Basic Placement Algorithms

Jens Vygen

Basic Placement Algorithms - Table of Contents -

- Problem Formulations (page 3-28)
- Global Placement Approaches (page 29-63)
- More on Quadratic Placement (page 64-90)
- Detailed Placement (page 91-110)
- References (page 111-121)

1. Problem Formulations

- Design Objectives
- Feasible Placements
- Netlength Measures
- Global Placement Problem Formulation
- Detailed Placement Problem Formulation

1.1. Design Objectives

- Routability
- Timing
  (meet timing constraints/minimize cycle time)
- Power consumption
- Production cost (die size, yield)
- Time-to-market (turn-around time)
- Allow for late netlist changes

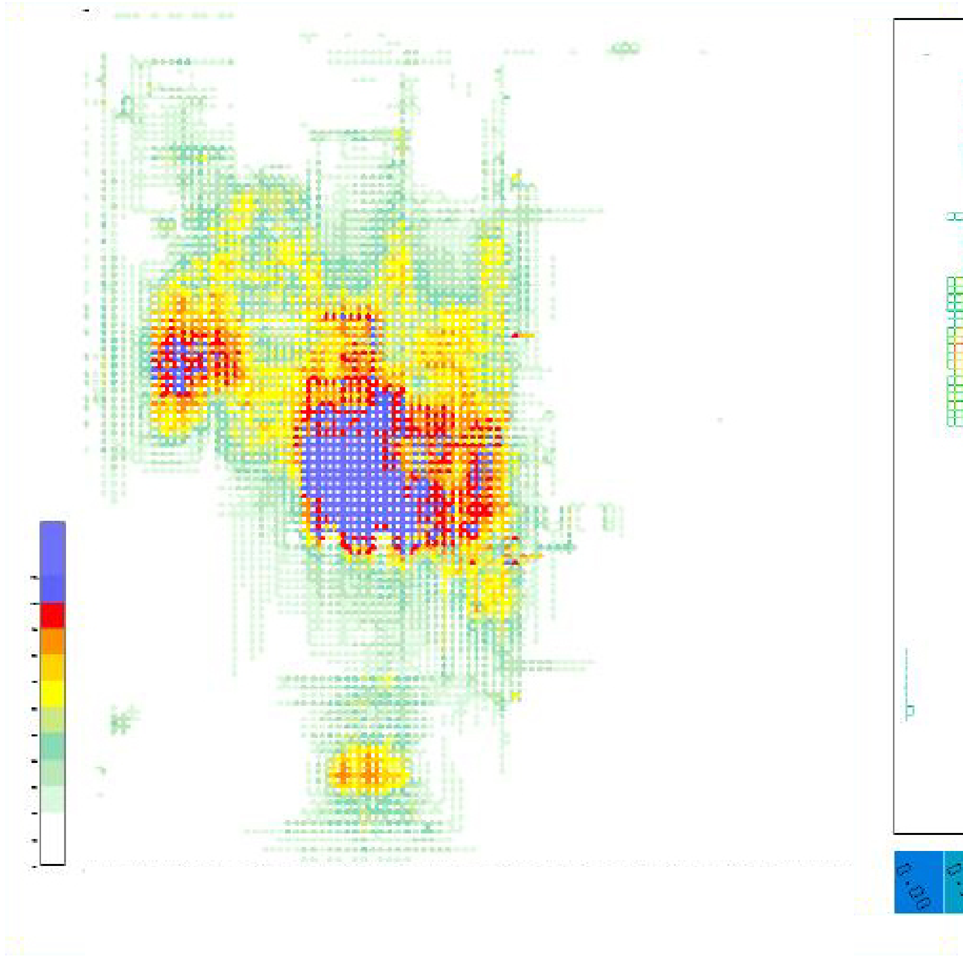
1.1.1. Routability

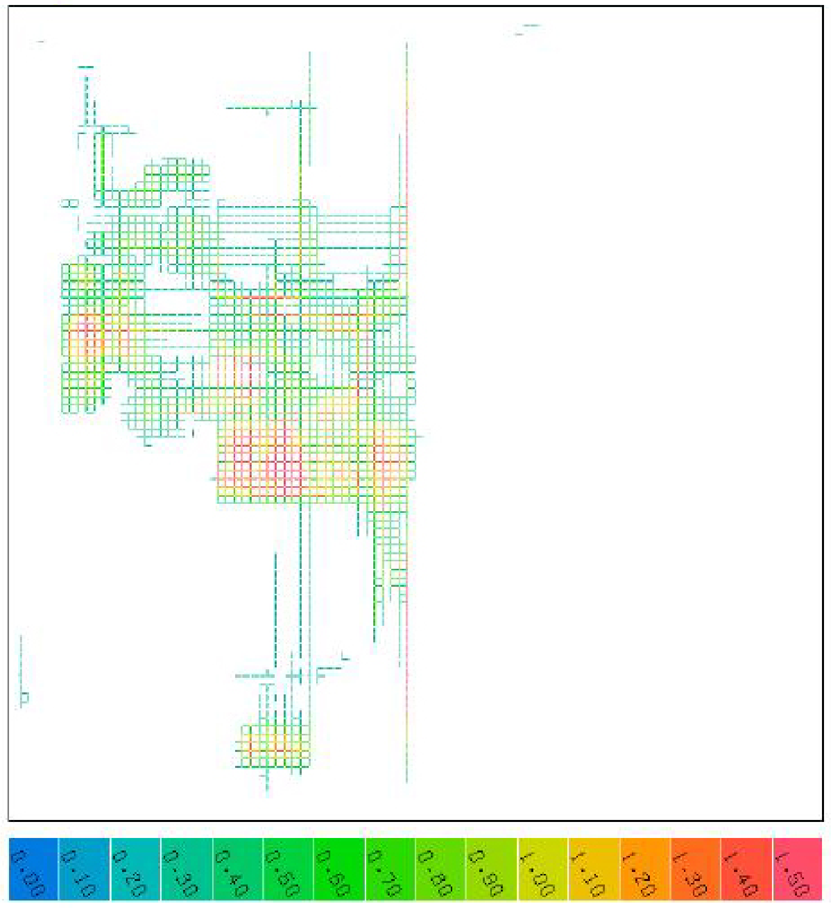
- Routability is a must
- Number of routing planes is usually fixed
- If a placement is difficult to route, then:
  - routing will take more time
  - more coupling capacitance/noise/detours will be necessary.

How to measure routability?

- Routability depends on the router
- An LP-relaxation approach gives a provably excellent measure of routability [Alb01], but takes a couple of hours for complex designs
- Very simple and fast estimates often correlate quite well [BR02], [HMS02]

Near-optimum global routing versus simple congestion estimate





1.1.2. Timing

- Timing constraints bound the delay on each
  - path (with a fixed clock schedule)
  - cycle (with a variable clock schedule)
- Minimizing the cycle time corresponds to minimizing the
  - delay on the critical path (fixed clock schedule)
  - average delay on the critical cycle (variable clock schedule) [AKSV99].

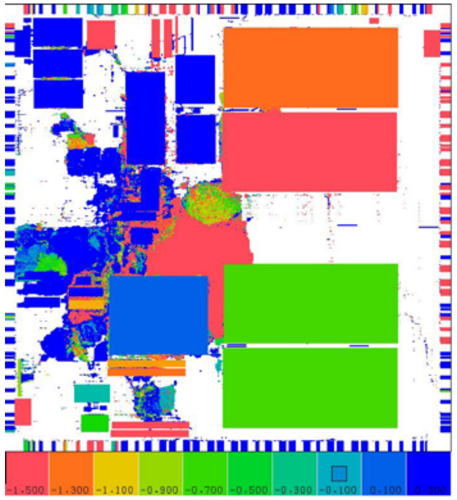
Placement-dependent delays

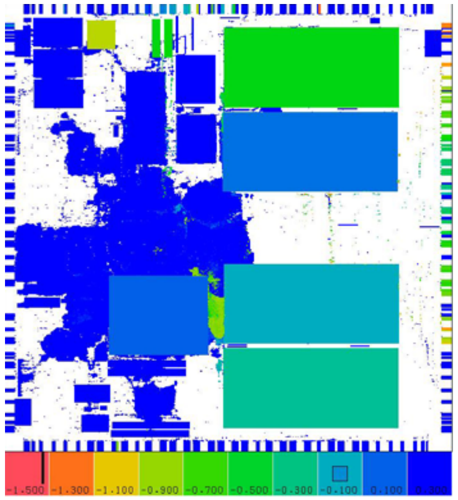
- For fixed netlist and device sizes, the delay along a path/cycle mainly depends on the capacitances of the nets
- For critical nets the capacitance can be assumed to be proportional to the length
- Thus, weighted sum of net lengths is a reasonable metric (for FPGAs this is not as good as for ASICs, but still used [Jay01]).

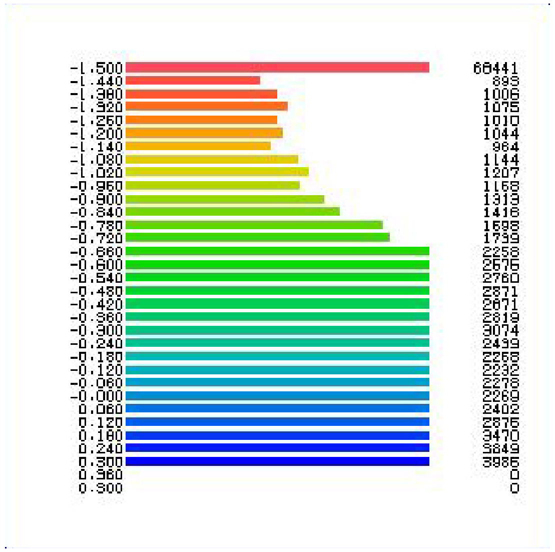
Timing optimization is necessary

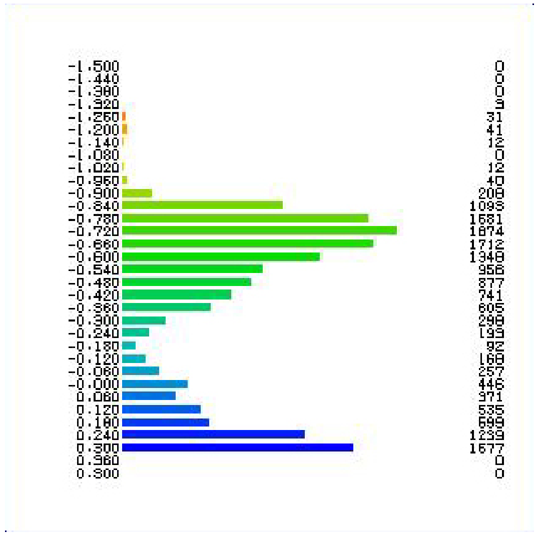
- But: to evaluate a placement (or intermediate results produced by a placement algorithm), timing analysis with fixed netlist and device sizes is not sufficient
- At least gate sizing and buffering must be done, based on the placement, to get meaningful results
- Thus, any timing analysis within placement must be preceded by timing optimization.

Timing before optimization versus after optimization









Static Timing Analysis

- Purpose: Check timing constraints, compute slacks
- Propagates timing info through an acyclic timing graph. Problems:
  - Classical STA assumes a fixed clock schedule (e.g. zero skew) and no transparent latches
  - The slew-dependency is not modelled properly [LOSW01], [Vyg01]
  - Slacks are not computed properly [Vyg01]
- Main problem: how to use timing information for placement.

Using slacks for net weighting

- Standard approach: define net weights depending on the slack
- Problems: no difference between long and short paths. No sensitivity analysis. No matter how many critical paths pass a net (but see [Kon02])
- Can be interpreted as Lagrange relaxation (see [TK91])
- Works quite well in iterative net weighting schemes
- Widely used in practice in various ways.

Slack Distribution

- Compute individual slacks (or delay budgets) for the nets, such that corresponding delays can be tolerated simultaneously
- Examples are [HNY87], [YLS92], [AKSV99], [CYS00], [KF00] and [Hel01] (optimum slack distribution in a natural well-defined sense)
- Problem: over-constrained (often infeasible) formulation
- Useful for detailed placement.

1.1.3. Power Consumption

- The power consumption (as far as influenced by placement) depends on the weighted sum of all capacitances of all nets
- If a placement is well-routable, then only few detours are necessary
- Hence the weighted sum of optimum rectilinear Steiner tree lengths for all nets is a good measure for power consumption, although it does not take coupling capacitance between wires into account.

1.1.4. Production Cost

- Production cost is influenced by placement in three ways:
  - Die size
  - Number of masks (depends on the number of wiring layers needed to route the placement)
  - Yield (depends on die size and feature density)
- However, the die size is often determined by IOs, DRAMs, etc. Often, the image and the number of wiring planes has to be determined quite early.

1.1.5. Time-to-Market

- As little manual interaction as possible
- Fast turn-around time. Need (almost) linear-time algorithms
- No try-and-error process. If possible, produce a routable placement meeting all timing constraints in a single run
- If this is impossible, we would like to know why.

1.1.6. Allow for late netlist changes

- Retain flexibility. No rigid structure
- Algorithms should produce similar results on similar inputs (stability)
- Legalizing a placement with minimum movement is an important task.

1.2. Feasible Placements

- Problem Instance:
  - a (rectangular) chip area
  - a set of (rectangular) blockades
  - a set of circuits/cells/modules/components to be placed, each associated with a (rectangular) area
  - netlist and further information on technology, timing, routing, yield impact, etc.

Overlap-free placements

- A **placement** of the cells consists of
  - x- and y-offset for each cell
  - possibly mirroring and rotation flags
- and is **feasible** if
  - each cell is within the chip area
  - no cell overlaps any blockade
  - no two cells overlap each other
- Disjointness of *n* axis-parallel rectangles can be checked in $O(n \log n)$ time.

Row-wise Placement

- To facilitate power supply, most cells have a unit height, and these should be placed within rows (standard cells)
- Realizing a certain logical function in gate arrays at a certain place can be regarded as placing a cell there. Gate array designs, too, are usually row-based
- Sometimes there are further local constraints (in particular for FPGAs [Jay01],[SB02]) that must be considered in detailed placement.

1.3. Netlength Measures

- Power consumption can be estimated by the weighted netlength:

$$\sum_{N \in \mathbf{N}} w(N) \cdot \mathrm{STEINER}(N)$$

- where **N** is the set of nets, *w(N)* is the weight of net *N*, and STEINER*(N)* is the length of an optimum rectilinear Steiner tree for *N*
- Often estimates are used instead of STEINER*(N)*.

Why other net models?

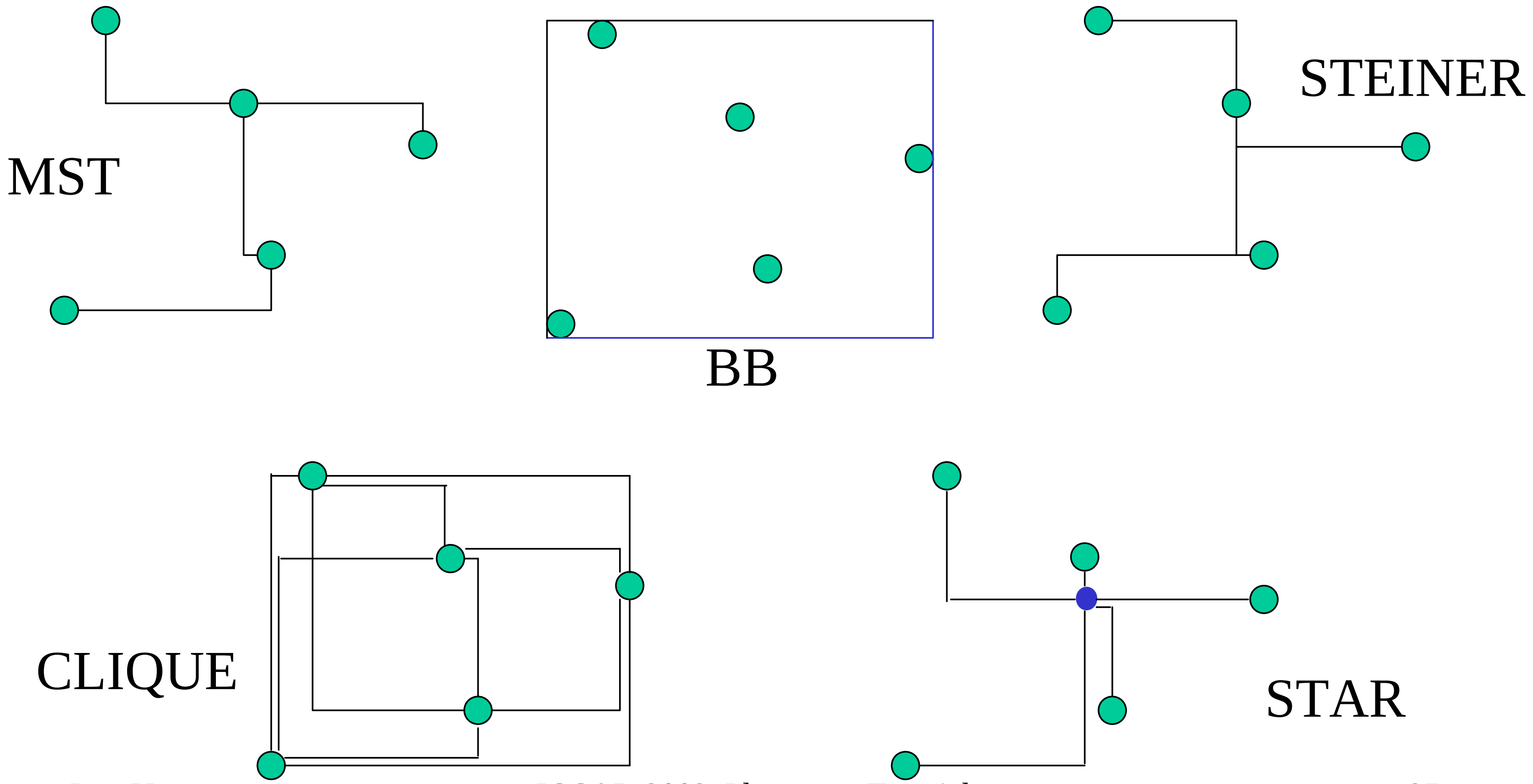
- The length of an optimum rectilinear Steiner tree is NP-hard to compute [GJ77], but this is not really a problem as fast algorithms exist for small terminal sets [WWZ00]
- Simpler estimates are quite accurate
- Steiner trees with variable topology are hard to deal with in optimization.

Commonly used net models

- BB(*N*): half perimeter of bounding box
- MST(*N*): minimum spanning tree
- STAR(*N*): sum of distances of all pins to one auxiliary point
- CLIQUE(*N*): 1/(|*N*|-1) times the sum of distances of all pairs of pins in *N*

(some use factor 2/|*N*| instead of 1/(|*N*|-1).)

The 5 net models: an example



Properties of these net models

- BB and STAR can be computed in $O(|N|)$, CLIQUE and MST in $O(|N| \log |N|)$ time
- Accurate worst-case bounds are known for all five models [BV01]
- The CLIQUE model is the one with best worst-case ratios to STEINER, among all net models with fixed topology [BV01].

1.4 Global Placement Problem

- *Simplest yet useful formulation:* Find a feasible placement such that the weighted netlength is minimum
- One has to take into account (at least):
  - routability, e.g. by density constraints (minimizing netlength tends to produce local routability problems, hot spots)
  - timing constraints, e.g. by clever net weights.

1.5. Detailed Placement Problem

- *Simplest yet useful formulation:*
  Given an infeasible placement, find a feasible one with as little changes as possible
- A reasonable measure is the weighted sum of squared movements of all cells
- Delay budgets and local densities and local routability can be taken into account.

2. Global Placement Approaches

- Theoretical Results
- Local Search
- Recursive Partitioning, Min-Cut
- Analytical Placement
- Discussion

2.1. Theoretical Results

- Checking whether a feasible placement exists is NP-hard [Kar72] – but this is irrelevant
- Even a special case of one-dimensional placement (optimum linear arrangement) is NP-hard [GJS76]
- Unit size cells and restriction to 2-terminal nets do not seem make the problem easier.

Approximation algorithms

- The minimum area needed for a routable placement can be approximated up to a factor of $O(\log^4 n)$ in a special case [EGS00]
- Without considering routability, the netlength can be approximated up to a factor of $O(\log n \log \log n)$ [ENRS00]
- Total netlength and maximum length of a net can be approximated up to a factor of $O(\log^{3.5} n)$ simultaneously w.h.p. [Vem98,Bre00]
- Here n is the number of cells to be placed.

Conclusions from Theory

- The above-mentioned results do not help at all!
- It is not known whether it is NP-hard to give better approximation guarantees
- There are no reasonable lower bounds for placement! For example, if you have placed your chip with 500 m total netlength, you cannot prove that 50 m is impossible!
- This makes it very hard to judge placement quality.

Approaches in Practice

- Local search, in particular simulated annealing
- Recursive partitioning, in particular min-cut
- Analytical (most notably, quadratic) placement
- These are the only three basic approaches used in practice, for more than 20 years.

2.2. Local Search

- Start with an arbitrary placement
- Successively check randomly (or heuristically) chosen modifications:
- Accept them if they improve the placement
- Otherwise you may or may not accept them
- Always keep in mind the best placement obtained so far and restore it if appropriate.

Local Search: General Remarks

- Widely used for discrete optimization problems in the absence of better algorithms
- Advantage: flexible, easy to implement, many constraints and objective functions can be incorporated
- Disadvantage: Often poor convergence, very long runtimes to obtain good results, lack of stability (to be explained later).

Local Search Metaheuristics

- Classical Local Search: Only accept solution if better
- Metropolis filter: Accept solution if better, otherwise, if cost increases by $p>0$, we accept with probability $e^{-p/T}$ [Met53]
- Simulated Annealing: Start with very large T, and then decrease T (the „temperature“) down to 0 [Kir83]
- Many variations have been tried: „Tabu Search“, „Genetic/Evolutionary“ Algorithms, ...

Simulated Annealing

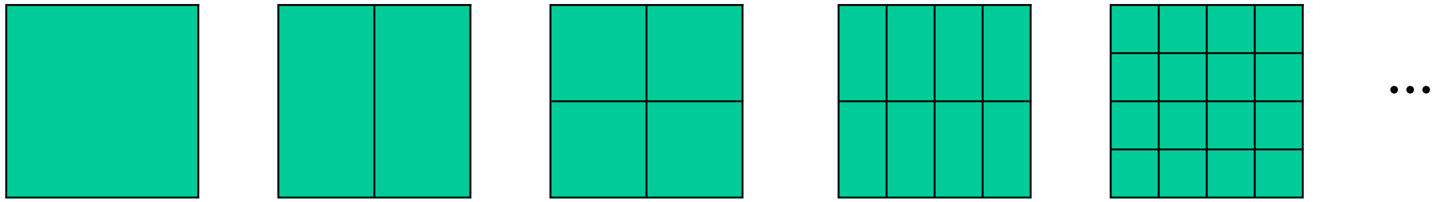
- Applied to placement for a long time: [SSV85], [SL87], [WLL88], [Sec88], [Len90]
- Provably converges to optimum solution if cooling schedule is slow enough (see [Len90] for details)
- However, then the running time is not much better than complete enumeration!
- Tends to work quite well for small instances (floorplanning) and complicated objectives
- Too slow for flat placement of today‘s chips.

Local Search for Postopt

- Local Search can be quite effective in improving a placement obtained by a different technique [FPZ01]
- But even here it is not trivial to obtain significant improvements fast enough
- Complex objectives can be incorporated, but changes have to be evaluated very fast.

2.3. Recursive Partitioning

- Standard approach: Divide-and-conquer
- Divide the chip area into 2 or 4 parts and assign each component to one of the parts
- Iterate this until the parts are small enough.



Partitioning Problem

- Given a set **C** of components, and a number size(C)>0 for component C, an integer m>1, and capacities cap(1),...,cap(m),
- Find a partition f : $\mathbf{C} \rightarrow \{1,...,m\}$ such that

$$\sum_{C\in\mathbf{C}: f(C)=i} \text{size}(C) \le \text{cap}(i)$$

for i=1,...,m (called a feasible partition).

Feasible Partitions

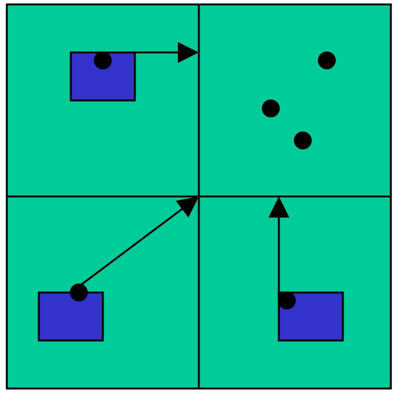
- Capacity of a region „cap(*i*)“ may be less than the usable area (avoiding too high density, possibly causing routing congestion)
- Area consumption „size(*C*)“ may be more than the actual size (anticipating routing problems if placed too densely)
- Deciding if a feasible partition exists is NP-hard [Kar72], but this is irrelevant as slight capacity violations can be tolerated.

What is a good partition?

- How do we measure the quality of a partition?
- Most popular objective: Min-Cut [Bre77],[DK85], [CKM00],[WYS00],[YM01]
- Minimize the number (or total weight) of nets that are separated. For m>2, take the number of parts to which the net´s pins are assigned into account.
- Other objective: minimize changes of analytical placement (see below)

Terminal Propagation

- If a net contains pins outside the region to be partitioned, these are treated as belonging to the nearest subregion to which we partition
- If we recursively apply min-cut bipartitioning, then the sum of the objective values equals the final BB netlength.



Disadvantages of Min-Cut

- Minimizing the first cut may lead to very poor subsequent cuts
- Nobody knows how to find a min-cut (s.b.)
- „1-dim approach to a 2-dim problem“ (does not apply to 4-way min-cut)
- „discrete approach to an almost continuous problem“
- Lack of stability (to be explained later).

Special Case: Graph Bisection

- Given a graph G, partition $V(G)=V_1 \cup V_2$ with $|V_1|=|V_2|$ and minimum number of edges between V_1 and V_2
- NP-hard [GJS76] and not appproximable with an additive error of $|V(G)|^r$ for any $r<2$ unless P=NP [BJ92]
- However, not known to be MAXSNP-hard
- Best approximation algorithm achieves a multiplicative error of $O(\log^2|V(G)|)$. [FK00]

Min-Cut Heuristics

- Exchange heuristics based on Kernighan-Lin [KL70] and Fiducccia-Mattheyses [FM82], often combined with multi-level clustering ([CS93],[HL95]), are most commonly used
- Hundreds of variants, but no performance guarantee can be proved for any of them
- Often 4-way instead of 2-way partitioning [SK89]
- [AK95] is a good survey on partitioning.

2.4. Analytical Placement

- Main Idea: First minimize netlength without caring about overlaps
- Then – somehow – work to remove overlaps
- Works only if there are some fixed pins (primary I/Os, possibly macros).

Linear netlength minimization

Theorem: BB netlength can be minimized in $O(n \log n\ (m+n \log n))$ time, where n is the number of components/nets and m is the number of pins

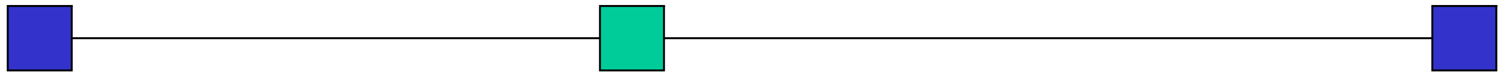
Proof: The linear program is the dual of an uncapacitated minimum cost flow problem (see [AMO93, KV02]). Apply the fastest min-cost flow algorithm [Orl93,Vyg99].

Applicability of linear netlength minimization

- Used in early times when chips were small [JK88]
- Today far too slow
- But usable for small subsets of circuits, combined with local search [HL99],[HL00]

Linear Netlength Minimization: Drawbacks

- Too slow for state-of-the-art chips
- The optimum solution is not unique:



- For a basic optimum solution, many components will have exactly the same position. In this case we get no information on left/right/bottom/top relation.

Quadratic Netlength Minimization

$$\min \sum_{N \in \mathbf{N}} \frac{w(N)}{|N|-1} \sum_{p,q \in N} \left(x(p) + x(C(p)) - x(q) - x(C(q)) \right)^2$$

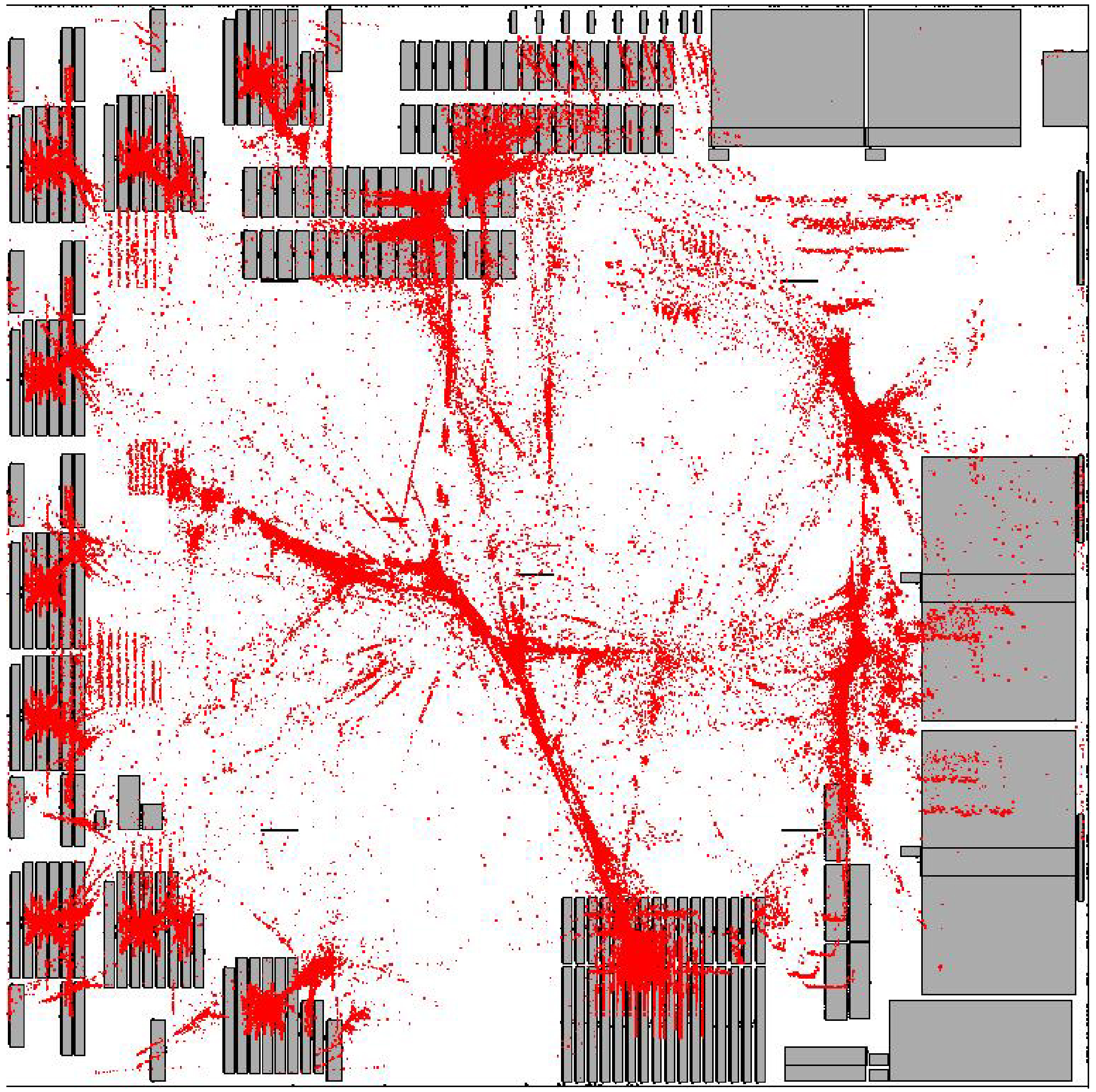
where

- **N** is the set of nets,
- |*N*| the number of elements (pins) of net *N*,
- *x(p)* is the horizontal offset of pin *p* w.r.t. *C(p)*,
- *C(p)* is the component to which pin *p* belongs; *C(p)*=0 if pin is fixed (e.g. primary I/O), *x(0)=0.*

Quadratic Placement

- Optimize x- and y-coordinates separately: 2 QPs
- If each connected component of the netlist contains at least one fixed pin, then these QPs are strictly convex and have unique optimum solutions *(x,y)*
- We call this solution the **quadratic placement**
- It can be found fast, e.g. by the conjugate gradient method [HS51] (theoretically *O(nm)* time, but we get very good approximations in *O(m)* time).

A typical result of quadratic placement (grey macros fixed)



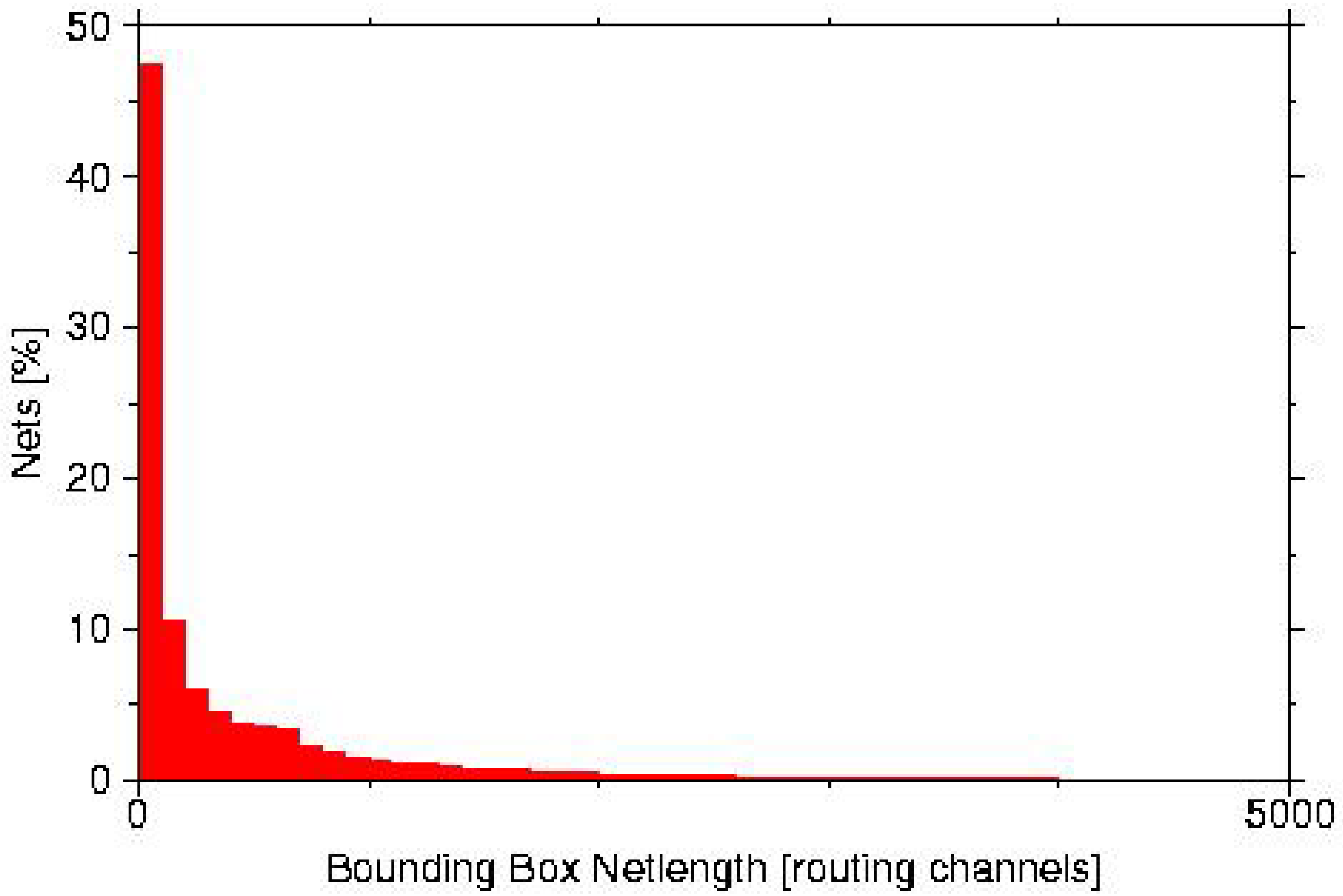
Quadratic Placement

- The basic idea is probably due to Tutte [Tut63], who used it for drawing planar graphs.
- Has been used in layout automation for a very long time; see e.g. [FCW67],[CK84]
- Many tools are based on QP, for example [TKH88], [KSJA91], [Vyg97], [EJ98].

Advantages of QP

- Fast (10 minutes for 3M movable objects)
- Unique optimum solution, stable
- Gives a lot of information on relative positions
- 2-dimensional, continuous approach. Particularly appropriate when there are many, small components (and large ones are preplaced on the edge of the chip).

Netlength Distribution on Salvina (1.7M Nets)



(1 routing channel = 0.4 µm – chip image 13.8 x 13.9 mm)

Disadvantages of QP

- Optimizes quadratic instead of linear netlength
- Result changes by buffer insertion
- Note that power consumption and delay (with optimum buffer insertion) grow only linearly with the netlength.

Try to combine LP and QP

- QP is appropriate after optimum buffering
- However, this is not known initially
- GordianL: Approximate LP successively by QPs [SDJ91]. But: substantial runtime overhead even with ideas of [ACK+98]
- Split up nets (e.g. at cuts [Vyg97]) or, equivalently, insert buffers when nets become too long.

Remove overlaps after QP

- Linear Assignment Problem (poor results)
- Recursive Partitioning
  - either using min-cut again (e.g. GORDIAN, [KSJA91])
    - but then QP is not really necessary ([ACC+99])
  - or using just the geometric information: minimizing movement
- Or modify the QP: „adaptive forces“ [EJ98]

We will discuss the last two approaches in detail.

2.5. Discussion

- Approaches: local search, min-cut, QP
- Criteria: quality of results, flexibility, turn-around time, ability to work without fixed pins, stability

	quality	flex.	TAT	free IO	stability
Local search	+	+ +	– –	+	– –
Min-cut	+	+	+	+	– –
QP	+ +	+	+	–	+

Stability

- The placement process itself should be stable (as local objectives will change). This is more or less the case with all approaches
- Similar netlists should lead to similar placements. Let T be a placement tool, and let C_1 and C_2 be two versions of a chip (say C_2 results from C_1 by a few netlist changes). Then we want $T(C_2)$ to be similar to $T(C_1)$.

Stability of Placement Approaches

- QP is stable: The discrepancy of the QPs of two similar netlists can be bounded
- Min-cut and local search are instable. Even an algorithm always computing the optimum placement can be very instable: For a single netlist, several completely different optimum placements can exist! [Vyg01],[Vyg02]
- This is a reason to favour quadratic placement.

Placement of Macros

- In practice, very large macros (cores or memory arrays) are usually placed manually in advance
- Most automatic placement tools can handle large objects, but often do not very well
- A good solution, e.g. a „planning tool“ that skilfully enumerates many macro placements and estimates the effect accurately would be very useful in practice
- However, this problem seems to be very hard.

3. More on Quadratic Placement

- Recursive Partitioning Based on QP
- QP with Respect to Regions
- Force-Directed Placement
- Concluding remarks on global placement

3.1. Recursive Partitioning Based on QP

- Min-cut: but then QP may be obsolete [ACC+99]
- Alternative: find a feasible partition *f* with minimum total movement

$$\sum_{C \in \mathbf{C}} \mathrm{size}(C)\, \mathrm{d}(C, f(C))$$

where d(C,i) denotes the L_1-distance from the position of C to region i

- As it is NP-hard to find any feasible partition, we relax the problem:

Fractional Partitions

- We ask for a fractional partition

  $g : C \times \{1,\ldots,m\} \to [0,1]$ with $\sum_{i=1}^{m} g(C,i) = 1 \ (C \in \mathbf{C})$
- meeting the capacity constraints

  $\sum_{C \in \mathbf{C}} \mathrm{size}(C) g(C,i) \le \mathrm{cap}(i)$ for $i = 1,\ldots,m$
- and minimizing total movement

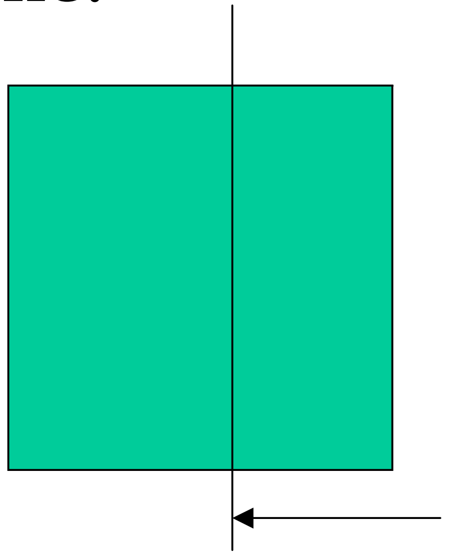
$$\sum_{C \in \mathbf{C}} \mathrm{size}(C) \sum_{i=1}^{m} \mathrm{d}(C,i) g(C,i)$$

Optimum fractional partition

- Can be found by min-cost flow for any *m* in $O(n^2 \log^2 n)$ time, where *n* is the number of components
- Too slow for today´s chips ($n > 10^6$)
- Only *m-1* objects will get a fractional value – these can be assigned arbitarily.

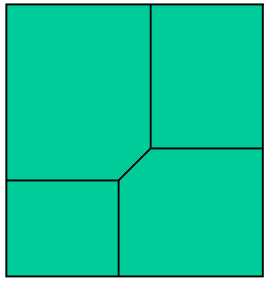
Optimum Bipartioning

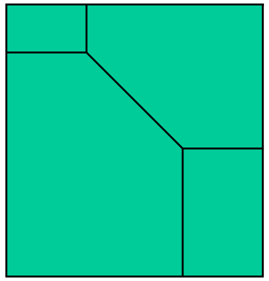
- *m*=2: „moving cut line“ finds the optimum. Equivalently: weighted median. Optimum can be found in *O(n)* time. [Dan57], [BFPRT73], see [KV02]
- Used by [TKH88]
- But: which coordinate first?

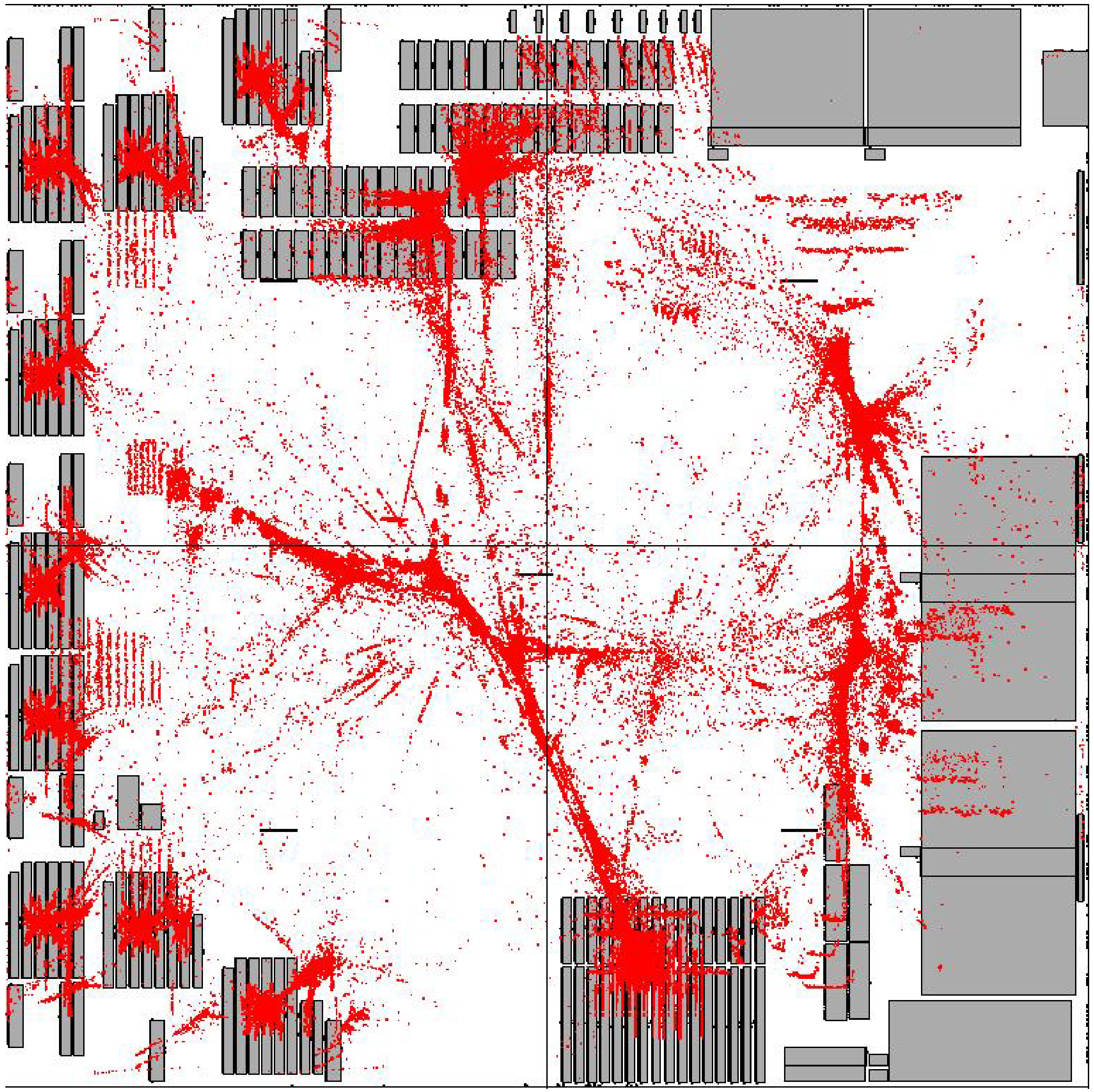


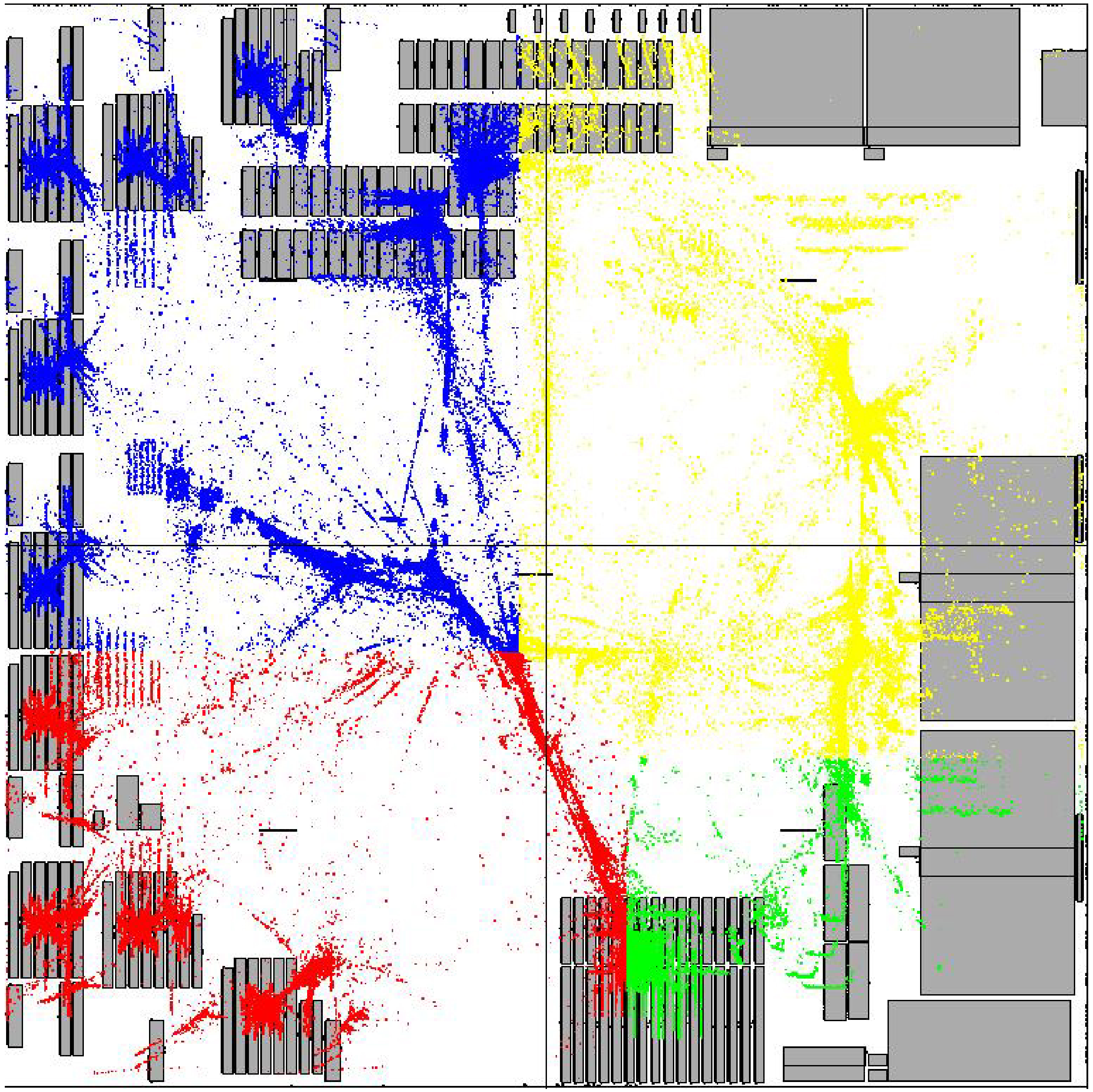
Quadrisection

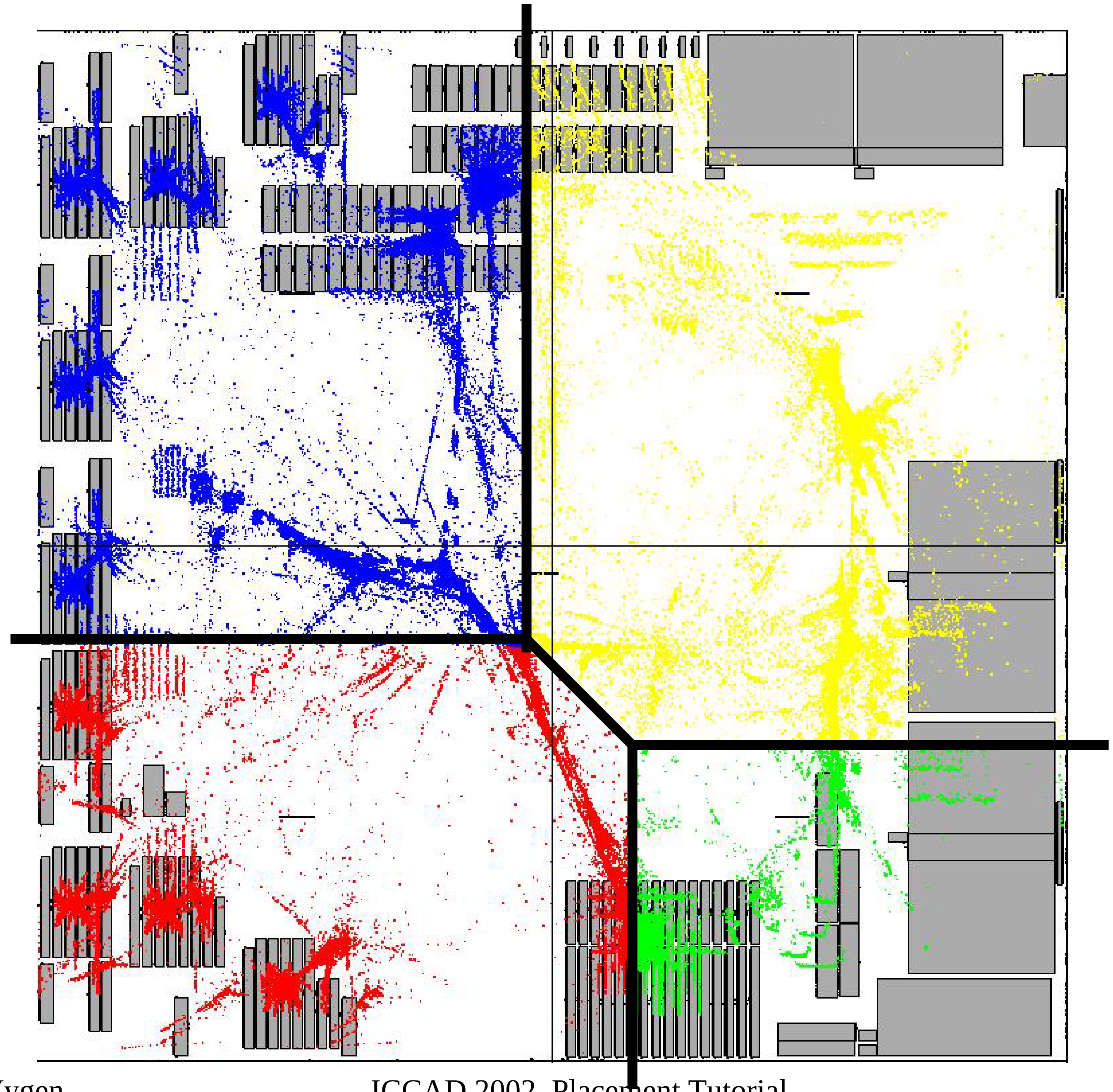
- For m=4 (2x2-partition), the optimum fractional partition has the structure of an „American map“
- It can be found in $O(n)$ time
- Only 3 objects will get a fractional value. Round these arbitrarily
- See [Vyg96,Vyg97,Vyg00]
- Much better than three times bipartitioning!





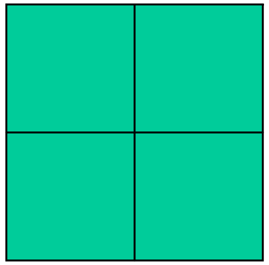






3.2. QP with Respect to Regions

When components are assigned to $m>1$ regions of the chip, how to solve the QP without losing this information?

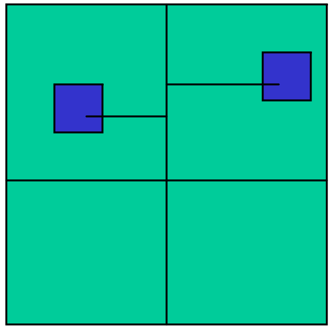


Ad-hoc approaches:

- *m* independent QPs with terminal propagation (but then we get poor results)
- Add linear inequality constraints (but then the QP is much harder to solve).

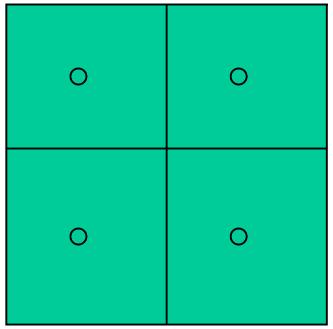
Better Ideas

- One QP, splitting up nets at cuts [Vyg97]: if p,q in different columns, replace $(x(p)+x(C(p))-x(q)-x(C(q)))^2$ by $(x(p)+x(C(p))-x_{cut})^2+(x_{cut}-x(q)-x(C(q)))^2$



- One QP with center-of-gravity constraints [KSJA91]:

$$\sum_{C\in\mathbf{C}: f(C)=i} \text{size}(C)x(C) = \sum_{C\in\mathbf{C}: f(C)=i} \text{size(C)}\,\text{center}(i)$$



Advantages

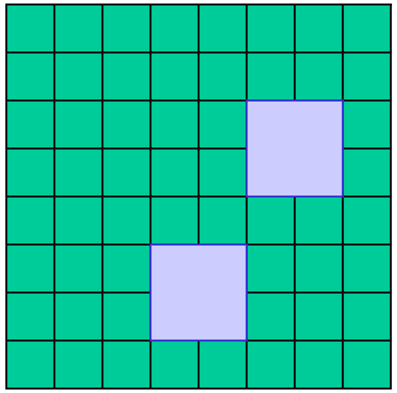
- Splitting up nets at cuts
  - No too full regions can occur
  - Implicit linearization of the objective function
  - QP will be solved even faster
  - No even distribution on the whole chip area is assumed.
- Center-of-gravity constraints
  - Components may move to other regions, leaving the possibility of revoking early decisions
  - More information due to better spreaded placement.

Some Hints

- Use both: first split up nets at cuts, then add cog-constraints where too little information
- The cog should not always be the region center
- Be careful with the CG method with cog-constraints: don´t lose sparsity of your matrix (see [KSJA91])
- Replace large cliques equivalently by stars!

Repartitioning

- The possibility of revising early decisions is essential
- Repartioning:
  - consider all 2x2-windows
  - do a single QP without splitting up nets internally,
  - apply a new Quadrisection
  - apply another QP with splitting up nets according to new partition
  - accept new partition if better.



3.3. Force-Directed Placement

- Idea: QP is equivalent to solving linear equation system $Ax=b$
- The placement x^* we look for (without any overlaps) satisfies $Ax^*=b^*$
- Successively modify b (hopefully towards b^*) in order to remove more and more overlaps.

Adding Forces

- To each component we apply a force, pulling it to a direction of „most unused space“
- The force applied to component *C* depends only on the current position $(x(C), y(C))$ of *C*
- A force into the direction $(f_x(C),f_y(C))$ is modelled as a 2-terminal net pulling *C* to the direction of the force.

How to apply forces

- Current horizontal placement x is a solution to the linear equation system $Ax=b$
- Adding a 2-terminal net of weight w pulling component C to a point that is $f_x(C)$ to the right of the (new) position of C, resulting in a new placement x', means adding w to $A_{C,C}$ and adding $w(x'(C)+f_x(C))$ to b_C. We get the system $A^+x'=b^+$
- Subtracting $wx'(C)$ from both sides of $A^+x'=b^+$ yields $Ax'=b'$, where b' results from b by adding $wf_x(C)$ to the component corresponding to C.

Adaptive Forces Algorithm [EJ98],[Eis99]

(1) Compute matrix A and r.h.s. vectors b_x,b_y as usual

(2) Compute solution to $Ax=b_x$ and $Ay=b_y$

(3) If only few overlaps, stop (and legalize)

(4) Compute force vectors f_x and f_y

(5) Set $b_x := b_x + wf_x$ and $b_y := b_y + wf_y$

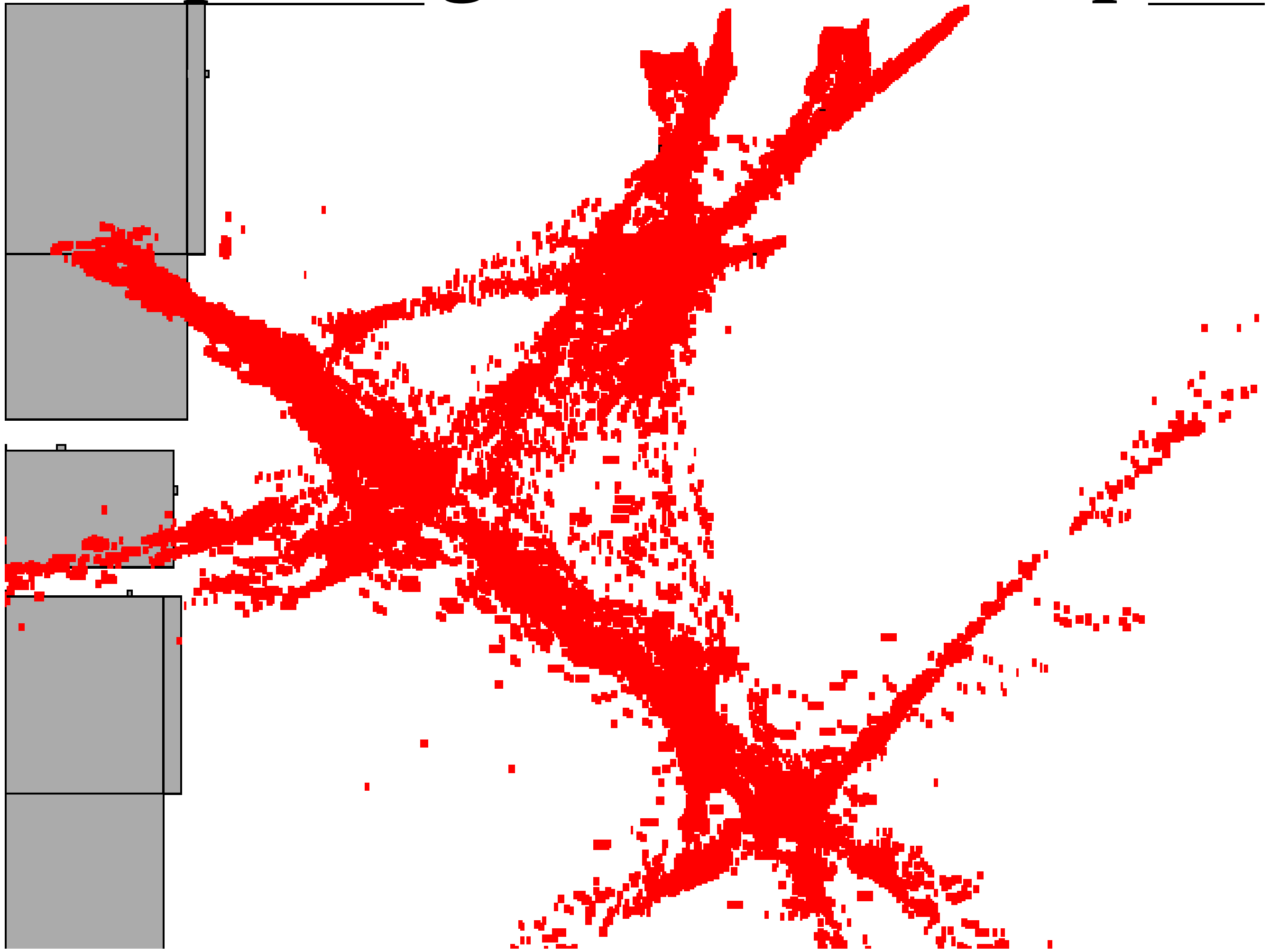
(6) Go to (2).

Where do forces apply?

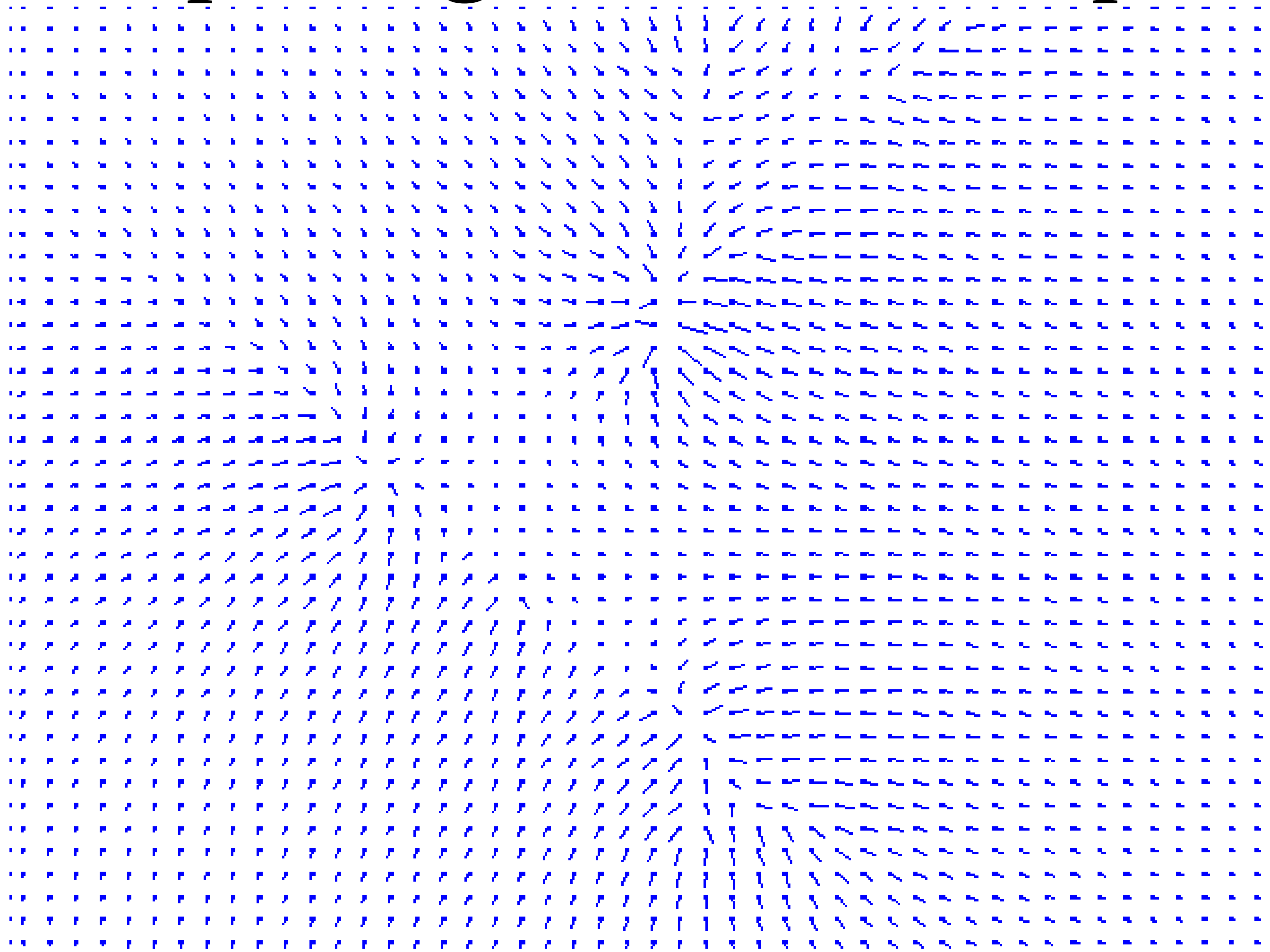
- Let *D(p)* be the number of circuits that overlap point p minus the target number (e.g. 0.7 if p is in a usable area with target density 70%)
- Each point p applies a force to each other point p´ (attracting or repelling), namely

$$D(p)\frac{p'-p}{\|p'-p\|^2}$$

Repelling forces: example



Repelling forces: example



How to compute the forces

- The total force applied to a component at position *p´* is

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} D(p_x, p_y)\frac{p' - (p_x, p_y)}{\left\| p' - (p_x, p_y) \right\|^2}\,\mathrm{d}p_x\mathrm{d}p_y$$

- The integral can be computed in *O(n)* time, where *n* is the number of rectangles (movable and fixed components). All forces in $O(n^2)$ time exactly or in *O(n)* time approximately with the fast multipole method [GR87]
- Details in [EJ98],[Eis99],[Bre00].

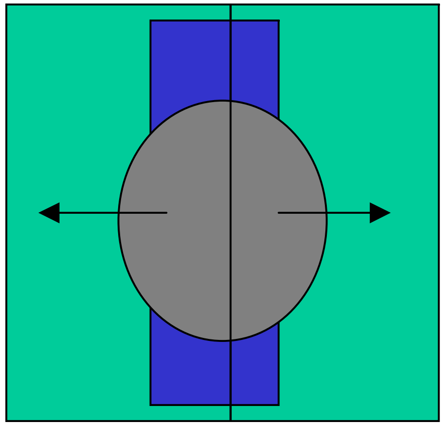
Summary: QP-Based Algorithms

- QP+Quadrisection: faster, probably better results except for large macros, implicit linearization of objective function, no target densities needed
- Adaptive Forces: better (though not perfect) handling of macros, even more stable
- Disadvantages of both: need fixed I/O pins, stick to the QP order quite strictly, can separate random logic groups when macros are in their way.

A situation where min-cut is appropriate

- After QP, a highly connected group of random logic lies right on top of a macro
- With Quadrisection and Adaptive Forces, this group may be separated into two parts, which is usually bad
- In such a case, min-cut

  seems appropriate.



3.4. Combining global placement strategies – a possible way

- Use QP plus Quadrisection in general.
- Apply a few steps of the Adaptive Forces algorithm when there is too little information
- Use min-cut in special situations where 0-1 decisions have to be made
- Use local search for local improvement in the end.

A remark on benchmarks

- Most benchmarks for which results are reported are old, small, and artificial
- Results are often not comparable [Mad01]
- Industry is reluctant publishing netlists
- But even with proper data, placement tools are very hard to compare
- Often, designers faced with most complex chips know best which tools are best.

So how to measure quality?

- There are three ways:
  - convincing new ideas, theory or algorithms
  - proven use for solving real problems
  - benchmarks
- The third possibility is usually the least convincing one. Much more care should be used than often seen whenever benchmarks shall be the main proof of relevance.

4. Detailed Placement

- Detailed Placement Approaches
- Min-Cost Flow Techniques
- Single-Row Placement

4.1. Detailed Placement Approaches

- Low temperature simulated annealing (can be quite time-consuming), possibly based on sequence-pair representation
- Greedy approaches (can be quite bad)
- Min-Cost-Flow-based algorithms
- Single-row algorithms
- Special FPGA algorithms, e.g. [SB02].

Objectives in Detailed Placement

- Main goal: make placement feasible with as little movement as possible (legalization)
- But on this detailed level we may also take concrete objectives into account (in particular timing and routability)
- Here we concentrate mainly on legalization
- We assume that macros (components that do not fit into a row) are fixed.

Legalization with Minimum Movement

- Is also useful in case of ECOs
- Timing optimization after placement will usually require another legalization
- Clocktree synthesis is usually done after global placement. It is important that critical clock drivers (as well as other critical components) are not moved far away
- Give these components a higher weight.

Placement Within Rows

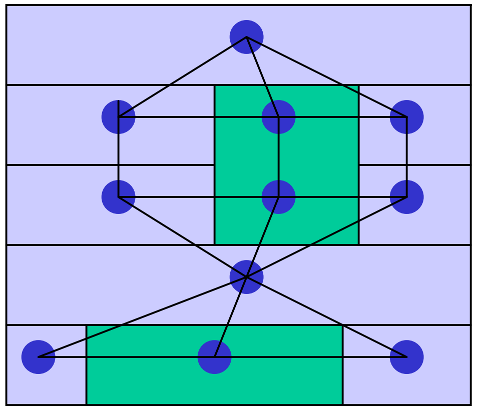
- We have to place all components within rows
- Define a *zone* to be a maximal part of a row that is completely usable or completely blocked
- Goal of the first phase: assign components to zones such that a legal placement within each zone is possible
- Second phase: place components within each zone separately.

4.2. Min-Cost-Flow: Direct Assignment

- If all components have unit size and we want to minimize weighted total movement, this is a linear assignment problem, which can be solved by min-cost-flow in $O(n^2p)$ time, where n and p are the number of components and positions, respectively
- But: too slow. Moreover, it is better to move 10 objects by 1 unit rather than 1 object by 10 units
- Nevertheless the basic idea is used in several detailed placement algorithms: [DJA94], [KWO+97],[Vyg98],[NC99],[BV02].

Min-Cost Flow between Zones

- Define a graph, whose vertices are zones and blockades, and an edge joins two vertices that have a common border
- Some zones contain too many components (the sources), and others have free space (the sinks)
- We look for a flow from the sources to the sinks.



Min-Cost Flow Problem

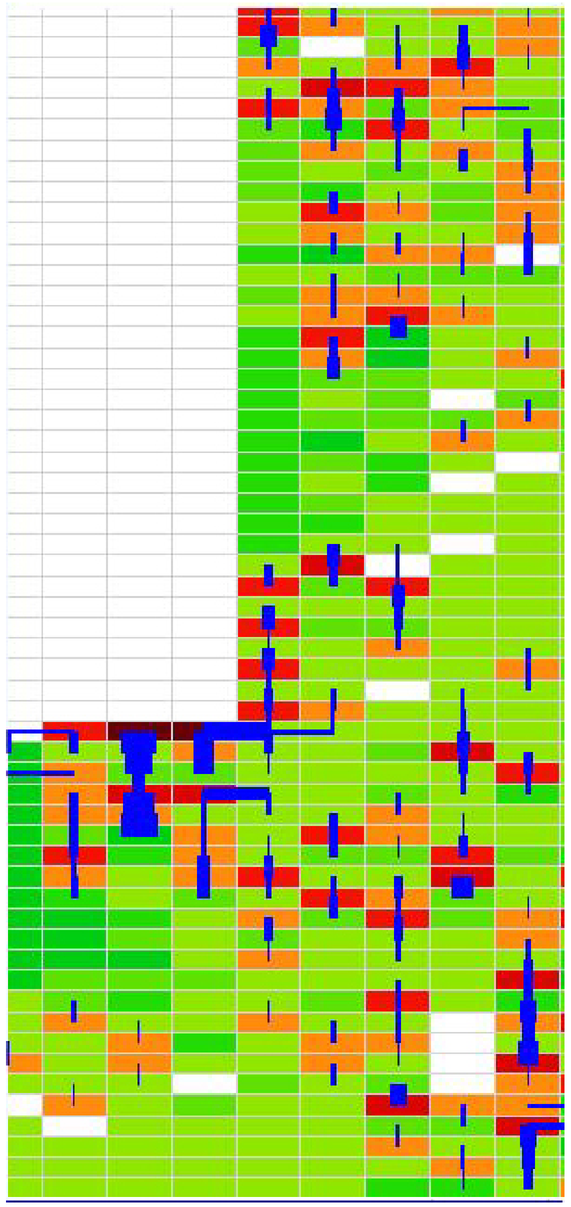
- For each vertex *v* let

$$b(v) := \sum_{C \in v} \text{size}(C) - \text{cap}(v)$$

- *v* is a source if *b(v)* is positive
- Let the cost of an edge be an estimate of the cost of moving 1 unit along this edge (should be >0). All capacities are infinite
- This defines a min-cost flow instance.

Moving flow

- The min-cost flow problem can be solved in $O(n^2 \log n)$ time, where n is the number of zones
- The edges carrying flow (in a specific direction) form an acyclic digraph
- Scan these edges in topological order
- For each edge: realize the flow.



Example: flow between regions

red/orange: source regions
green: sink regions
blue: flow

Realizing the flow

- For a flow of k units from vertex v to adjacent vertex w, we have to move components of total size k from v to w.
- Note: this is not always possible exactly.
- The best subset of components to move can be computed efficiently by solving a knapsack problem (by dynamic programming).

Problems and Solutions

- Non-exact realization can cause too full zones again (iterate, use heuristics to guarantee termination)
- Components may have to be moved from the „far end“ of a zone. The zones can be very wide!
  - Subdivide wide zones
  - „Soft boundary“ between subzones
- See [Vyg98],[BV02] for details.

4.3. Single-Row Placement

- Even the Optimm Linear Arrangement problem (a special case of one-dimensional placement) is NP-hard [GJS76], but has at least an $O(\log n)$-approximation [RR98]
- One-dimensional placement can be solved in polynomial time
  - for trees (but this is quite useless)
  - for general netlists if the order is fixed.

Single Row, Fixed Order

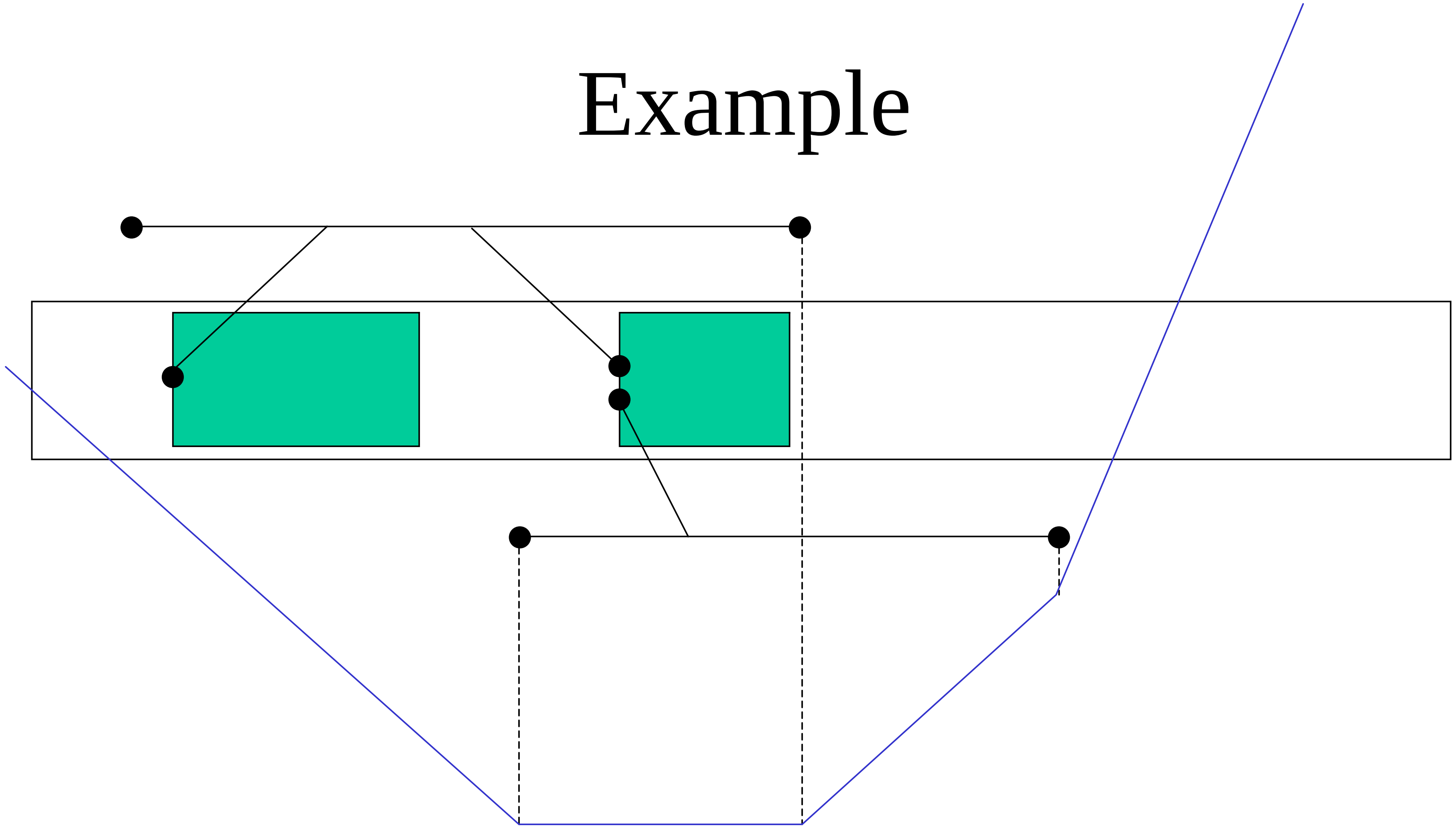
- Fix the horizontal order according to the last QP.
- Then minimizing BB netlength or total movement is the dual of a min-cost flow problem
- Thus it can be solved in $O(n^2\log^2 m)$ time, where *m* is the number of nets involved
- This even works for all zones simultaneously (saves ~2% when minimizing BB netlength)
- Faster: each zone separately. Then a dynamic programming algorithm [KTZ99] can be used.

Dynamic Programming Algorithm

- Minimize BB netlength (movement is easier)
- The BB netlength – as far as it can be optimized by placement of current zone – can be written as $\sum_{i=1}^{n} \mathrm{cost}_i(x_i)$

  where x_i denotes the coordinate of the *i*-th component and cost_i is a piecewise linear convex function ($i=1,...,n$).

Example



Clumping Algorithm [KTZ99]

- For $i=1$ to n: PLACE(i).
- PLACE(i):
  - If i can be placed to the right of $i\text{-}1$ such that $cost_i$ is minimized, then do so (in case of a tie leftmost)
  - Otherwise combine $i\text{-}1$ and i to a single movable object $i´$ and PLACE($i´$).

Running time

- Efficient implementation [BV00,BV02] makes this algorithm very fast:
- Minimizing BB netlength or total movement takes
  - $O(m \log m \log \log m)$ with units weights
  - $O(m \log^2 m)$ with arbitrary weights
- Total squared movement can be minimized in linear time.

Postoptimization

- Optimum cell flipping is NP-hard [CYH91], but heuristics can be useful for routability [BHMR99]
- A legal placement can be optimized by exchange heuristics (e.g. [HL99], [FPZ01])
- For timing optimization, it is useful to consider the critical path/cycle. But it is already NP-hard to place a single object with three pins optimally [Vyg01]. Simplifications (e.g. weighted BB netlength) must be used.

PostOpt Heuristic

- Take a subset of circuits
- Place them optimally (say w.r.t. weighted BB netlength) without caring about overlaps
- Legalize the resulting placement

- [HL99],[HL00] proposed variants of this heuristic for the 1- and 2-dim placement (minimizing netlength), in a local search framework.

References

[ACC+99] Alpert, C.J., Caldwell, A.E., Chan, T.F., Huang, D.J.-H., Kahng, A.B., Markov, I.L., Moroz, M.S.: Analytical engines are unnecessary in top-down partitioning-based placement. VLSI Design 10 (1999), 99-116

[ACK+98] Alpert, C.J., Chan, T.F., Kahng, A.B., Markov, I.L., Mulet, P.: Faster minimization of linear wirelength for global placement. IEEE Trans. on CAD 17 (1998), 3-13

[AMO93] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice-Hall, Englewood-Cliffs 1993

[Alb01] Albrecht, C.: Global routing by new approximation algorithms for multicommodity flow. IEEE Trans. on CAD of Integrated Circuits and Systems 20 (2001), 622-632.

[AKSV99] Albrecht, C., Korte, B., Schietke, J., Vygen, J.: Cycle time and slack optimization for VLSI chips. Proc. ICCAD 1999, 232-238. Extended version to appear in Discr.Appl.Math. 123 (2002), 103-127

[AK95] Alpert, C.J., Kahng, A.B.: Recent directions in netlist partitioning: a survey. Integration, the VLSI journal 19 (1995), 1-81

[BFPRT73] Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., Tarjan, R.E.: Time bounds for selection. J.Comp.Sys Sci. 7 (1973), 448-461

[BHMR99] Boros, E., Hammer, P.L., Minoux, M., Rader jr., D.J.: Optimal cell flipping to minimize channel density in VLSI design and pseudo-Boolean optimization. Discr.Appl.Math. 90 (1999), 69-88

[BJ92] Bui, T.N., Jones, C.: Finding good approximate vertex- and edge-partitions is NP-hard. Inf.Proc.Letters 42 (1992), 153-159

[Bre77] Breuer, M.: Min-cut placement. J. of Design, Automation and Fault-Tolerant Computing 1 (1977), 343-382

[Bre00] Brenner, U.: Plazierung im VLSI-Design. Diploma thesis, University of Bonn 2000

[BR02] Brenner, U., Rohe, A.: An effective congestion-driven placement framework. ISPD 2002, 6-11

[BV00] Brenner, U., Vygen, J.: Faster optimal single-row placement with fixed ordering. Proc. DATE 2000, 117-121

[BV01] Brenner, U., Vygen, J.: Worst-case ratios of net-works in the rectilinear plane. Networks 38 (2001), 126-139

[BV02] Brenner, U., Vygen, J.: Legalizing a placement with minimum total movement. Manuscript. To appear in IEEE TCAD 23 (2004)

[CK84] Cheng, C.K., Kuh, E.S.: Module placement based on resistive network optimization. IEEE Trans. on CAD 1984, 218-225

[CKM00] Caldwell, A.E., Kahng, A.B., Markov, I.L.: Can recursive bisection alone produce routable placements? Proc. DAC 2000, 153-158

[CS93] Cong, J., Smith, M.L.: A parallel bottom-up clustering algorithm with applications to circuit partitioning in VLSI design. Proc. DAC 1993, 755-760

[CYH91] Cheng, C.-K., Yao, S.Z., Hu, T.C.: The orientations of modules based on graph decomposition. IEEE Trans. on Computers 40 (1991), 774-780

[CYS00] Chen, Y., Yang, X., Sarrafzadeh, M.: Potential slack: an effective metric of combinational circuit performance. Proc. ICCAD 2000, 198-201

[Dan57] Dantzig, G.B.: Discrete-variable extremum problems. Operations Research 5 (1957), 266-277

[DJA94] Doll, K., Johannes, F.M., Antreich, K.J.: Iteratice placement improvement by network flow methods. IEEE Trans. on CAD of IC and Systems 13 (1994), 1189-1199

[DK85] Dunlop, A.E., Kernighan, B.W.: A procedure for placement of standard cell VLSI circuits. IEEE Trans. on CAD 4 (1985), 92-98

[Eis99] Eisenmann, H.: Ein universelles Plazierverfahren für integrierte Schaltungen. PhD thesis, Technical University of Munich 1999

[EJ98] Eisenmann, H., Johannes, F.M.: Generic global placement and floorplanning. Proc. DAC 1998, 269-274

[EGS00] Even, G., Guha, S., Schieber, B.: Improved approximations of crossings in graph drawings and VLSI layout areas. Proc. Symp. on the Theory of Computing 2000, 296-305

[ENRS00] Even, G., Naor, J., Rao, S., Schieber, B.: Divide-and-conquer approximation algorithms via spreading metrics. J. of the ACM 47 (2000), 585-616

[FCW67] Fisk, C.J., Caskey, D.L., West, L.E.: ACCEL: automated circuit card etching layout. Proc. of the IEEE 55 (1967), 1971-1982

[FK00] Feige, U., Krauthgamer, R.: A polylogarithmic approximation of the minimum bisection. Proc. Symp. On Foundations of Computer Science 2000, 105-115

[FM82] Fiduccia, C.M., Mattheyses, R.M.: A linear-time heuristic for improving network partitions. Proc. DAC 1982, 175-181
[FPZ01] Faroe, O., Pisinger, D., Zachariasen, M.: Local search for final placement in VLSI design. Proc. ICCAD 2001, 565-572
[GJ77] Garey, M.R., Johnson, D.S.: The rectilinear Steiner tree problem is NP-complete. SIAM J.Appl.Math. 32 (1977), 826-834
[GJS76] Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete graph problems. Theor.Comp.Sci. 1 (1976), 237-267
[GR87] Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comput. Phys. 73 (1987), 325-348
[Hel01] Held, S.: Algorithmen für Potential-Balancierungs-Probleme und deren Anwendungen im VLSI-Design. Diploma thesis, University of Bonn 2001
[HL95] Hendrickson, B., Leland, R.: A multilevel algortihm for partitioning graphs. Proc. Supercomputing 1995
[HL99] Hur, S.-W., Lillis, J.: Relaxation and clustering in a local search framework: application to linear placement. Proc. DAC 1999, 360-366

[HL00] Hur, S.-W., Lillis, J.: Mongrel: hybrid techniques for standard cell placement. Proc. ICCAD 2000, 165-170

[HMS02] Hu, B., Marek-Sadowska, M.: Congestion estimation during placement without estimation. Proc. ICCAD 2002

[HNY87] Hauge, P.S., Nair, R., Yoffa, E.J.: Circuit placement for predictable performance. Proc. ICCAD 1987, 88-91

[HS51] Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. of Research of the National Bureau of Standards 49 (1952), 409-439

[Jay01] Jayaraman, R.: Physical Design for FPGAs. Proc. ISPD 2001, 214-221

[JK88] Jackson, M.B., Kuh, E.S.: Performance-driven placement of cell-based Ics. Proc. DAC 1988, 370-375

[Kar72] Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations (R.E. Miller, J.W. Thatcher, eds.), Plenum Press, New York 1972, pp. 85-103

[KF00] Kourtev, I.S., Friedman, E.G.: Timing optimization through clock skew scheduling. Kluwer, Boston 2000

[Kir83] Kirkpatrick, S., Gelatt jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 230 (1983), 671-680

[KL70] Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell System Technical J. 49 (1970), 291-307

[Kon02] Kong, T.: A novel net weighting algorithm for timing-driven placement. Proc. ICCAD 2002

[KSJA91] Kleinhans, J.M., Sigl, G., Johannes, F.M., Antreich, K.J.: GORDIAN: VLSI placement by quadratic programming and slicing optimization. IEEE Trans. on CAD of Integrated Circuits and Systems 10 (1991), 356-365

[KTZ] Kahng, A.B., Tucker, P., Zelikovsky, A.: Optimization of linear placements for wirelength minimization with free sites. Proc. ASPDAC 1999, 241-244

[KV02] Korte, B., Vygen, J.: Combinatotial Optimization: Theory and Algorithms. Springer 2000 (2nd edition 2002)

[KWO+97] Koide, T., Wakabayashi, S., Ono, M., Nishimaru, Y., Yoshida, N.: A timing-driven placement algorithm with the Elmore delay model for row-based VLSIs. VLSI journal 24 (1997), 53-77

[LOSW01] Lee, J.-F., Ostapko, D.L., Soreff, J., Wong, C.K.: On the signal bounding problem in timing analysis. Proc. ICCAD 2001, 507-514
[Len90] Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. Teubner/Wiley, Chichester 1990
[Mad01] Madden, P.H.: Reporting of standard cell placement results. Proc. ISPD 2001, 30-35
[Met53] Metropolis, N., Rosenblut, A., Rosenbluth, M., Teller, A., Teller, E.: Equation of state calculation by fast computing machines. J.Chem.Phys. 21 (1953), 1087-1092
[NC99] Nag, S., Chaudhary, K.: Post-placement residual-overlap removal with minimal movement. Proc. DATE 1999, 581-586
[Orl93] Orlin, J.B.: A faster strongly polynomial minimum cost flow algorithm. Operations Reserach 41 (1993), 338-350
[RR98] Rao, S., Richa, A.W.: New approximation techniques for some ordering problems. Proc. Symp. on Discrete Algorithms 1998, 211-218
[SB02] Singh, D.P., Brown, S.D.: Incremental placement for layout-driven optimizations on FPGAs. Proc. ICCAD 2002

[SDJ91] Sigl, G., Doll, K., Johannes, F.M.: Analytical placement: a linear or quadratic objective function? Proc. DAC 1991, 427-432
[Sec88] Sechen, C.: VLSI Placement and Routing using Simulated Annealing. Kluwer, Boston 1988
[SK89] Suaris, P.R., Kedem, G.: A quadrisection-based combined place and route scheme for standard cells. IEEE Trans. on CAD of IC and Systems 8 (1989), 234-244
[SL87] Sechen, C., Lee, K.W.: An improved simulated annealing algorithm for row-based placement. Proc. ICCAD 1987, 478-481
[SSV85] Sechen, C., Sangiovanni-Vincentelli, A.L.: The TimberWolf placement and routing package. IEEE J. of Solid-State Circuits 20 (1985), 510-522
[TK91] Tsay, R.-S., Koehl, J.: An analytic net weighting approach for performance optimization in circuit placement. Proc. DAC 1991, 620-625
[TKH88] Tsay, R.-S., Kuh, E., Hsu, C.-P.: Proud: a sea-of-gate placement algorithm. IEEE Design and Test of Computers 1988, 44-56
[Tut63] Tutte, W.T.: How to draw a graph. Proc. of the London Math. Soc. 13 (1963), 743-767

[Vem98] Vempala, S.: Random projection: a new approach to VLSI layout. Proc. Symp. on Foundations of Computer Science 1998, 389-395

[Vyg96] Vygen, J.: Plazierung im VLSI-Design und ein zweidimensionales Zerlegungsproblem. PhD thesis, University of Bonn, 1996

[Vyg97] Vygen, J.: Algorithms for large-scale flat placement. Proc. DAC 1997, 746-751

[Vyg98] Vygen, J.: Algorithms for detailed placement of standard cells. Proc. DATE 1998, 321-324

[Vyg99] Vygen, J.: On dual minimum cost flow algorithms. Report No. 99889, Research Institute for Discrete Mathematics, University of Bonn 2002. To appear in Math. Meth. of Oper. Res. Extended Abstract in: Proc. Symp. On the Theory of Computing 2000, 117-125

[Vyg00] Vygen, J.: Geometric Quadrisection in Linear Time, with Application to VLSI placement. Manuscript, to appear in Discrete Optimization

[Vyg01] Vygen, J.: Theory of VLSI Layout. Habilitation thesis, University of Bonn 2001

[Vyg02] Vygen, J.: New theoretical results on quadratic placement. Manuscript, to appear in Integration – a VLSI Journal

[WLL88] Wong, D.F., Leong, H.W., Liu, C.L.: Simulated Annealing for VLSI Design. Kluwer, Boston 1988

[WWZ00] Warme, D.M., Winter, P., Zachariasen, M.: Exact algorithms for plane Steiner tree problems: a computational study. In: Advances in Steiner trees (D.Z. Du, J.M. Smith, eds.), Kluwer, Boston 2000, pp. 81-116

[WYS00] Wang, M., Yang, X., Sarrafzadeh, M.: Dragon2000: fast standard-cell placement for large circuits. Proc. ICCAD 2000, 260-263

[YLS92] Youssef, H., Lin, R.-B., Shragowitz, E.: Bounds on net delays for VLSI circuits. IEEE Trans. On Circuits and Systems II 39 (1992), 815-824

[YM01] Yildiz, M.C., Madden, P.H.: Global objectives for standard cell placement. Proc. Great Lakes Symp. On VLSI 2001