A High-Performance Triple Patterning Layout Decomposer with Balanced Density

Bei Yu[1] Yen-Hung Lin[2] Gerard Luk-Pat[3] Duo Ding[4]
Kevin Lucas[3] David Z. Pan[1]

[1]ECE Dept., University of Texas at Austin [3]Synopsys Inc.

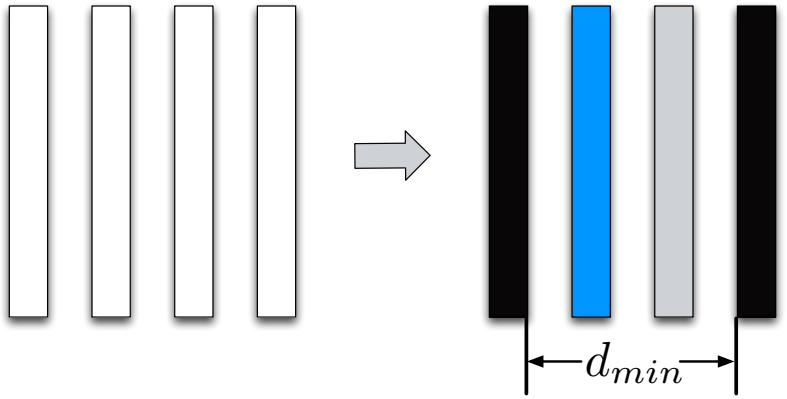
[2]CS Dept., National Chiao Tung University [4]Oracle Corp. Austin

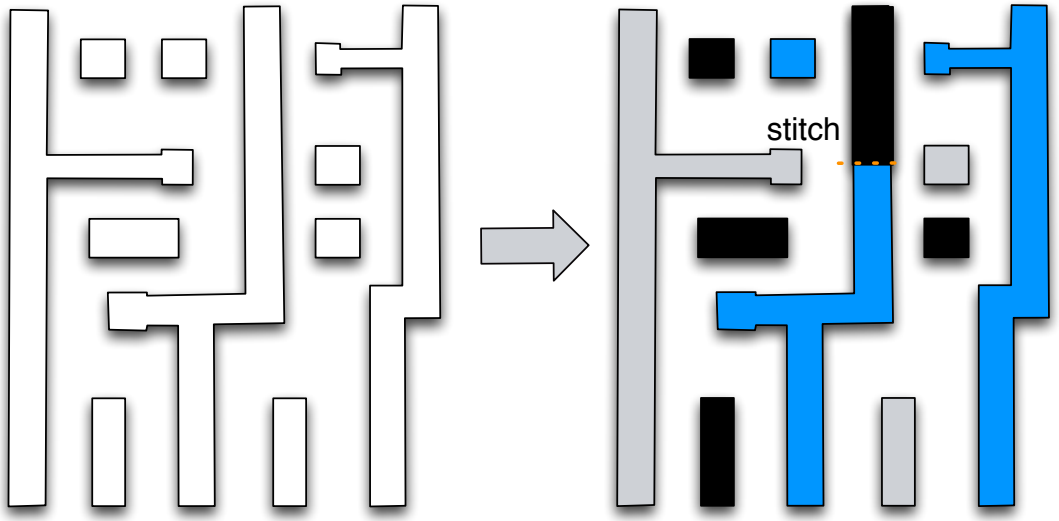
SYNOPSYS® ORACLE®

Triple Patterning Lithography (TPL)

ITRS roadmap

- 28nm single-patterning
- 20nm double-patterning
- 14nm triple-patterning / EUV
- 10nm quadruple-patterning / EUV





TPL Decomposition Works

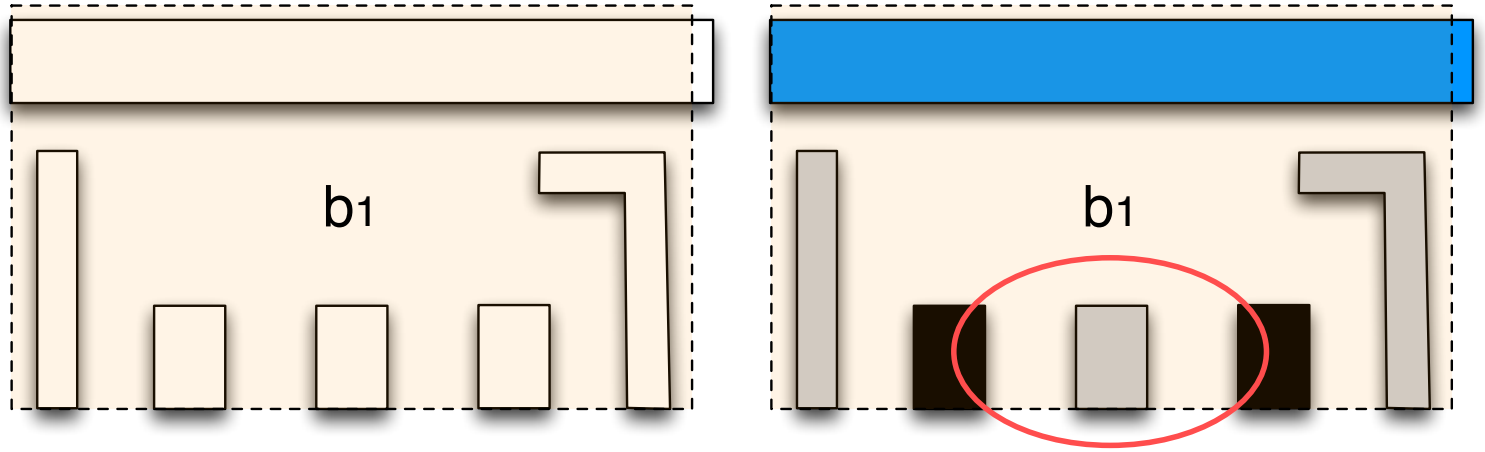
– **ILP or SAT**
[Cork+,SPIE'08][Yu+,ICCAD'11][Cork+,SPIE'13]

– **Graph Search for Row based Layout**
[Tian+, ICCAD'12][Tian+,SPIE'13]

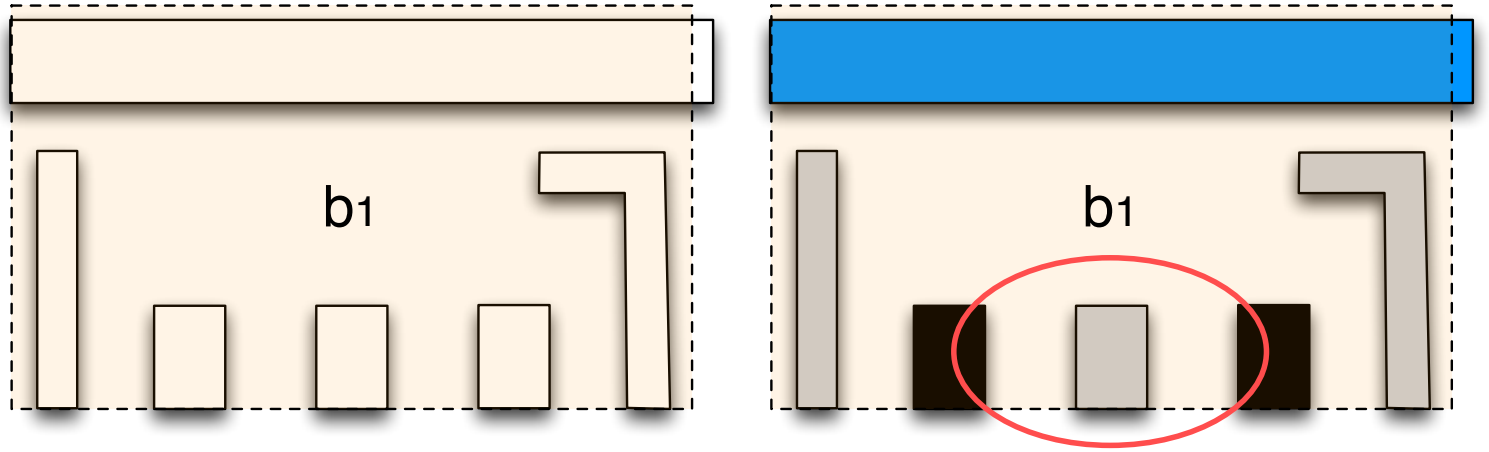
– **Heuristic**
[Ghaida+,SPIE'11][Fang+,DAC'12][Chen,ISQED'13]
[Kuang+,DAC'13][Tang+,Patent'13]

– **Semidefinite Programming (SDP)** (trade-off)
[Yu+, ICCAD'11][Yu+,ICCAD'13]

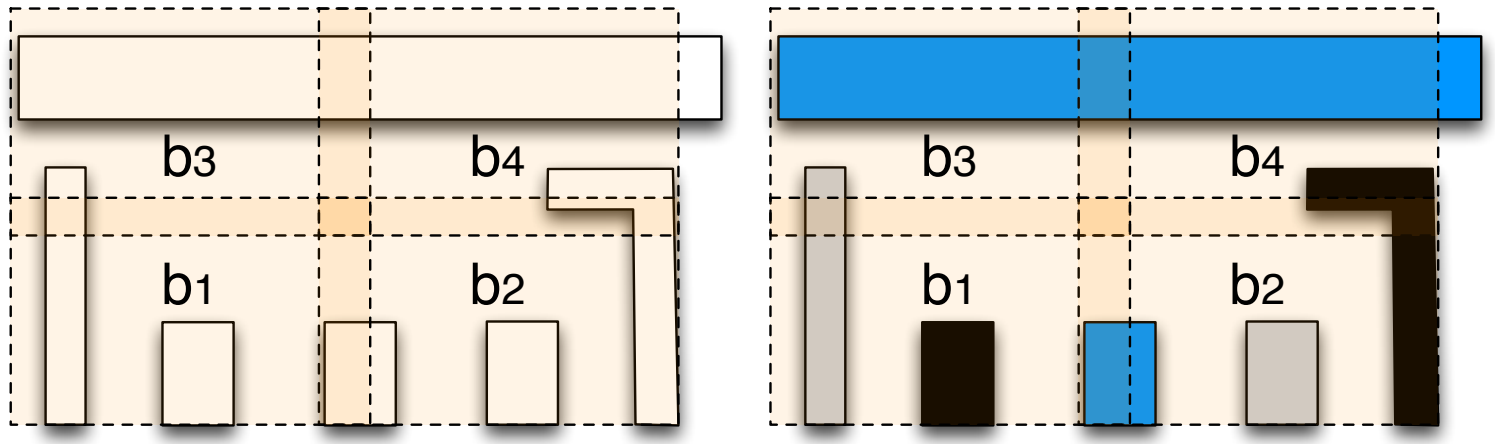
- Global Balanced Density?



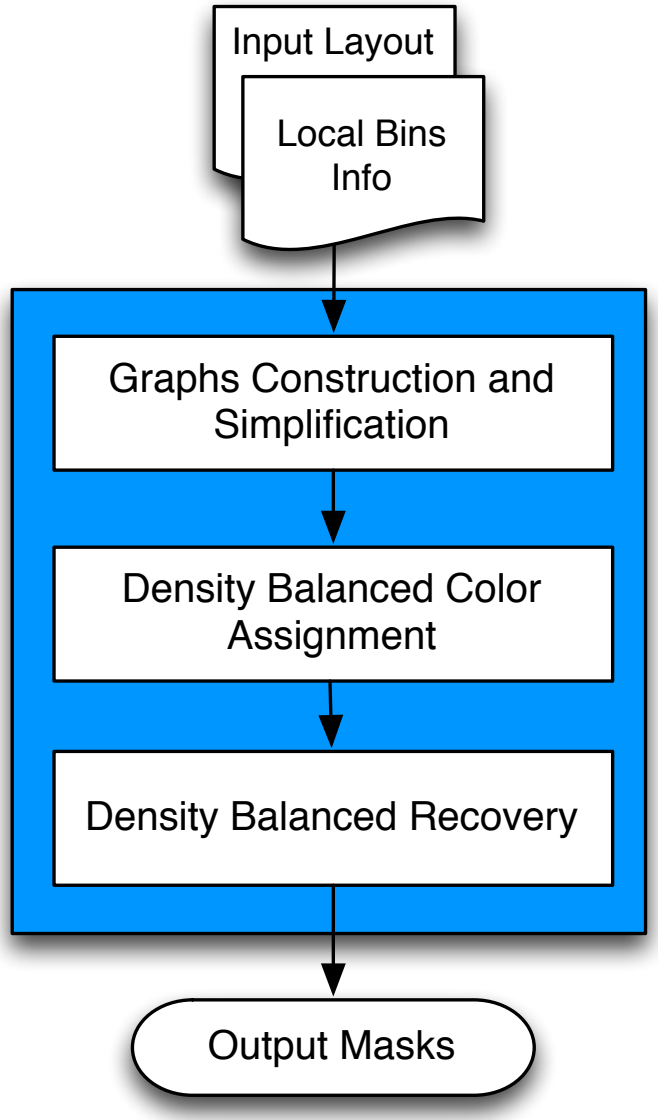
- Global Balanced Density?

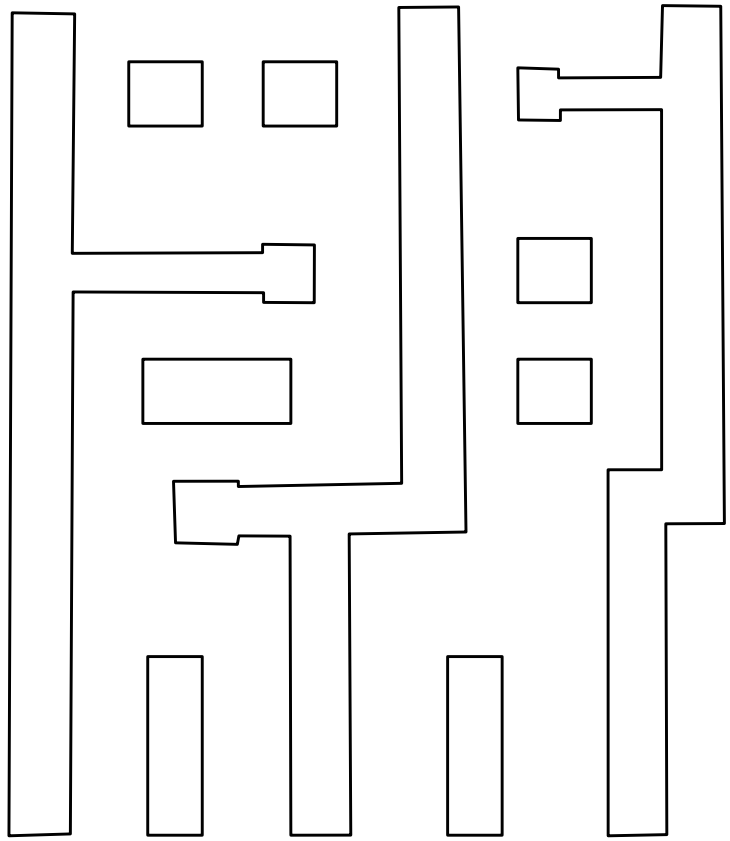


- Local Balanced Density!

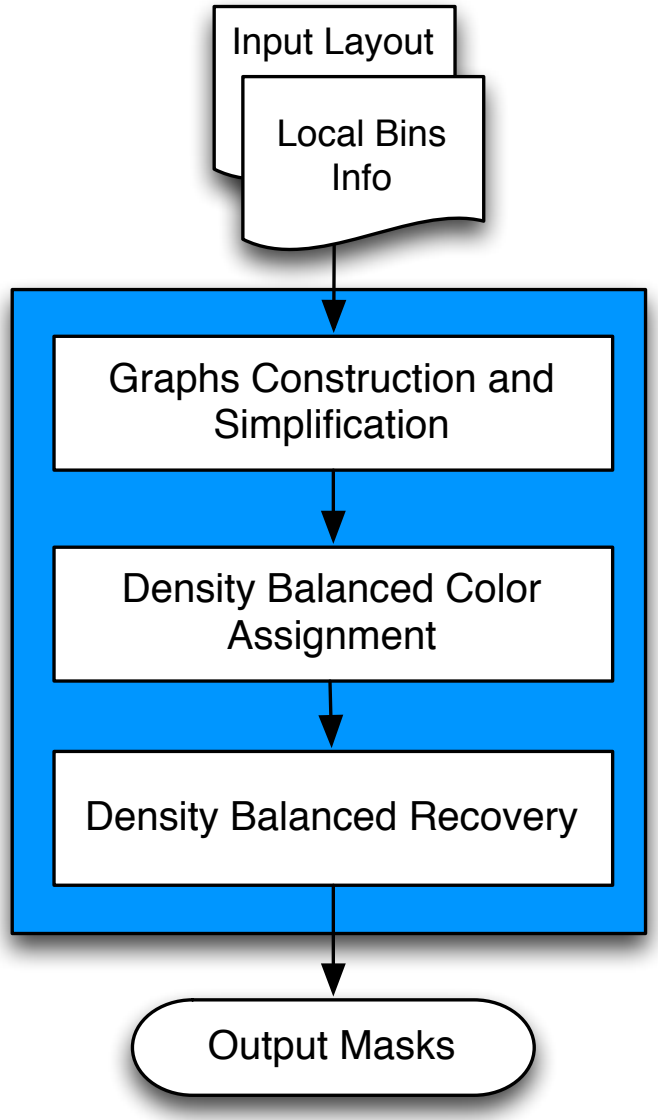


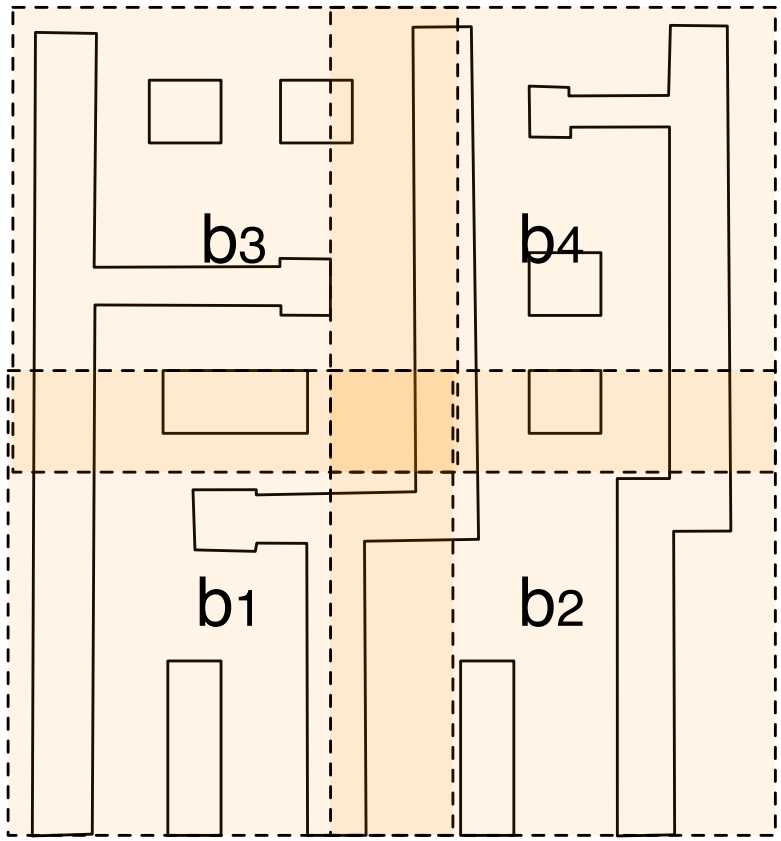
Overall Flow



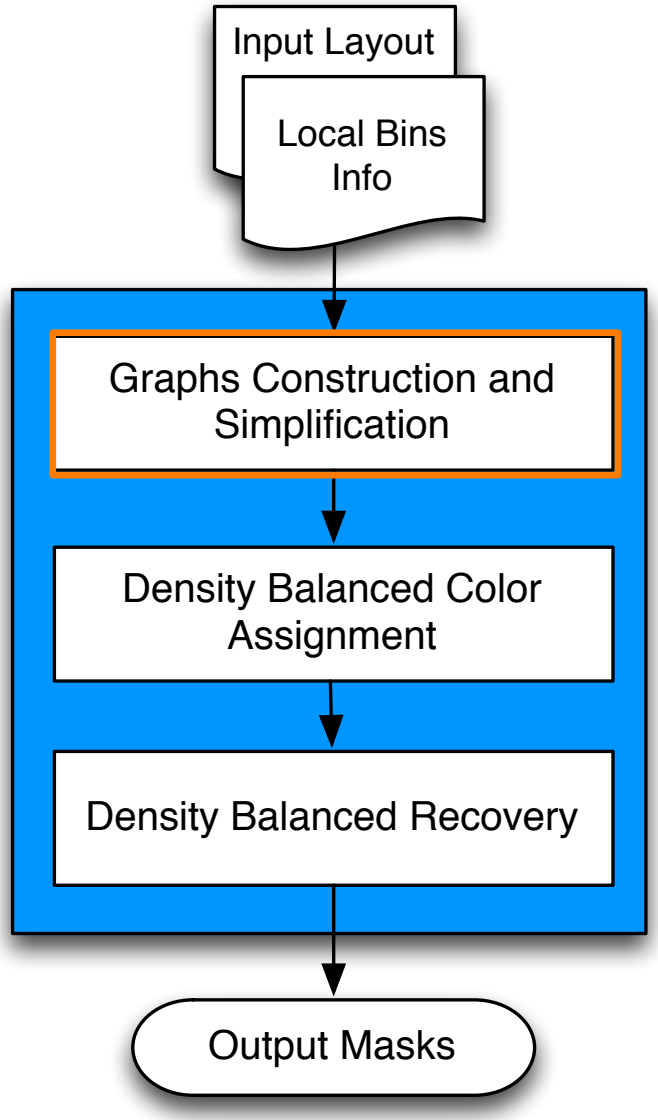


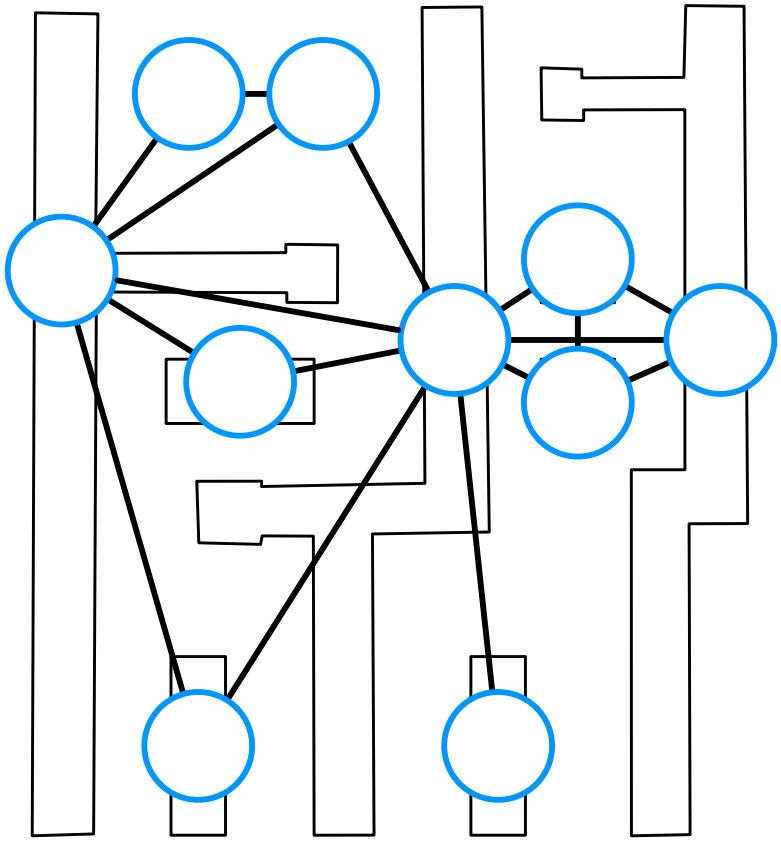
Overall Flow



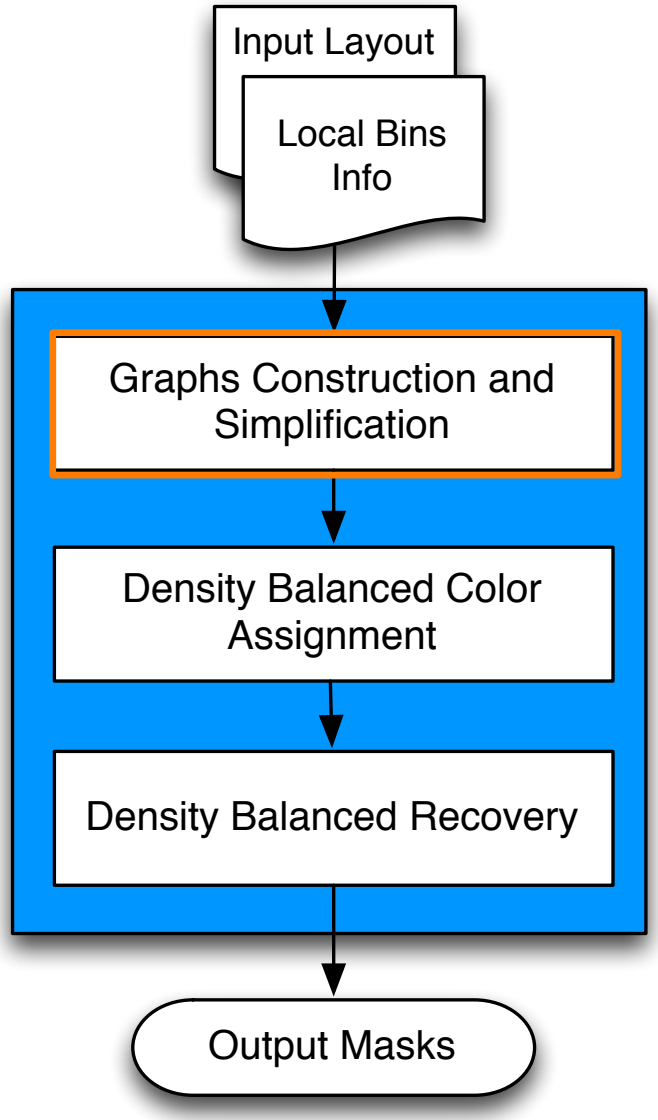


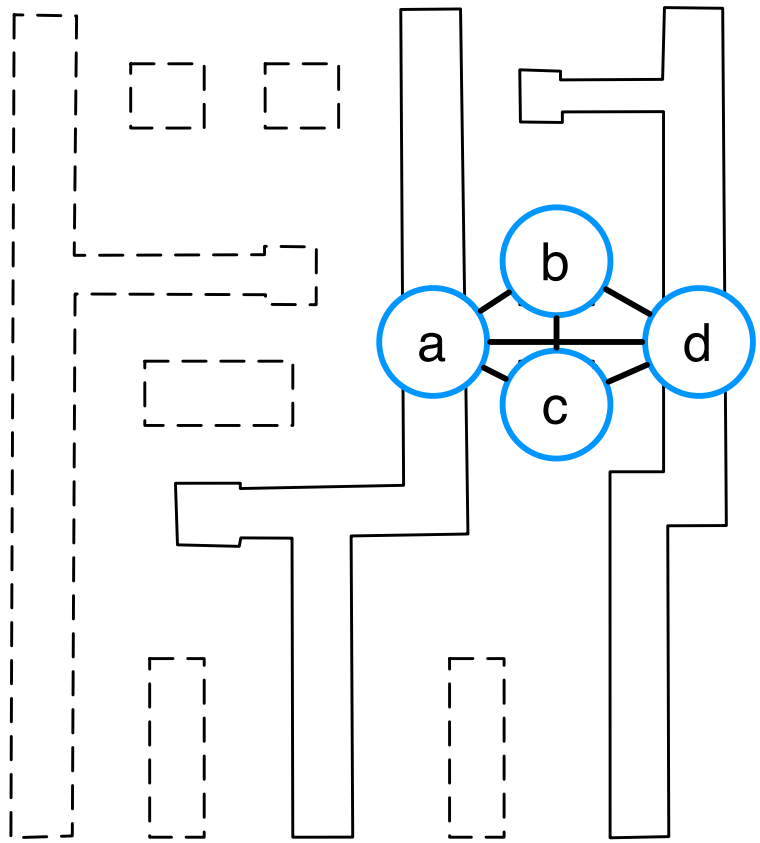
Overall Flow



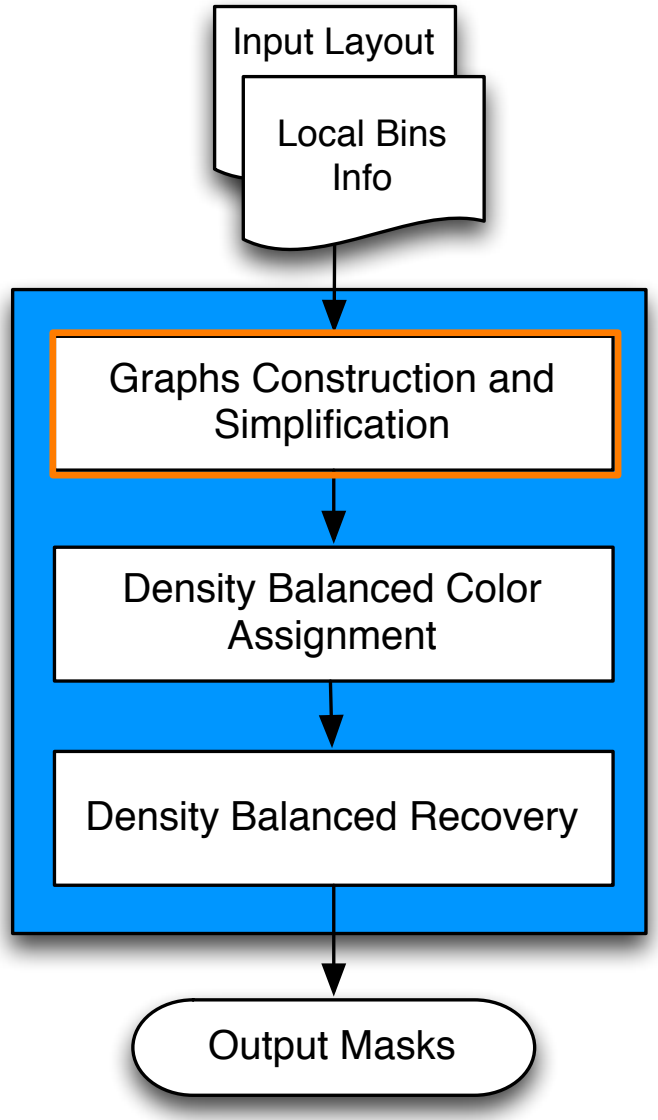


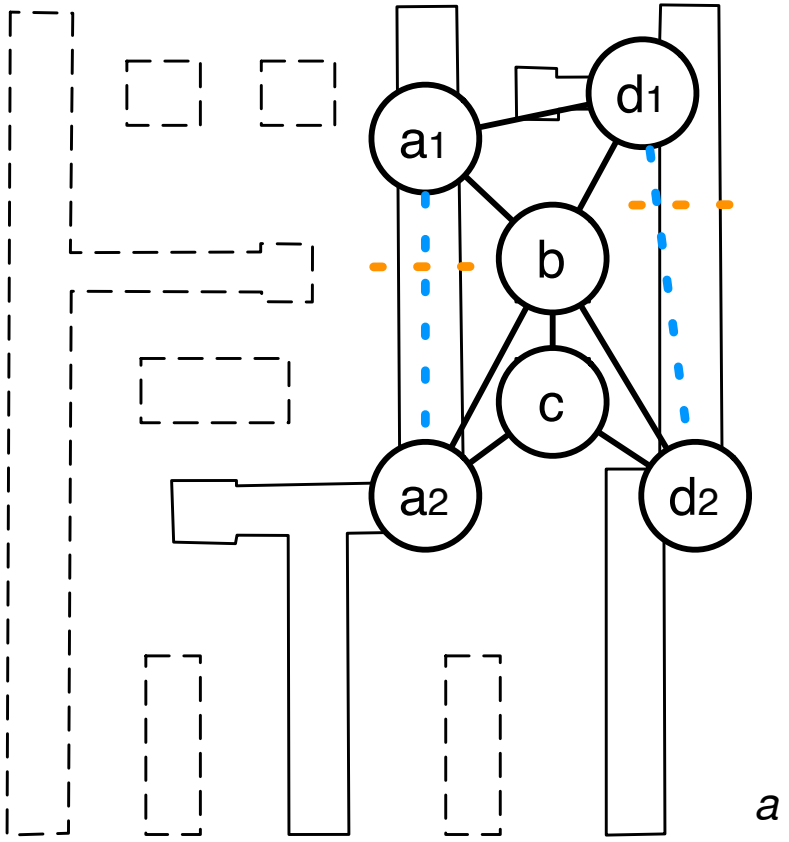
Overall Flow





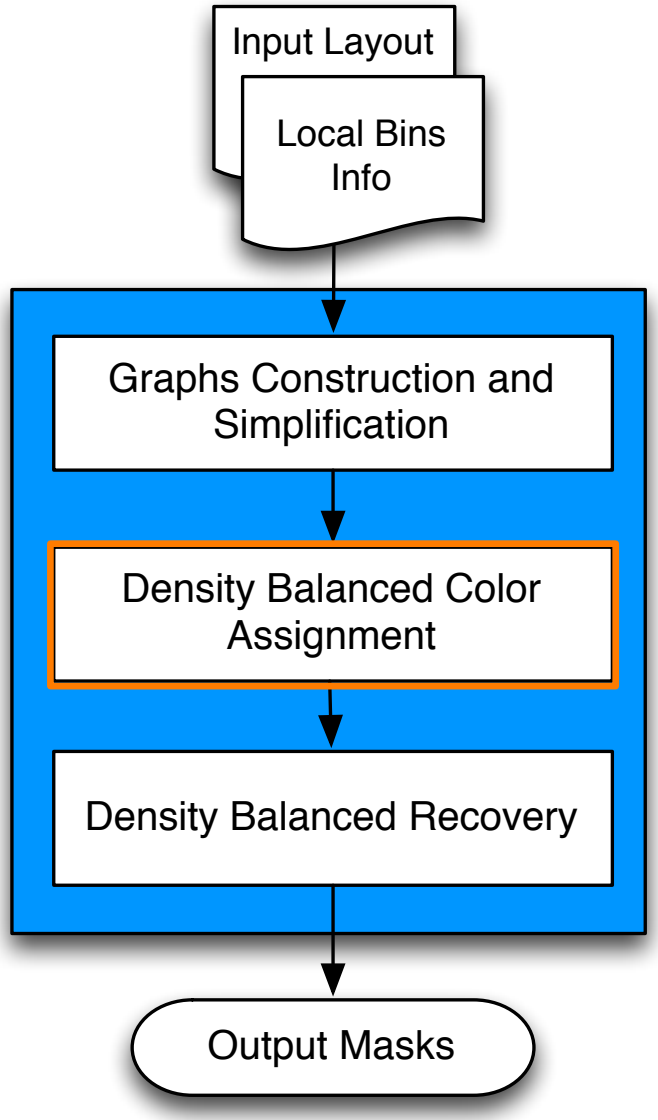
Overall Flow

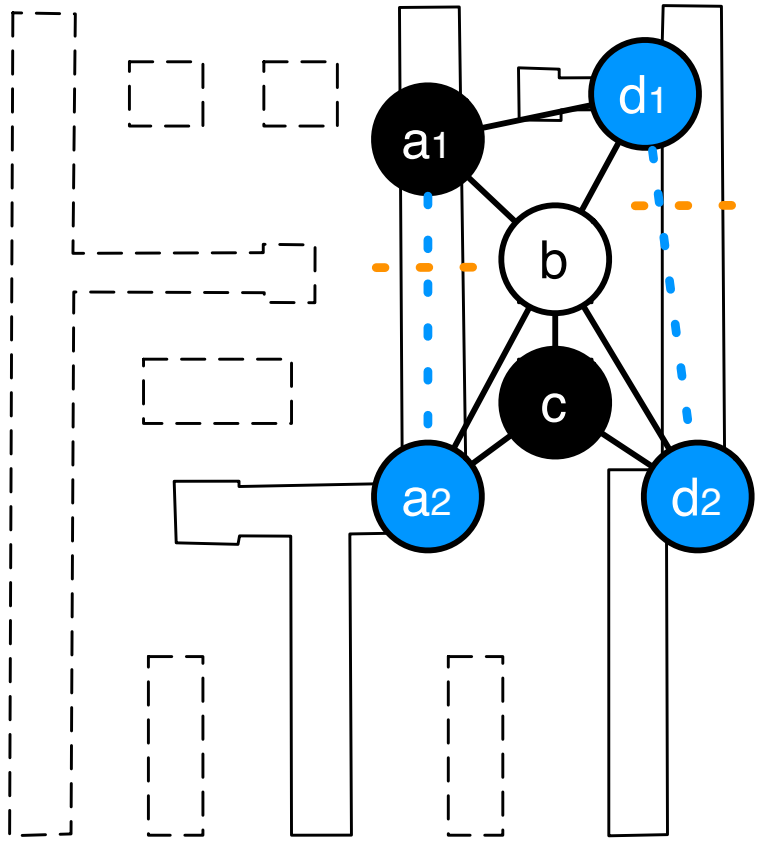




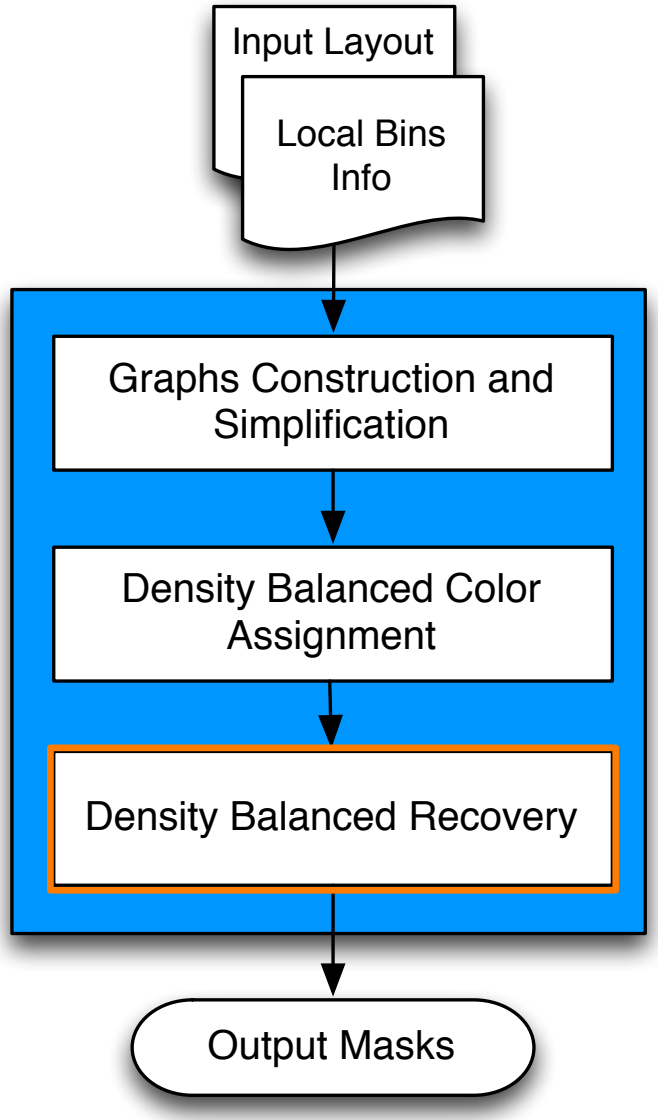
[a]Stitch candidate generation [Kuang+,DAC'13]

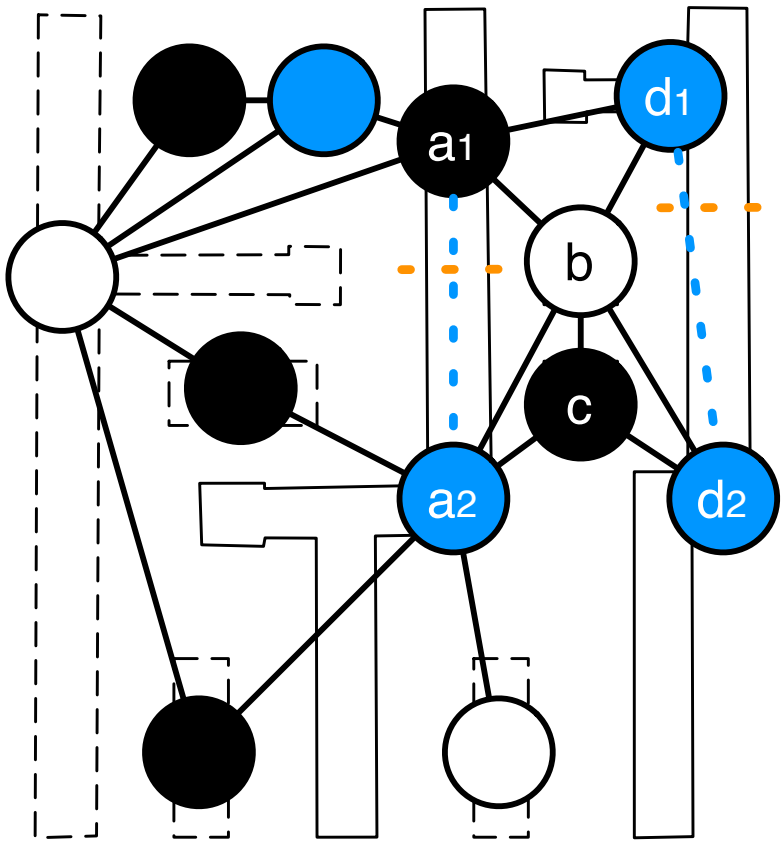
Overall Flow



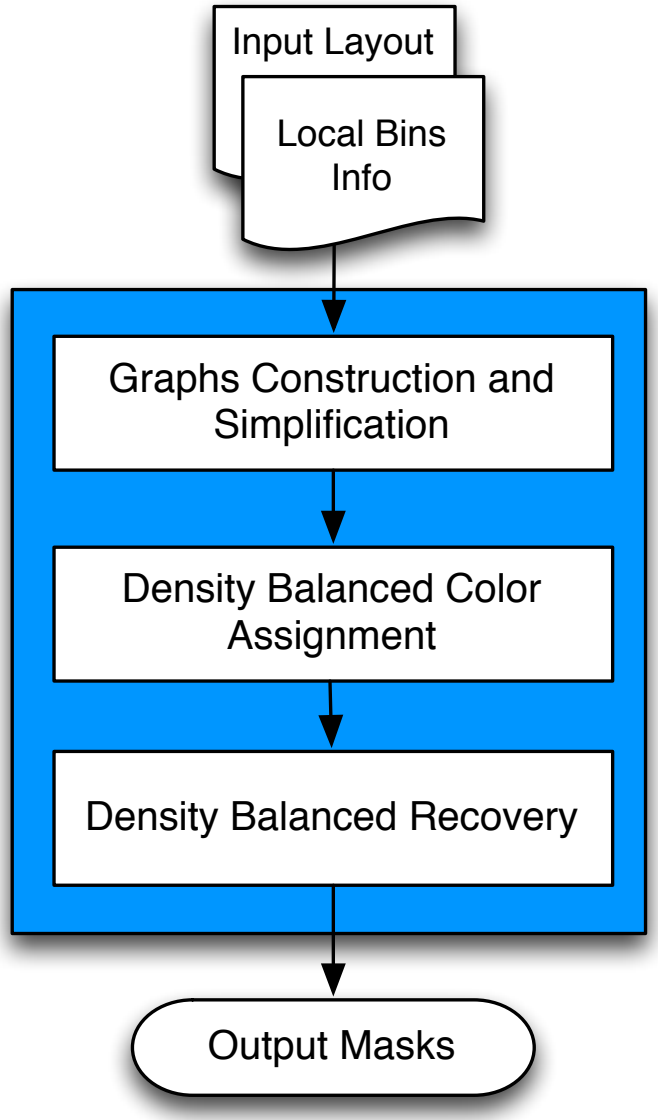


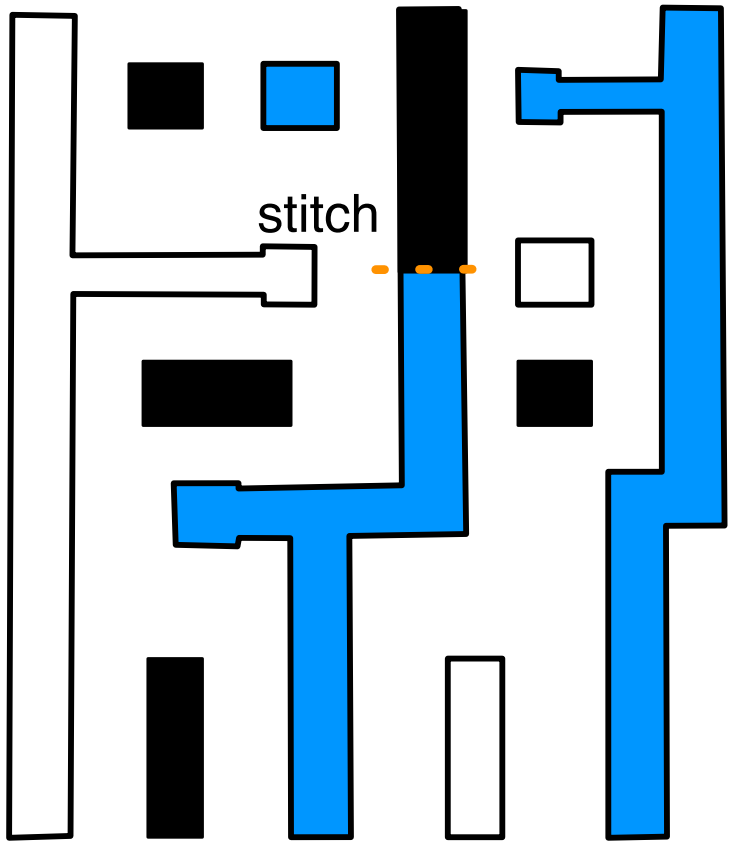
Overall Flow

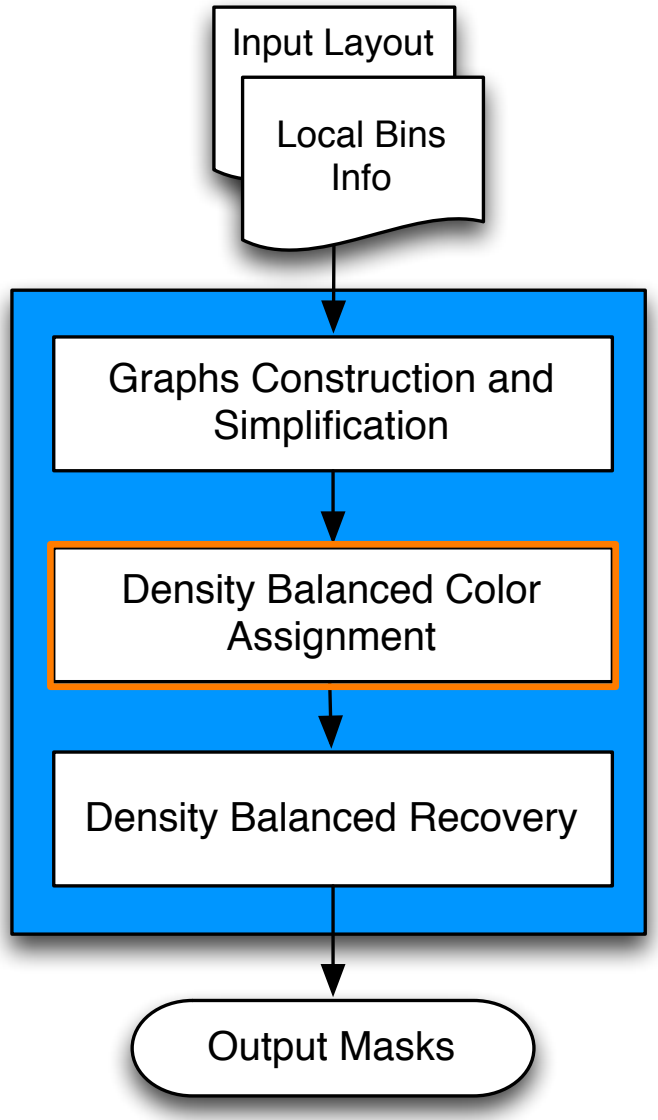




Overall Flow






Input Layout
Local Bins Info
Graphs Construction and Simplification
Density Balanced Color Assignment
Density Balanced Recovery
Output Masks

Problem Formulation

Local density uniformity in bin b_k

$$DU_k = d_{k1} \cdot d_{k2} + d_{k1} \cdot d_{k3} + d_{k2} \cdot d_{k3}$$

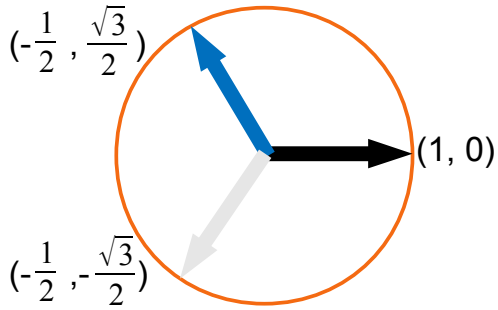
Lemma: Maximizing DU_k can achieve better density balance.

Density Balanced Color Assignment

Input	Graph model
Output	Color Assignment to the graph
Objective	min conflict#, stitch#, and max DU_k

Color representation

- Three unit vectors [Yu+,ICCAD'11]
- same color: $\vec{v}_i \cdot \vec{v}_j = 1$
- different color: $\vec{v}_i \cdot \vec{v}_j = -1/2$



Our Vector Programming

$$\min \sum_{e_{ij} \in CE} (\vec{v}_i \cdot \vec{v}_j) - \alpha \sum_{e_{ij} \in SE} (\vec{v}_i \cdot \vec{v}_j) - \beta \cdot \sum_{b_k \in B} DU_k$$

$$\text{s.t.} \quad \vec{v}_i \in \{(1,0), (-\frac{1}{2}, \frac{\sqrt{3}}{2}), (-\frac{1}{2}, -\frac{\sqrt{3}}{2})\}$$

$$DU_k = -\sum_{i,j \in V} den_{ki} \cdot den_{kj} \cdot (\vec{v}_i \cdot \vec{v}_j) \quad \forall b_k \in B$$

den_{ki}: density of feature r_i in bin b_k

Relax to Semidefinite Programming (SDP)

$$\begin{aligned} \text{SDP: } \min \quad & A \bullet X \\ & X_{ii} = 1, \quad \forall i \in V \\ & X_{ij} \geq -\frac{1}{2}, \quad \forall e_{ij} \in CE \\ & X \succeq 0 \end{aligned}$$

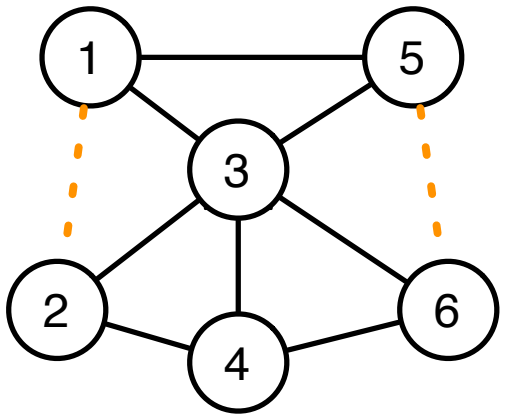
$$A_{ij} = \begin{cases} 1 + \beta \cdot \sum_k den_{ki} \cdot den_{kj}, & \forall b_k \in B, e_{ij} \in CE \\ -\alpha + \beta \cdot \sum_k den_{ki} \cdot den_{kj}, & \forall b_k \in B, e_{ij} \in SE \\ \beta \cdot \sum_k den_{ki} \cdot den_{kj}, & \text{otherwise} \end{cases}$$

Output matrix X:

- If X_{ij} close to 1, i, j same color
- If X_{ij} close to -0.5, i, j different colors

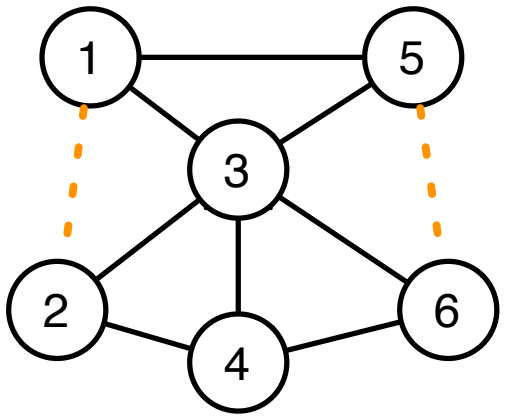
Mapping: From SDP to Color Assignment

$$X = \begin{pmatrix} 1.0 & 0.43 & -0.5 & 0.21 & -0.5 & 0.15 \\ & 1.0 & -0.5 & -0.5 & 0.15 & 0.95 \\ & & 1.0 & -0.5 & -0.5 & -0.5 \\ & & & 1.0 & 0.21 & -0.5 \\ & \ldots & & & 1.0 & 0.43 \\ & & & & & 1.0 \end{pmatrix}$$

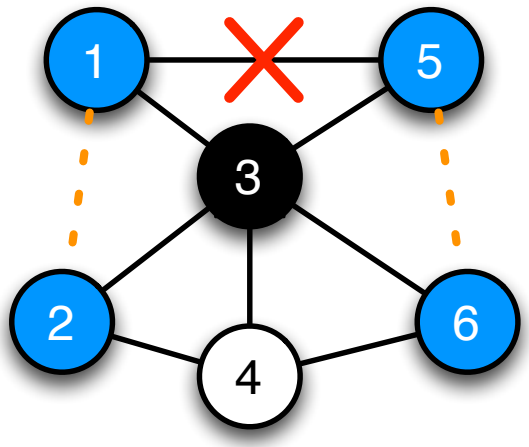


Mapping: From SDP to Color Assignment

$$X = \begin{pmatrix} 1.0 & 0.43 & -0.5 & 0.21 & -0.5 & 0.15 \\ & 1.0 & -0.5 & -0.5 & 0.15 & 0.95 \\ & & 1.0 & -0.5 & -0.5 & -0.5 \\ & & & 1.0 & 0.21 & -0.5 \\ & \ldots & & & 1.0 & 0.43 \\ & & & & & 1.0 \end{pmatrix}$$

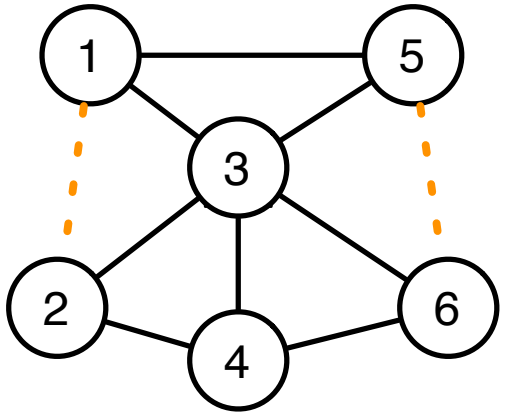


- Greedy may lose optimality
- 0.95? 0.43?

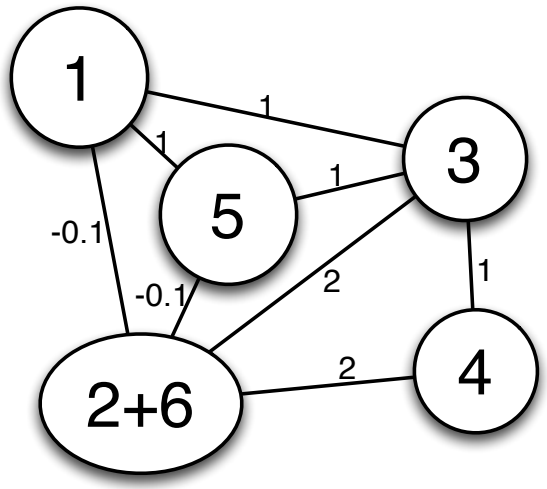


Mapping: 3-Way Max-Cut

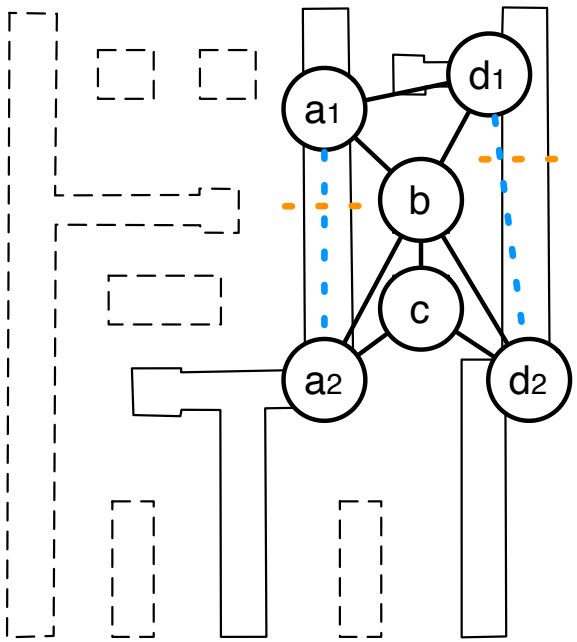
$$X = \begin{pmatrix} 1.0 & 0.43 & -0.5 & 0.21 & -0.5 & 0.15 \\ & 1.0 & -0.5 & -0.5 & 0.15 & 0.95 \\ & & 1.0 & -0.5 & -0.5 & -0.5 \\ & & & 1.0 & 0.21 & -0.5 \\ & \ldots & & & 1.0 & 0.43 \\ & & & & & 1.0 \end{pmatrix}$$

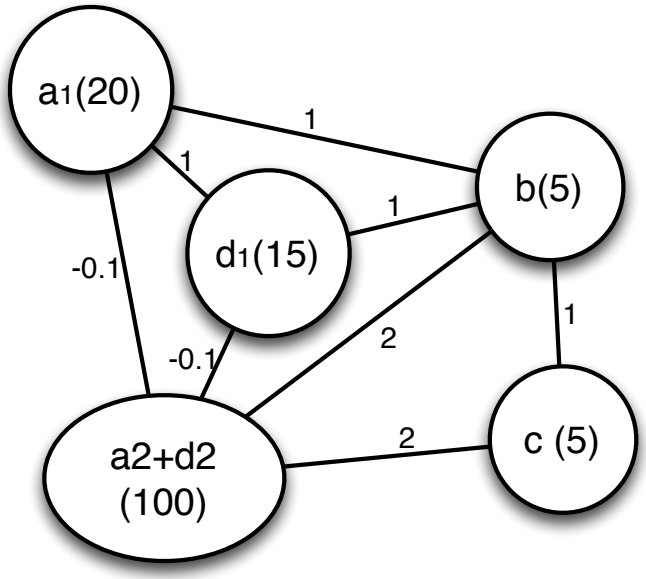


- Smaller merged graph
- 3-Way Max-Cut
- FM Heuristic v.s. Search

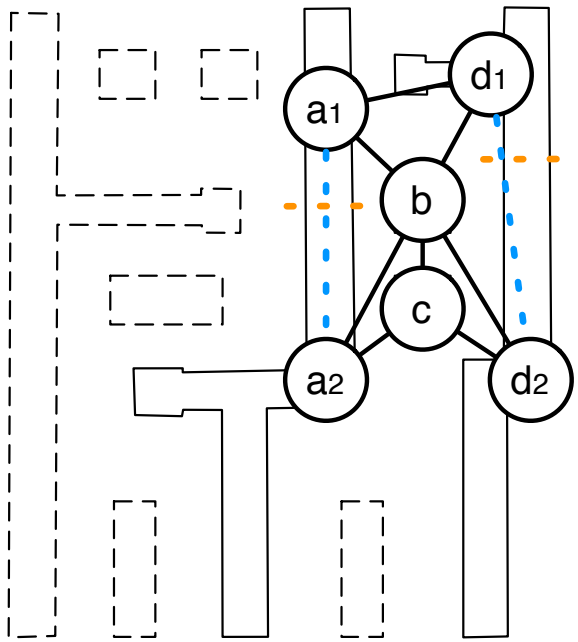


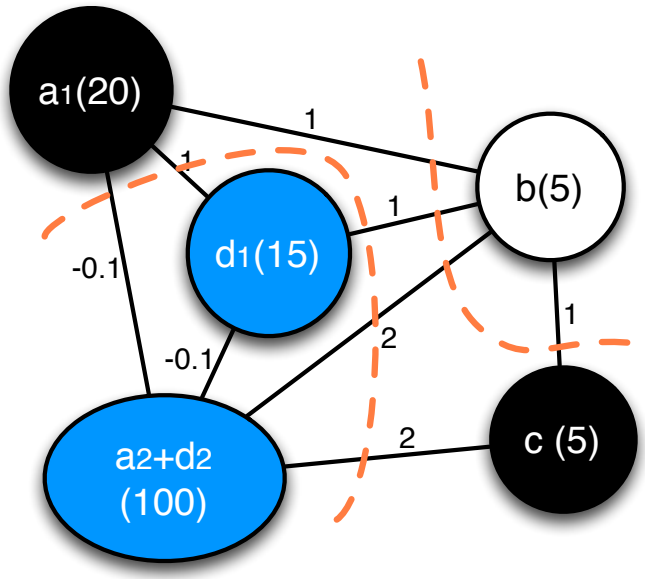
Mapping: Extend to Density Balance

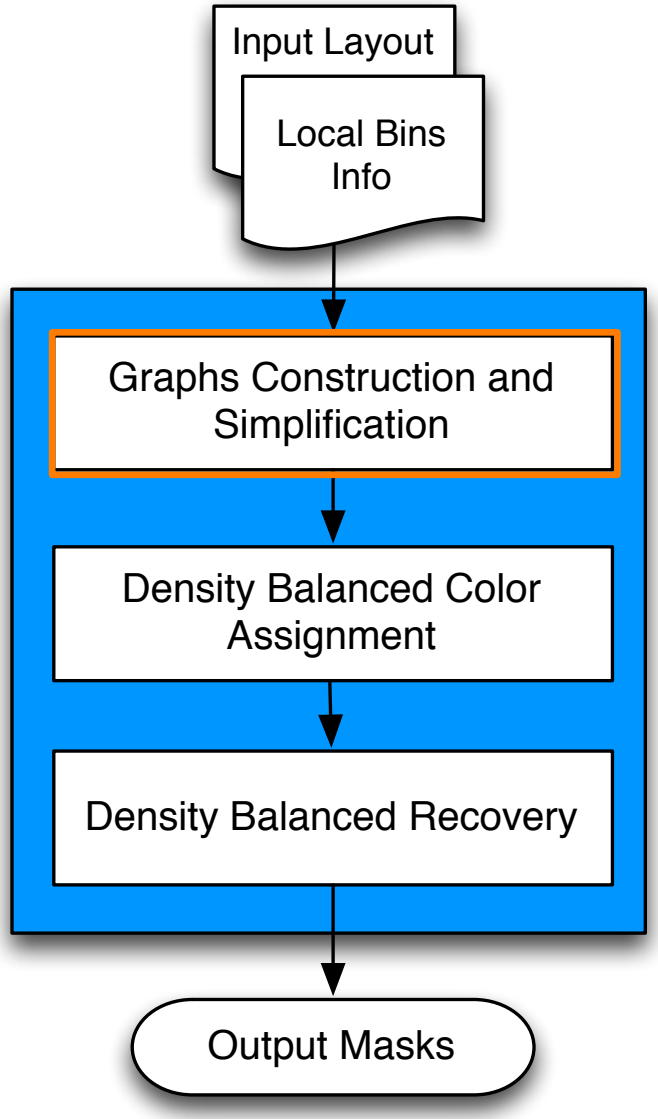




Mapping: Extend to Density Balance



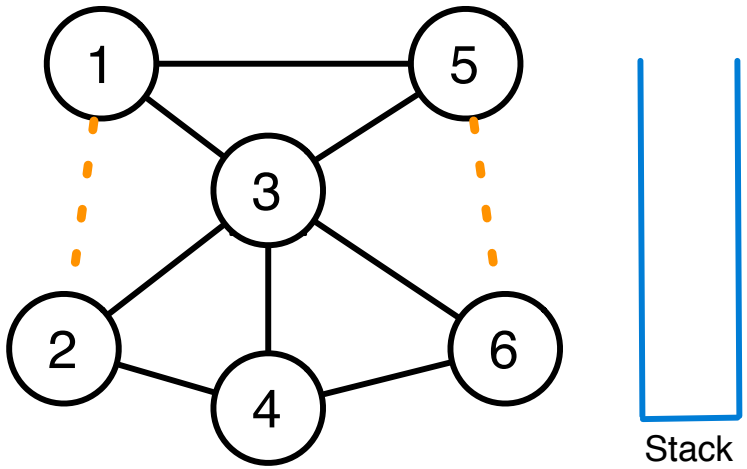




- Implement techniques
  [Yu+,ICCAD'11]
  [Fang+,DAC'12]
  [Kuang+,DAC'13]
- 3 new techniques
- Integrate density balance

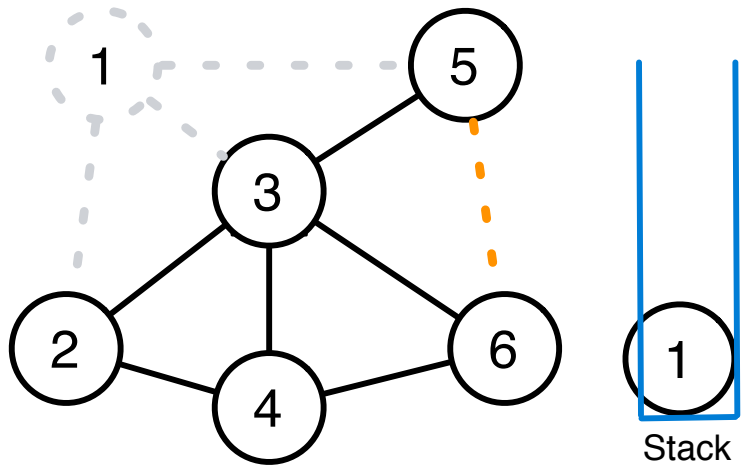
Fast Color Assignment Trial

- Iteratively remove vertex with:
  - Conflict degree < 3
  - Stitch degree < 2
- Linear runtime
- Keep conflict # optimality



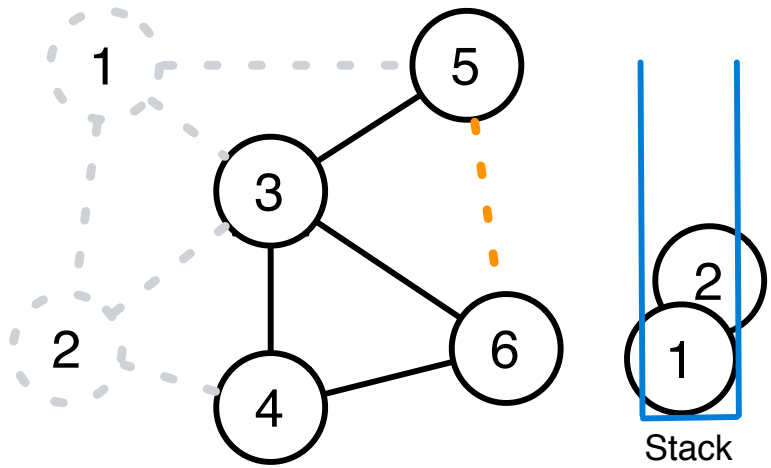
Fast Color Assignment Trial

- Iteratively remove vertex with:
  - Conflict degree < 3
  - Stitch degree < 2
- Linear runtime
- Keep conflict # optimality



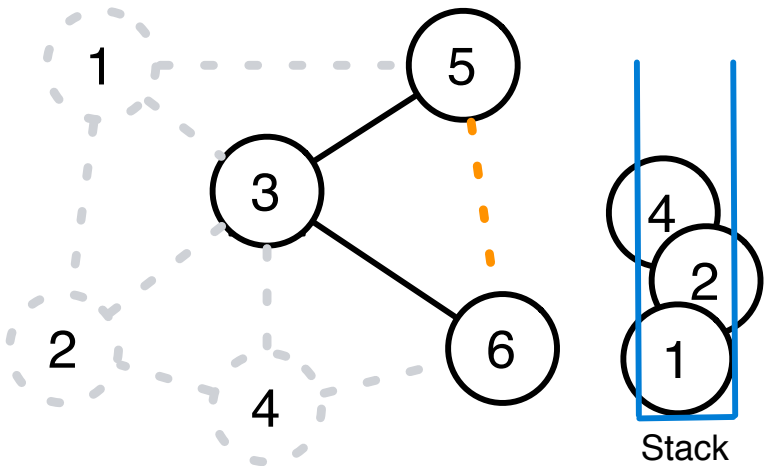
Fast Color Assignment Trial

- Iteratively remove vertex with:
  - Conflict degree < 3
  - Stitch degree < 2
- Linear runtime
- Keep conflict # optimality



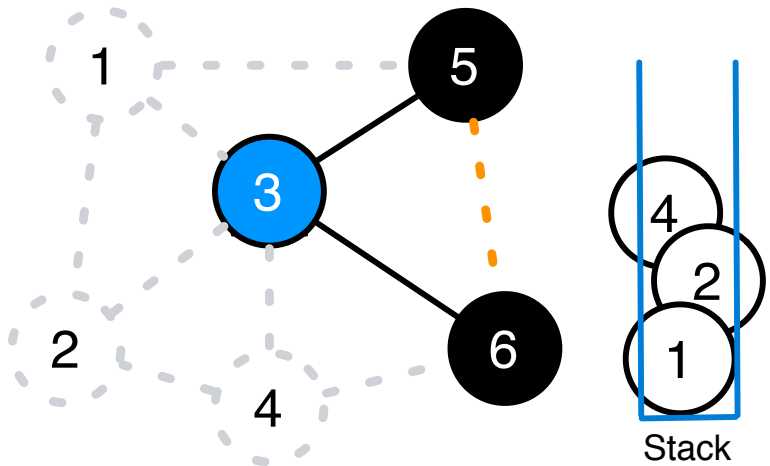
Fast Color Assignment Trial

- Iteratively remove vertex with:
  - Conflict degree < 3
  - Stitch degree < 2
- Linear runtime
- Keep conflict # optimality



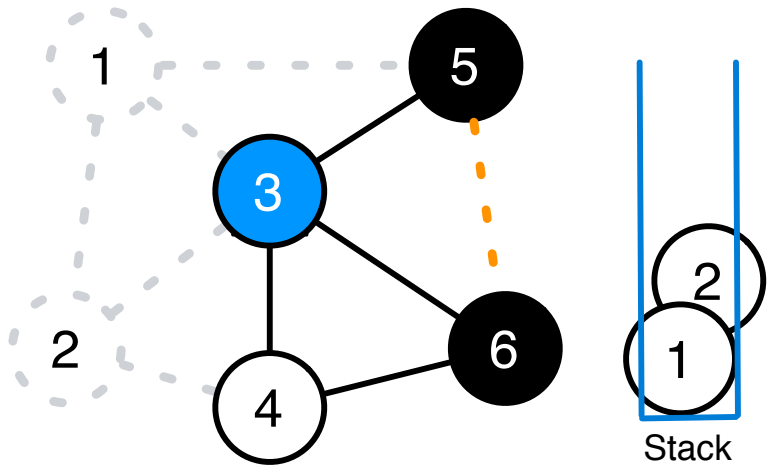
Fast Color Assignment Trial

- Iteratively remove vertex with:
  - Conflict degree < 3
  - Stitch degree < 2
- Linear runtime
- Keep conflict # optimality



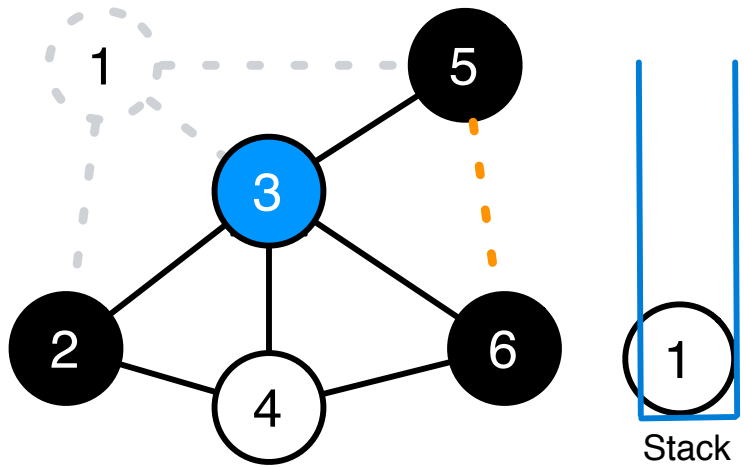
Fast Color Assignment Trial

- Iteratively remove vertex with:
  - Conflict degree < 3
  - Stitch degree < 2
- Linear runtime
- Keep conflict # optimality



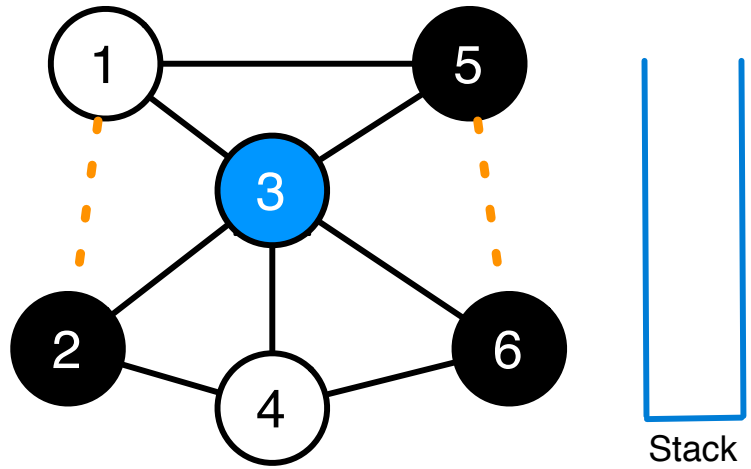
Fast Color Assignment Trial

- Iteratively remove vertex with:
  - Conflict degree < 3
  - Stitch degree < 2
- Linear runtime
- Keep conflict # optimality



Fast Color Assignment Trial

- Iteratively remove vertex with:
  - Conflict degree < 3
  - Stitch degree < 2
- Linear runtime
- Keep conflict # optimality



Cut Vertex Stitch Forbiddance

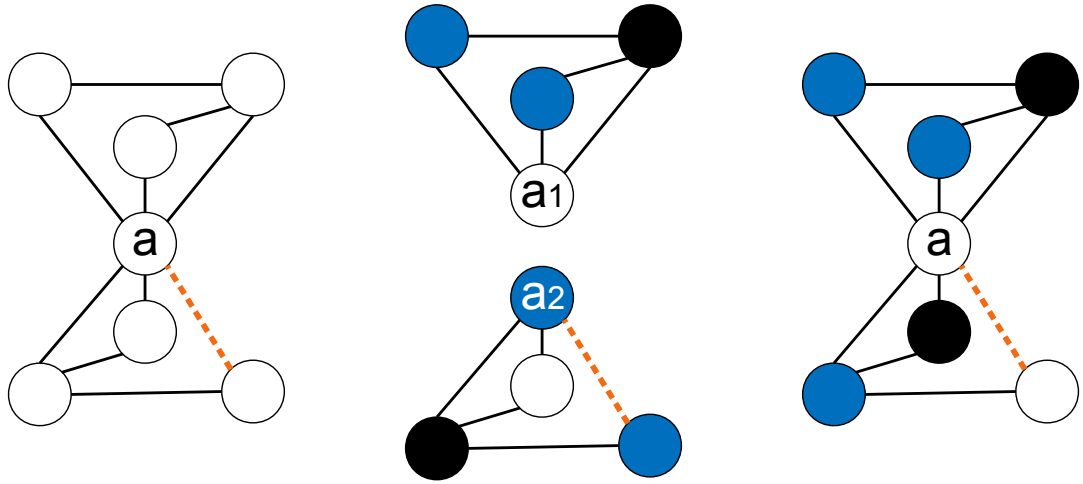


Figure : Cut Computation [Fang+,DAC'12]

Cut Vertex Stitch Forbiddance

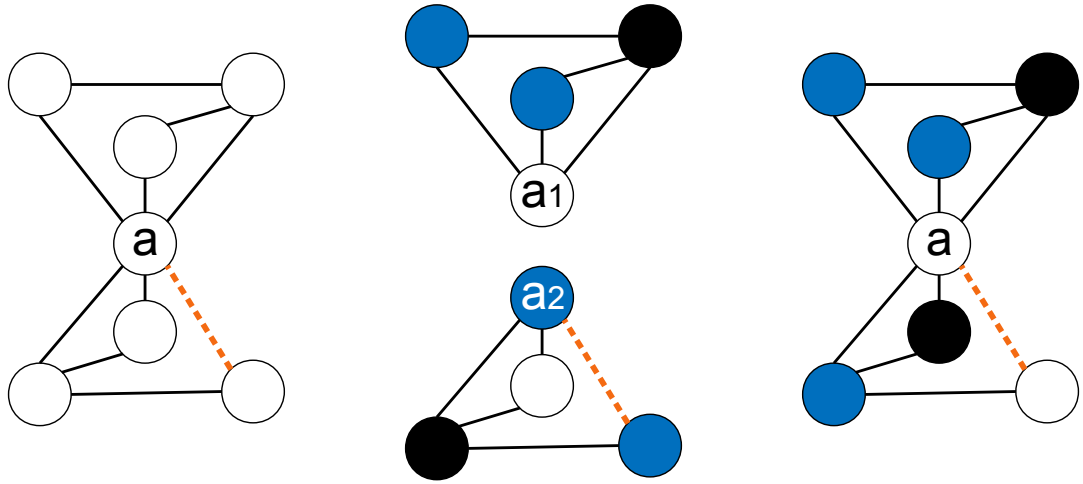
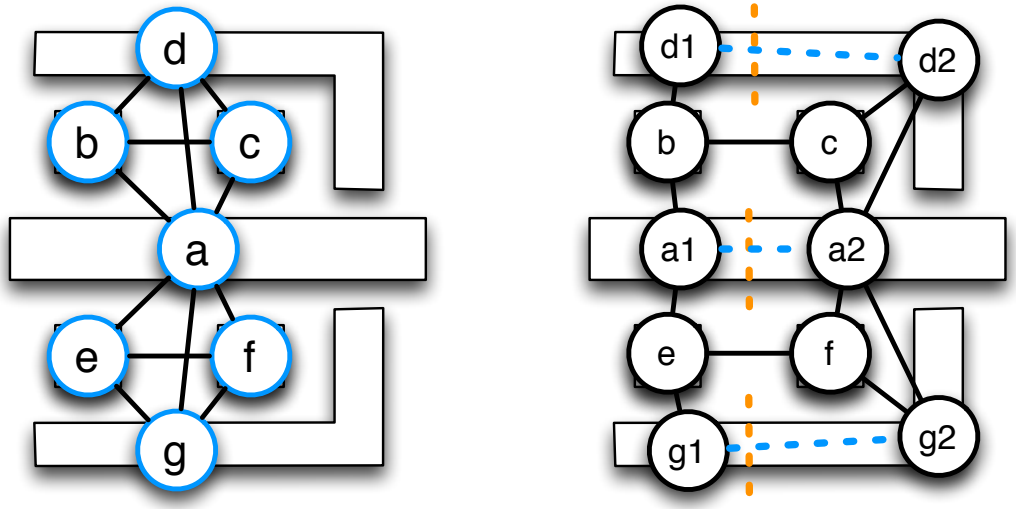


Figure : Cut Computation [Fang+,DAC'12]



Cut Vertex Stitch Forbiddance

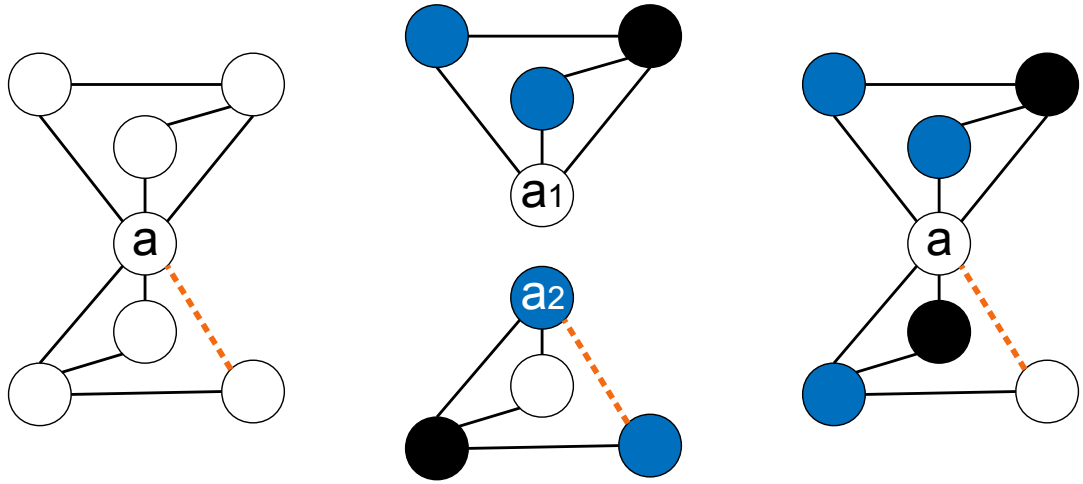
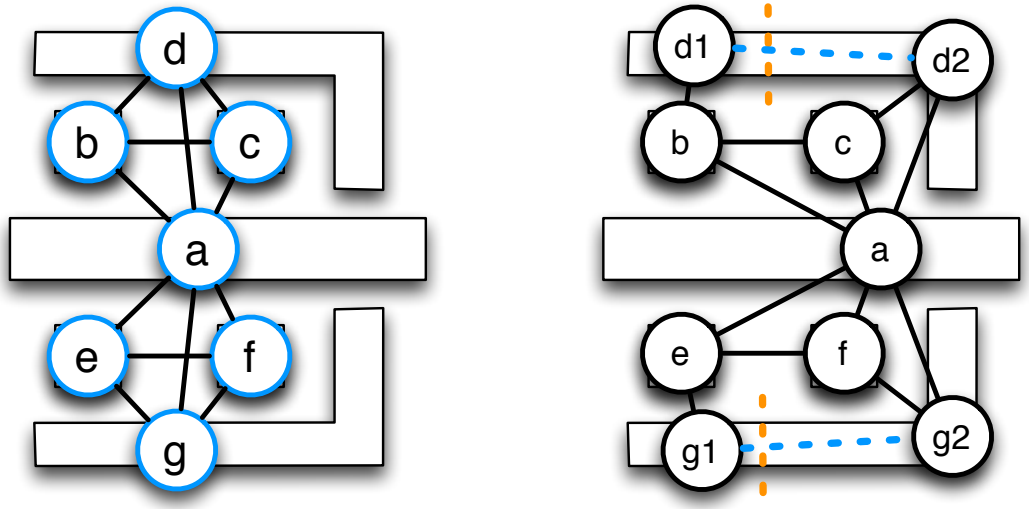


Figure : Cut Computation [Fang+,DAC'12]



Cut Vertex Stitch Forbiddance

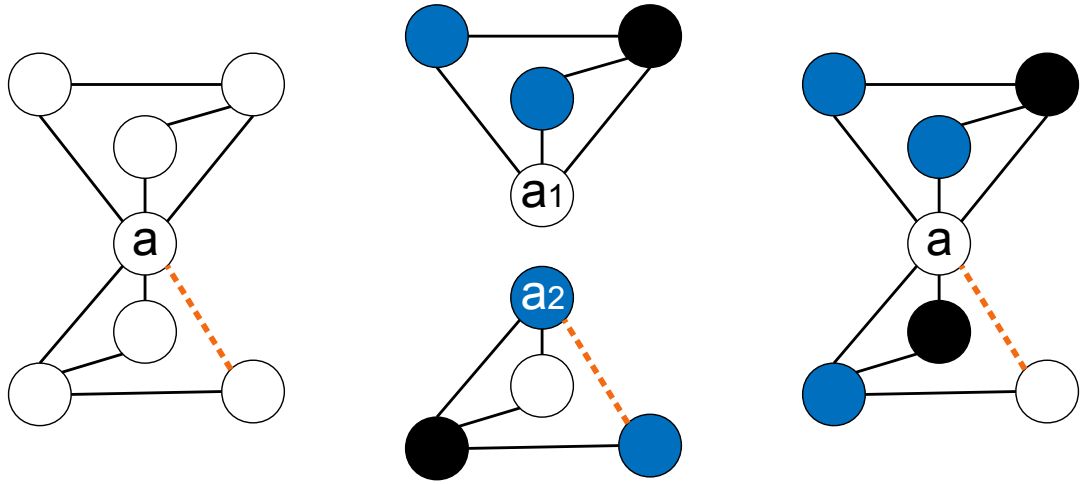
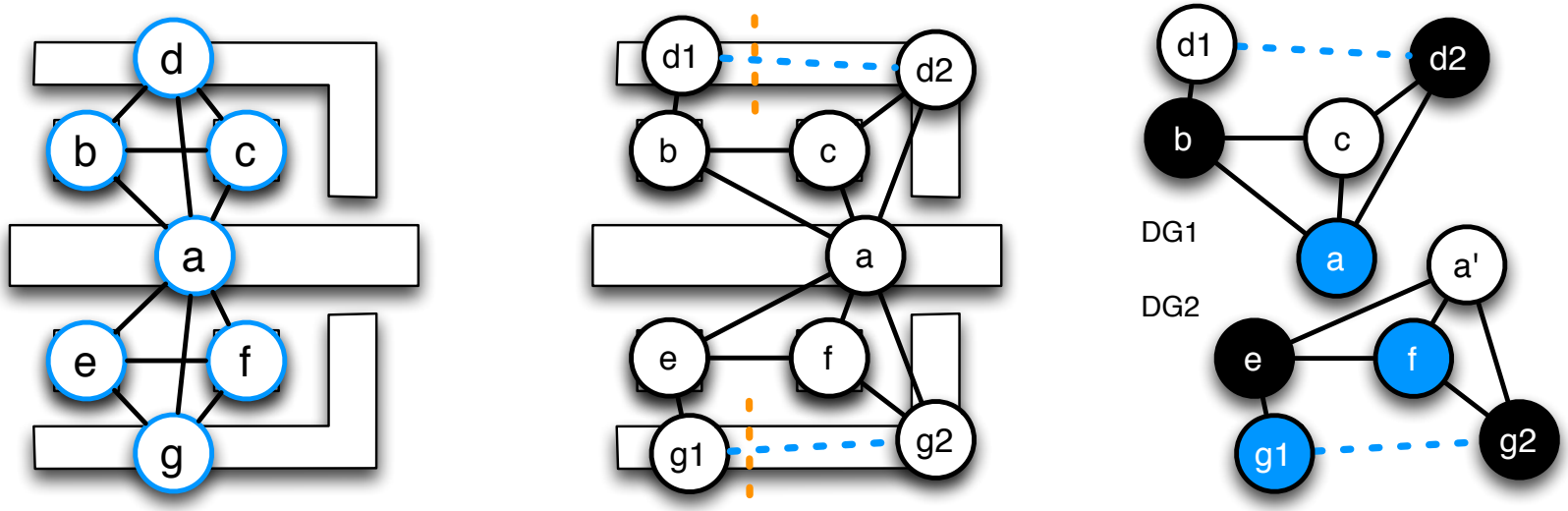
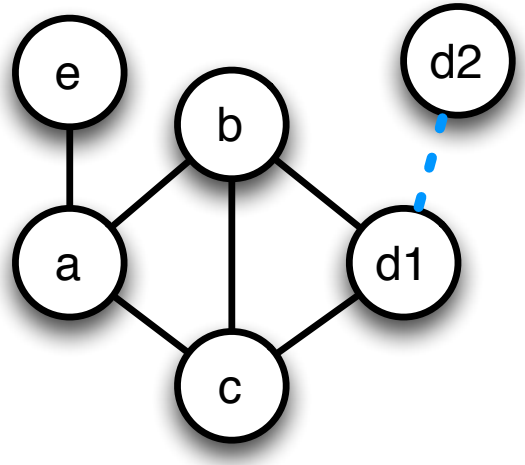


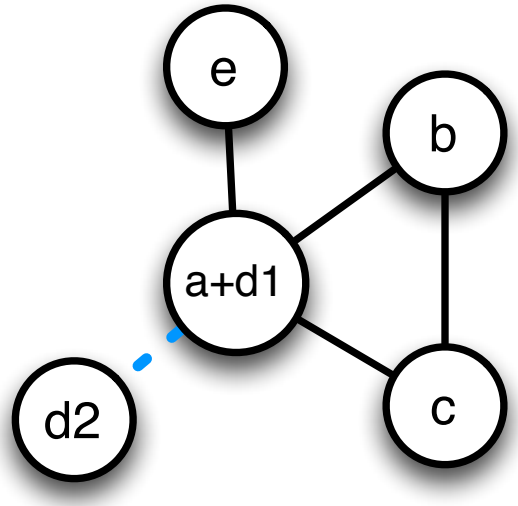
Figure : Cut Computation [Fang+,DAC'12]



Vertex Clustering



(a)

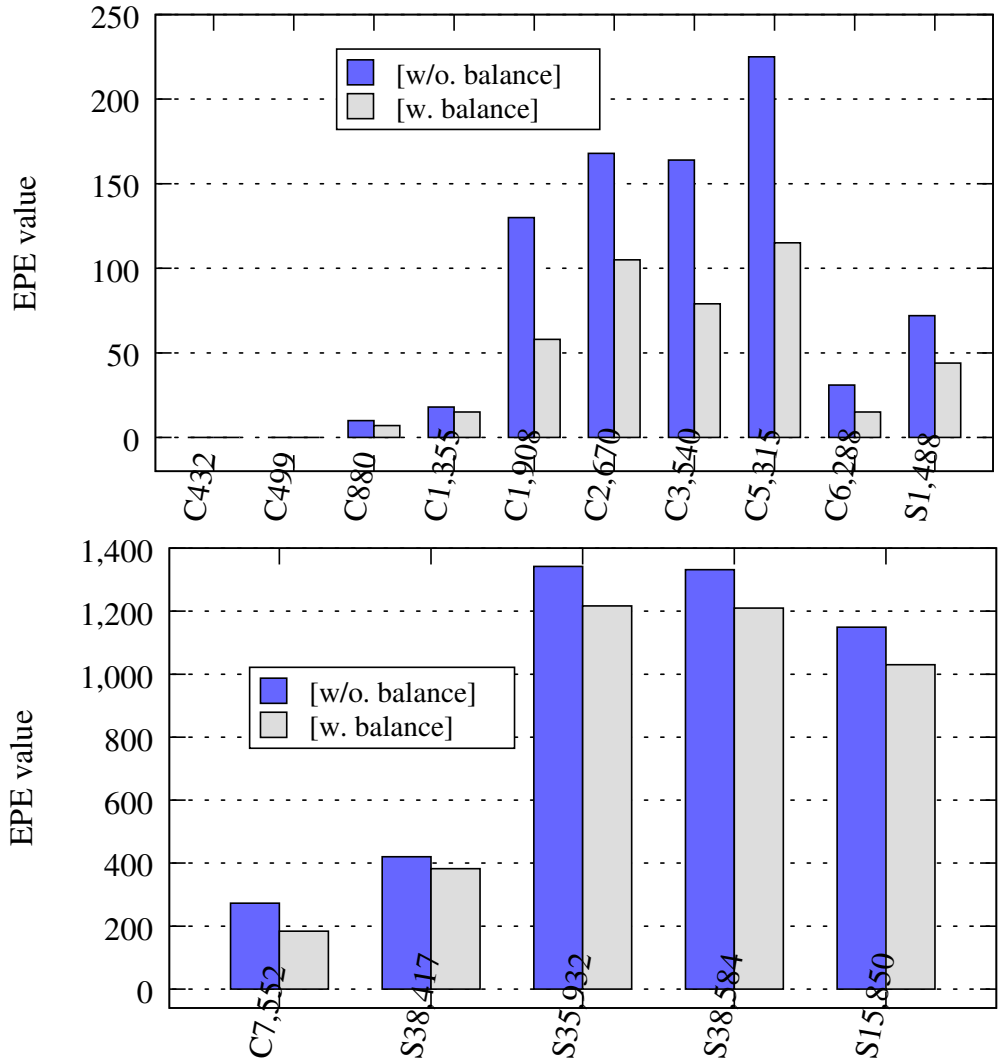


(b)

Experimental Results– Setup

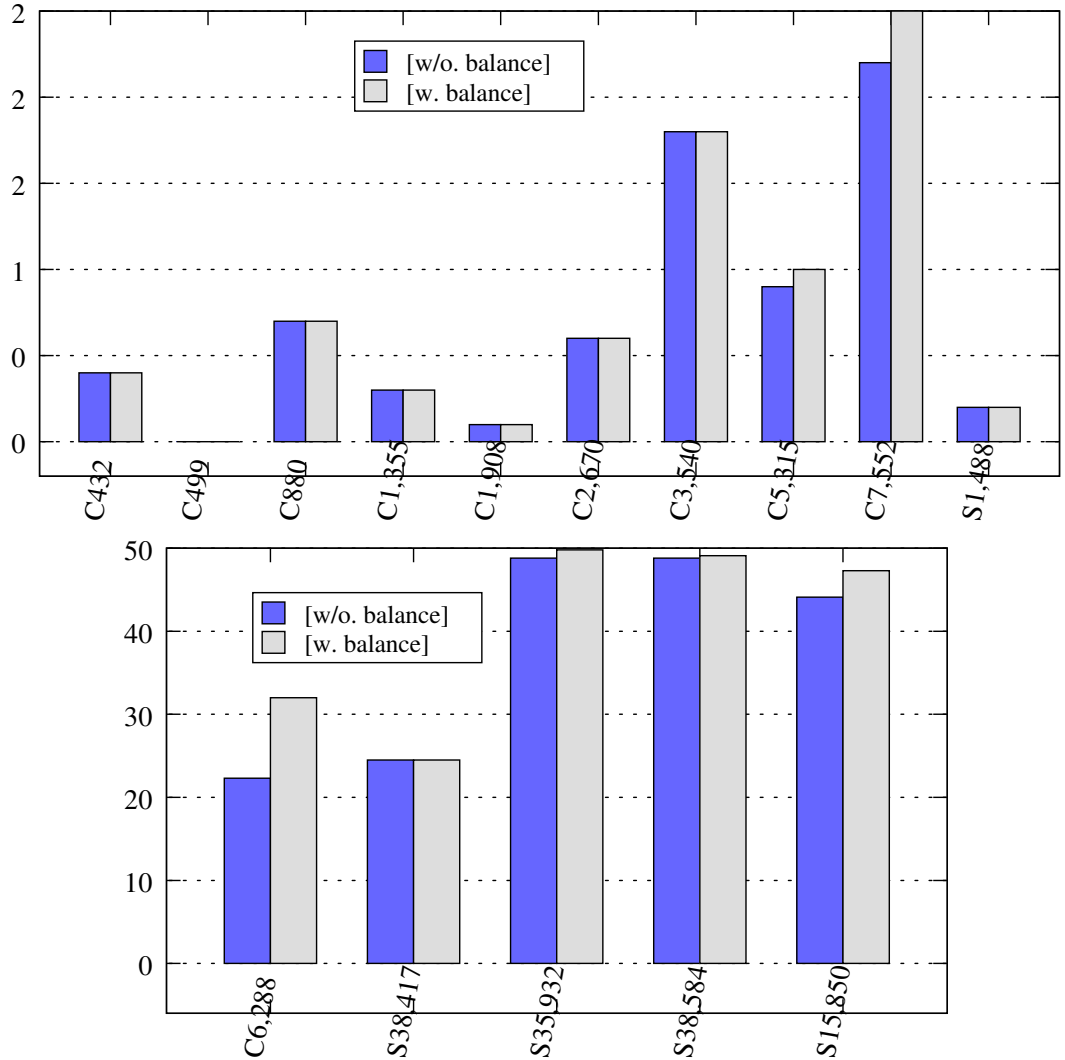
- Using C++ on 3.0GHz Linux machine
- **CSDP** as SDP solver
- Benchmarks
  - ISCAS benchmarks from [Yu+,ICCAD'11]
  - Two OpenSPARC T1 benchmarks mul_top and exu_ecc
  - Six dense benchmarks c9_total – s5_total

With or Without Density Balance?



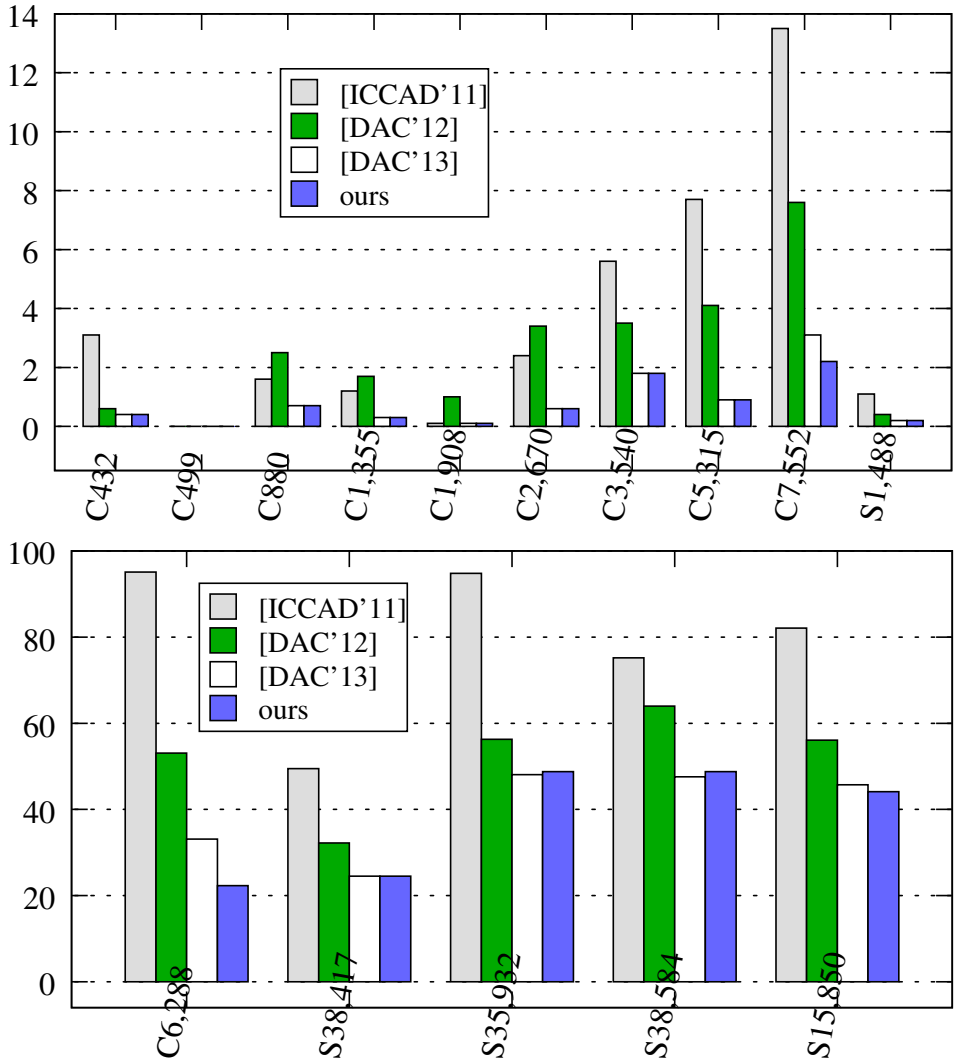
- Considering balance, -14% EPE#
- EPE: Edge placement error

With or Without Density Balance? (cont.)



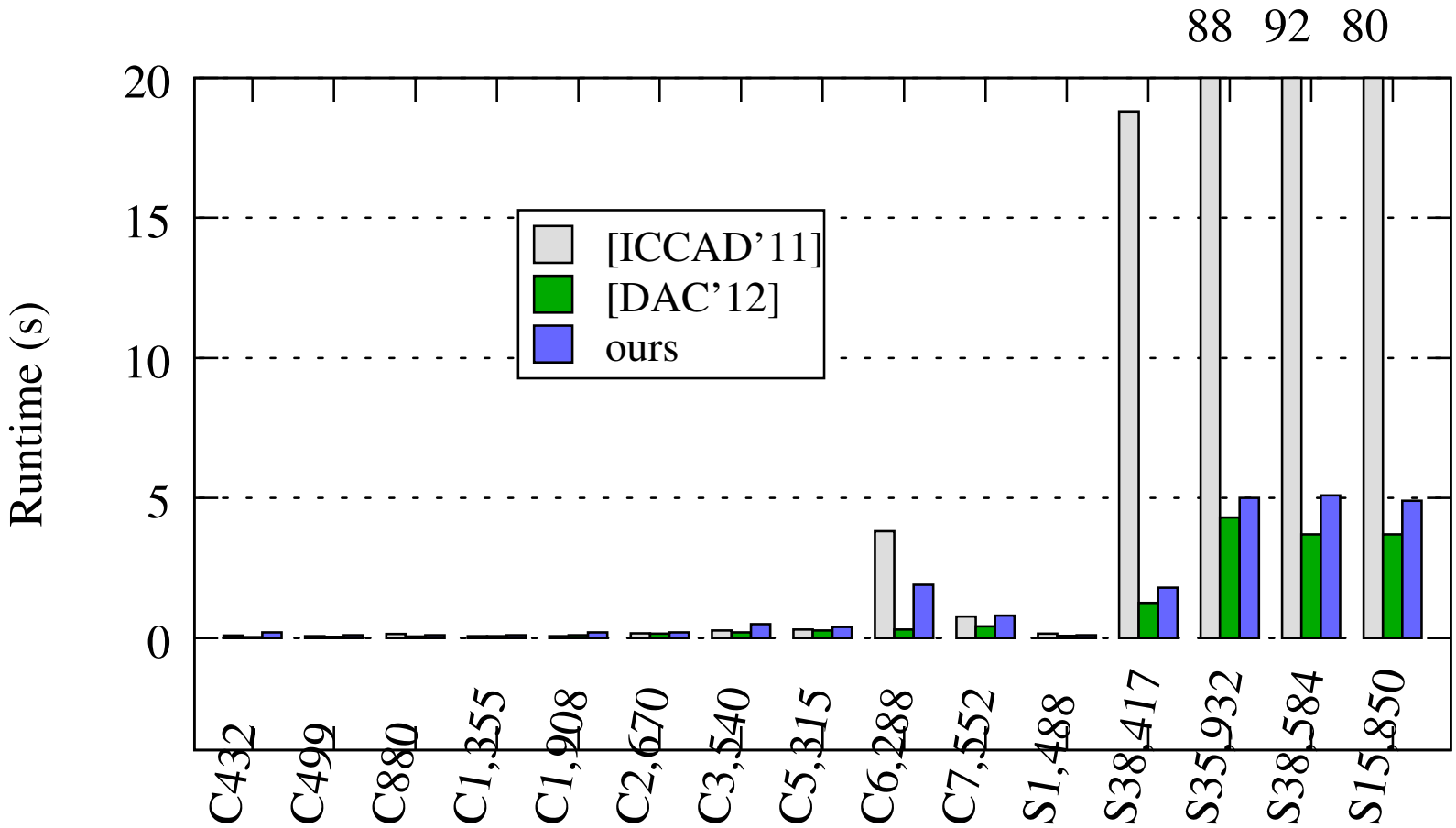
- Considering balance, +4% cost penalty
- Cost = conflict# + 0.1 stitch#

V.S. Other Decomposers ([Yu+,ICCAD'11] cases)



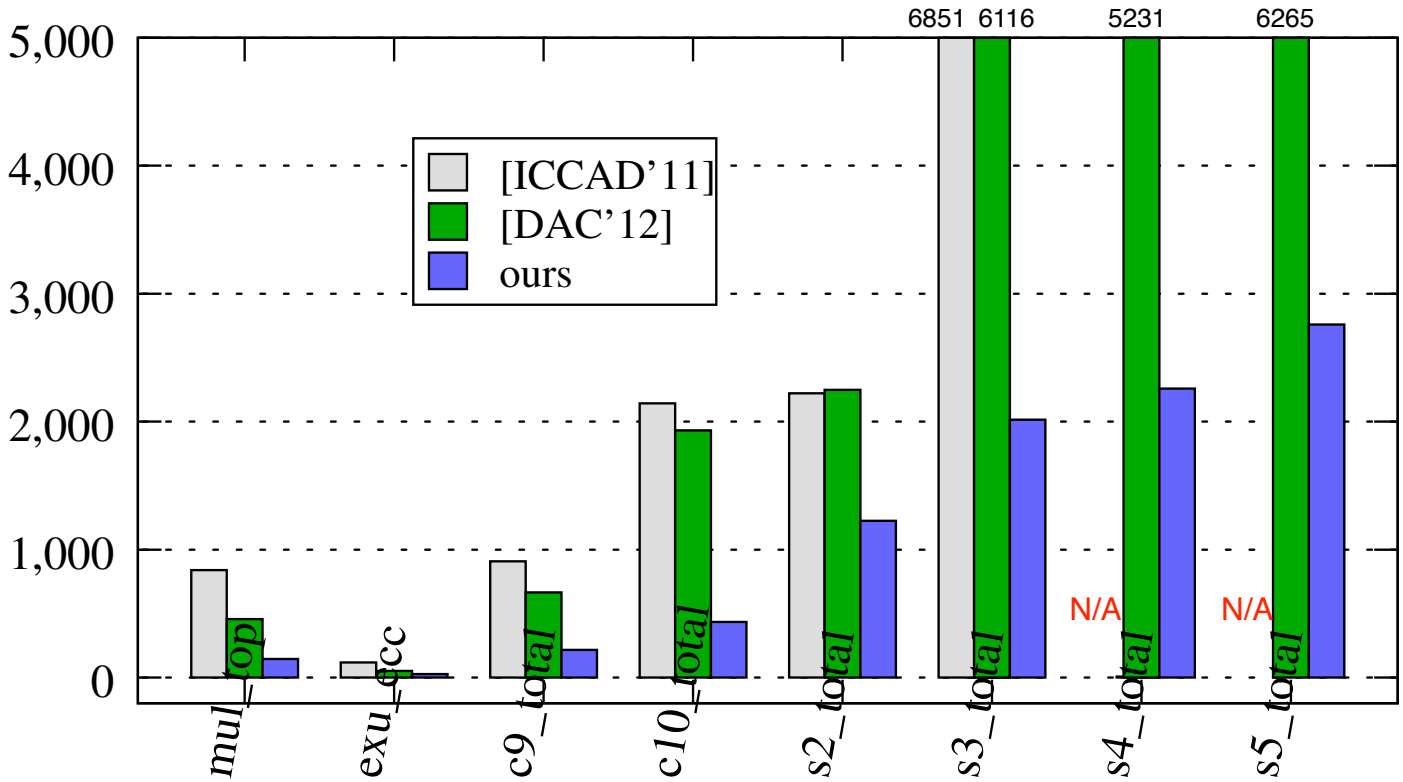
- Cost: -55% (v.s. ICCAD'11), -25% (v.s. DAC'12), -5% (v.s. DAC'13)

V.S. Other Decomposers ([Yu+,ICCAD'11] cases)



V.S. Other Decomposers (other cases)

Cost comparison

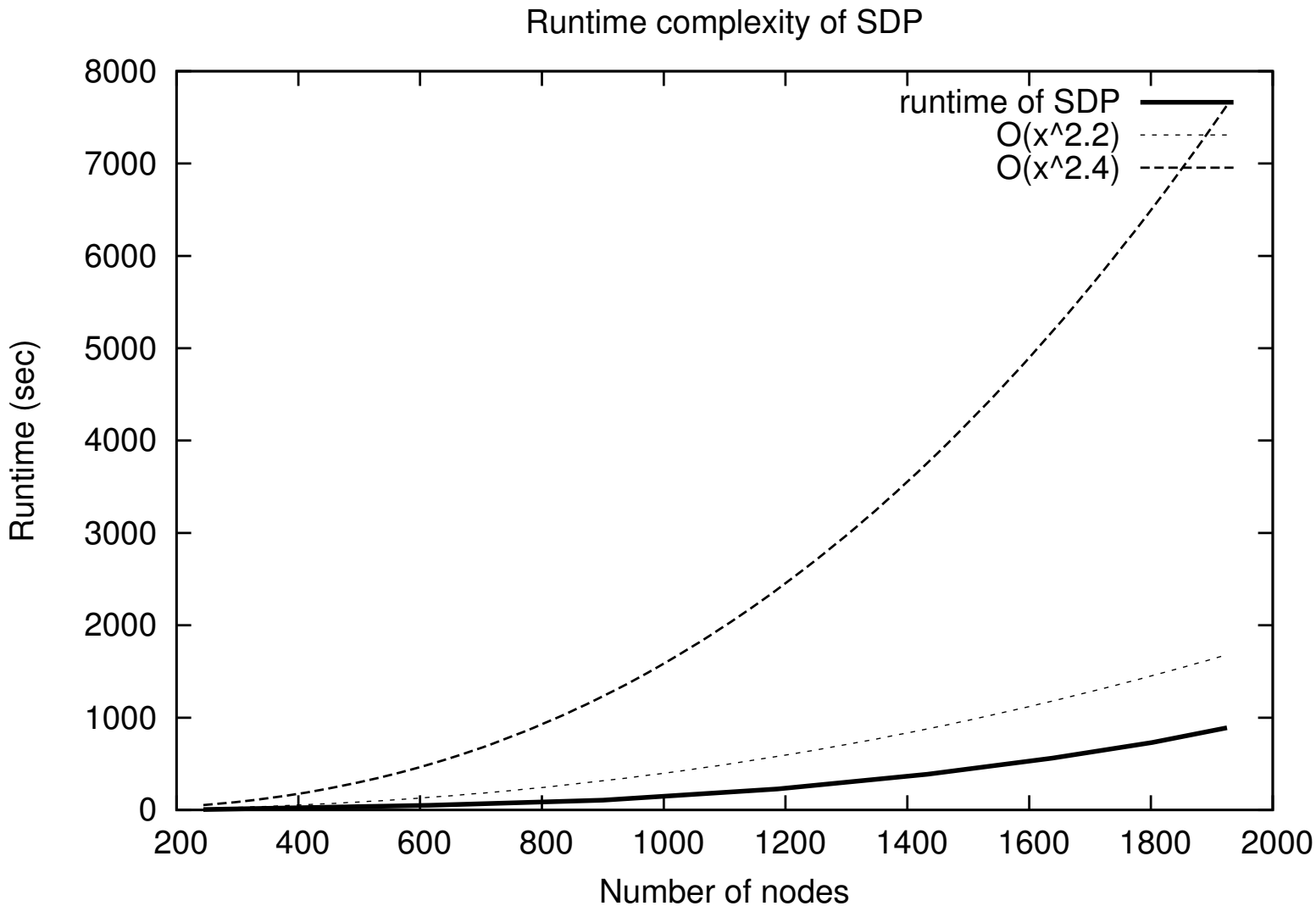


- -60% (v.s. ICCAD'11), -60% (v.s. DAC'12)

Runtime comparison

- ICCAD'11 : DAC'12 : ours = 3600s : 6s : 134s

Scalability of SDP



Conclusions

- Integrate density balance
  - SDP formulation
  - Mapping
  - Graph simplification
- High performance
  - Mapping
  - Graph model
- Faster
  - A set of graph simplification

Thank You !

Acknowledgement

Supported by IBM Scholarship, NSF, CCF, SRC, and NSFC