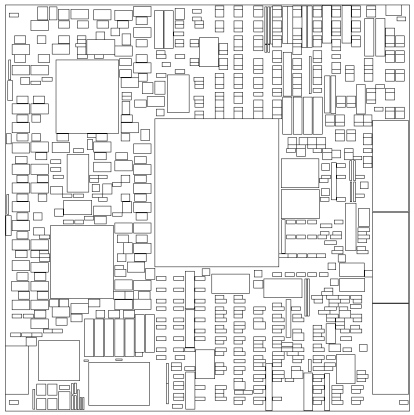
The ISPD05 benchmarks



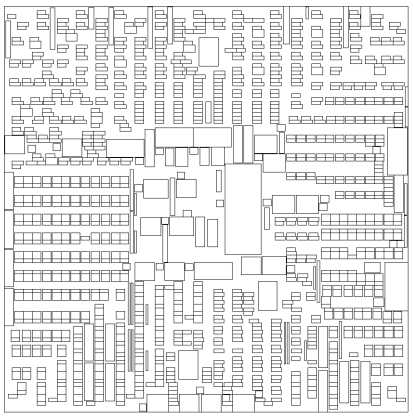
Adaptec1



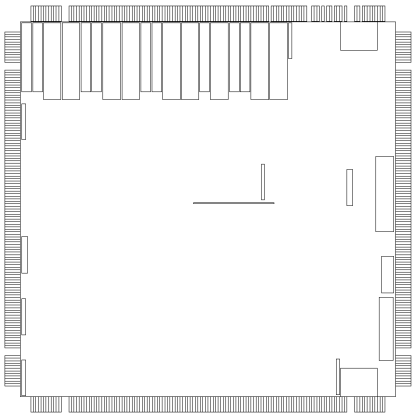
Adaptec2



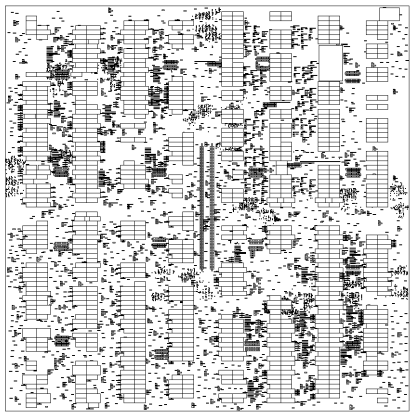
Adaptec3



Adaptec4

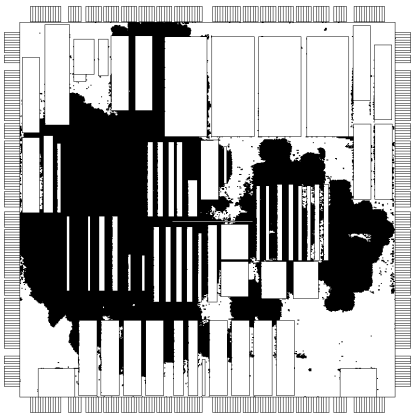


Bigblue1

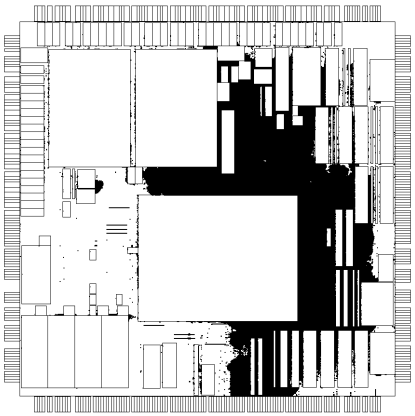


Bigblue2

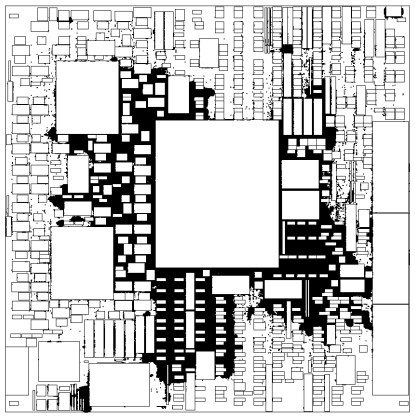
ISPD05 placements



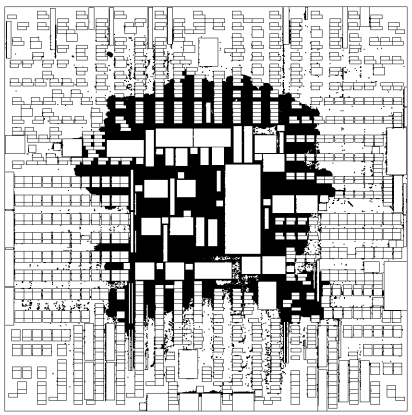
Adaptec1



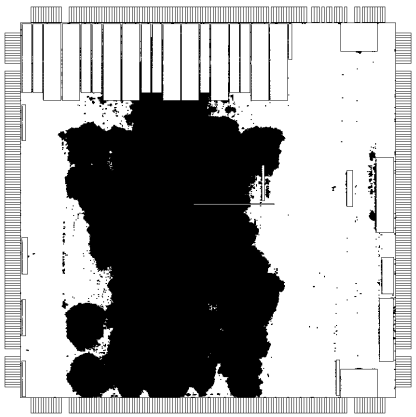
Adaptec2



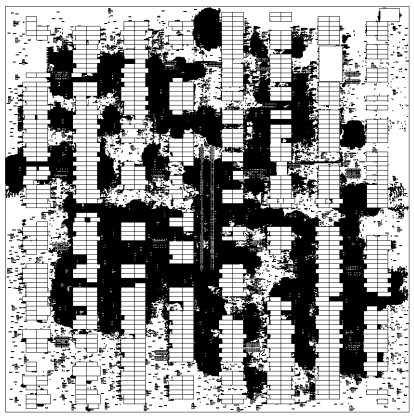
Adaptec3



Adaptec4



Bigblue1



Bigblue2

Not bad

Problem	ePlace	Best quadratic	Coloquinte	%
Adaptec1	**74.63**	76.36 (Maple)	76.49	+2.49
Adaptec2	**84.84**	86.16 (POLAR)	87.90	+3.61
Adaptec3	**194.57**	201.30 (POLAR)	202.12	+3.88
Adaptec4	**179.02**	179.91 (Maple)	182.27	+1.82
Bigblue1	**90.99**	93.74 (Maple)	94.76	+4.14
Bigblue2	141.83	143.95 (POLAR)	**141.16**	**-0.47**
Bigblue3	**308.77**	317.17 (Bonn)	#	#
Bigblue4	**753.20**	775.71 (Maple)	#	#

Quality comparison: HPWL (10^6 units)

Further work: placement

Further work: placement

- Switch to Nesterov optimization

Further work: placement

- Switch to Nesterov optimization
- More legalization improvements

Further work: placement

- Switch to Nesterov optimization
- More legalization improvements
- Performance tuning

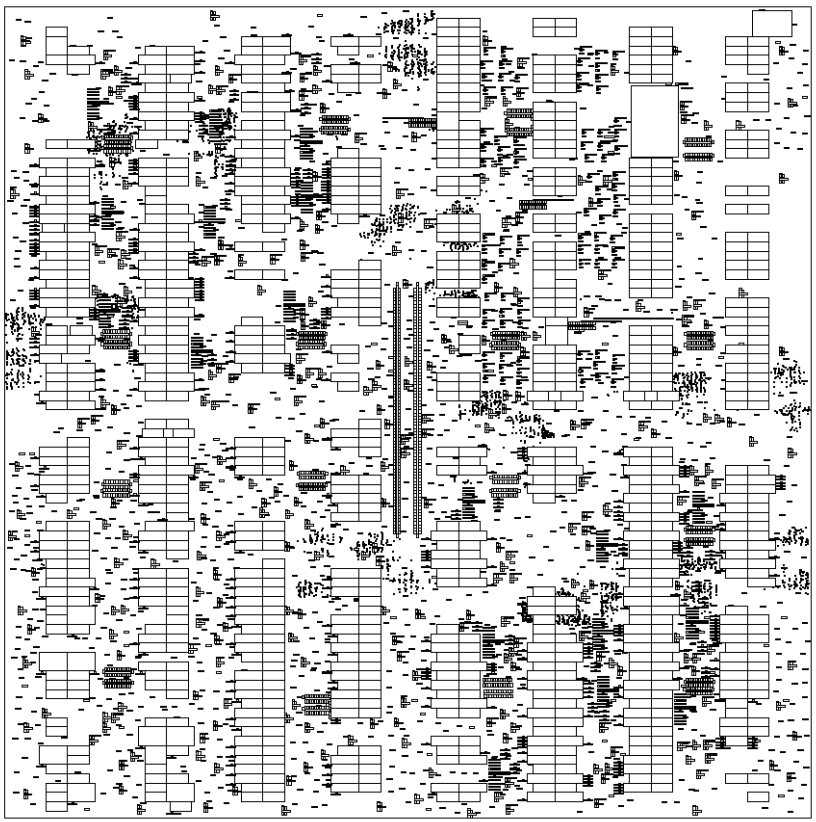
Further work: placement

- Switch to Nesterov optimization
- More legalization improvements
- Performance tuning
- Integer programming in detailed placement

Further work: routing, synthesis

- Gridless routing
- Specialized multiplier synthesis
- Placement-aware resynthesis

Questions



References I


C.J. Alpert, T.F. Chan, A.B. Kahng, I.L. Markov, and P. Mulet.
Faster minimization of linear wirelength for global placement.
Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 17(1):3–13, Jan 1998.


Myung-Chul Kim, Dong-Jin Lee, and Igor L Markov.
Simpl: An effective placement algorithm.
Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 31(1):50–60, 2012.


Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, D. Huang, Yufeng Luo, Chin-Chi Teng, and Chung-Kuan Cheng.
eplace-ms: Electrostatics-based placement for mixed-size circuits.
Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 34(5):685–698, May 2015.

References II

Yunpeng Pan and L. Shi.

Dual constrained single machine sequencing to minimize total weighted completion time.

Automation Science and Engineering, IEEE Transactions on, 2(4):344–357, Oct 2005.

J.A. Roy and I.L. Markov.

Seeing the forest and the trees: Steiner wirelength optimization in placement.

Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 26(4):632–644, April 2007.

C. Sechen and A. Sangiovanni-Vincentelli.

The timberwolf placement and routing package.

Solid-State Circuits, IEEE Journal of, 20(2):510–522, April 1985.

References III

Peter Spindler, Ulf Schlichtmann, and Frank M. Johannes.

Abacus: Fast legalization of standard cell circuits with minimal movement.

In *Proceedings of the 2008 International Symposium on Physical Design*, ISPD '08, pages 47–53, New York, NY, USA, 2008. ACM.