UCLA COMPUTER SCIENCE DEPARTMENT

Large Scale Circuit Placement: Gap and Progress

Tony Chan[2], Jason Cong[1], *Joe Shinnerl[1]*, Kenton Sze[2], Min Xie[1]

[1]VLSI CAD Lab

[1]Computer Science Department [2]Mathematics Department

University of California, Los Angeles

http://cadlab.cs.ucla.edu/~cong

cong@cs.ucla.edu

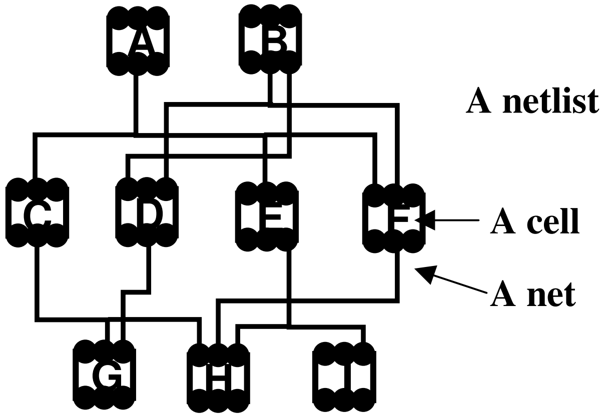
Outline

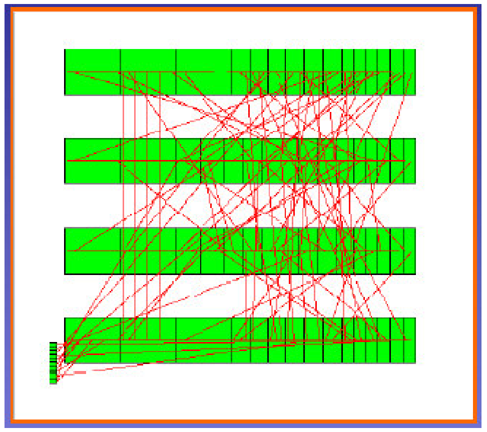
- Introduction
  - Problem Description
  - Popular Methods
- Gap Analysis of Existing Placement Algorithms
  - PEKO Benchmark Construction
  - Experiment Results
- UCLA mPL5
  - Multiscale Optimization Framework
  - Generic Force-Directed Formulation
  - Multiscale Nonlinear-Programming Solution

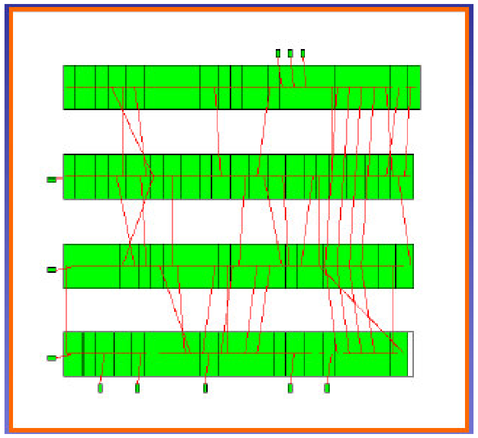
Outline

- Introduction
  - Problem Description
  - Popular Methods
- Gap Analysis of Existing Placement Algorithms
- UCLA mPL5

Circuit Placement Problem Statement




Bad placement


Good placement

- Given
  - A set of cells (modules) of fixed dimensions and the interconnections between them – a netlist
- Find
  - The position of each cell, such that
    - no overlap (and enough routing space)
    - minimize total length of all interconnections
    - minimize routing congestion, delay, …

Popular Placement Methods

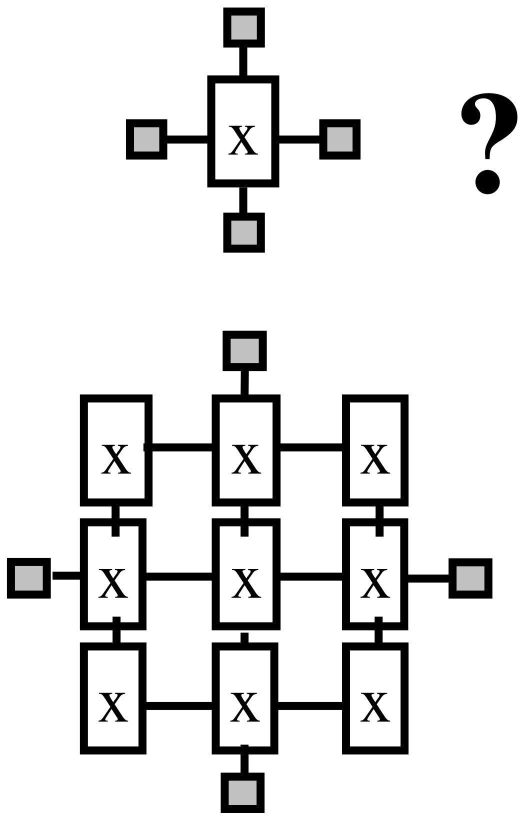
- Iterative improvement (Timberwolf, iTools)
  - Repeatedly rearrange small subsets of modules
  - E.g. Simulated annealing
- Min-cut based placement (Capo, Feng-Shui)
  - Recursively bi-partition modules in a way that minimize connections between partition blocks
- Quadratic placement with recursive legalization (Gordian, BonnPlace, FastPlace, Kraftwerk, …)
  - Initial solution by unconstrained quadratic wirelength minimization
  - Gradually spread cells out to remove overlap
- Multiscale (Ultra-fast VPR, mPL, Dragon, …)

Outline

- Introduction
- Gap Analysis of Existing Placement Algorithms
  - PEKO Benchmark Construction
  - Experiment Results
- Highlights from UCLA mPL5

Optimality and Scalability Study--- Related Work

- Quantified Suboptimality of VLSI Layout Heuristics [L. Hagen et al, 1995]

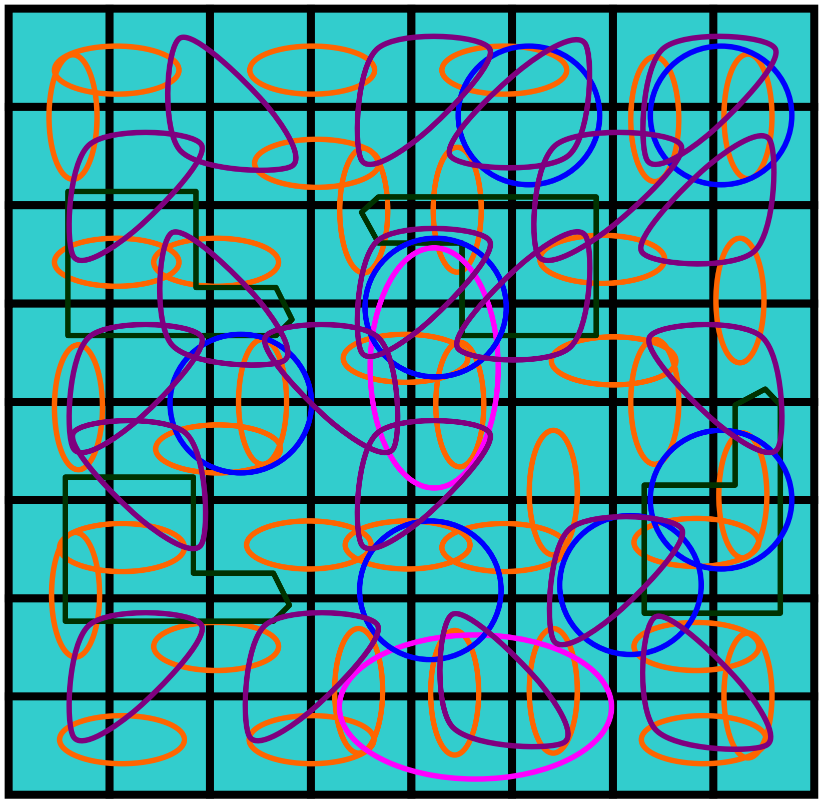


- Construct scaled instance with known upperbound
- Over 10% area suboptimality in TimberWolf
- Notable wirelength suboptimality in GORDIAN-L
- But test cases are small, the largest netlist is less than 40K

Construction of Placement Examples with Known Optimal Wirelength (PEKO Examples)

- **Idea: construct synthetic benchmarks matching netlist characteristics of industrial benchmarks**
- **Input**
  - Desired number of placeable modules *t*
  - *Net Distribution Vector* (NDV) $D = (d_2, d_3, \ldots d_p)$, d_k is the # of *k*-pin nets in the circuit,
  - *t and D* are extracted from a real circuit
- **Output**
  - Cell library *L*
  - Netlist *N* with known optimal wirelength
- **Constraint**
  - *N* has *D* as its NDV

Placement Examples with Known Optimal Wirelength [Chang et al, 2003]



- All the modules are of equal size, and there is no space between rows and adjacent modules
- For 2-pin nets , connect any two adjacent modules
- For each *n*-pin net , connect the *n* modules in a rectangular region close to a square, i.e., the length of each side is close to sqrt(*n*)
- The wirelength is of each *n*-pin net is given by $\lceil \sqrt{n} \rceil + \lceil n / \lceil \sqrt{n} \rceil \rceil - 2$
- Net degree distributions extracted from real industrial benchmarks

PEKO Characteristics

PEKO Suite1 (12.5k – 210k)

ckt	#cell	#net	#row	Optimal WL
Peko01	12506	13865	113	8.14E+05
Peko02	19342	19325	140	1.26E+06
Peko03	22853	27118	152	1.50E+06
Peko04	27220	31683	166	1.75E+06
Peko05	28146	27777	169	1.91E+06
Peko06	32332	34660	181	2.06E+06
Peko07	45639	47830	215	2.88E+06
Peko08	51023	50227	227	3.14E+06
Peko09	53110	60617	231	3.64E+06
Peko10	68685	74452	263	4.73E+06
Peko11	70152	81048	266	4.71E+06
Peko12	70439	76603	266	5.00E+06
Peko13	83709	99176	290	5.87E+06
Peko14	147088	152255	385	9.01E+06
Peko15	161187	186225	402	1.15E+07
Peko16	182980	189544	429	1.25E+07
Peko17	184752	188838	431	1.34E+07
Peko18	210341	201648	460	1.32E+07

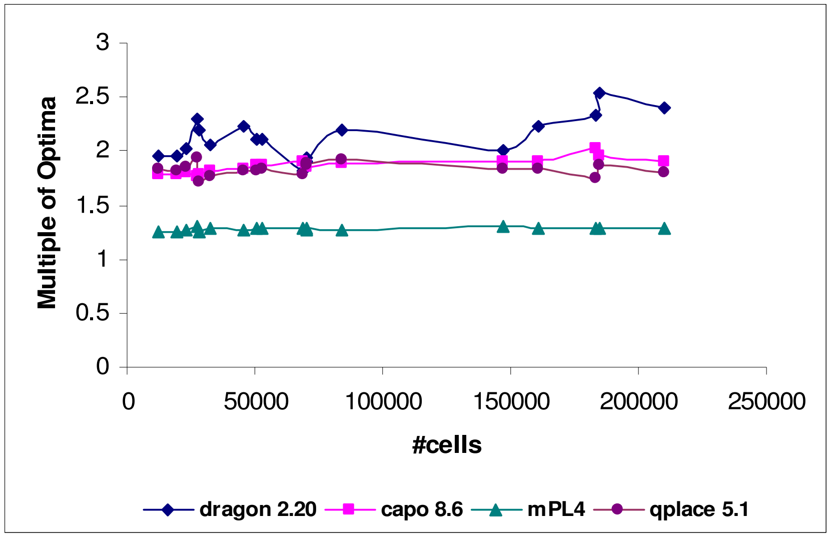
PEKO Suite2 (125k – 2.1M)

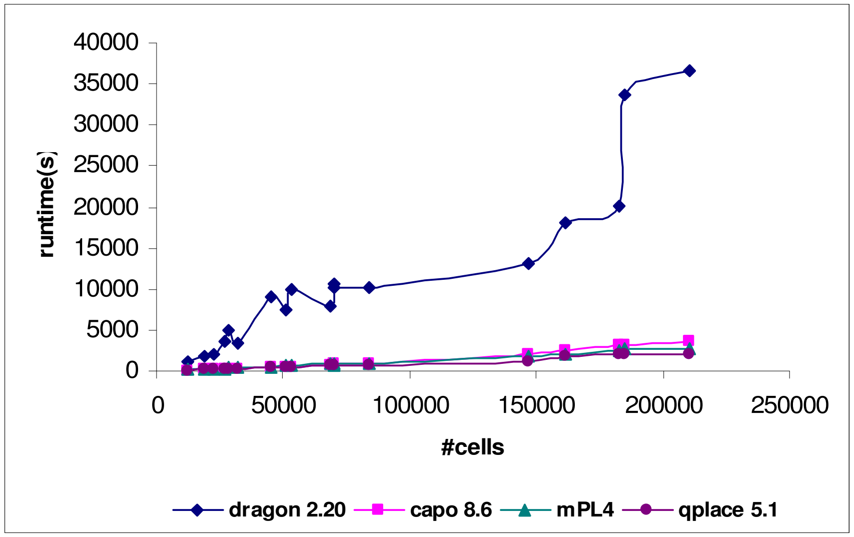
ckt	#cell	#net	#row	Optimal WL
Peko01x10	125060	138650	335	8.14E+06
Peko02x10	193420	193250	441	1.26E+07
Peko03x10	228530	271180	479	1.50E+07
Peko04x10	272200	316830	523	1.75E+07
Peko05x10	281460	277770	532	1.91E+07
Peko06x10	323320	346600	570	2.06E+07
Peko07x10	456390	478300	677	2.88E+07
Peko08x10	510230	502270	715	3.14E+07
Peko09x10	531100	606170	730	3.64E+07
Peko10x10	686850	744520	830	4.73E+07
Peko11x10	701520	810480	839	4.71E+07
Peko12x10	704390	766030	840	5.00E+07
Peko13x10	837090	991760	916	5.87E+07
Peko14x10	1470880	1522550	1214	9.01E+07
Peko15x10	1611870	1862250	1271	1.15E+08
Peko16x10	1829800	1895440	1354	1.25E+08
Peko17x10	1847520	1888380	1360	1.34E+08
Peko18x10	2103410	2016480	1451	1.32E+08

Studied Four State-of-the-Art Placers

- Capo [A. Caldwell et al, 2000]
  - Based on multilevel partitioner
  - Aims to enhance the routability
- Dragon [M. Wang et al, 2000]
  - Uses hMetis for initial partition
  - SA with bin-based swapping
- mPL [T. Chan et al, 2000]
  - Multilevel placer using NLP on the coarsest level
  - Goto based relaxation
- QPlace [Cadence Inc.]
  - Leading edge industrial placer
  - Component of Silicon Ensemble

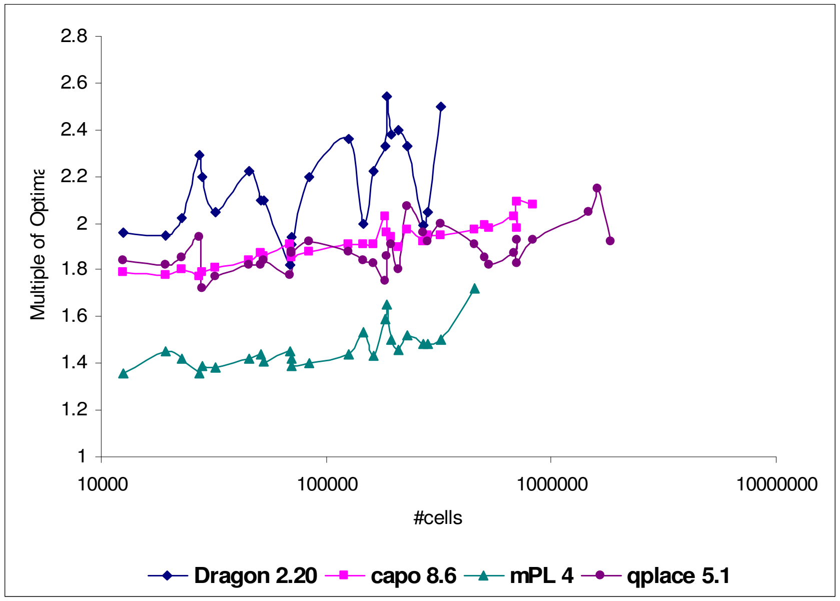
Experiment Results on PEKO, July 2004

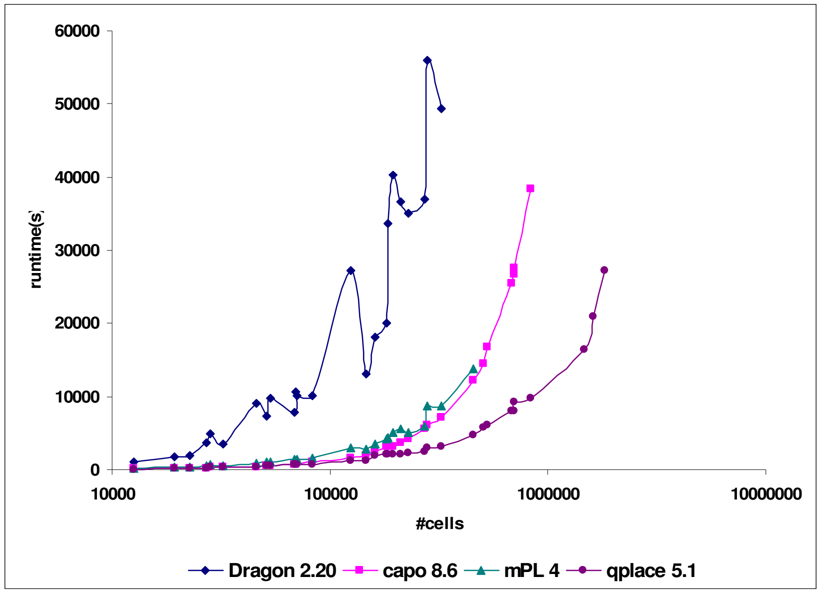




- Existing algorithms are 30-153% away from the optimal on PEKO
  - There is significant room for improvement in placement algorithms!
- ROI can be huge – 30% wirelength reduction is equivalent to
  - Move from aluminum to copper, or
  - One process generation shrink

Experiment with State-of-the-Art Placers Using PEKO Suite1 & Suite2 (July 2004)





- Capo, QPlace and mPL scales well in runtime
- Average solution quality of each tool shows deterioration by an additional 4% to 25% when the problem size increases by a factor of 10
- QoR of the existing placement algorithms can be 40% - 160% away from the optimal for large designs

Limitations of the PEKO Examples

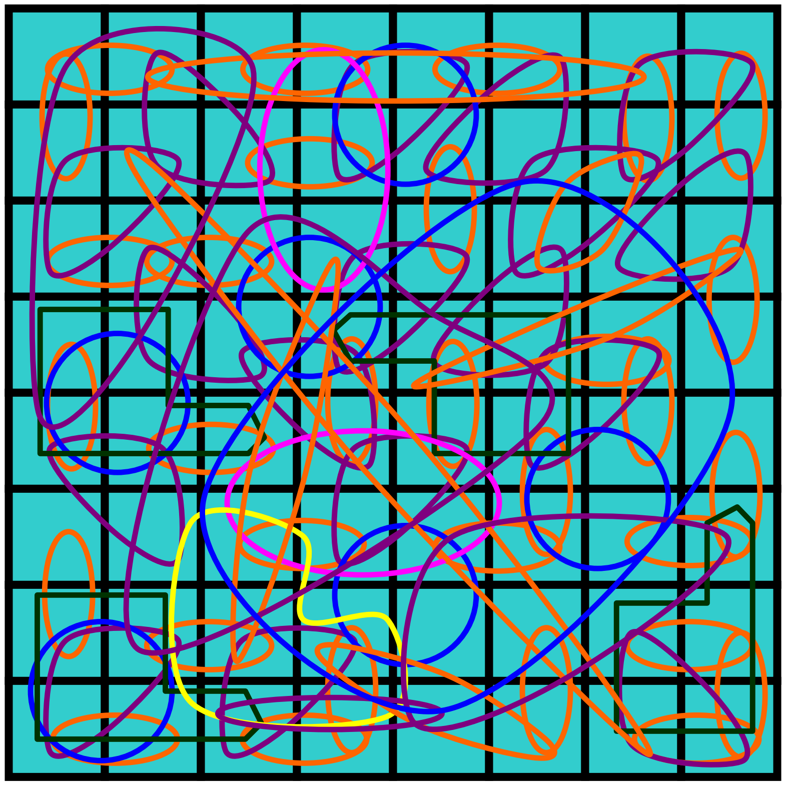
- Optimal solution includes local nets only
  - Unlikely for real designs
- Measure wirelength only
  - Timing and routability are important objectives for placement algorithms as well

Impact of Global Connections in Real Examples

circuit	height	width	WL of longest net	WL contribution of longest 10%
ibm01	8158	4530	7148	51%
ibm02	8158	6430	14224	46%
ibm03	8158	6740	10624	58%
ibm04	8158	9140	15171	53%
ibm05	8158	11055	19064	47%
ibm06	8158	8715	13966	61%
ibm07	8158	14605	14051	51%
ibm08	8158	15895	16142	60%
ibm09	8158	16395	13780	55%
ibm10	8158	27890	30755	53%
ibm11	16350	10925	19234	59%
ibm12	16350	15545	26748	52%
ibm13	16350	12230	19539	59%
ibm14	16350	25475	26370	61%
ibm15	16350	23785	27284	63%
ibm16	16350	34015	42860	59%
ibm17	16283	38895	45686	56%
ibm18	16350	37065	52846	64%

- Produced by Dragon on ISPD98
- The wirelength contribution from global connections can be significant!
- Need to consider the impact of global connections

Placement Examples with Known Upperbounds (PEKU)



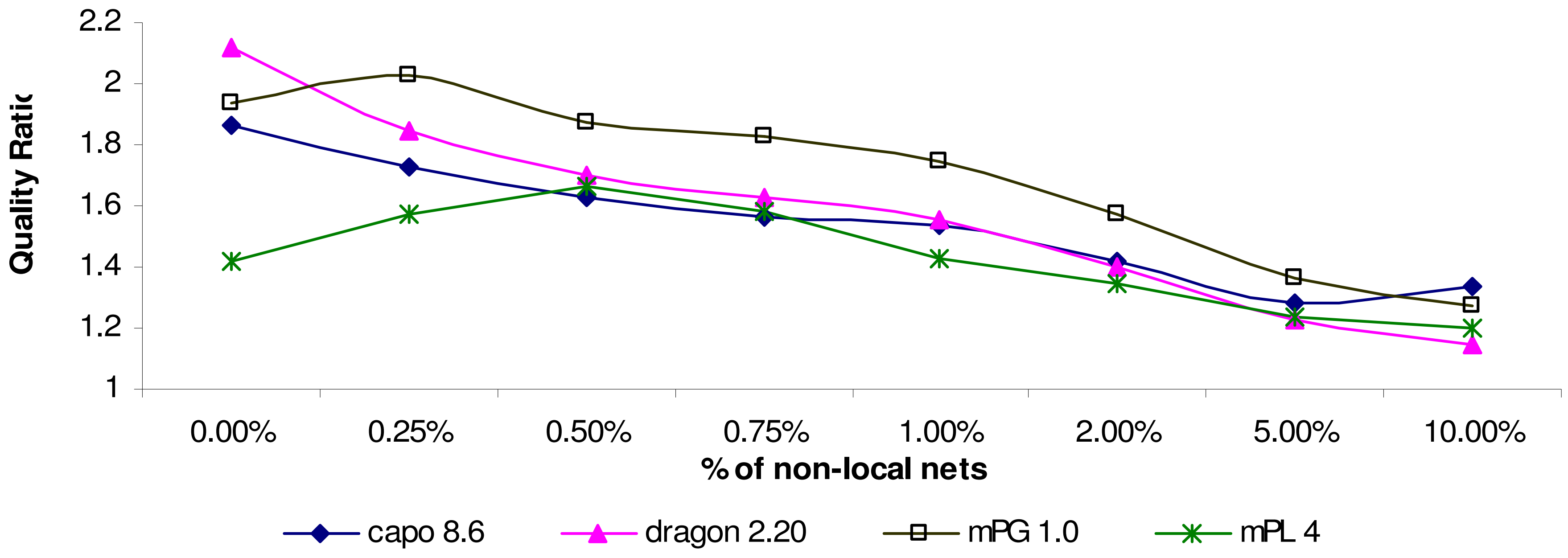
- Generate nets with optimal wirelength as in Peko
- Add random connections with emulate global nets

PEKU Suite

% non-local nets	circuit	#cell	#net	#row	Row utilization	LB	UB
0	Peku01	12506	14111	113	85%	8.14E+05	8.14E+05
	Peku05	28146	28446	169	85%	1.91E+06	1.91E+06
	Peku10	68685	75196	263	85%	4.73E+06	4.73E+06
	Peku15	161187	186608	402	85%	1.15E+07	1.15E+07
	Peku18	210341	201920	460	85%	1.32E+07	1.32E+07
0.25%	Peku01	12506	14111	113	85%	8.14E+05	9.23E+05
	Peku05	28146	28446	169	85%	1.91E+06	2.24E+06
	Peku10	68685	75196	263	85%	4.73E+06	6.17E+06
	Peku15	161187	186608	402	85%	1.15E+07	1.71E+07
	Peku18	210341	201920	460	85%	1.32E+07	2.01E+07
0.50%	Peku01	12506	14111	113	85%	8.14E+05	1.02E+06
	Peku05	28146	28446	169	85%	1.91E+06	2.63E+06
	Peku10	68685	75196	263	85%	4.73E+06	7.52E+06
	Peku15	161187	186608	402	85%	1.15E+07	2.30E+07
	Peku18	210341	201920	460	85%	1.32E+07	2.75E+07
Up to 10%	…						

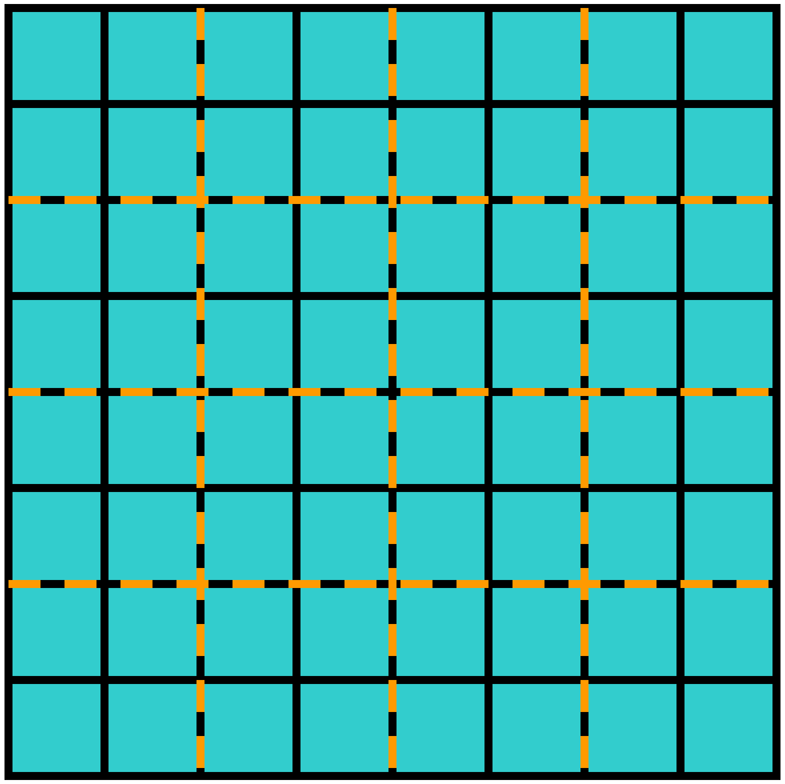
URL: http://cadlab.cs.ucla.edu/~pubbench/peku.htm

Experiment Results on PEKU, July 2004



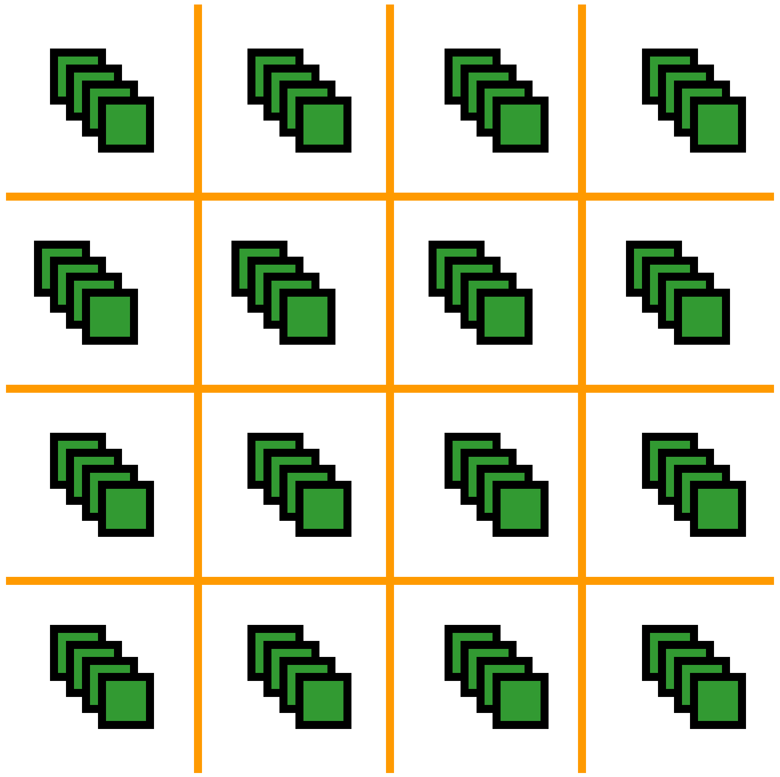
- Absolute value of the QRs may not be meaningful, but it helps to identify the technique that works best under each scenario
- No existing placer can consistently produce the best quality

PEKO-DP Detailed Placement Example Construction



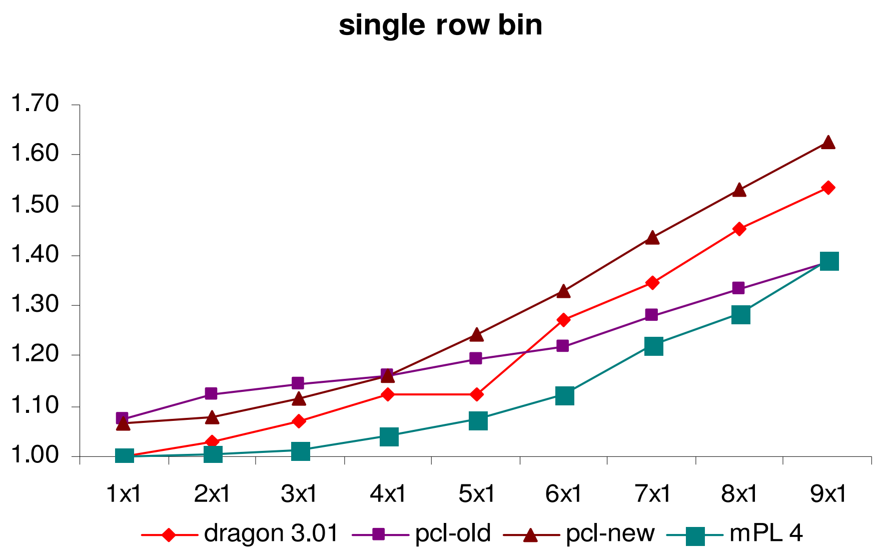
- Start from existing Peko examples [Chang et al, ASPDAC 03]
- Define a bin grid of user-specified size

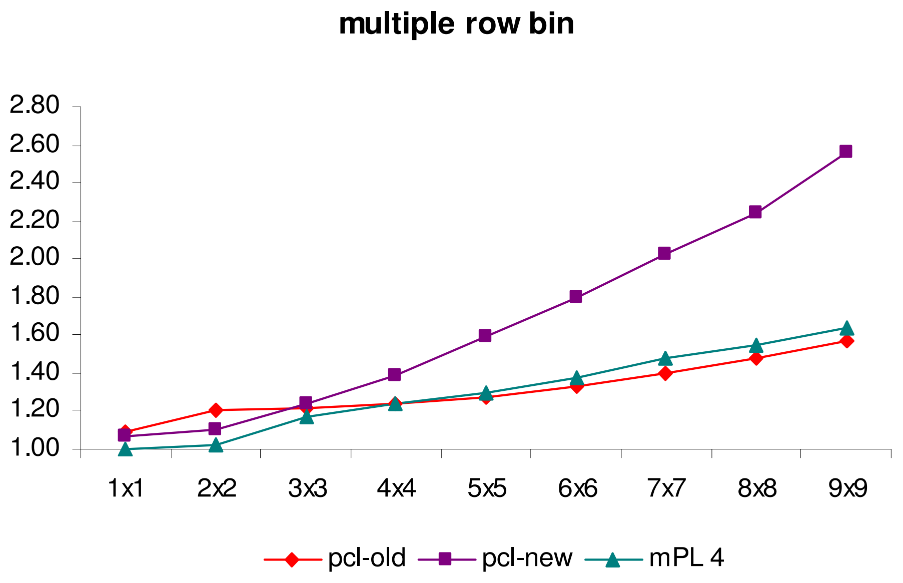
PEKO-DP Detailed Placement Example Construction



- Start from existing Peko examples [Chang et al, ASPDAC 03]
- Define a bin grid a user-specified size
- Snap cells to bin centers

Experiment Results on PEKO-DP, July 2004

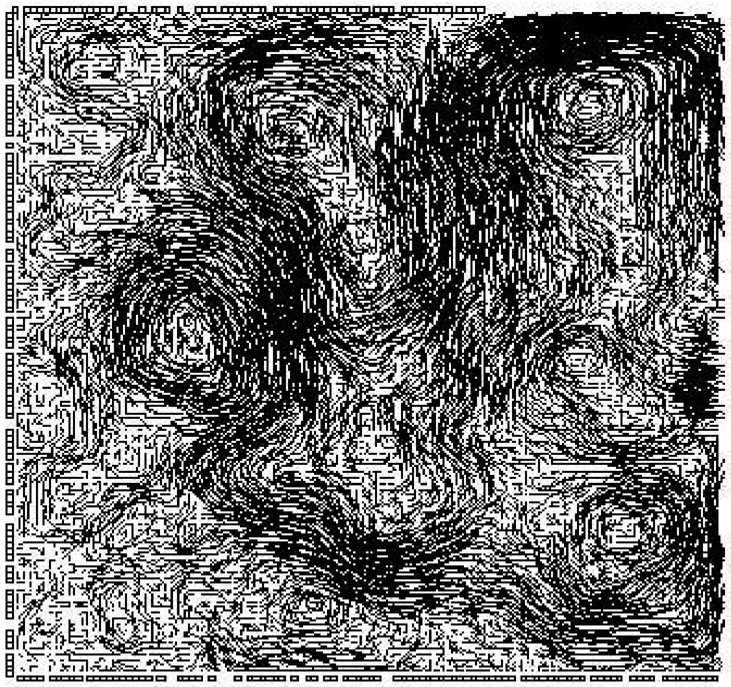




- Penalizing displacement from the global placement can consistently produce solutions close to the optimal given reasonably small bins
- QoR still degrades with the increase of bin size

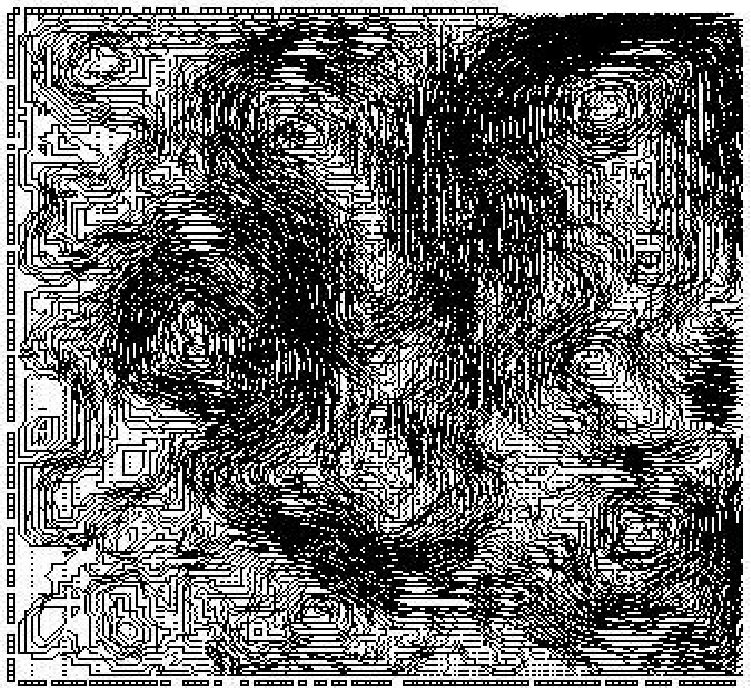
Displacement maps for mPL4 sol'n on PEKO

```
m PL HPWL =    961095, Opt HPWL =    828160.
# of steps from optimal location:
0 <= steps <= 2 : 6543 (52%), 2 < steps <= 4: 3814 (30%)
4 < steps <= 8 : 1924 (15%), 8 < steps <= 16: 225 (1%)
steps > 16 : 0(0%), Largest steps = 15
```



After Global Placement

```
m PL HPWL =    1046762, Opt HPWL =    828160.
# of steps from optimal location:
0 <= steps <= 2 : 5497 (43%), 2 < steps <= 4: 4241 (33%)
4 < steps <= 8 : 2468 (19%), 8 < steps <= 16: 300 (2%)
steps > 16 : 0(0%), Largest steps = 14
```



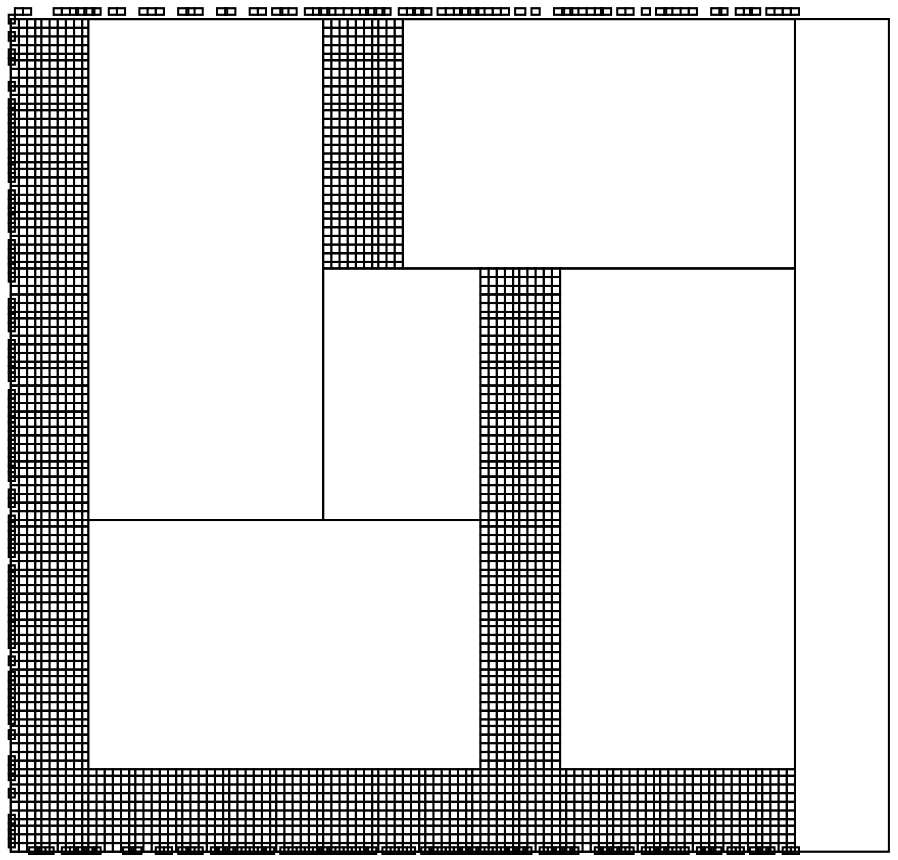
After Detailed Placement

Localized moves may not be enough to correct large errors

In Preparation: PEKO-MS (Mixed-Size PEKO)

Center-to-center HPWL = 1029536.

Pin-to-pin HPWL = 264944.



As of March 2005, the best result of mPL5 on this benchmark is still over 6X greater than optimal (in pin-to-pin half-perimeter wirelength)!

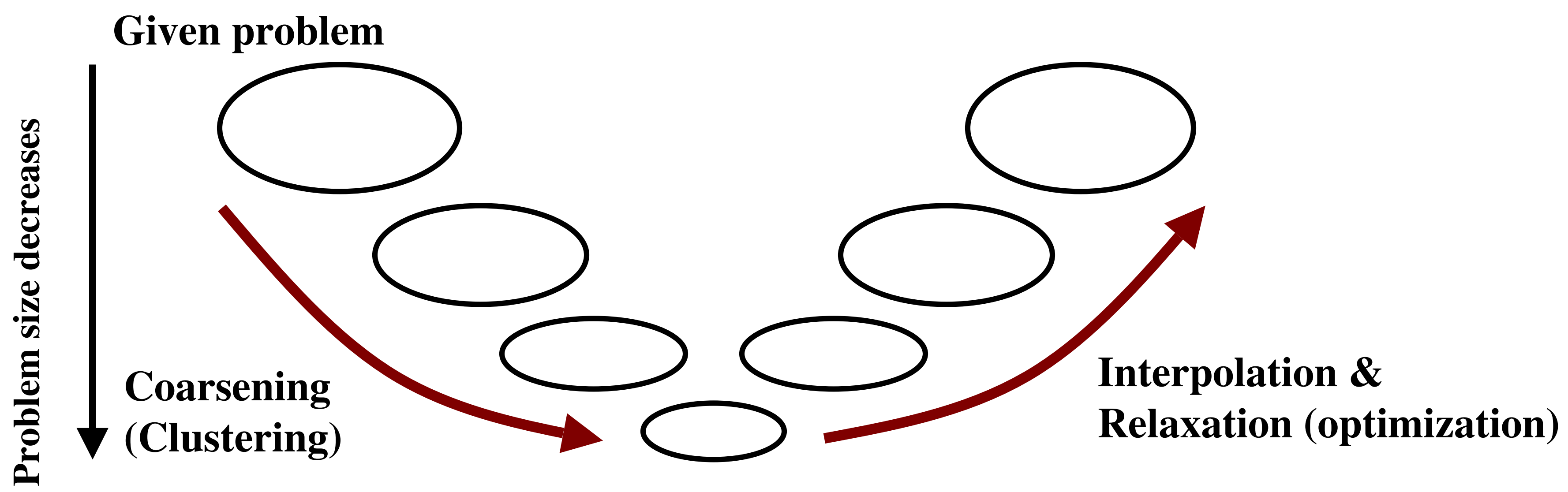
Observations from Gap Analysis

- Significant opportunity in placement
  - Existing algorithms may produce solutions far away from the optimal
  - The quality result of the same placer varies for circuits of similar size but different characteristic
  - Scalability problem in runtime and solution quality
- Significant ROI
  - Benefit equal to one to two generations of process scaling
  - But without requiring multi-billion dollar investment (we hope!)

Outline

- Introduction
- Gap Analysis of Existing Placement Algorithms
- **Highlights from UCLA mPL5**
  - Multiscale Optimization Framework
  - Generic Force-Directed Formulation
  - Multiscale Nonlinear-Programming Algorithm

Multilevel Optimization Framework



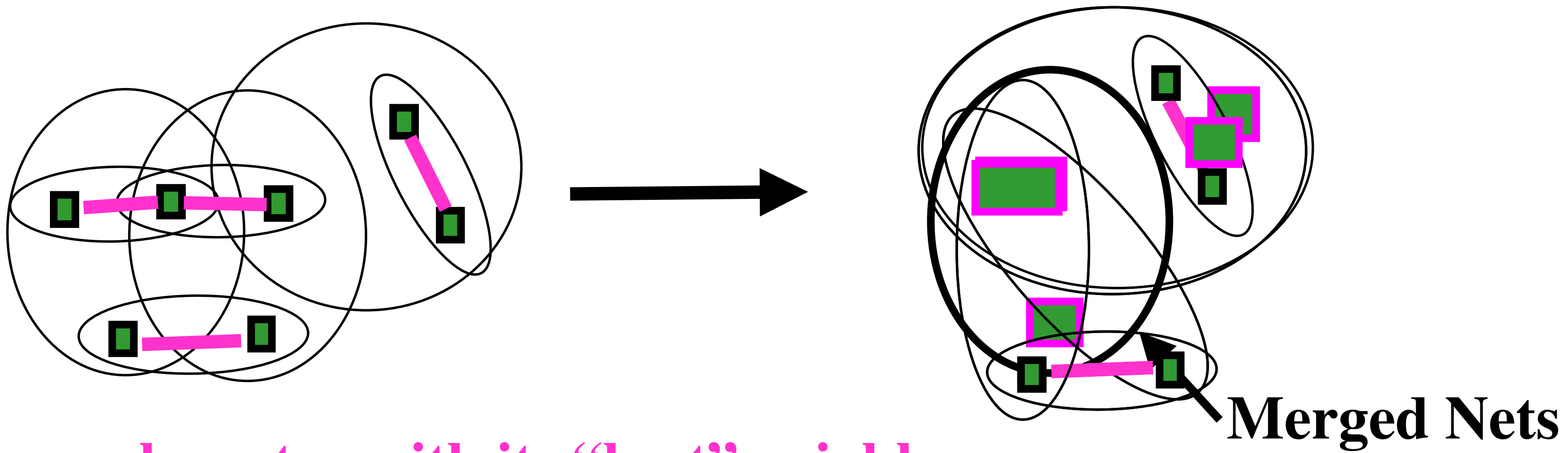
- Multilevel coarsening generates smaller problem sizes at coarser levels → faster optimization at coarser levels
- May explore different aspects of the solution space at different levels
- Gradual refinement on good solutions from coarser levels is very efficient
- Successful in many applications
  - Originally developed for PDEs
  - Recent success in VLSI CAD: partitioning, placement, routing

Multilevel Placement

- Coarsening: build a hierarchy of problem approximations by generalized clustering
- Relaxation: improve the placement at each level by iterative optimization
- Interpolation: transfer coarse-level solution to adjacent, finer level (generalized declustering)
- Multilevel Flow: multiple traversals over multiple hierarchies (V-cycle variations)

Multilevel Methods: Coarsening by Recursive Aggregation

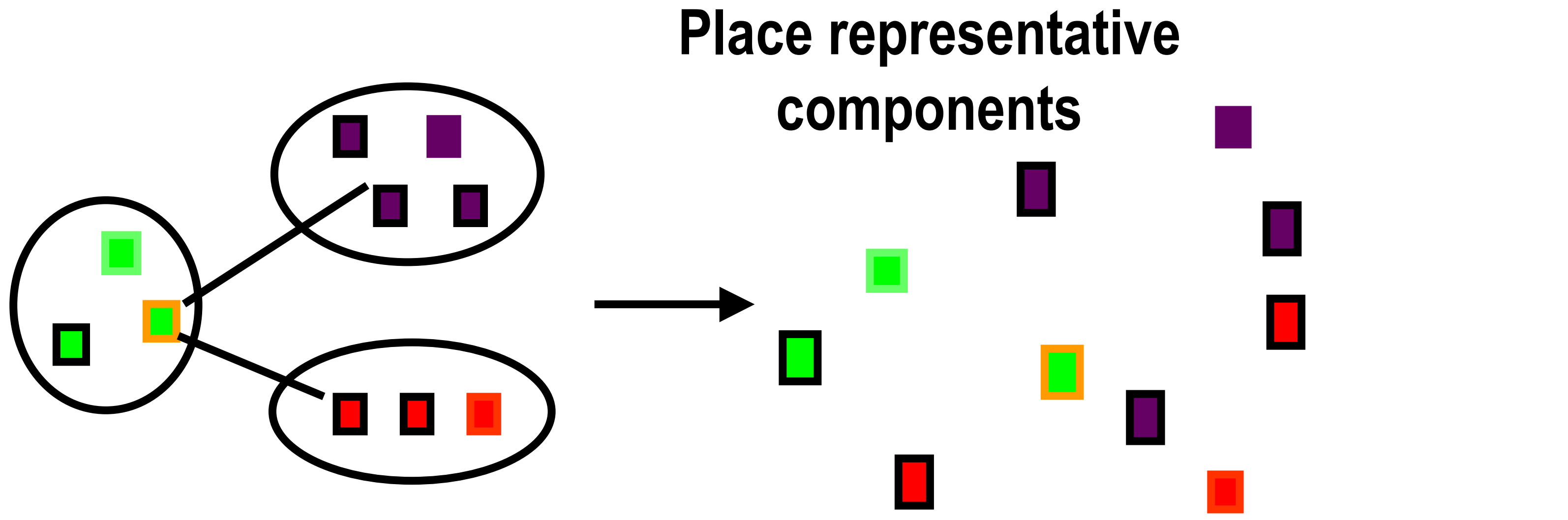
- Recursive aggregation defines the hierarchy.
- Different aggregation algorithms can be used on different levels and/or in different V-cycles.
- Example: First-Choice Clustering (hMetis [Karypis 1999]).



Merge each vertex with its "best" neighbor

Multilevel Methods: Interpolation (Generalized Declustering)

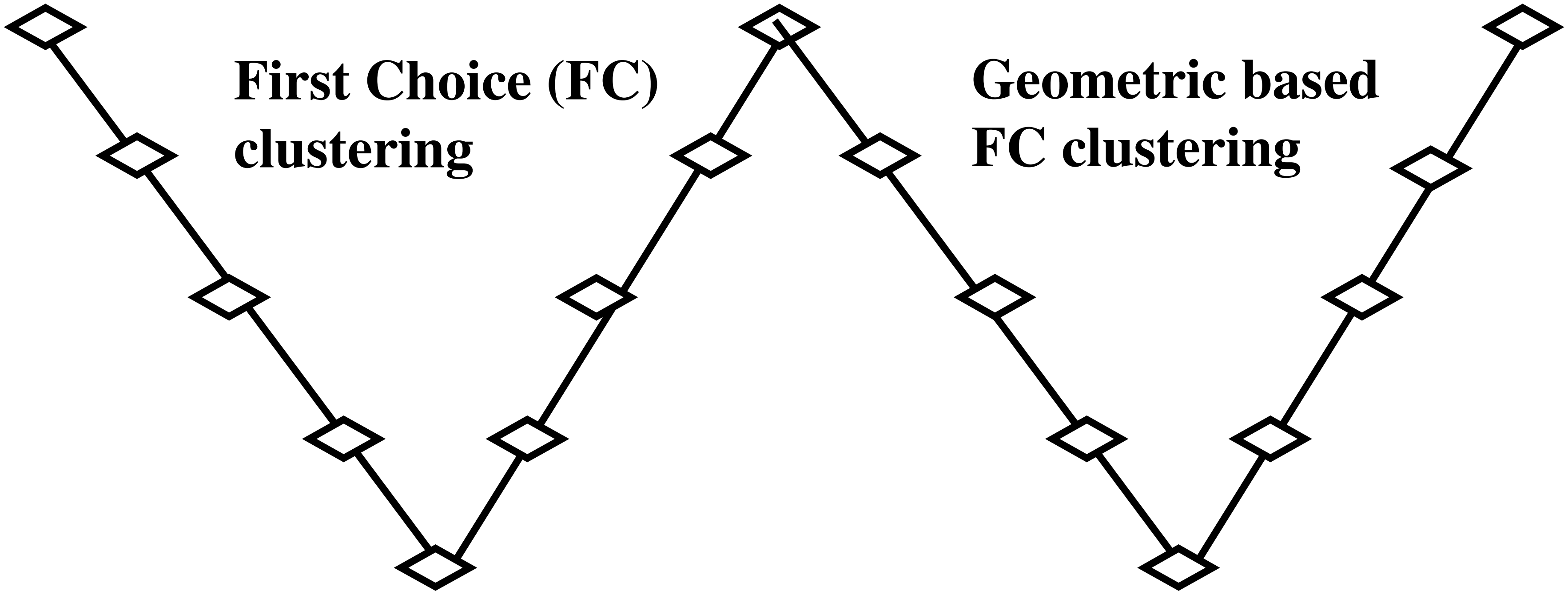
- Transfer a partial solution from a coarser level to its adjacent finer level
- Example: place a component (🟩) at the weighted average of the positions of the clusters containing its neighbors



Place others by weighted interpolation

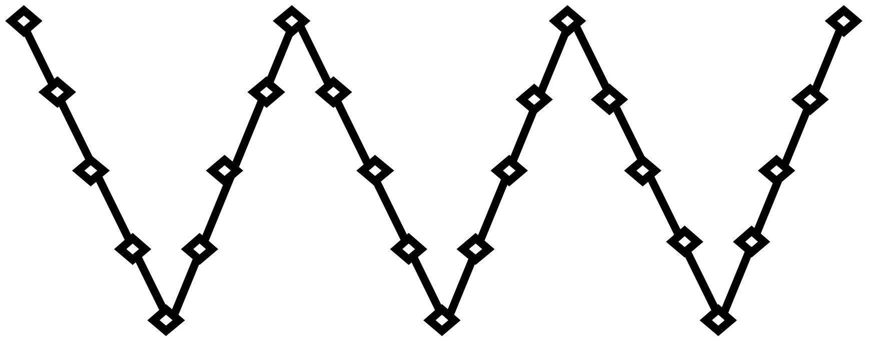
Iterated Multilevel Flow

Make use of placement solution from 1st V-cycle

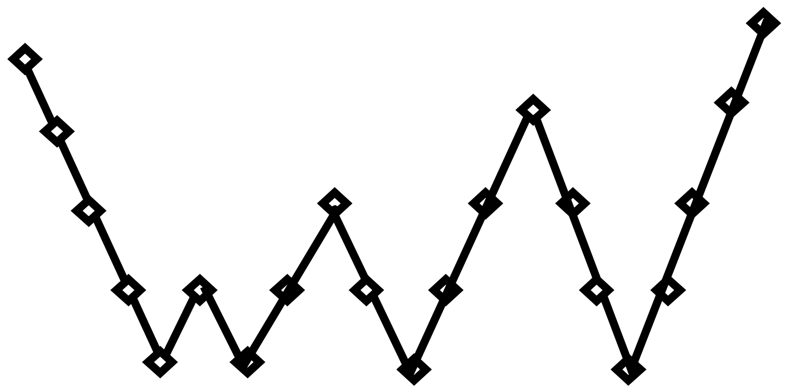


Iterated Multilevel Flow

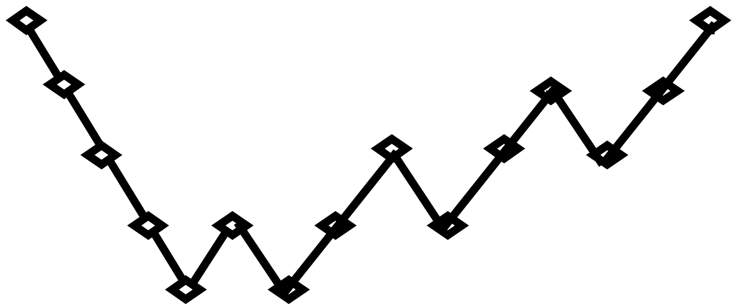
Iterated V-Cycles



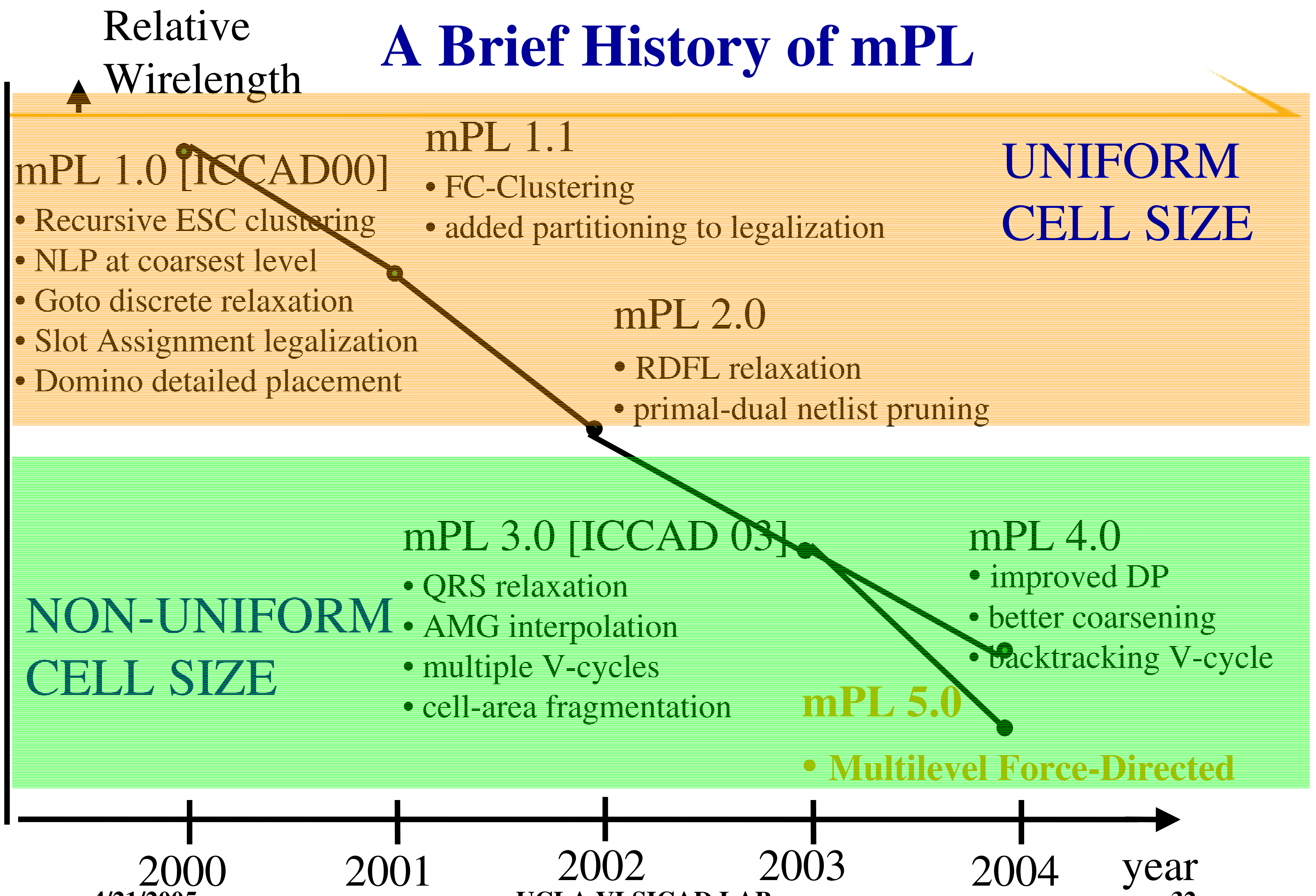
F-Cycle



Backtracking V-Cycle



A Brief History of mPL



Kraftwerk Framework for Force-Directed Placement [Eisenmann and Johannes 98]

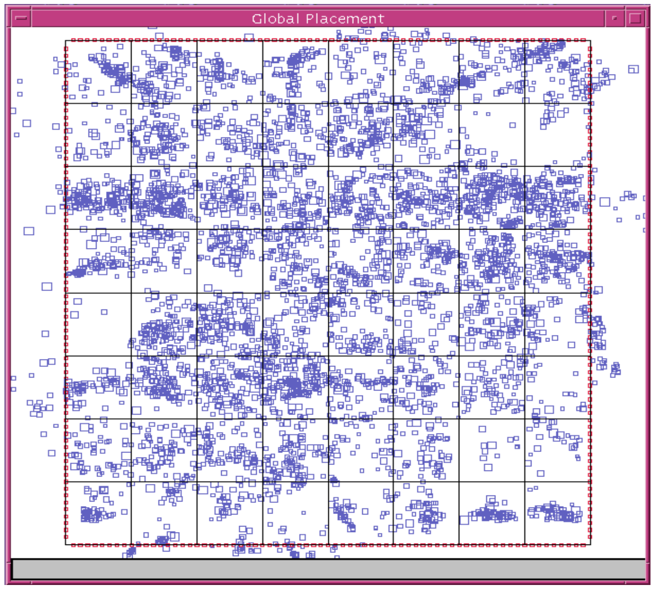
- **Minimize quadratic wirelength** $\frac{1}{2}v^T A v + r^T v, \quad v = (x,\ y).$
- **Incorporate density-gradient forces (f_k) acting on cells into the optimality condition:** $Av_{k+1} = -r + f_k$
- **Assume forces are zero at infinity.**
- **Iteratively update v_k and f_k.**

Compute f_k as $f_k = \nabla\phi$, where

$$\Delta\phi = \frac{\partial^2}{\partial x^2}\phi + \frac{\partial^2}{\partial y^2}\phi = d(v_k);$$

$d(v_k)$ is the density function for placement v_k.

- **Key limitation: extensive tuning required for proper force scaling.**



Cell density is a continuous but NON-SMOOTH function of position

mPL5 Generalized Force-Directed Placement

- **Smooth the density constraints by solving a Poisson Equation:**

$$\begin{array}{ll} \text{min} & W(v) \\ \text{s.t.} & \phi(v) = \kappa, \\ \text{where} & \phi(v) = \Delta^{-1} d(v),\ \kappa = \Delta^{-1} c. \end{array}$$

- **Assume Neumann boundary conditions: forces pointing outside the chip boundary are zero.**

- **Log-sum-exp smooth approximation to half-perimeter wirelength [Naylor 2001; Kahng and Wang 2004]:**

$$W(v) = \gamma \sum_{\text{nets } e} \Bigg(\log \sum_{\text{nodes } v_k \in e} \exp(x_k/\gamma) + \log \sum_{v_k \in e} \exp(-x_k/\gamma) + \log \sum_{v_k \in e} \exp(y_k/\gamma) + \log \sum_{v_k \in e} \exp(-y_k/\gamma) \Bigg)$$

mPL5 Nonlinear-Programing Solution

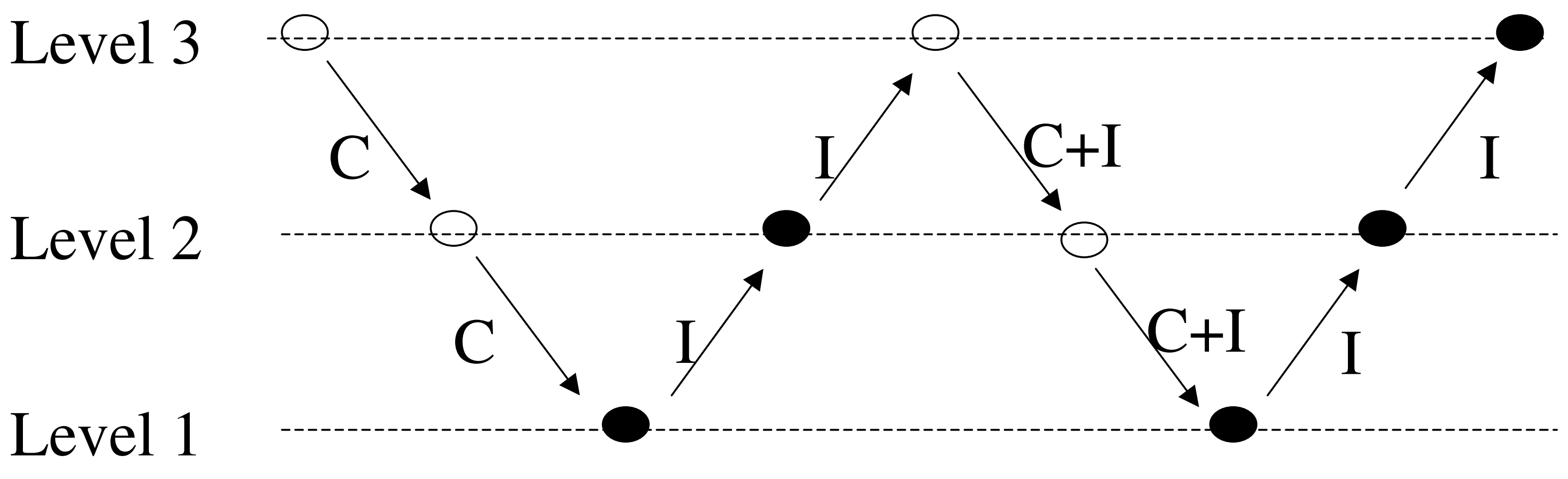
- Using the Uzawa algorithm to solve the above nonlinear constrained minimization problem, we iteratively solve

$$\nabla W(v_{k+1}) = \lambda_k \cdot \nabla \phi(v_k) \equiv \lambda_k \cdot f_k$$

$$\lambda_{k+1} = \lambda_k - \alpha(\phi(v_k) - \kappa))$$

- No matrix storage and no second derivatives are computed.
- Use multilevel approach to speed-up computation and better quality

mPL5 Framework



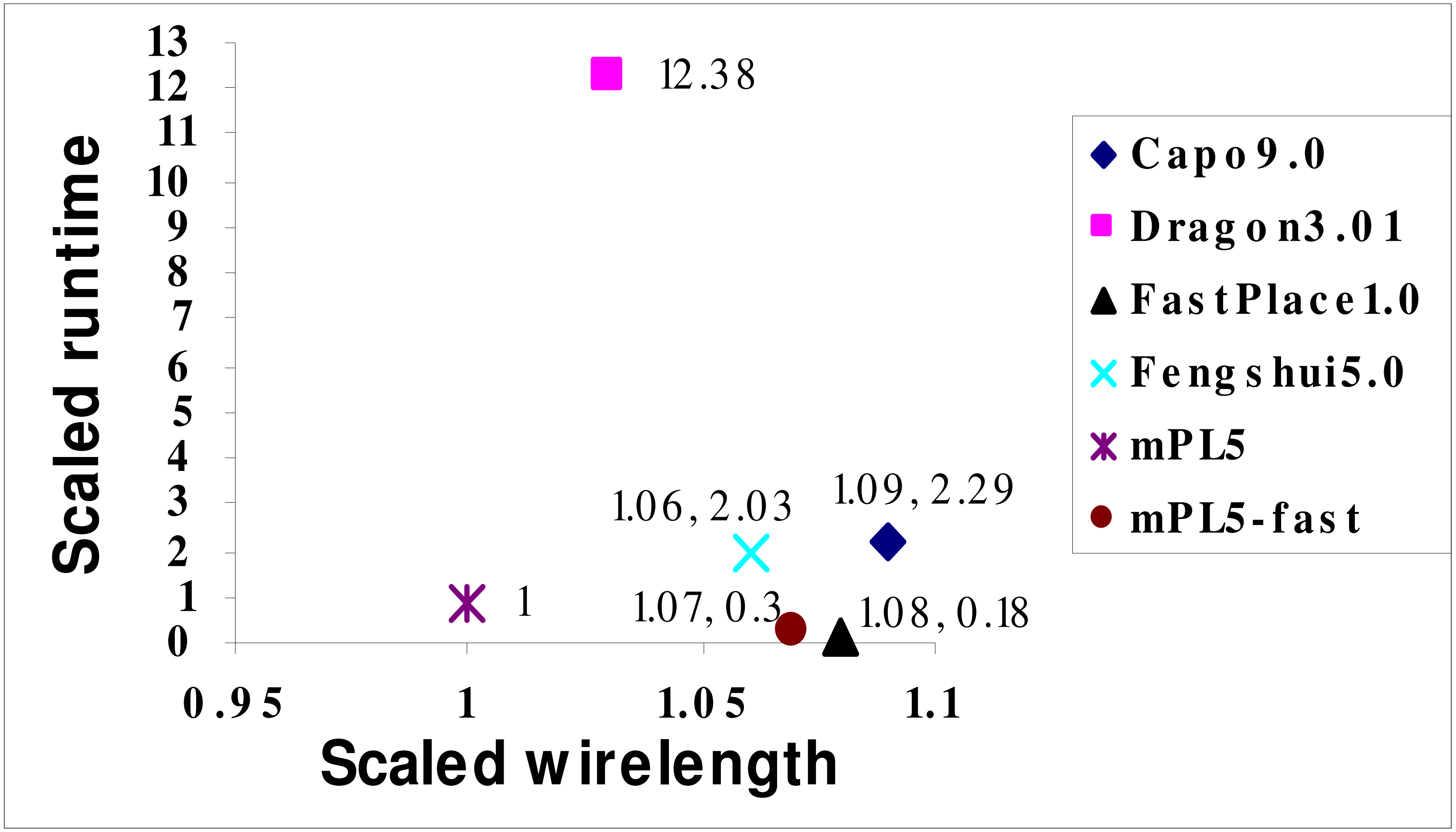
● Level at which GFD is applied

C Coasening

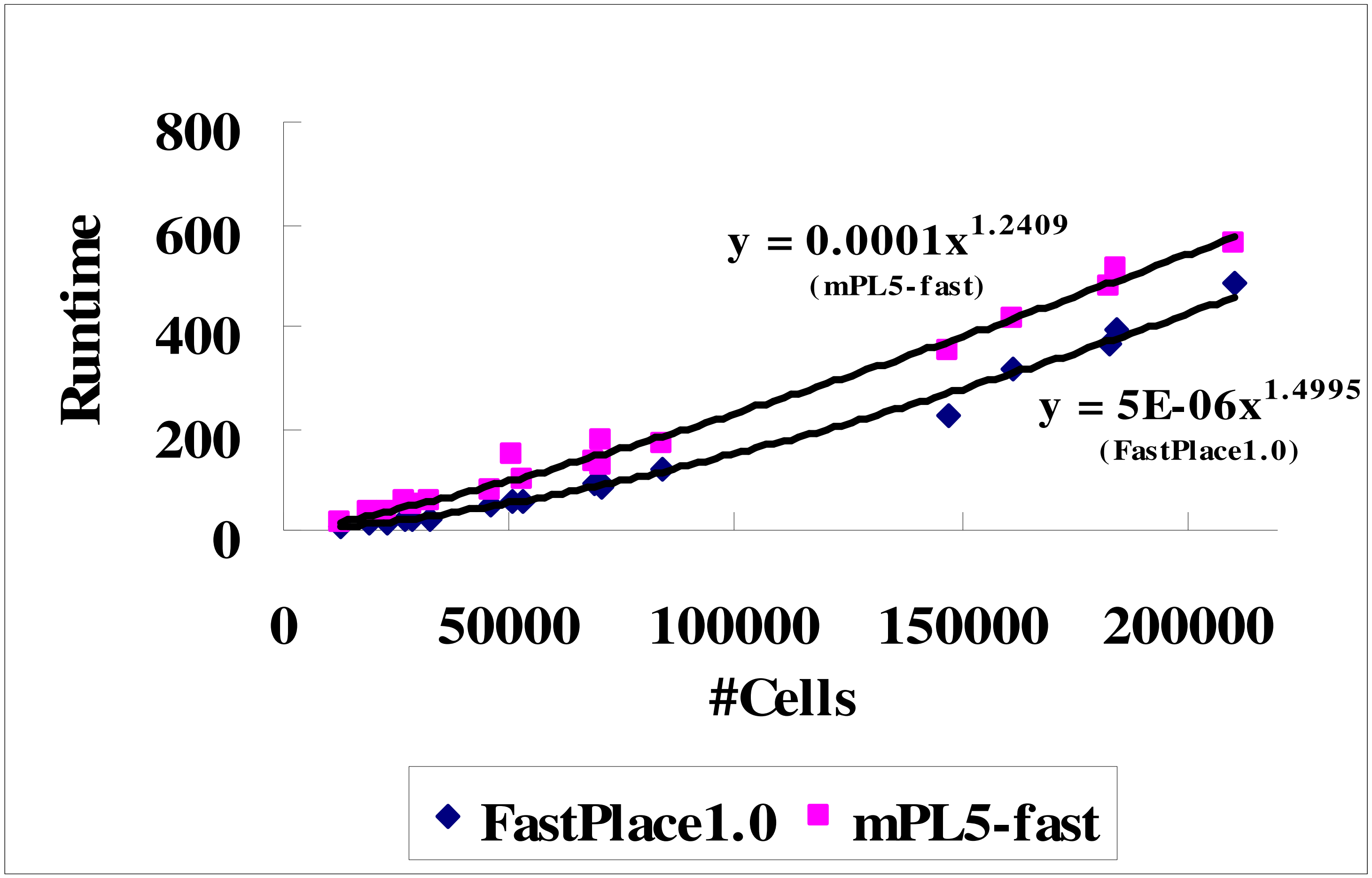
I Interpolation

Keep coarsening until # cells less than 500

mPL5 VS other state-of-the-art-placers on FastPlace IBM Standard Cell Placement Benchmarks (March 2005)

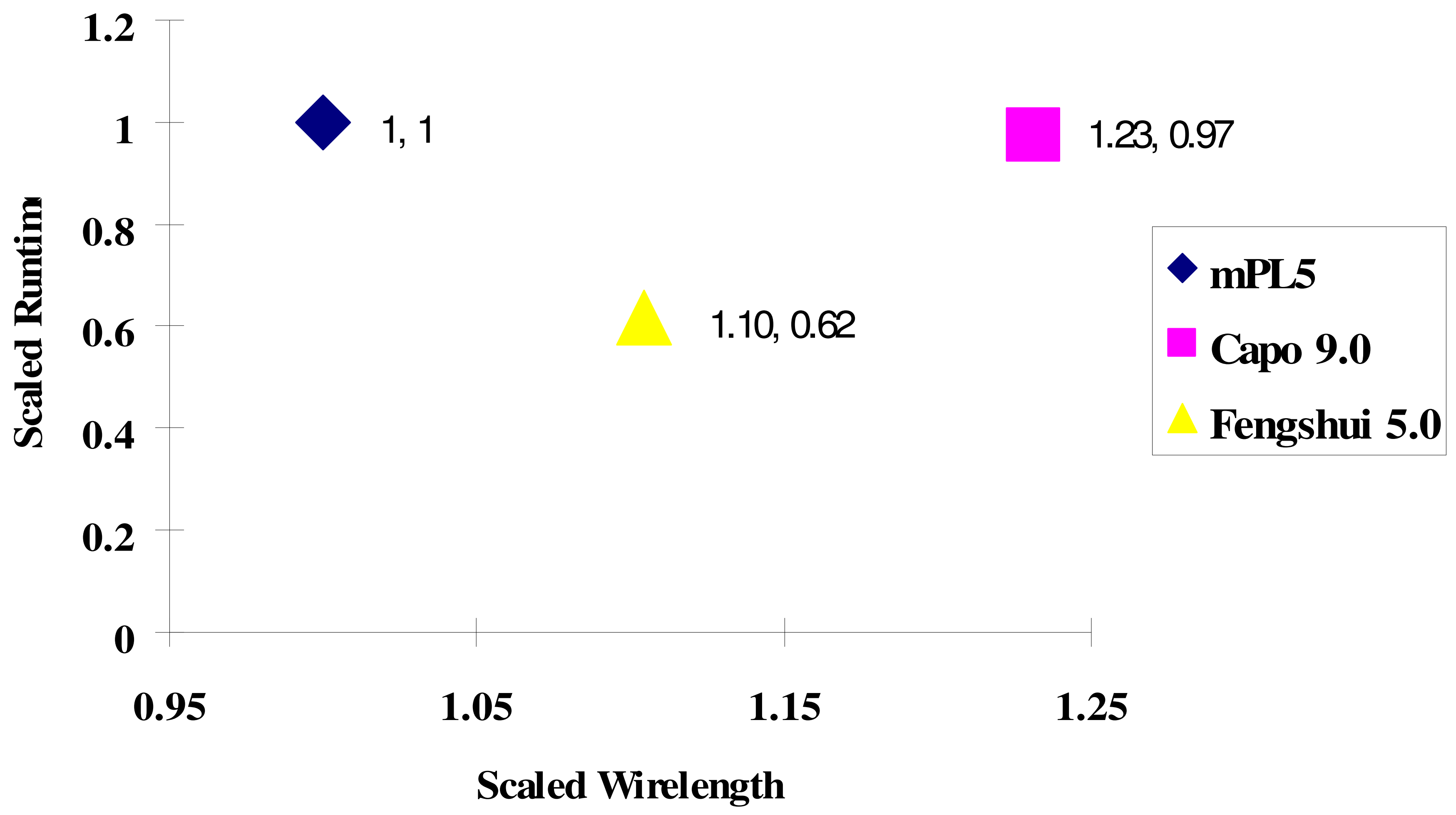


Scalability plot of mPL5-fast VS FastPlace1.0 on FastPlace IBM Benchmarks



mPL5-fast is slightly more scalable than FastPlace1.0

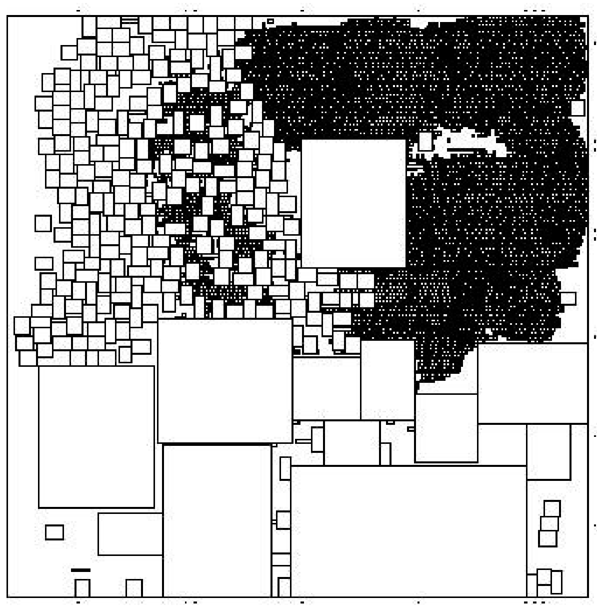
mPL5 VS Capo 9.0 and Fengshui 5.0 on ICCAD 2004 IBM Mixed-Size Placement Benchmarks



Placement Plot of Placers on IBM02

Center-to-center HPWL = 4706363.

Pin-to-pin HPWL = 4625567.

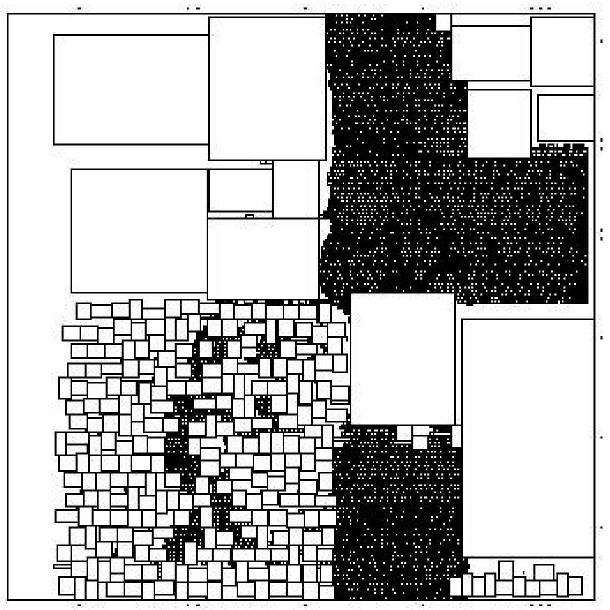


mPL5

Rel. WL = 1.00

Center-to-center HPWL = 5110913.

Pin-to-pin HPWL = 5114879.

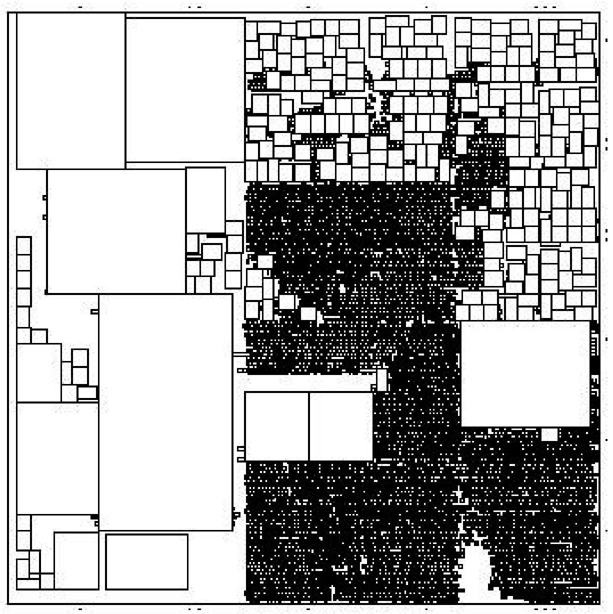


Fengshui 5.0

Rel. WL = 1.11

Center-to-center HPWL = 5489534.

Pin-to-pin HPWL = 5408726.



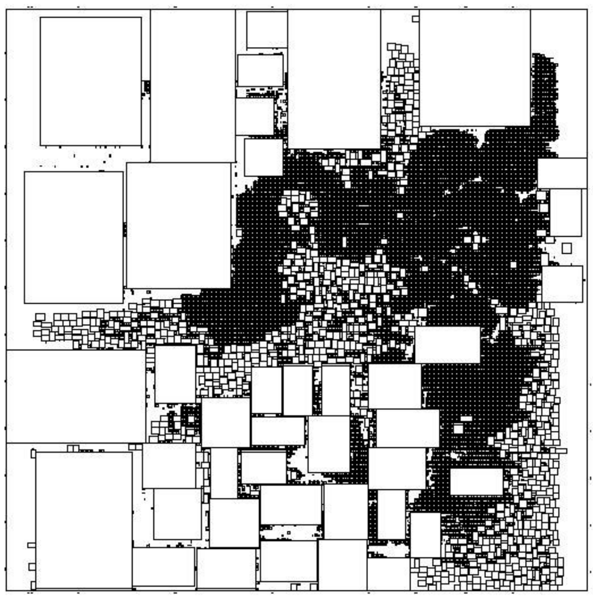
Capo 9.0

Rel. WL = 1.17

Placement Plot of Placers on IBM10

Center-to-center HPWL = 28961908.

Pin-to-pin HPWL = 28721684.

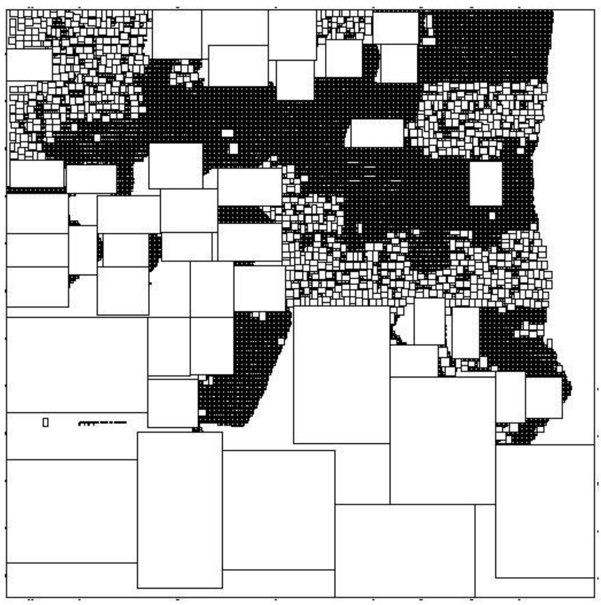


mPL5

Rel. WL = 1.00

Center-to-center HPWL = 32680859.

Pin-to-pin HPWL = 33044276.

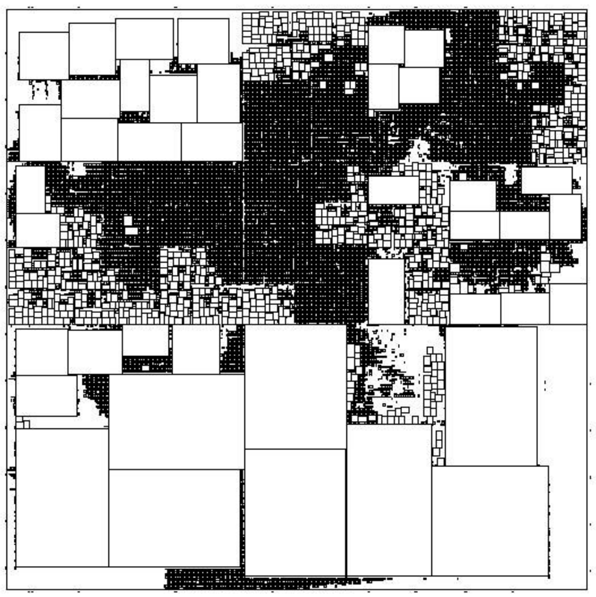


Fengshui 5.0

Rel. WL = 1.15

Center-to-center HPWL = 36997623.

Pin-to-pin HPWL = 36705667.



Capo 9.0

Rel. WL = 1.28

Concluding Remarks

- There is still significant opportunity to improve placement technologies.
- mPL5 achieves improvement by incorporating PDE-constrained nonlinear programming into a multilevel framework.

  •Multiscale Optimization Framework

  •Generic Force-Directed Formulation

  •Multiscale Nonlinear-Programming Algorithm