Triple/Quadruple Patterning Layout Decomposition via Novel Linear Programming and Iterative Rounding

Yibo Lin[1], Xiaoqing Xu[1], Bei Yu[2], Ross Baldick[1], David Z. Pan[1]
[1]ECE Department, University of Texas at Austin
[2]CSE Department, Chinese University of Hong Kong

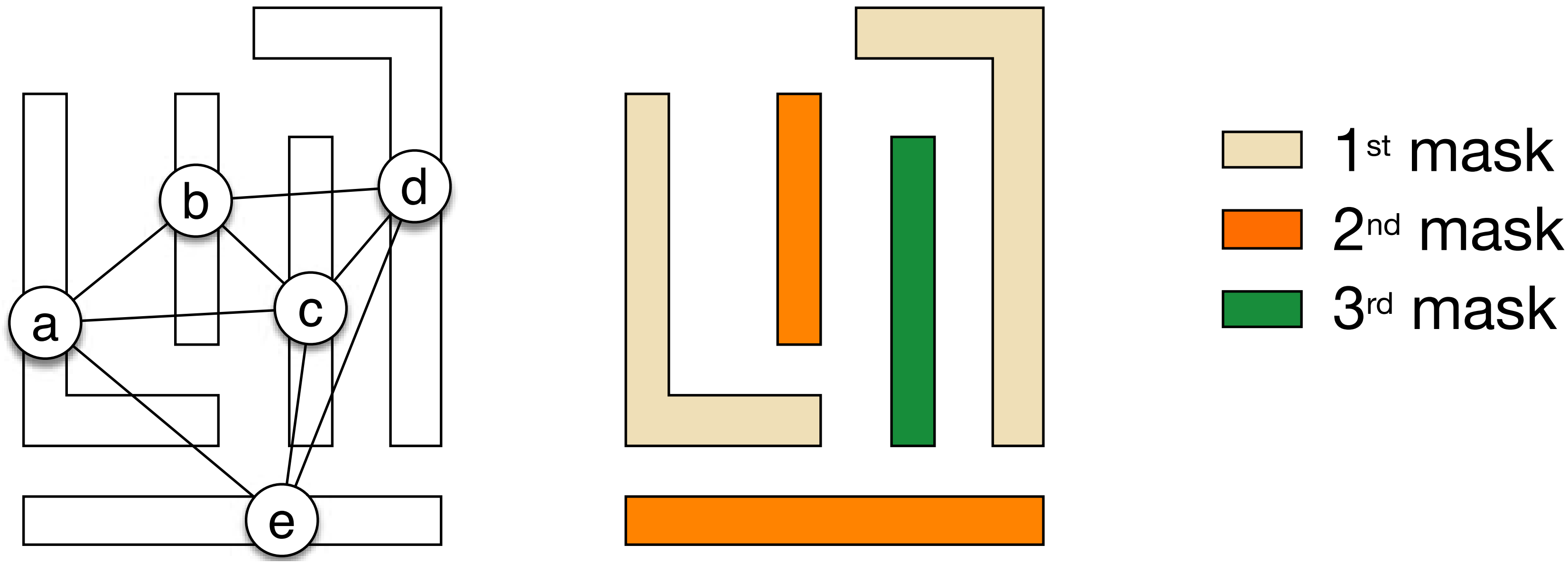
This work is supported in part by NSF and SRC

Outline

- Introduction
- A New Framework for Layout Decomposition
  - ILP → LP relaxation with iterative rounding
- Experimental Results
- Conclusion

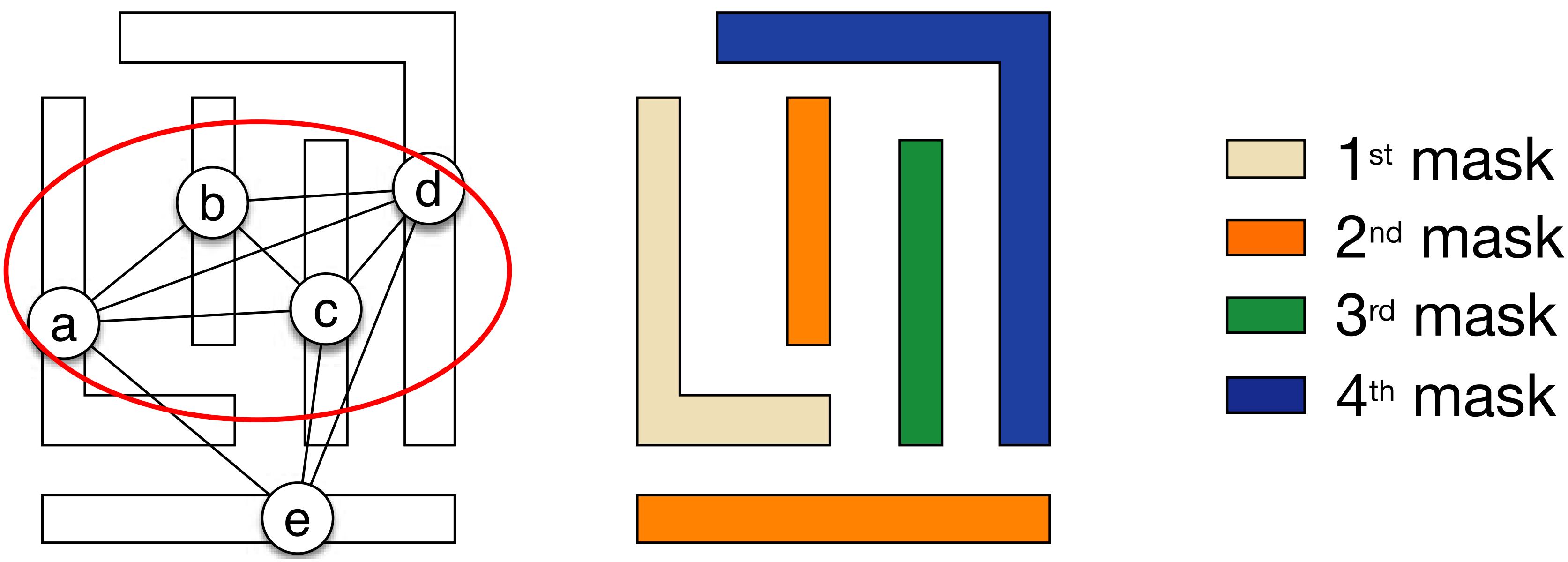
Triple Patterning Lithography (TPL)

- An example of TPL conflict graph and decomposition
- Layout decomposition is a fundamental problem for multiple patterning



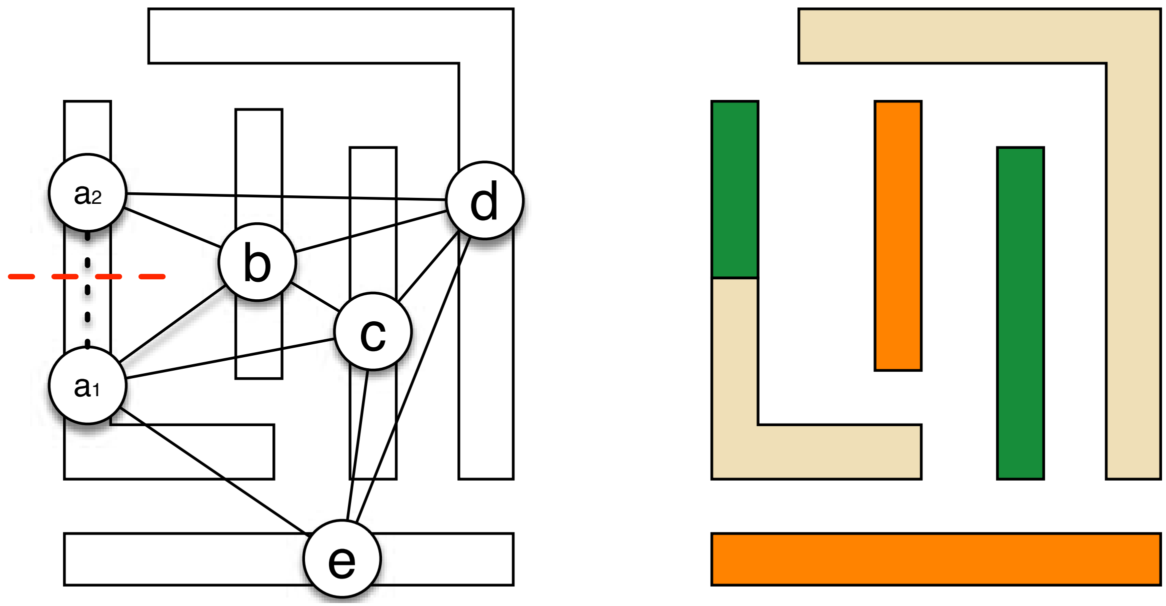
Quadruple Patterning Lithography (QPL)

- An example of QPL layout decomposition (coloring) and conflict graph



Stitch Insertion

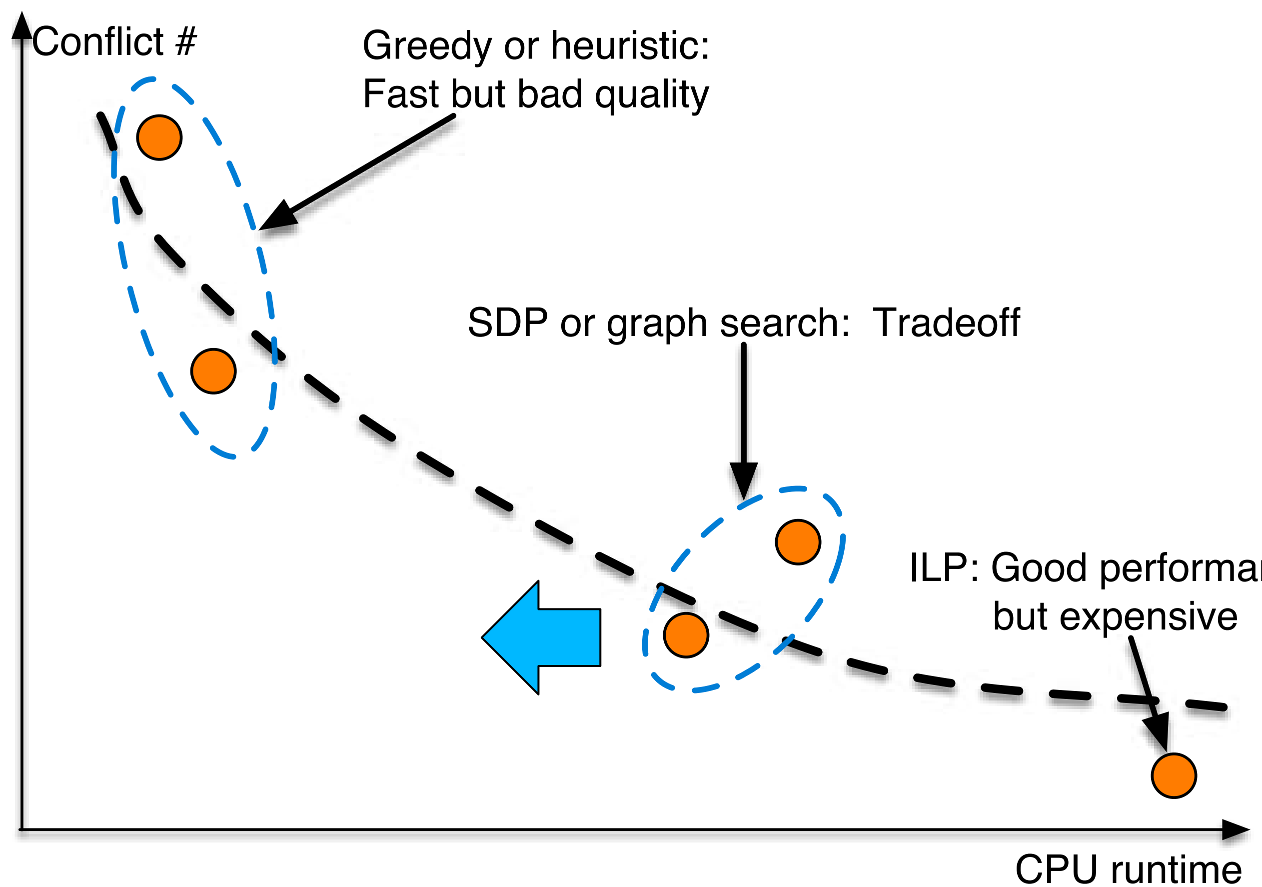
- Stitch may be inserted to resolve conflict



- However, strongly discouraged due to misalignment and yield loss
- In this work, we do not allow stitch insertion

Current State of MPL Decomposition

- **ILP or SAT:** [Cork+, SPIE'08], [Yu+, ICCAD'11], [Cork+, SPIE'13]
- **Greedy or heuristic:** [Ghaida+, SPIE'11], [Fang+, DAC'12], [Kuang+, DAC'13], [Fang+, SPIE'14]
- **SDP or graph search:** [Yu+, ICCAD'11], [Chen+, ISQED'13], [Yu+, ICCAD'13], [Yu+, DAC'14]



[Yu+, SPIE'14]

Major Contributions of This Work

- A new layout decomposition framework for TPL/QPL
- ILP → novel linear programming (LP) based algorithm with iterative rounding scheme
- An odd-cycle based technique to enhance LP solution quality (which can be better mapped to ILP solution)
- Our experiments obtain comparable quality cf. previous state-of-the-art, but are 26x to 600x faster than ILP, and 1.8x to 2.6x faster than SDP

Problem Formulation

- Input
  - Uncolored layout patterns
  - Minimum coloring distance d_c
  - Number of colors available (TPL or QPL)
- Output
  - Decomposed layout with color assignment for each pattern
  - TPL/QPL friendliness
  - Stitch insertion is not allowed

Initial ILP Formulation

- Represent color with two binary variables

$$(x_{i1}, x_{i2}) \rightarrow \text{color}$$

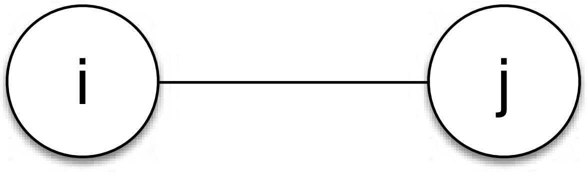
$$(0,0) \rightarrow 0$$

$$(0,1) \rightarrow 1$$

$$(1,0) \rightarrow 2$$

$$(1,1) \rightarrow 3$$

(x_{i1}, x_{i2}) (x_{j1}, x_{j2})



$$x_{i1}, x_{i2}, x_{j1}, x_{j2} \in \{0,1\}$$

(0, 0) (0, 0)

$$x_{i1} + x_{i2} + x_{j1} + x_{j2} \geq 1$$

(0, 1) (0, 1)

$$x_{i1} + (1 - x_{i2}) + x_{j1} + (1 - x_{j2}) \geq 1$$

...

Additional constraint for TPL

$$x_{i1} + x_{i2} \leq 1$$

ILP Formulation

- The goal is to meet all the constraints

Only for TPL

$$
\begin{aligned}
\min \quad & \mathit{Objective} && && (1a) \\
\text{s.t} \quad & \boxed{x_{i1} + x_{i2} \leq 1,} && && (1b) \\
& x_{i1} + x_{i2} + x_{j1} + x_{j2} \geq 1, && \forall e_{ij} \in E_c, && (1c) \\
& x_{i1} + \bar{x}_{i2} + x_{j1} + \bar{x}_{j2} \geq 1, && \forall e_{ij} \in E_c, && (1d) \\
& \bar{x}_{i1} + x_{i2} + \bar{x}_{j1} + x_{j2} \geq 1, && \forall e_{ij} \in E_c, && (1e) \\
& \bar{x}_{i1} + \bar{x}_{i2} + \bar{x}_{j1} + \bar{x}_{j2} \geq 1, && \forall e_{ij} \in E_c, && (1f) \\
& \bar{x}_{i1} = 1 - x_{i1}, \quad \forall i \in V, && && (1g) \\
& \bar{x}_{i2} = 1 - x_{i2}, \quad \forall i \in V, && && (1h) \\
& x_{i1}, x_{i2} \in \{0, 1\}, \quad \forall i \in V. && && (1i)
\end{aligned}
$$

LP Relaxation

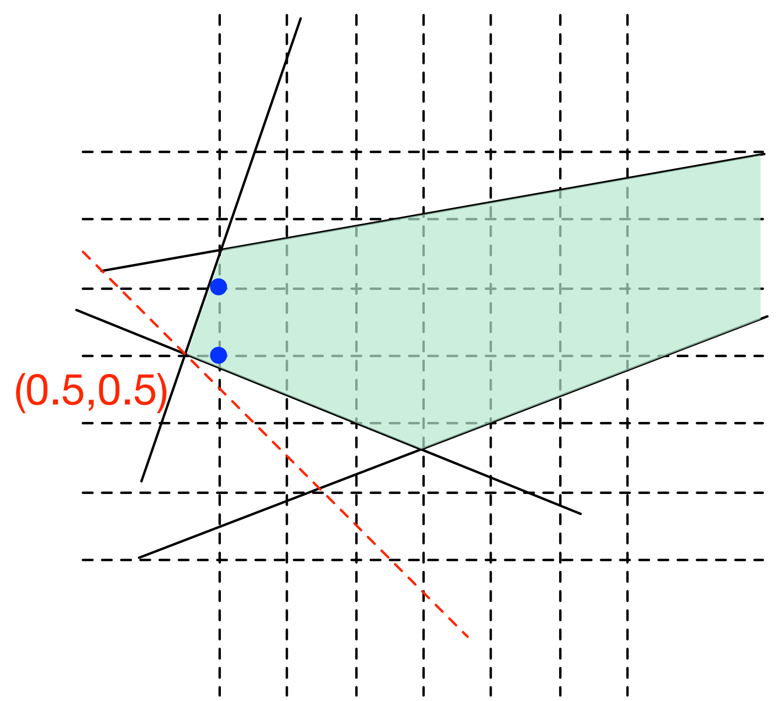
- Relax integer to continuous variables

$$
\begin{aligned}
\min \quad & Objective && (1a)\\
\text{s.t.} \quad & x_{i1} + x_{i2} \leq 1, && (1b)\\
& x_{i1} + x_{i2} + x_{j1} + x_{j2} \geq 1, \quad \forall e_{ij} \in E_c, && (1c)\\
& x_{i1} + \bar{x}_{i2} + x_{j1} + \bar{x}_{j2} \geq 1, \quad \forall e_{ij} \in E_c, && (1d)\\
& \bar{x}_{i1} + x_{i2} + \bar{x}_{j1} + x_{j2} \geq 1, \quad \forall e_{ij} \in E_c, && (1e)\\
& \bar{x}_{i1} + \bar{x}_{i2} + \bar{x}_{j1} + \bar{x}_{j2} \geq 1, \quad \forall e_{ij} \in E_c, && (1f)\\
& \bar{x}_{i1} = 1 - x_{i1}, \quad \forall i \in V, && (1g)\\
& \bar{x}_{i2} = 1 - x_{i2}, \quad \forall i \in V, && (1h)\\
& \cancel{x_{i1}, x_{i2} \in \{0,1\}, \quad \forall i \in V.} && \cancel{(1i)}\\
& 0 \leq x_{i1}, x_{i2} \leq 1, \quad \forall i \in V
\end{aligned}
$$

LPIR

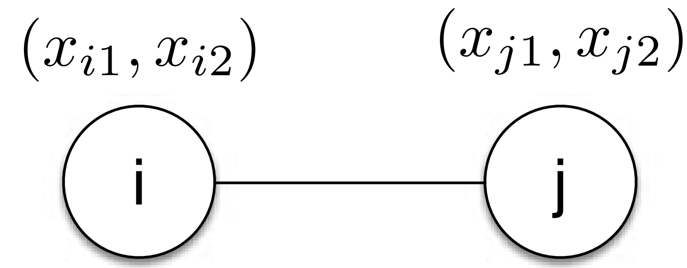
- Linear programming and iterative rounding (LPIR)
- Non-integer solutions
  - Fewer non-integers mean closer to optimal solutions of ILP
  - Prune non-integer solutions in the feasible set

Reduce non-integers



Simple Observation

- Suppose $x_{i1} = x_{j1} = 0$



$$x_{i1} + x_{i2} + x_{j1} + x_{j2} \geq 1, \quad \forall e_{ij} \in E_c, \tag{1c}$$

$$x_{i1} + \bar{x}_{i2} + x_{j1} + \bar{x}_{j2} \geq 1, \quad \forall e_{ij} \in E_c, \tag{1d}$$

$$\bar{x}_{i1} + x_{i2} + \bar{x}_{j1} + x_{j2} \geq 1, \quad \forall e_{ij} \in E_c, \tag{1e}$$

$$\bar{x}_{i1} + \bar{x}_{i2} + \bar{x}_{j1} + \bar{x}_{j2} \geq 1, \quad \forall e_{ij} \in E_c, \tag{1f}$$

$$\bar{x}_{i1} = 1 - x_{i1}, \quad \forall i \in V, \tag{1g}$$

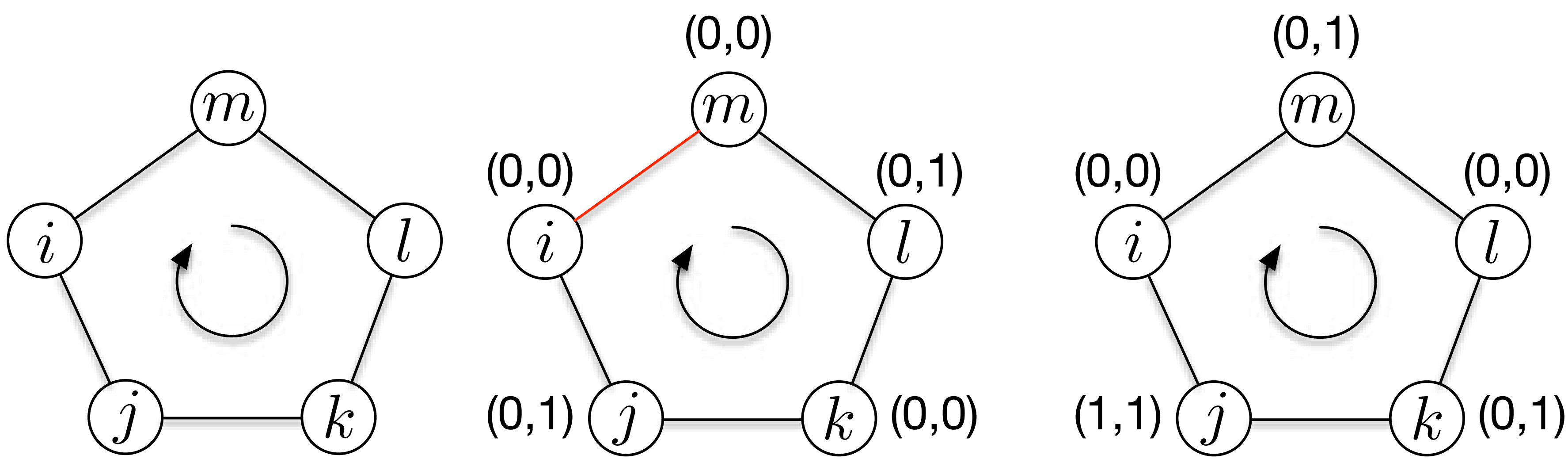
$$\bar{x}_{i2} = 1 - x_{i2}, \quad \forall i \in V \tag{1h}$$

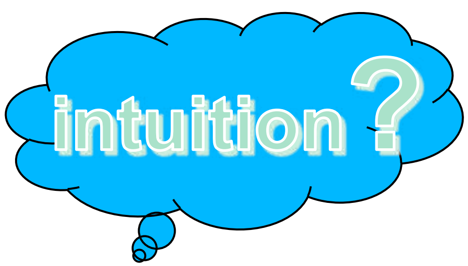
$$x_{i2} + x_{j2} = 1$$

The second bits must be different

Non-integers along Odd Cycles

- Consider the constraints along an odd cycle
- Suppose $x_{i1} = x_{j1} = x_{k1} = x_{l1} = x_{m1} = 0$

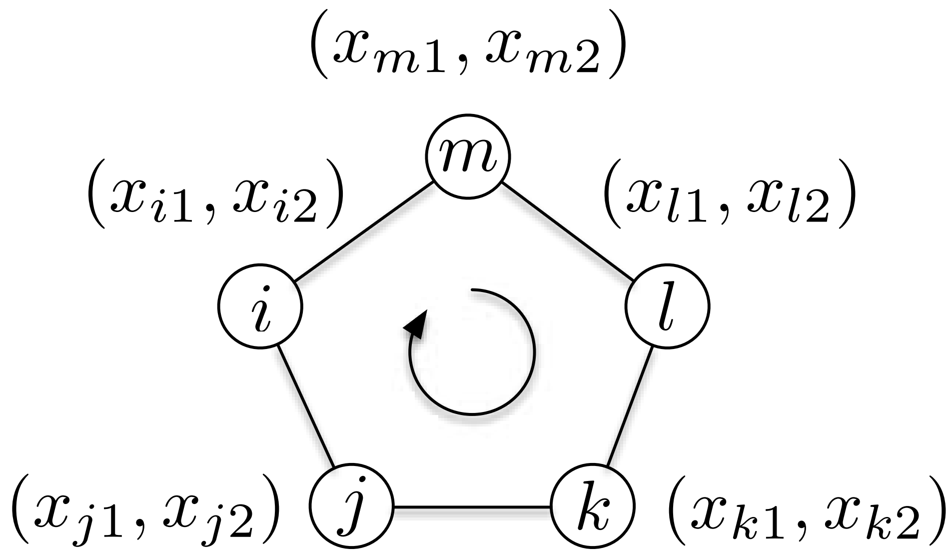




$$\begin{aligned} x_{i2} + x_{j2} &= 1, \\ x_{j2} + x_{k2} &= 1, \\ x_{k2} + x_{l2} &= 1, \\ x_{m2} + x_{i2} &= 1 \end{aligned} \quad \Rightarrow \quad x_{i2} = x_{j2} = x_{k2} = x_{l2} = x_{m2} = 0.5$$

LPIR – Add Odd Cycle Constraints

- Additional constraints
- Prune non-integer solutions from feasible set



s.t.

$$x_{i1} + x_{j1} + x_{k1} + x_{l1} + x_{m1} \geq 1,$$
$$(1 - x_{i1}) + (1 - x_{j1}) + (1 - x_{k1}) + (1 - x_{l1}) + (1 - x_{m1}) \geq 1$$

Help resolve potential non-integers in the second bits

LPIR – Objective Function Biasing

- Push non-integer solutions to integers by dynamically adapting the objective function
- If $x_i = 0.6$, it means x_i tends to be 1
- If $x_i = 0.4$, it means x_i tends to be 0

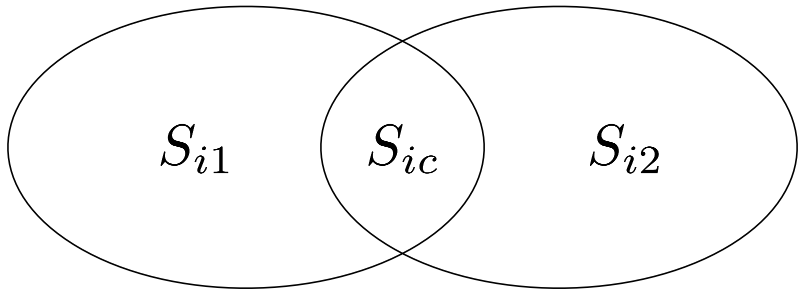
$$\text{If } x_i > 0.5, \ \text{obj} \leftarrow \text{obj} + (1 - x_i).$$
$$\text{If } x_i < 0.5, \ \text{obj} \leftarrow \text{obj} + x_i.$$

Cannot handle (0.5, 0.5)

LPIR – Binding Constraints Analysis

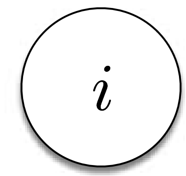
- Try to handle $(x_{i1}, x_{i2}) = (0.5, 0.5)$

Constraint set for x_{i1}



Constraint set for x_{i2}

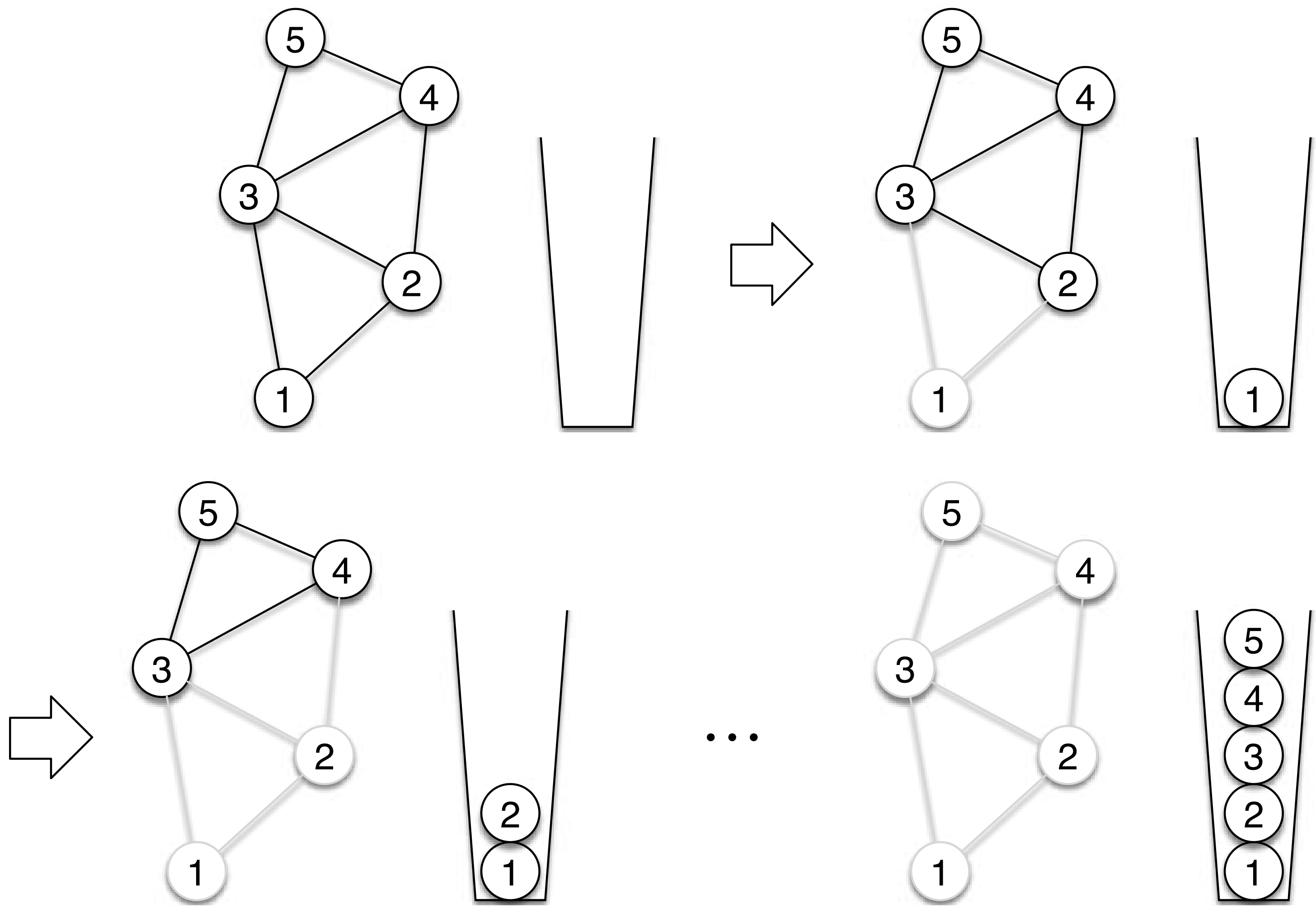
$$x_{i1} = 0.5$$
$$x_{i2} = 0.5$$



S_{i1}	S_{i2}	S_{ic}
$\ldots + x_{i1} + \ldots \leq c_1$	$\ldots + x_{i2} + \ldots \geq c_5$	$\ldots + x_{i1} + x_{i2} + \ldots \leq c_9$
$\ldots + x_{i1} + \ldots \leq c_2$	$\ldots + x_{i2} + \ldots \geq c_6$	$\ldots + x_{i1} - x_{i2} + \ldots \geq c_{10}$
$\ldots + x_{i1} + \ldots \leq c_3$	$\ldots + x_{i2} + \ldots \geq c_7$	$\ldots + x_{i1} + x_{i2} + \ldots \geq c_{11}$
$\ldots + x_{i1} + \ldots \leq c_4$	$\ldots + x_{i2} + \ldots \geq c_8$	
Try pushing x_{i1} to 0	Try pushing x_{i2} to 1	Check $(x_{i1}, x_{i2}) = (0, 1)$

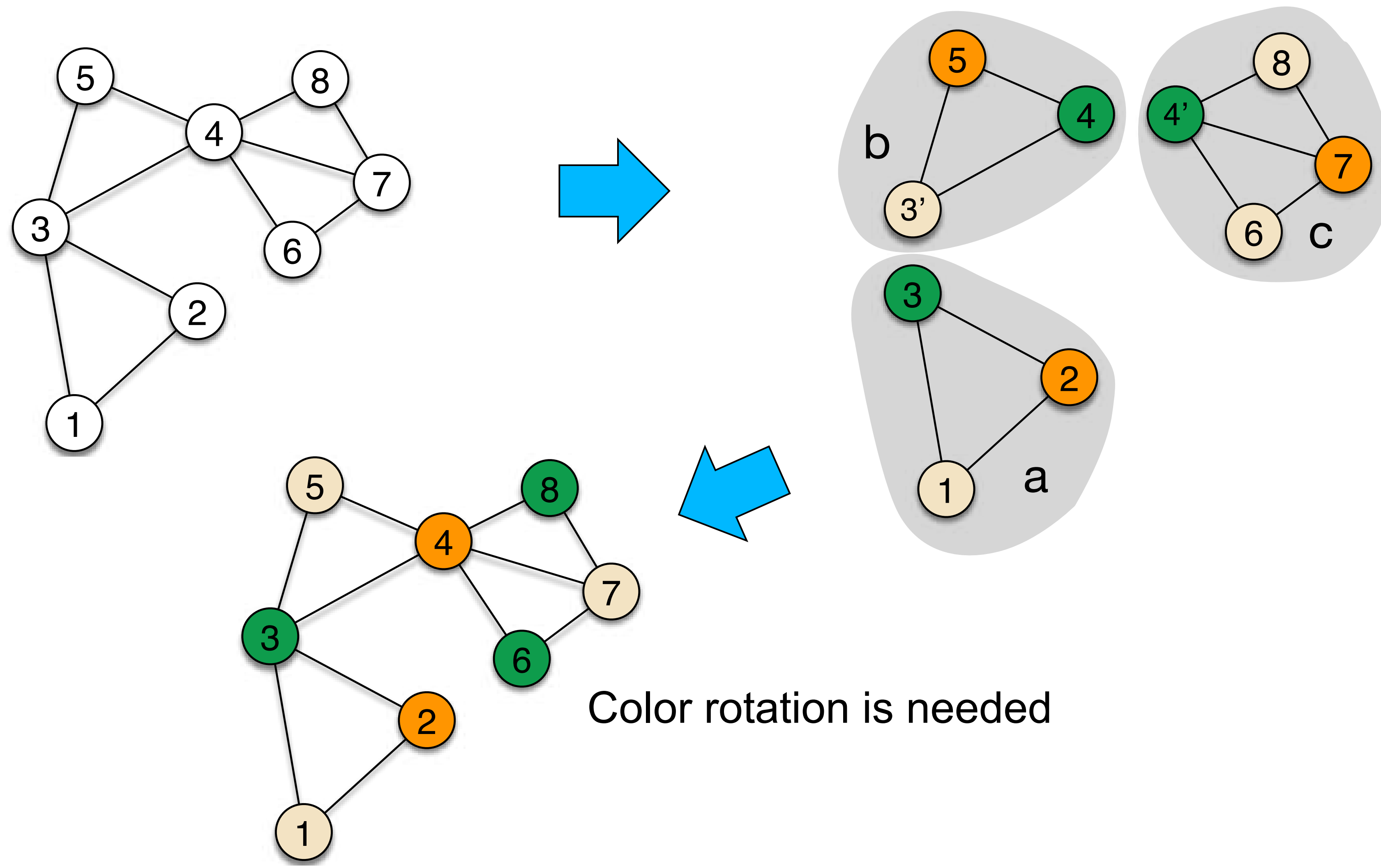
Graph Simplification – Iterative Vertex Removal

- Iterative vertex removal
- Density aware recovery

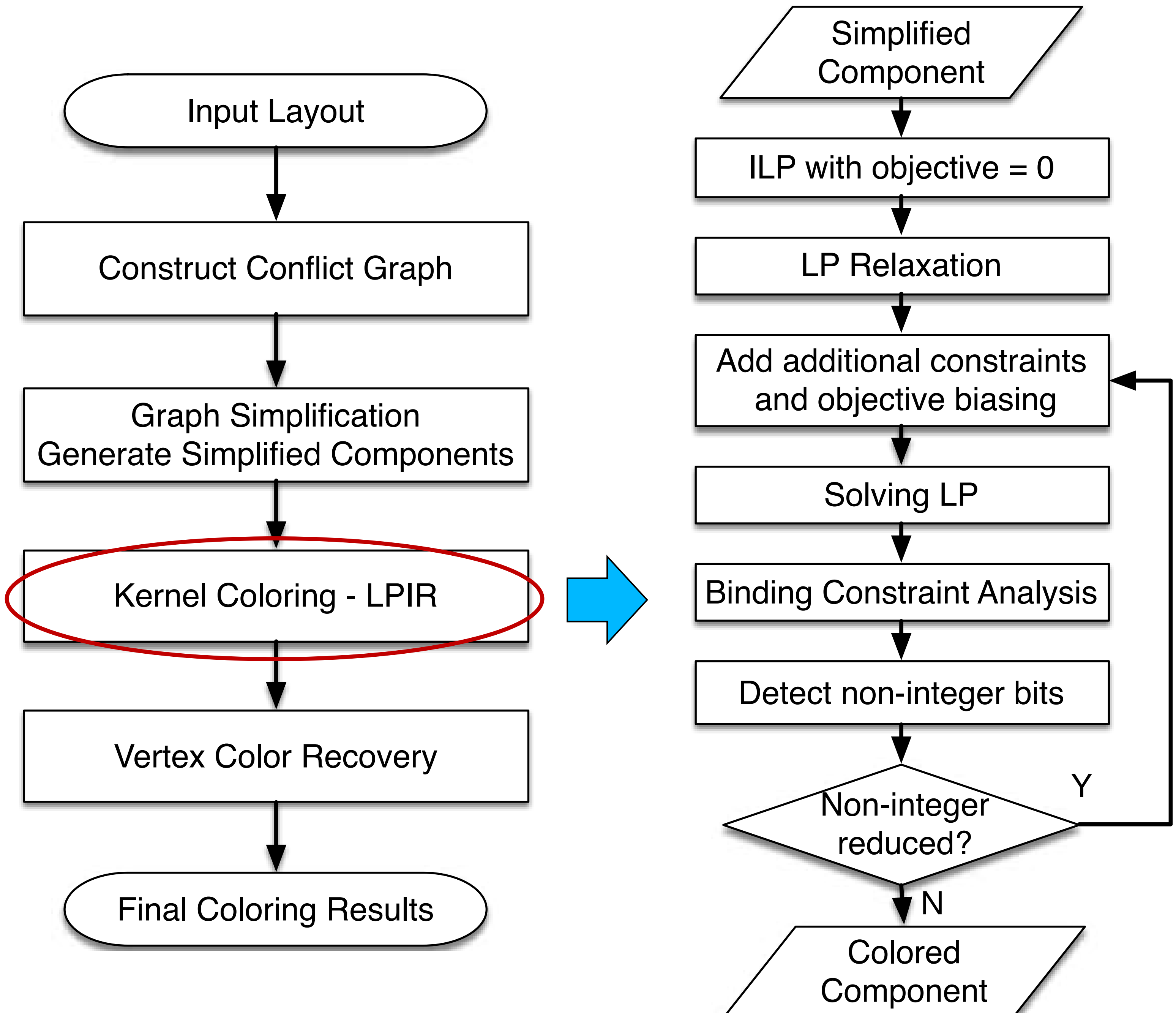


Graph Simplification: Bi-connected Component Extraction

- Color recovery
  - Color rotation on each component



Overall Flow

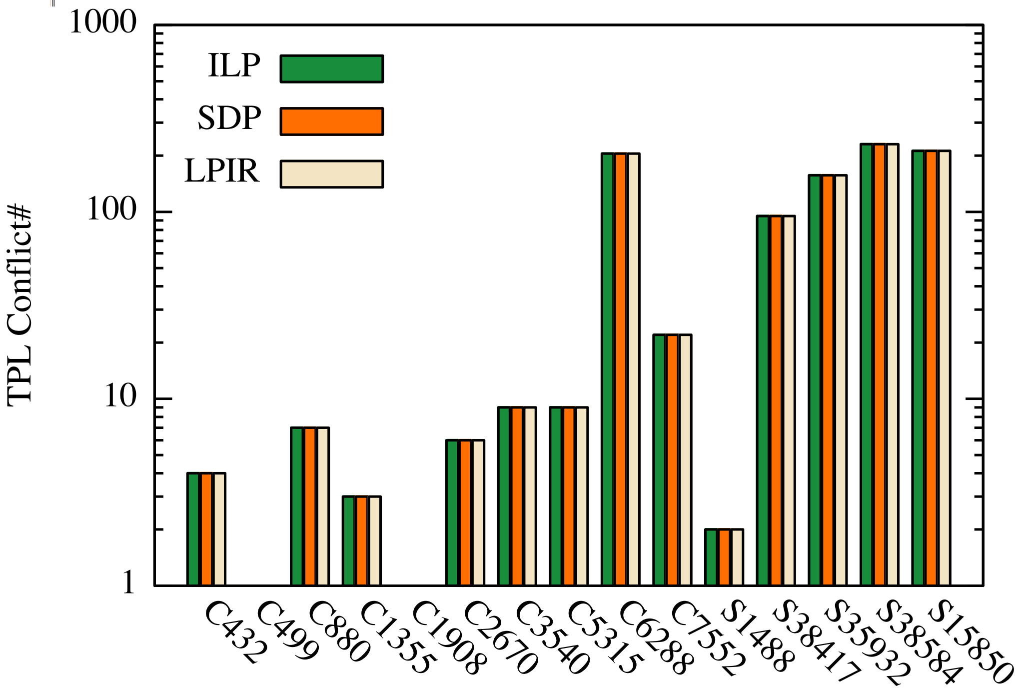


Experimental Environment Setup

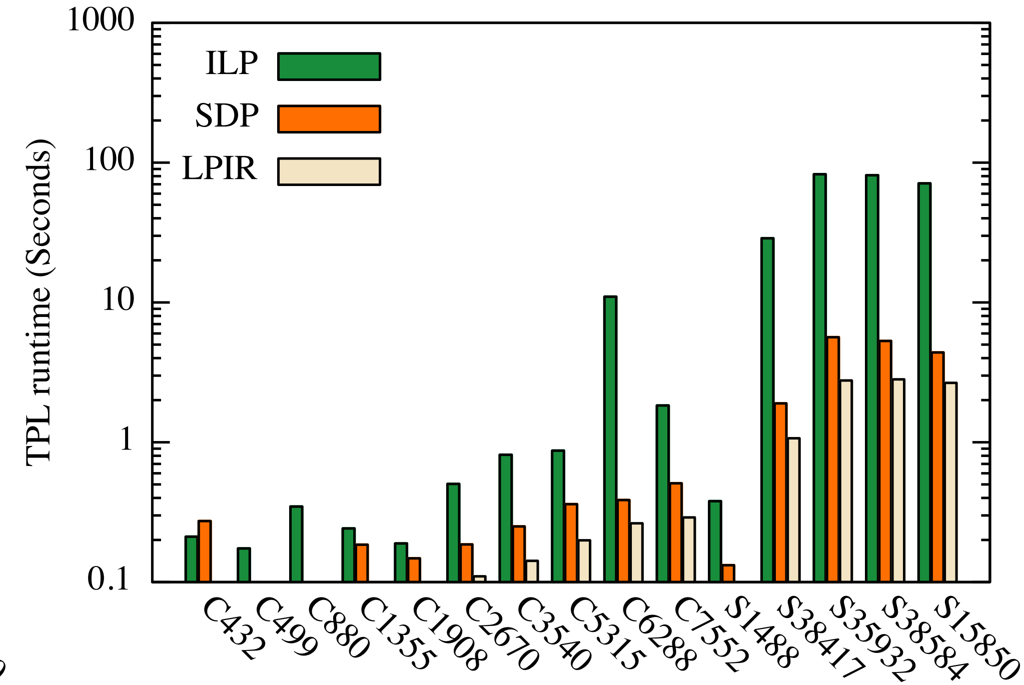
- Implemented in C++
- 8-Core 3.4GHz Linux server
- 32GB RAM
- ISCAS benchmark from [Yu+, TCAD'15]
- LP solver Gurobi was used

Experimental Results on TPL

TPL conflict#

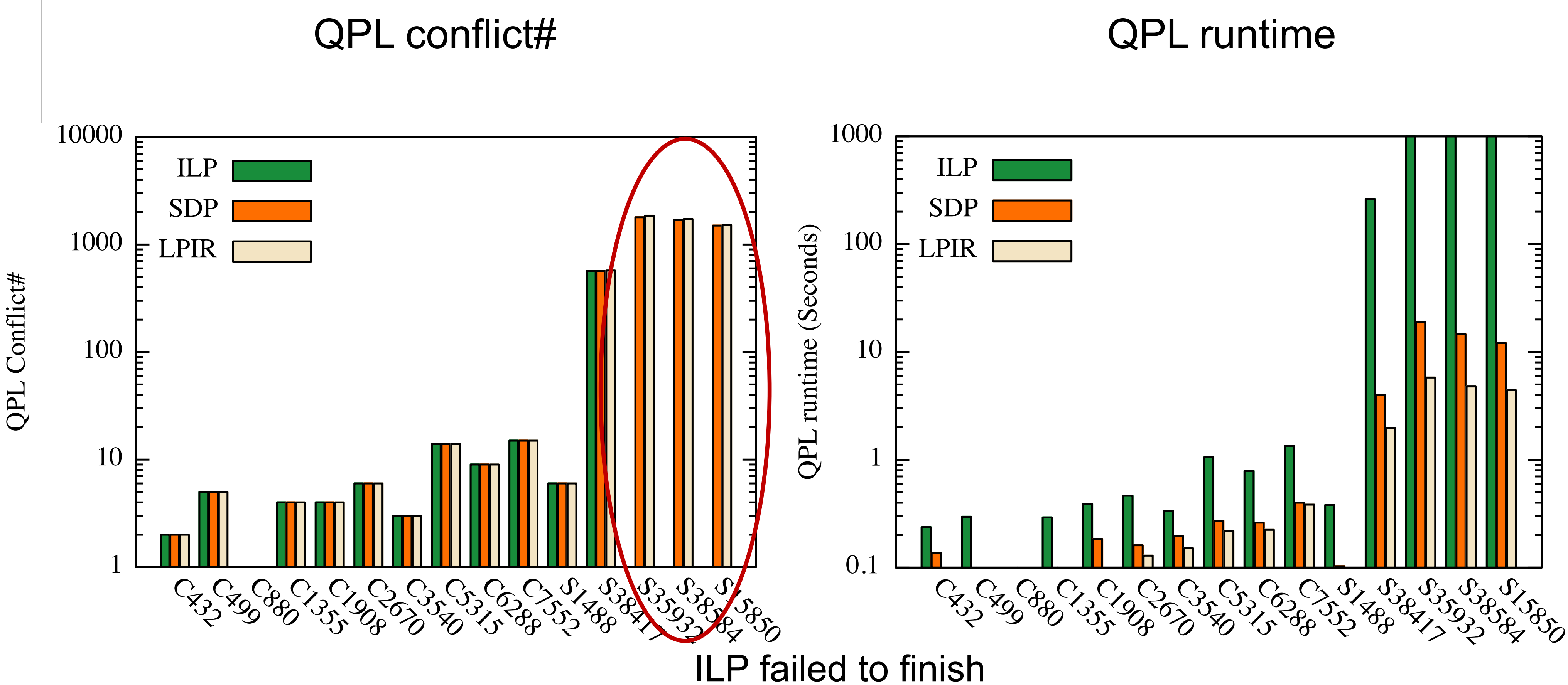


TPL runtime



Baseline 1: ILP [Yu+, TCAD'15]
Baseline 2: SDP [Yu+, TCAD'15]
LPIR achieves almost the same conflict numbers as ILP and SDP, but 26x faster than ILP and 1.8x faster than SDP

Experimental Results on QPL



Baseline 1: ILP [Yu+, DAC'14]
Baseline 2: SDP [Yu+, DAC'14]
LPIR achieves less than 2% degradation in conflict numbers than SDP, but 600x faster than ILP and 2.6x faster than SDP

Conclusion

- This paper proposes a new layout decomposition framework for TPL/QPL
  - Novel linear programming (LP) based algorithm with iterative rounding
  - Odd-cycle based pruning technique to enhance LP quality
  - Very good results cf. previous state-of-the-art decomposer
- Future work
  - Lithography impacts (e.g., hotspots) from different decomposition solutions
  - Decomposition friendliness from early design stages like placement and routing

Thanks!