UCLA COMPUTER SCIENCE DEPARTMENT

Highly Efficient Gradient Computation for Density-Constrained Analytical Placement Methods

Jason Cong and Guojie Luo

UCLA Computer Science Department

{ cong, gluo } @ cs.ucla.edu

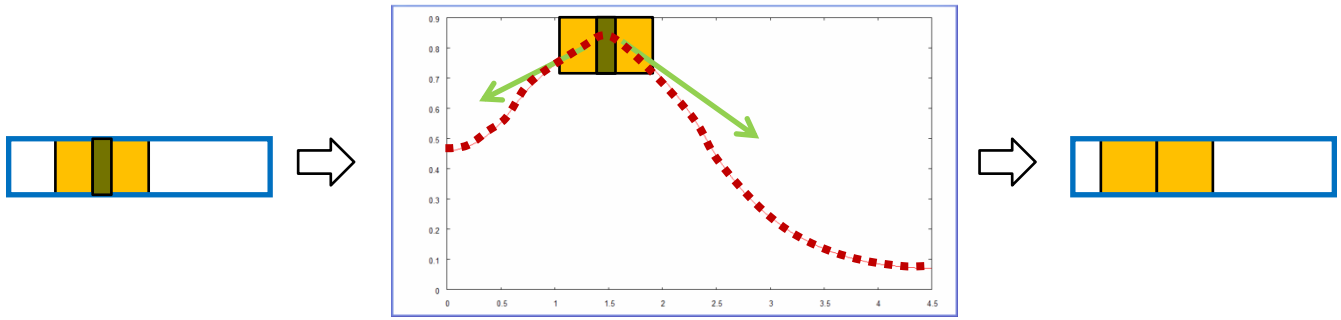
This work is partially supported by
NSF CCF-0430077 and CCF-0528583

Outline

- Related Work, Motivation & Contributions
- NLP Formulation and Gradient Computation
  - Nonlinear programming formulation
  - Iterative methods
  - Density and smoothing techniques
  - Efficient Gradient computation
- Experimental Results
- Summary and Future Work

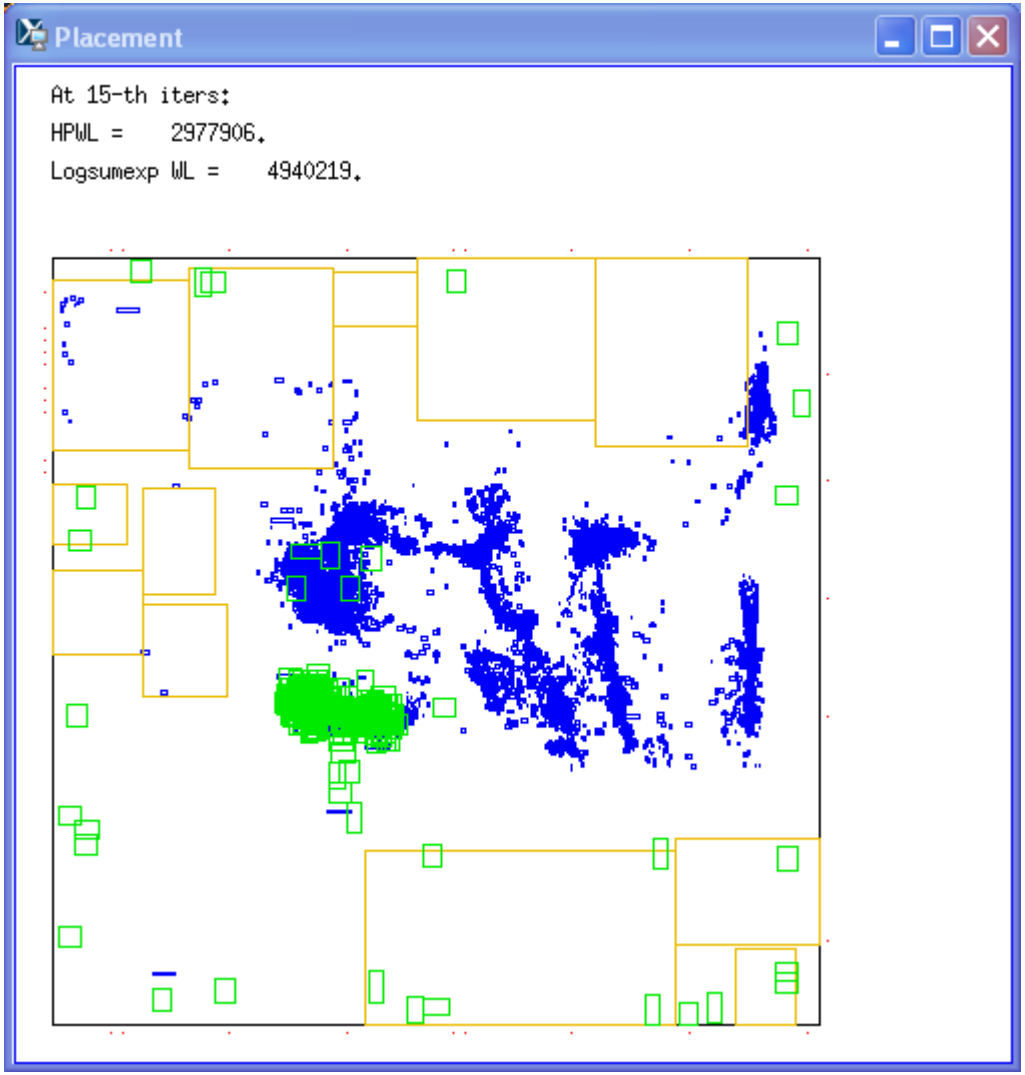
Related Work

- Top performing placers in ISPD'06 placement contest
  - Kraftwerk [Eisenmann & Johannes DAC' 98]
  - mPL6 [Chan, Cong, et al., ISPD'06]
  - NTUplace [Chen, Chang, et al. ICCAD'06]
- Iterative, density-driven overlap removal



  - Guided by "force",
  - or by descend direction of density penalty

An Example of Analytical Placement



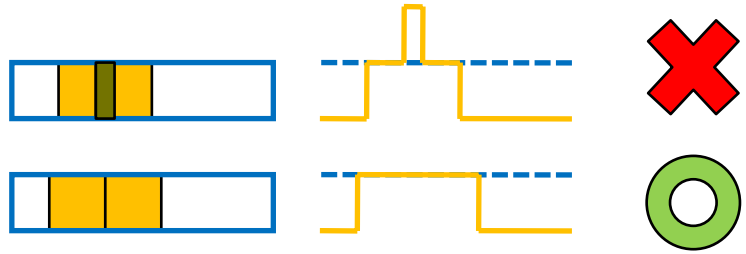
- Blue: standard cells
- Green: movable macros
- Yellow: fixed macros

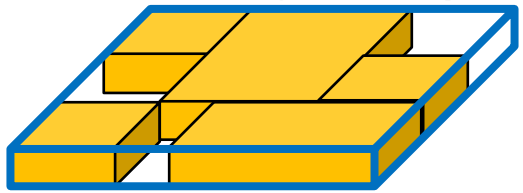
Motivation & Contributions

- Common strategies in analytical placers
  - Density smoothing techniques
    - Some smooth operators without closed-form formula involved
  - Iterative unconstrained optimizations
    - Frequent gradient computation is necessary
- Our contributions
  - Unify a wide range of smoothing techniques
  - Derive a highly efficient gradient computation method
  - Show good quality results for mixed-size placements

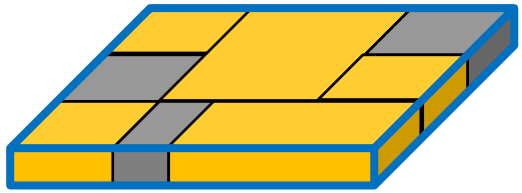
Nonlinear Programming Formulation for Placement

- Wirelength-driven placement with density constraints
  - NLP formulation with inequality constraints
    - minimize: total wirelength
    - subject to: cell_density $\leqslant$ capacity



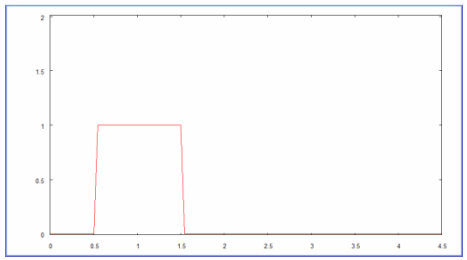


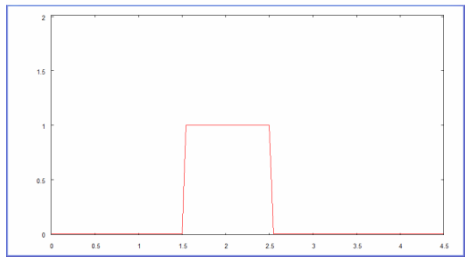
  - NLP formulation with equality constraints
    - minimize: total wirelength
    - subject to: cell_density + filler_density = capacity

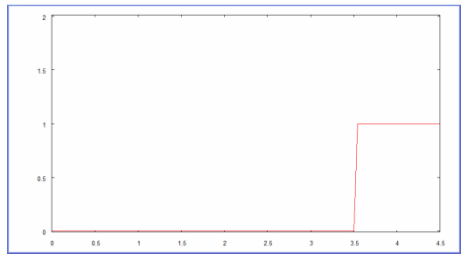


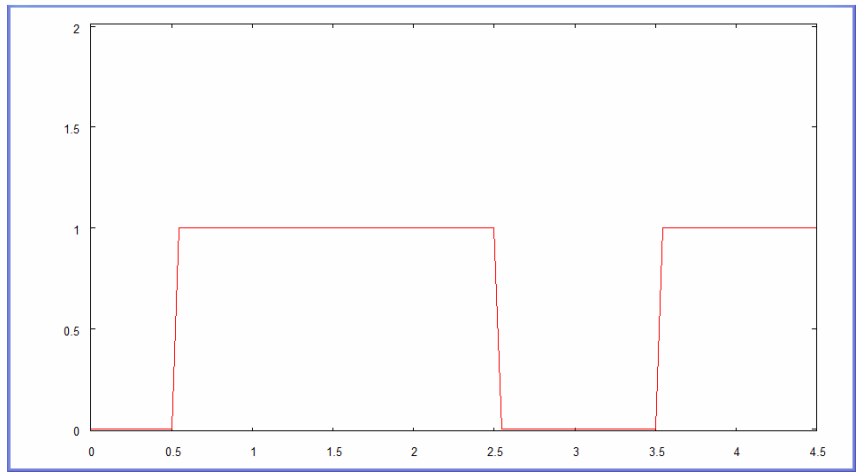
Density Functions (1/3)

- Original Density



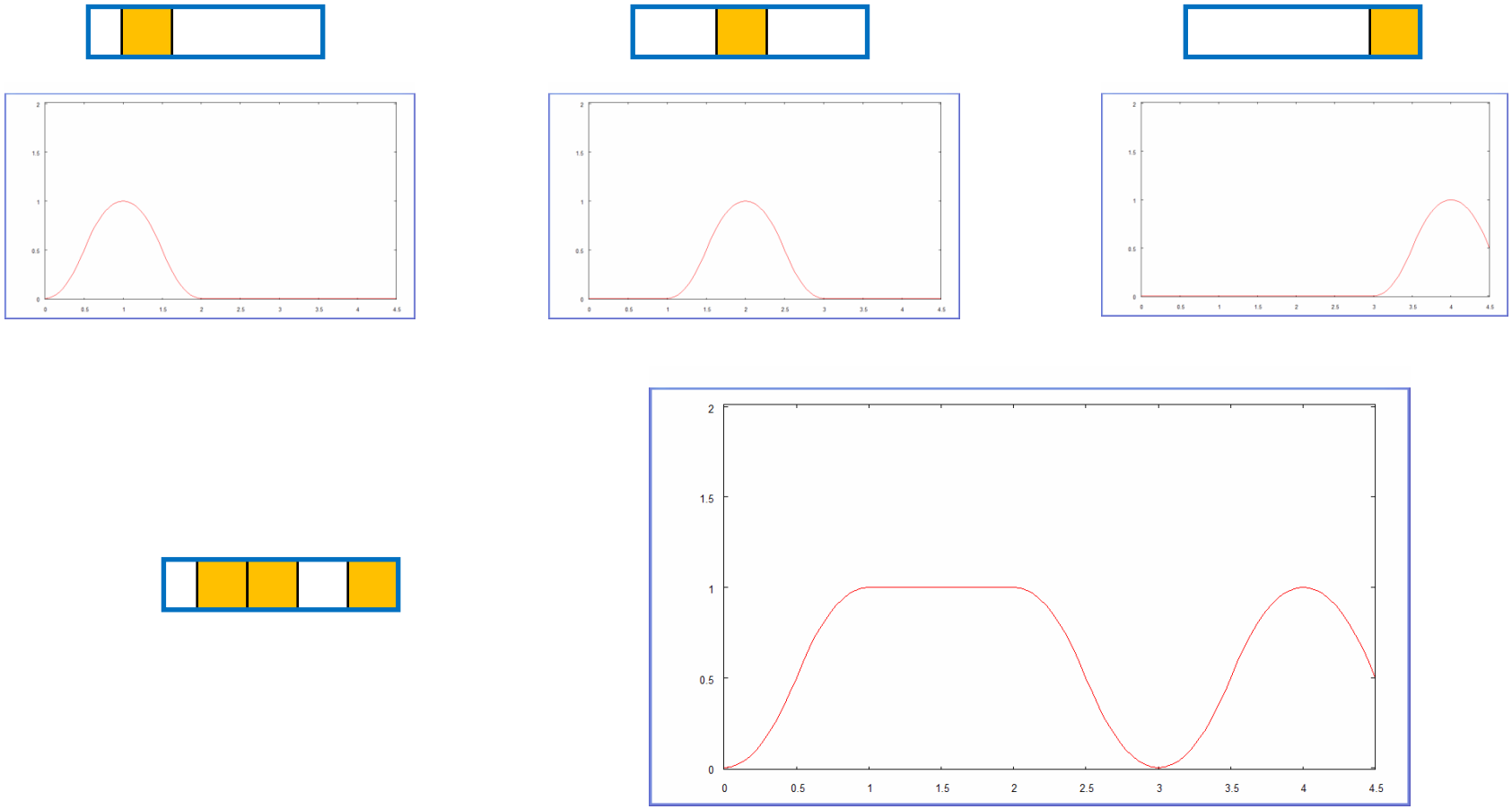






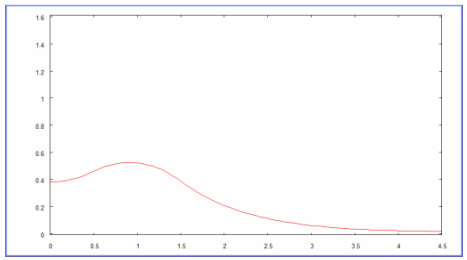
Density Functions (2/3)

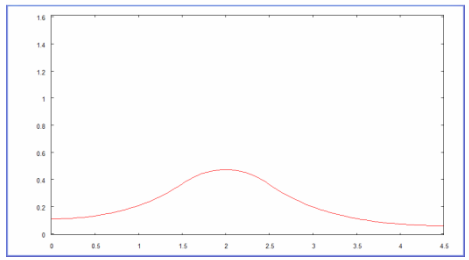
- Bell-shaped Density (Naylor, APlace, NTUplace)

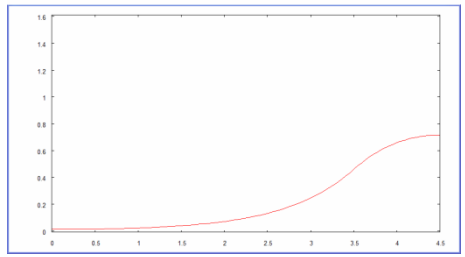


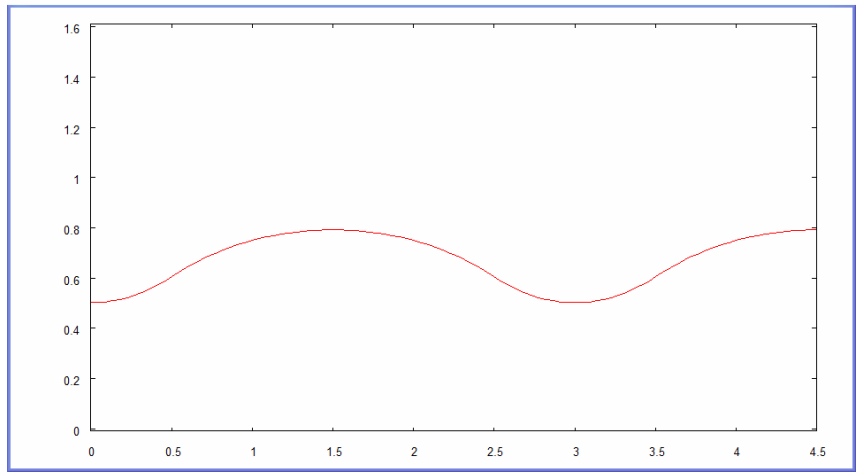
Density Functions (3/3)

- Poisson/Helmholtz-smoothed Density (Kraftwerk, mPL)







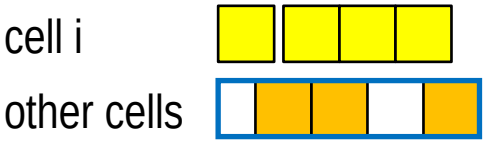
Iterative Methods

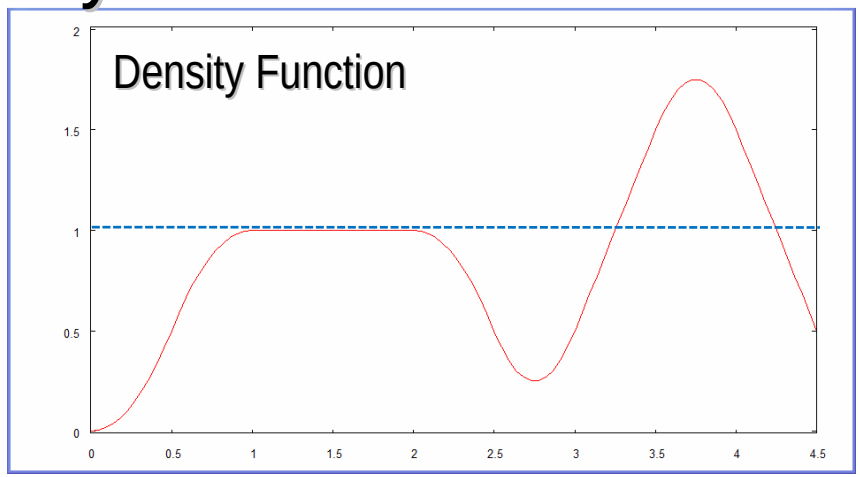
- Problem formulation
  - minimize: wirelength
  - subject to: cell_density = capacity
- Unconstrained problems
  - Penalized objective function
    - OBJ_{μ} = wirelength + μ • Penalty
  - Penalty function
    - e.g. squared deviation: Penalty = ● ^{2} + ● ^{2}

- Iterative method
  - loop { minimize OBJ_{μ} ; if (converge) break; else increase μ ; }

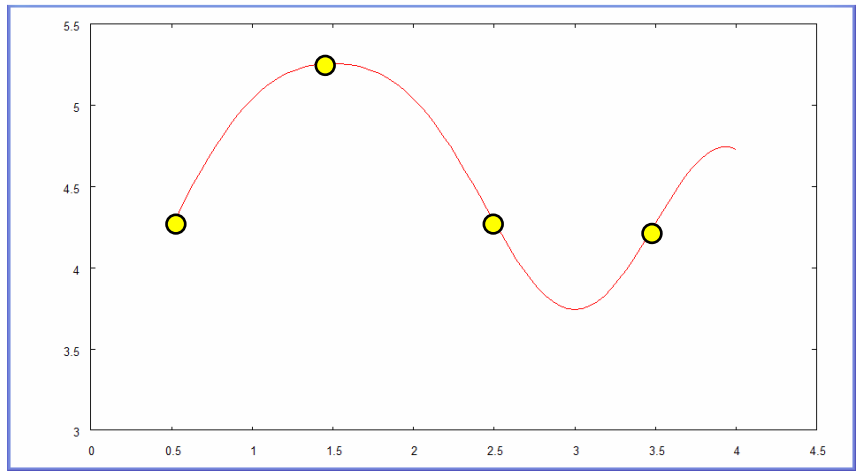
Density Penalty Functions (1/3)

- Bell-shaped density penalty function



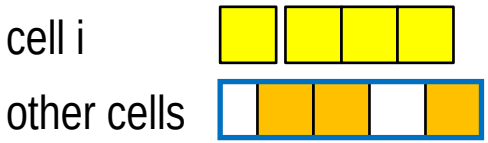


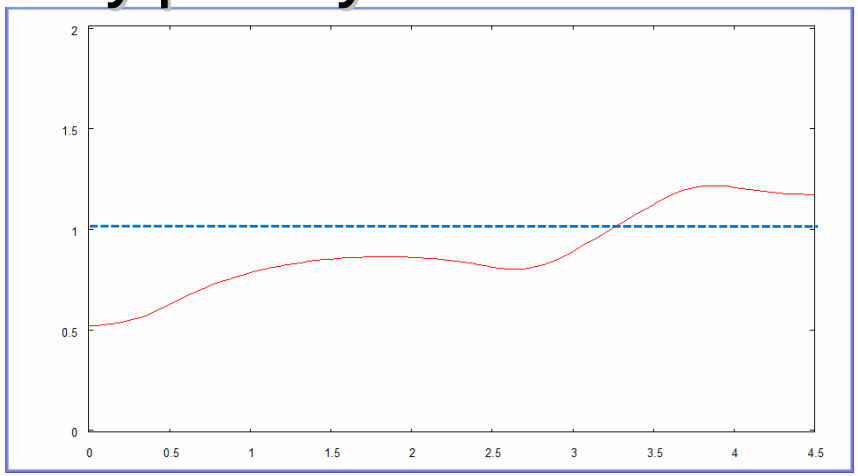
(high dimensional)
Density Penalty Function
(projected in x_i)



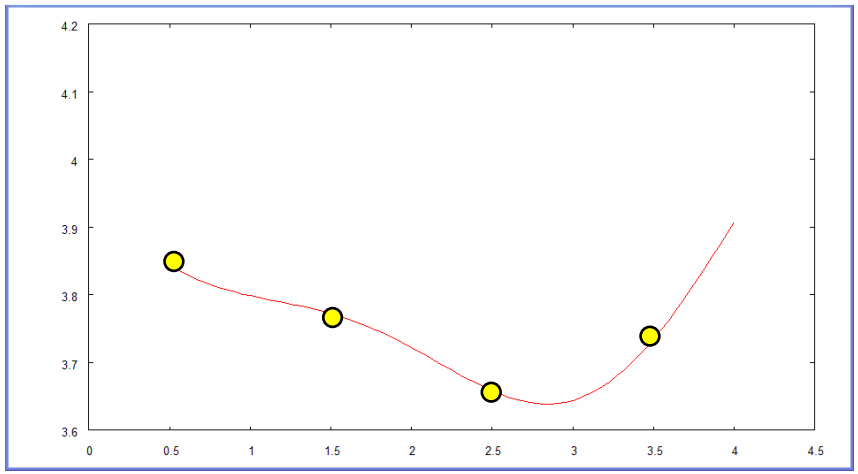
Density Penalty Functions (2/3)

- Helmholtz-smoothed density penalty function

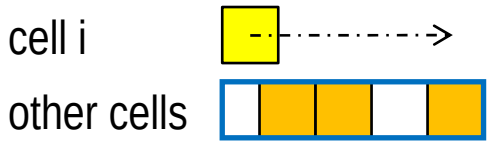




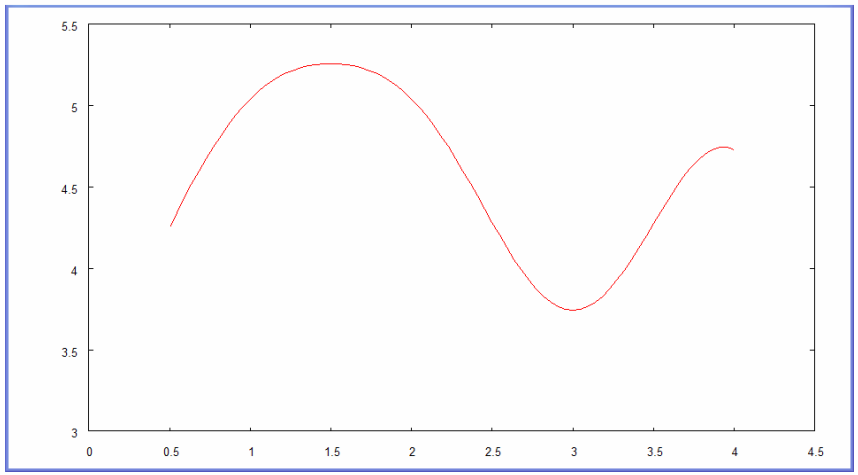
Density Penalty Function
(projection in x_i)



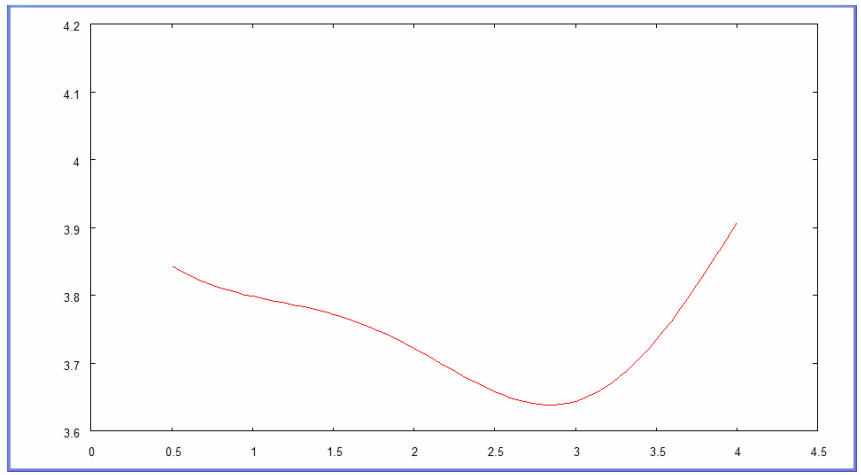
Density Penalty Functions (3/3)



Bell-shaped



Helmholtz-smoothed

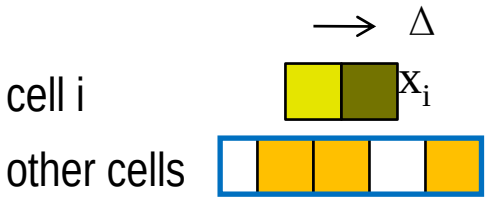


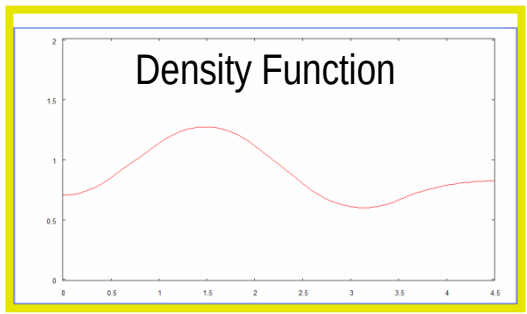
Density Smoothing Techniques

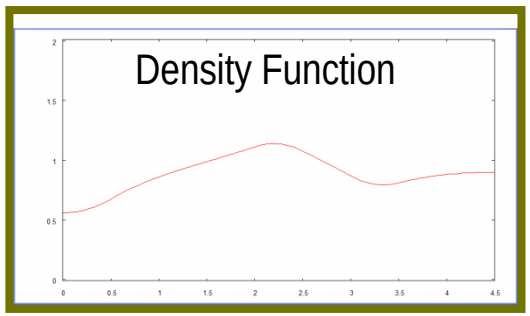
- Local smoothing
  - Locally smooth the non-differentiable edges
- Global smoothing
  - Globally distribute the original density but maintain consistency
  - We unified these global smoothing techniques
    - An integral transformation
      - $\psi(u,v) = \int_0^1 \int_0^1 D(u',v')G(u,v,u',v')du'dv'$
    - Including but not limited to
      - Poisson smoothing (Kraftwerk)
      - Helmholtz smoothing (mPL)
      - Gaussian smoothing (NTUplace)
    - Keywords: Elliptic PDE, Green's function
  - Our gradient computation works with such smoothing techniques

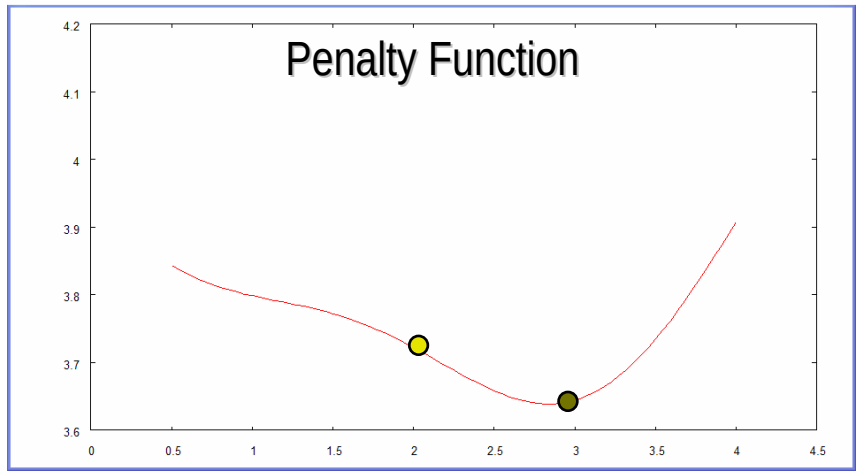
Gradient Computation (1/2)

- Naïve computation by finite difference scheme





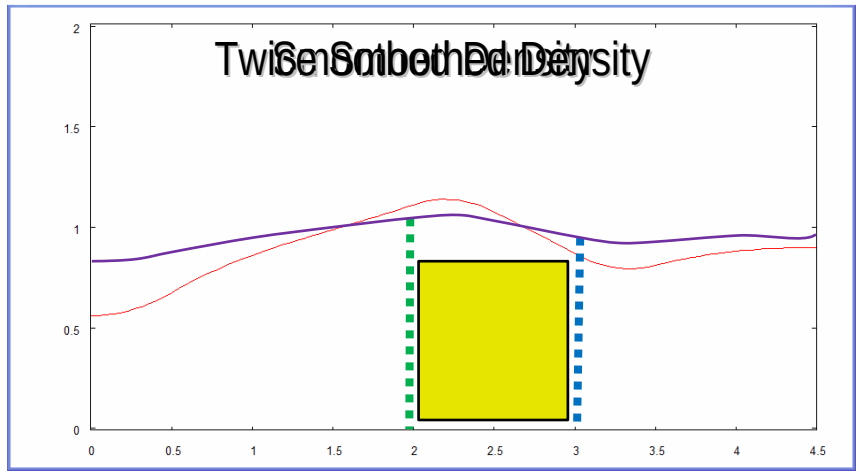




$$\frac{\partial \text{Penalty}}{\partial x_i} \approx \frac{\bullet - \circ}{\Delta x_i}$$

Gradient Computation (2/2)

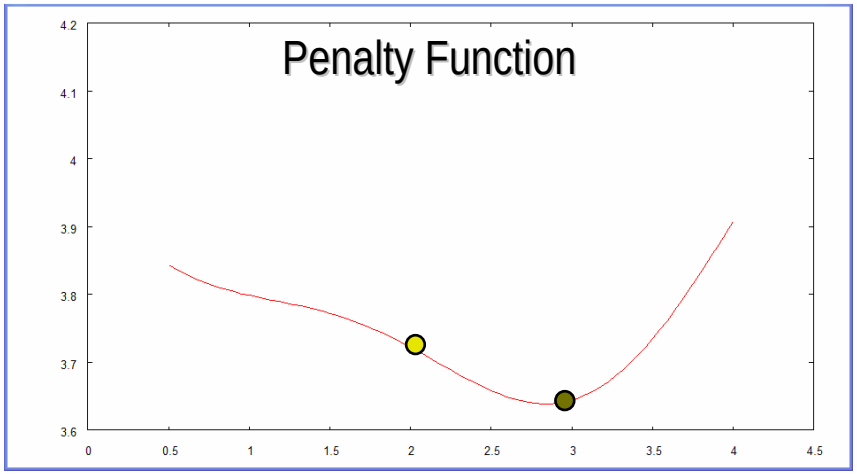
- Our efficient method



$$\frac{\partial \text{Penalty}}{\partial x_i} = \frac{\vdots - \vdots}{w_i}$$

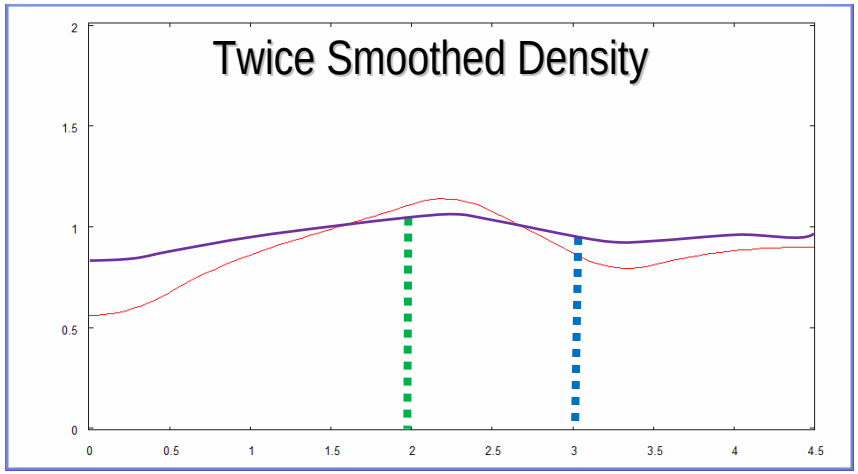
Comparison of the Gradient Computation Methods

- Naïve computation



$$\frac{\partial \text{Penalty}}{\partial x_i} \approx \frac{\bullet - \bullet}{\Delta x_i}$$

- Our efficient computation

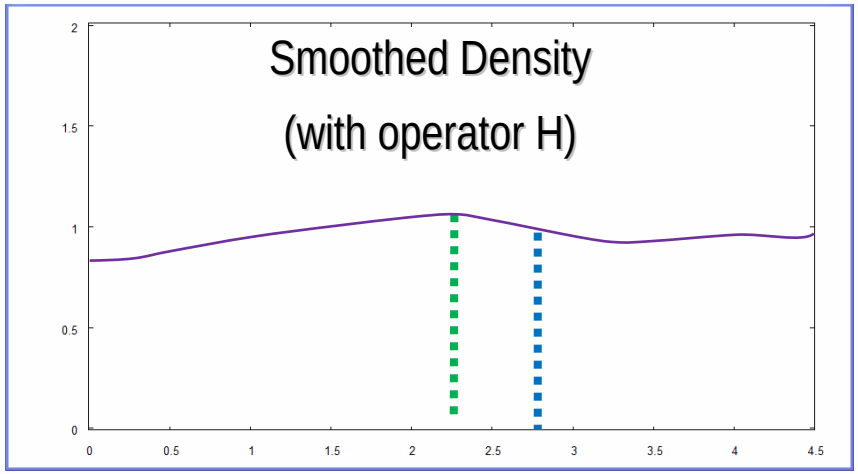


$$\frac{\partial \text{Penalty}}{\partial x_i} = \frac{\vdots - \vdots}{w_i}$$

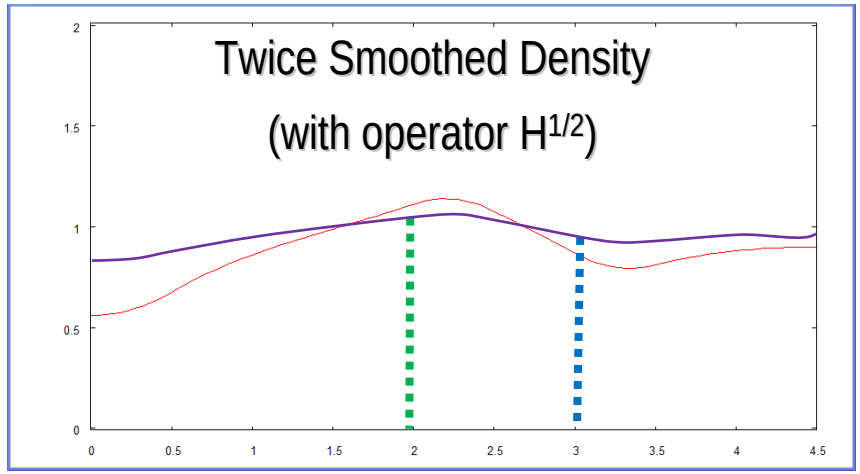
- Avoid exploring the high dimensional penalty function
- Reduce the computational time by a factor of *n*

Comparison with the mPL Implementation

- mPL implementation



- Our efficient computation



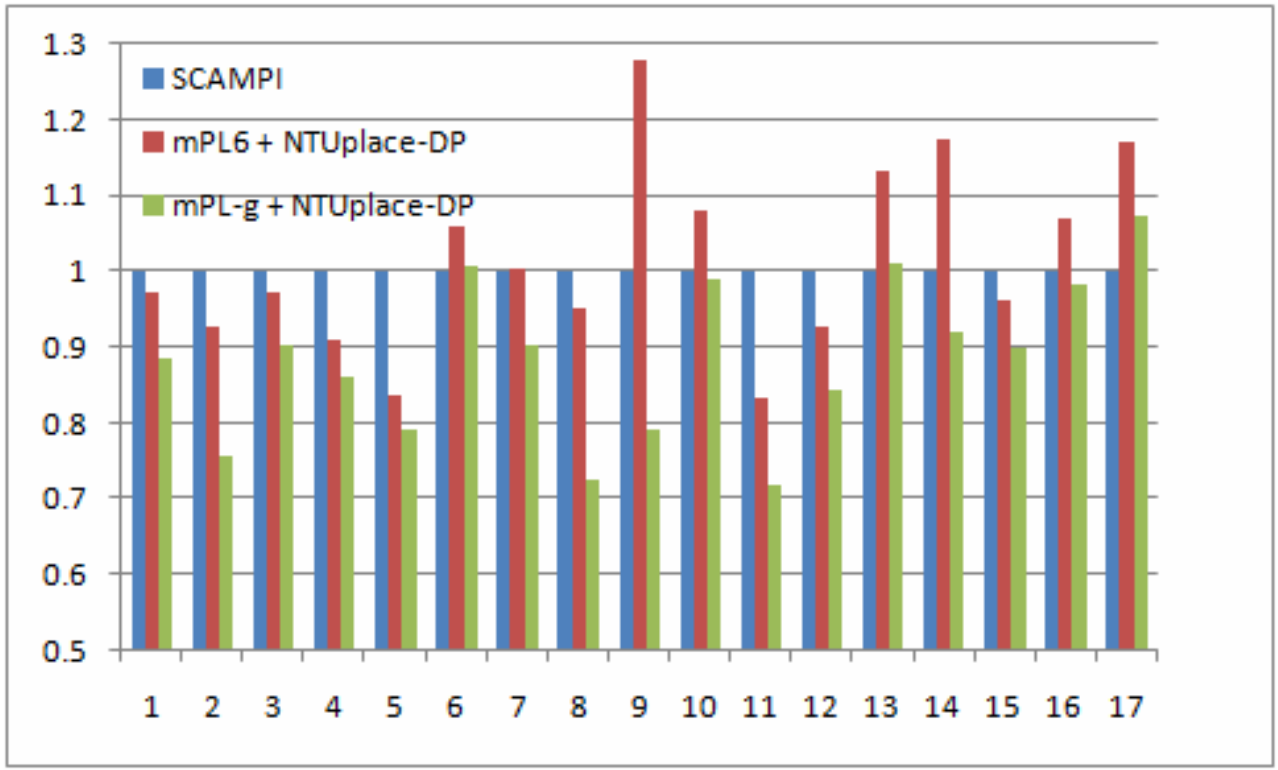
$$\frac{\partial \text{Penalty}}{\partial x_i} \approx \frac{\vdots - \vdots}{\Delta x_i} \cdot A$$

$$\frac{\partial \text{Penalty}}{\partial x_i} = \frac{\vdots - \vdots}{w_i}$$

- Essentially the same for small cells with proper scaling *A*
- Our method is directly applicable to large macros

Experimental Results on IBM-HB+ Benchmarks

- In average, wirelength is 13% shorter than SCAMPI [Ng, Markov, et al.]
- And 15% shorter than mPL6 [Chan, Cong, et al.]
  - NTUplace [Chen, Chang, et al.] as the detailed placer



Summary & Future Work

- We proposed an efficient gradient computation method
  - Applicable to a wide range of global smoothing techniques
  - Reduce the runtime by a factor of *n* compared to a naive method
  - Achieved good quality mixed-size placement results
- Future work
  - Apply the computation to test other iterative methods
    - Such as augmented Lagrangian method
  - Extend the nonlinear programming framework to handle complex objective and constraints
    - such as power density constraints
    - and TS via constraints in 3D placement

Thank You!

Backup Slide

$$\frac{\partial P(\vec{x},\vec{y};\mu)}{\partial x_k} = \quad - \quad$$

- An equivalent expression

$$\frac{\partial P(\vec{x},\vec{y};\mu)}{\partial x_k} \approx \frac{\partial \mu \int_0^1 \int_0^1 \left(\hat{D}_{(\vec{x},\vec{y})}(u,v) - \bar{C}(u,v)\right)^2 dudv}{\partial x_k}$$

$$= 2\mu \int_0^1 \int_0^1 \left(\hat{D}_{(\vec{x},\vec{y})}(u,v) - \bar{C}(u,v)\right) \frac{\partial \hat{D}_{(\vec{x},\vec{y})}(u,v)}{\partial x_k} dudv$$

$$\frac{\partial \hat{D}_{(\vec{x},\vec{y})}(u,v)}{\partial x_k} = \frac{\partial \int_0^a \int_0^b D_{(x_k,y_k)}(u',v') G(u,v,u',v') du'dv'}{\partial x_k}$$

Leibniz Integral Rule

$$= \int_{y_k - h_k/2}^{y_k + h_k/2} G(u,v,x_k + w_k/2, v')\, dv' - \int_{y_k - h_k/2}^{y_k + h_k/2} G(u,v,x_k - w_k/2, v')\, dv'$$

$$\frac{\partial P(\vec{x},\vec{y};\mu)}{\partial x_k} = 2\mu \left(\int_{y_k - h_k/2}^{y_k + h_k/2} \left(\hat{D}_{(\vec{x},\vec{y})}(u',v') - \bar{C}(u',v')\right) dv' \right) \Bigg|_{u'=x_k - w_k/2}^{u'=x_k - w_k/2}$$

$$2\mu\left(\hat{D}_{(\vec{x},\vec{y})}(u',v') - \bar{C}(u',v')\right)$$