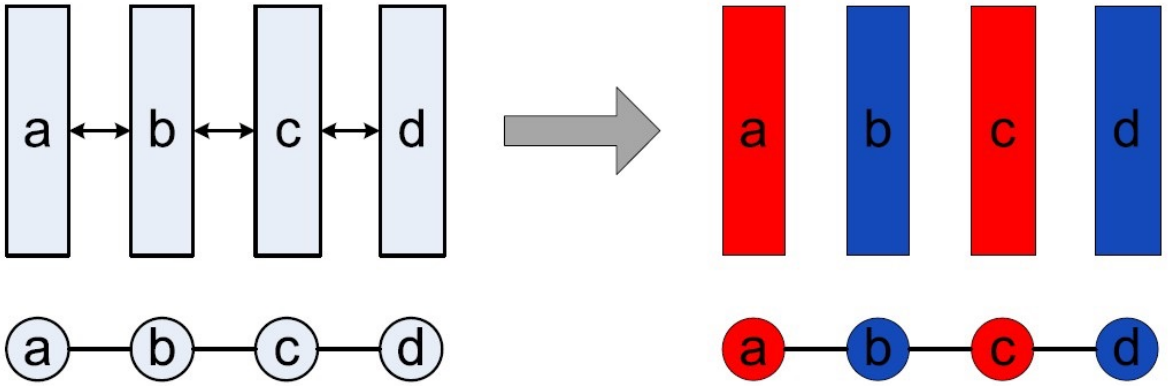
Triple Patterning Decomposition for Cell-Based Row-Structure

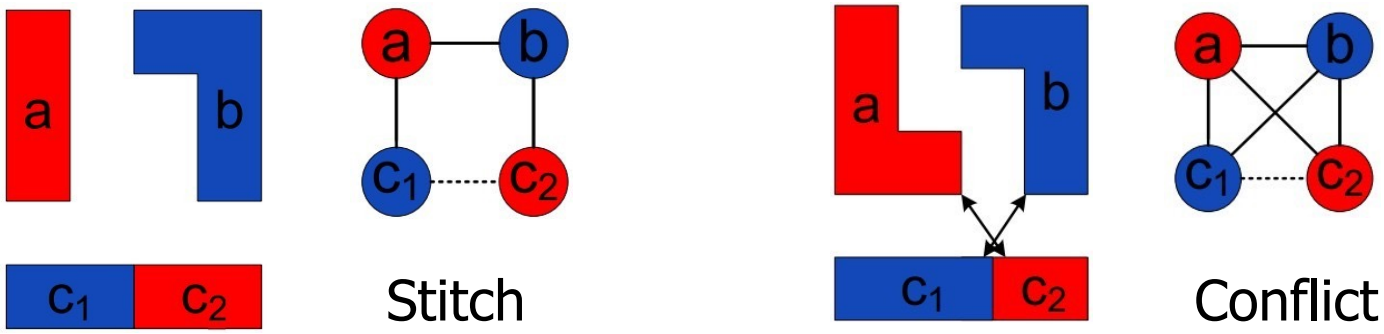
Double Patterning Lithography (DPL)

- DPL is a feasible litho process for 20nm node
- DPL decomposes features on one layer into two masks
  - Doubles printing pitch
  - Enhances resolution
- Color-conflicting features with spacing between them less than d_{min} have to be assigned to different masks (colors)



From DPL to TPL

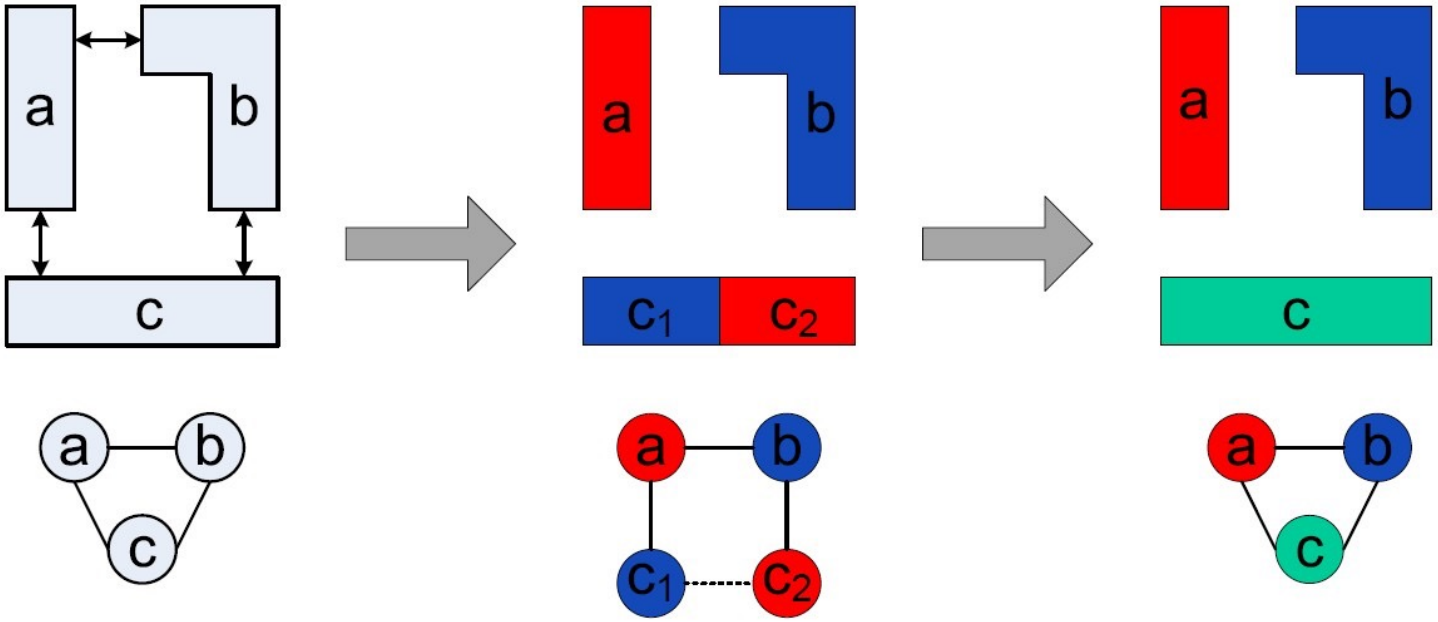
- As technology continues to scale to 14nm, DPL is being pushed to its limits
  - **Stitches:** introduce overlay error
  - **Color-Conflicts:** make layout indecomposable



- TPL is a practical extension of DPL
  - Three masks available to accommodate the features
  - Reduce stitches and resolve color-conflicts in 14nm node
  - Make 10nm node design manufacturable

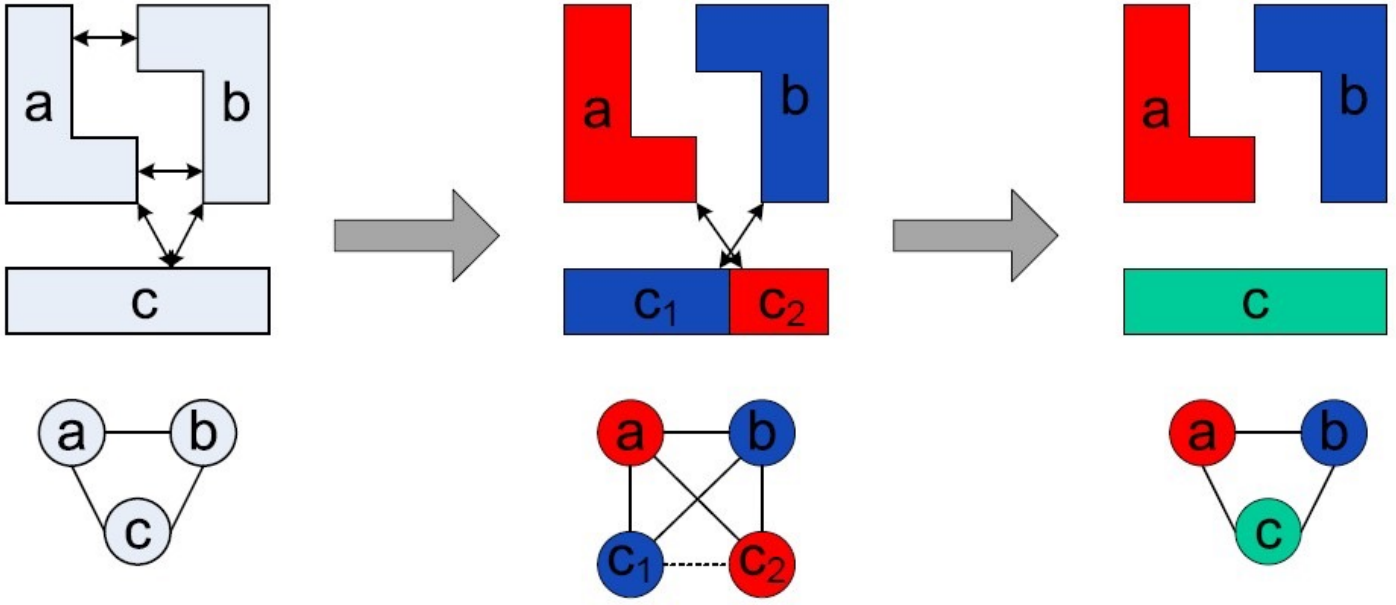
From DPL to TPL

- TPL can reduce the number of stitches in DPL



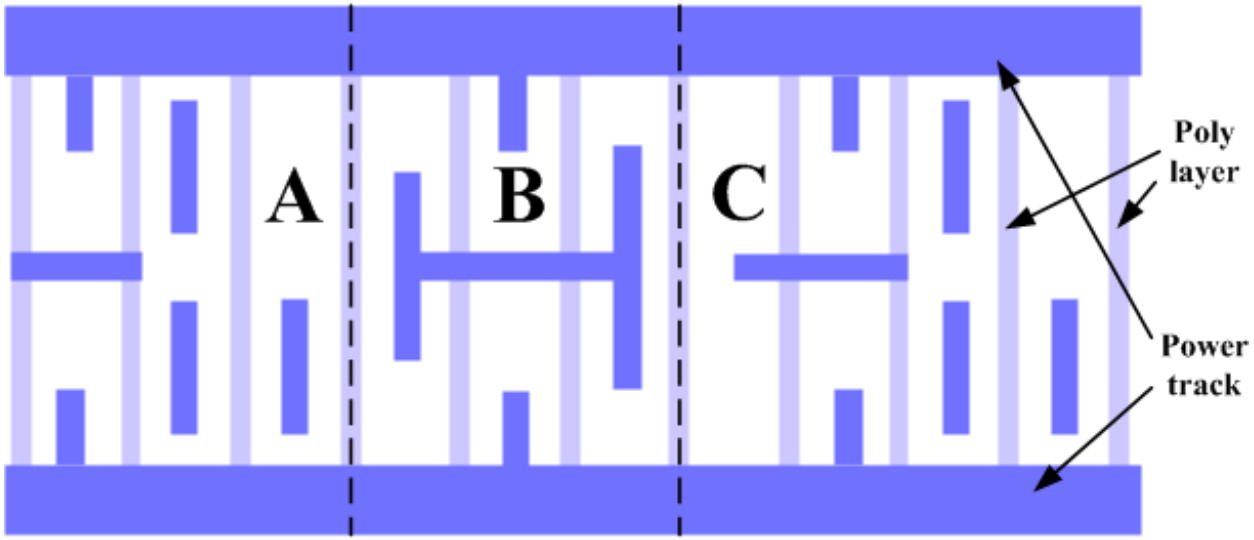
From DPL to TPL

- TPL can resolve certain conflicts in DPL



TPL for Standard Cell Design

- Cells with the same height
- Power track going from the left end to the right end
- Wires
  - M1
    - Most complex
    - Multiple exposures needed
  - M2+
    - preferred routing directions
- TPL needed for M1



Standard cell row with three cells

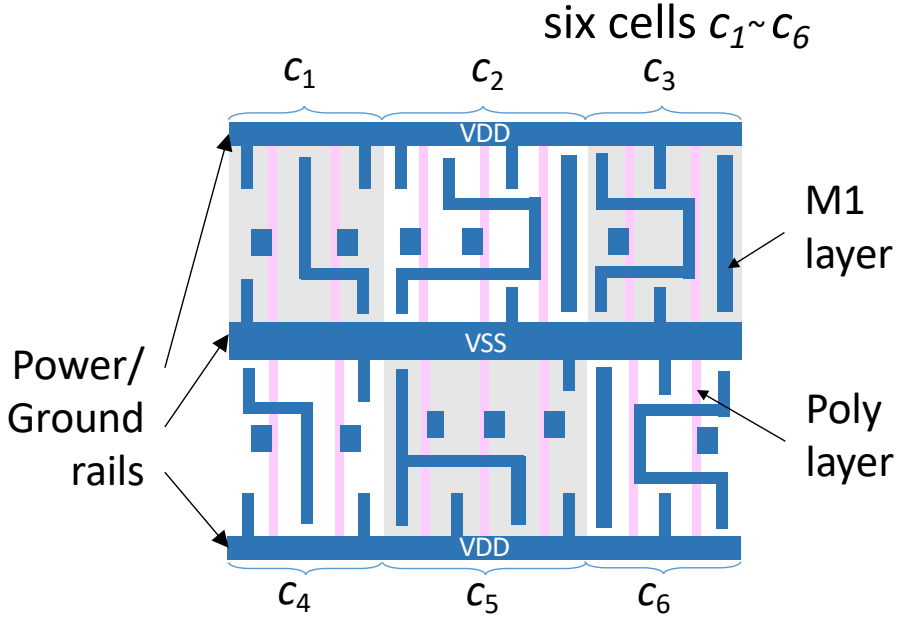
Special layout structure
Optimal graph coloring possible !

A Polynomial Time Triple Patterning Algorithm for Cell Based Row-Structure Layout

H. Tian, H. Zhang, Q. Ma, Z. Xiao, M.D.F. Wong

Problem Formulation of TPL Decomposition for Cell-Based Row-Structure

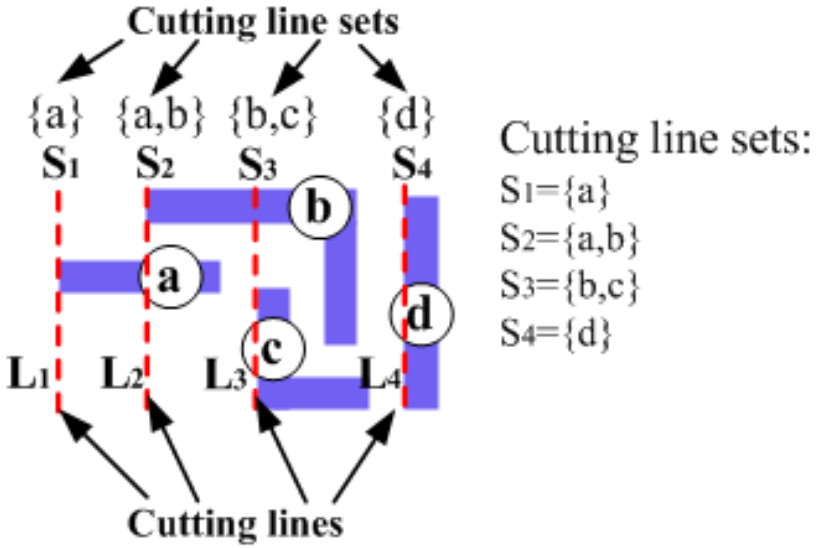
- Given
  - A cell-based row-structure M1 layout
  - A minimum coloring spacing d_{min}
- Objective
  - Find a feasible TPL decomposition of the layout

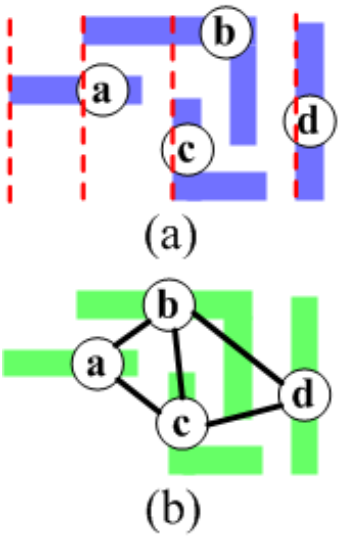


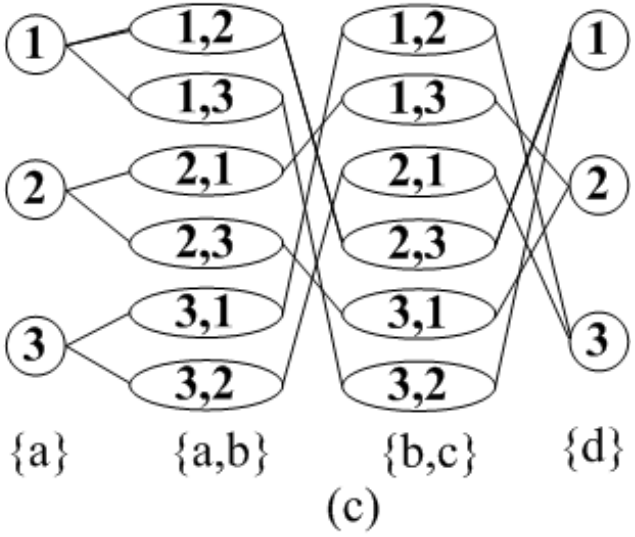
- Important observation

  Polygons separated by power/ground track in adjacent cell rows will not interact (i.e., more than d_{min} apart)

TPL Algorithm: basic concept







Cutting line

- a vertical line at the left boundary of a feature going from top of a cell to the bottom

Cutting line set

- polygons intersect with the same cutting line

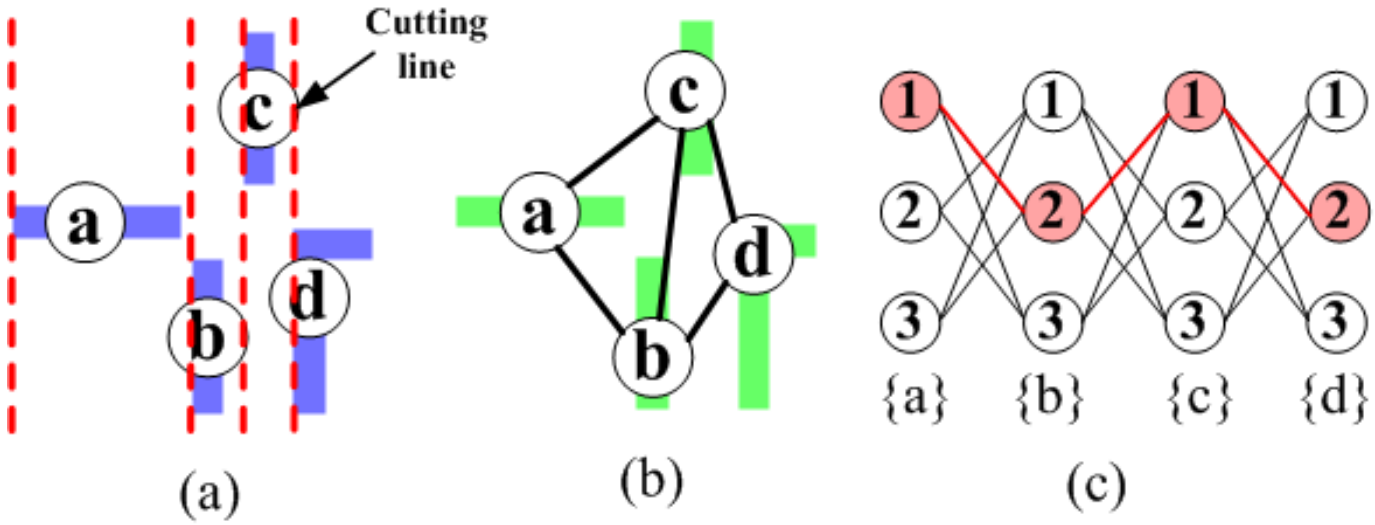
Conflict graph

- Nodes: polygons
- Edges: conflicting relations

Solution graph

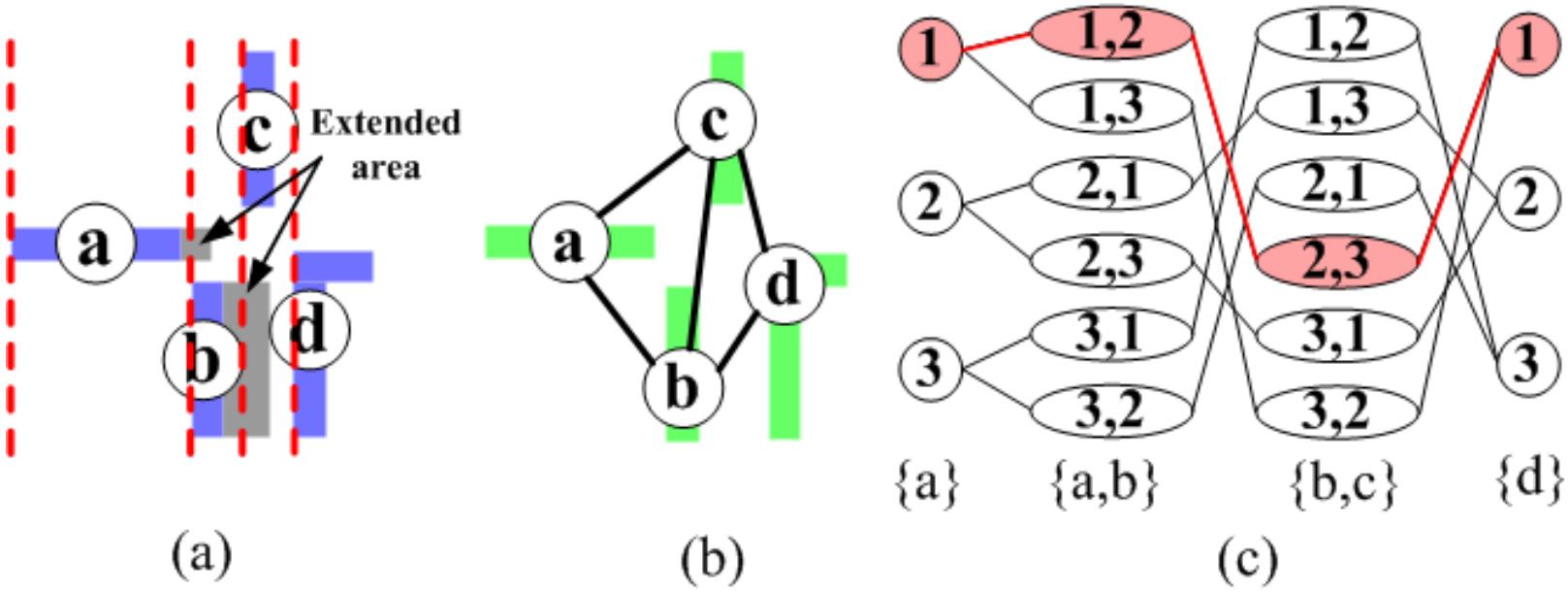
- Nodes: solutions of cutting line sets
- Edges: compatible solutions

TPL Algorithm: basic concept



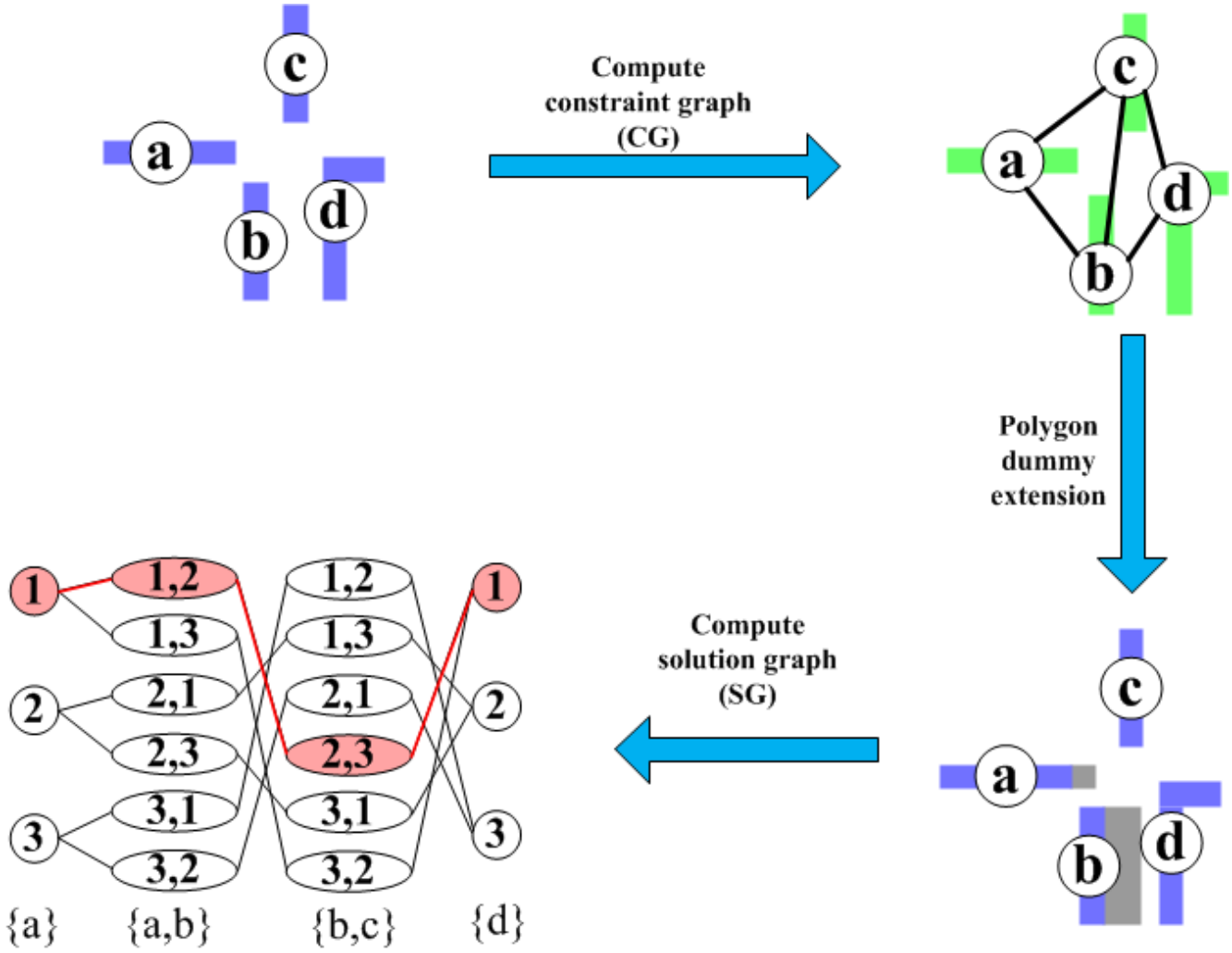
- Problem arises:
  - Four cutting line sets: {a}, {b}, {c}, and {d} respectively.
  - Solution graph shown in (c)
  - Coloring assignment {a(1), b(2), c(1), d(2)}: ***illegal!***

TPL Algorithm: basic concept



- Polygon dummy extension
  - Extend right boundary of each polygon pass all RHS conflicting polygons except the last one (why?)
- Solution graph after polygon dummy extension
  - Every path is guaranteed to be a valid one

TPL Algorithm

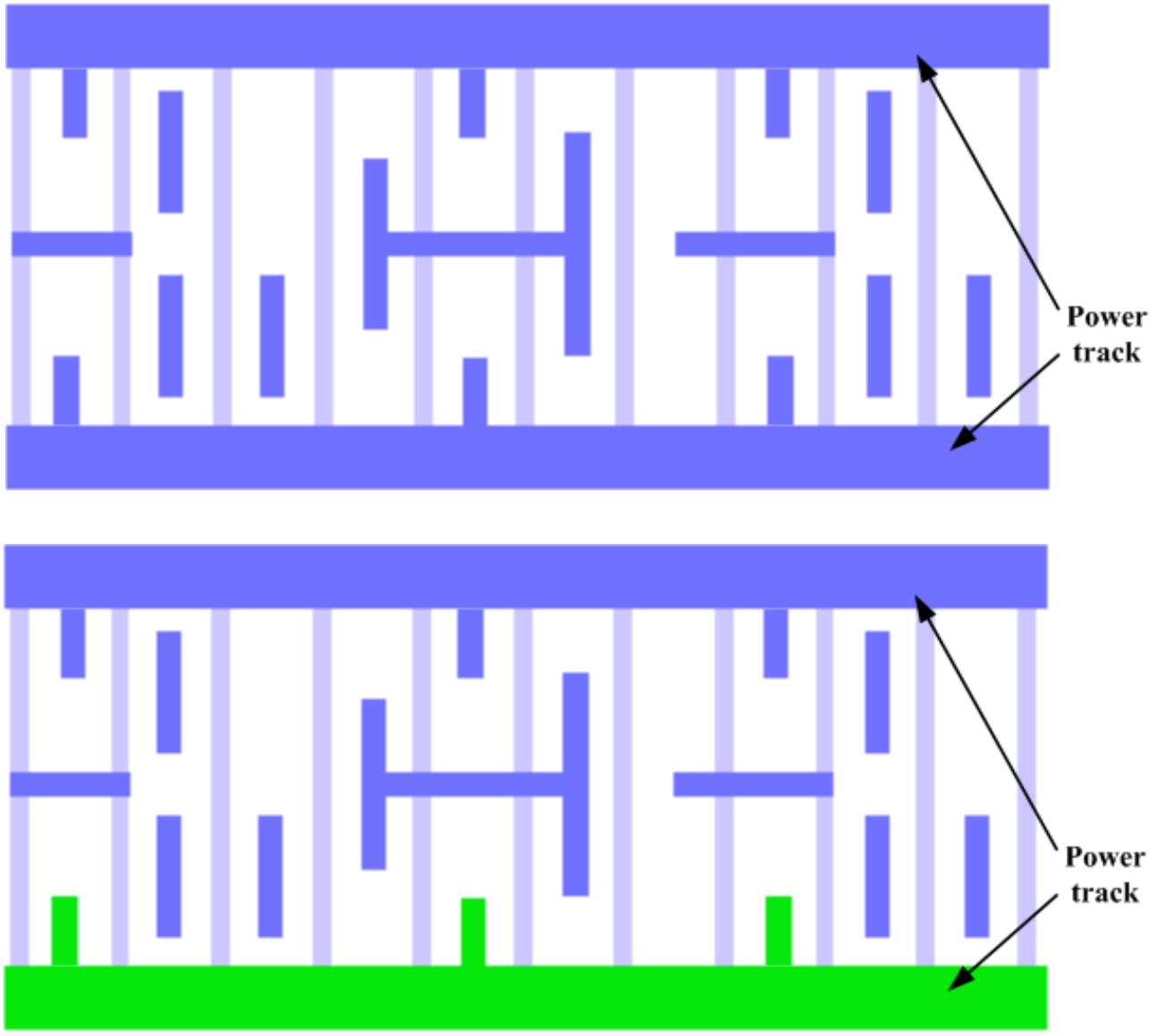


Solution: any path from left to right of the solution graph

Power/Ground Tracks

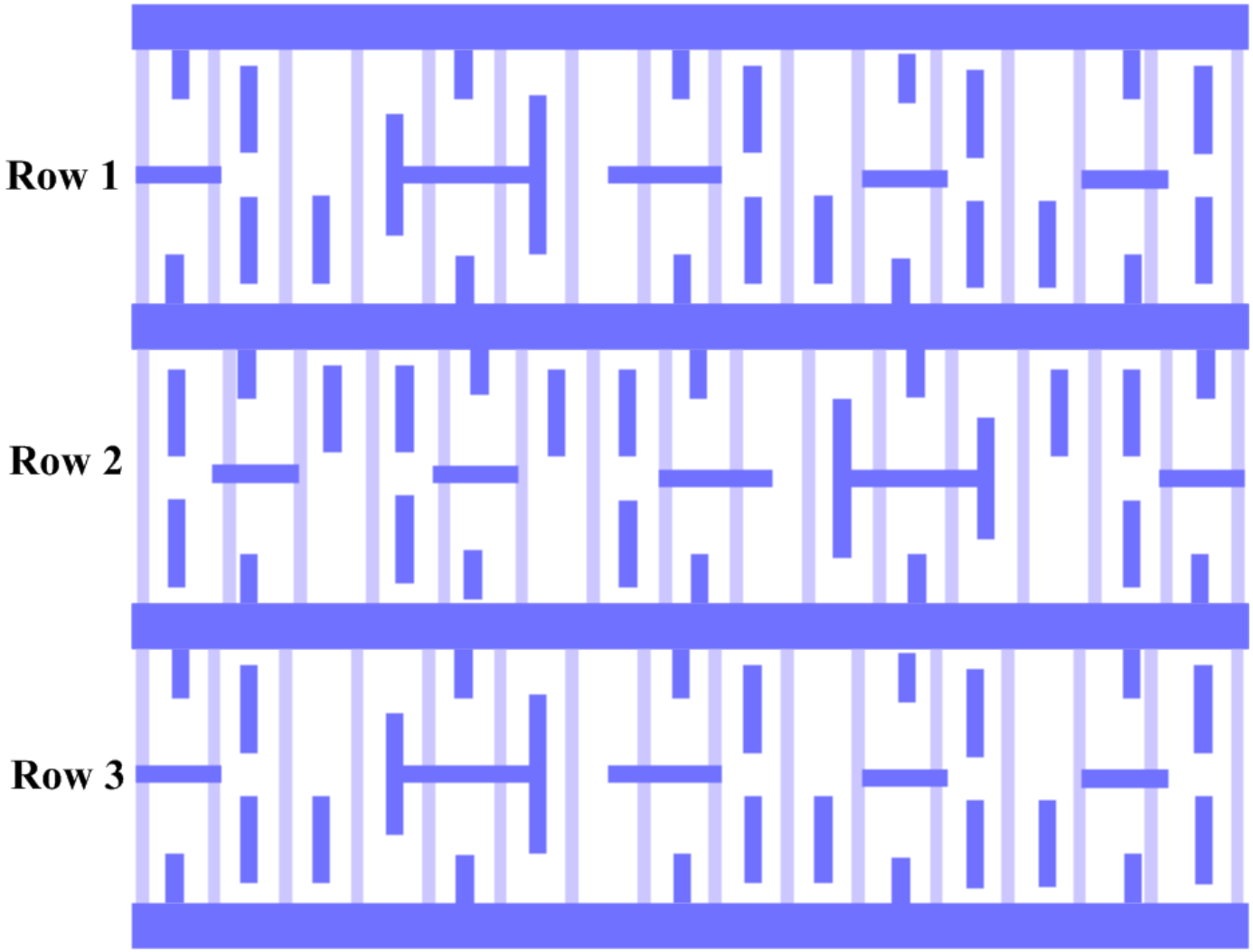
For two consecutive P/G tracks in layout

- choice 1:
  - Pre-color them with same color
- choice 2:
  - Pre-color them with different colors
- Choose the better one



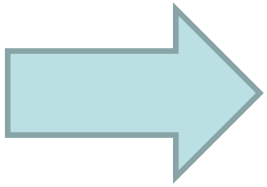
Obtain Optimal Solution for Multiple Rows

Solving row by row gives optimal result !



Hierarchical Approach

Standard Cell Based Designs

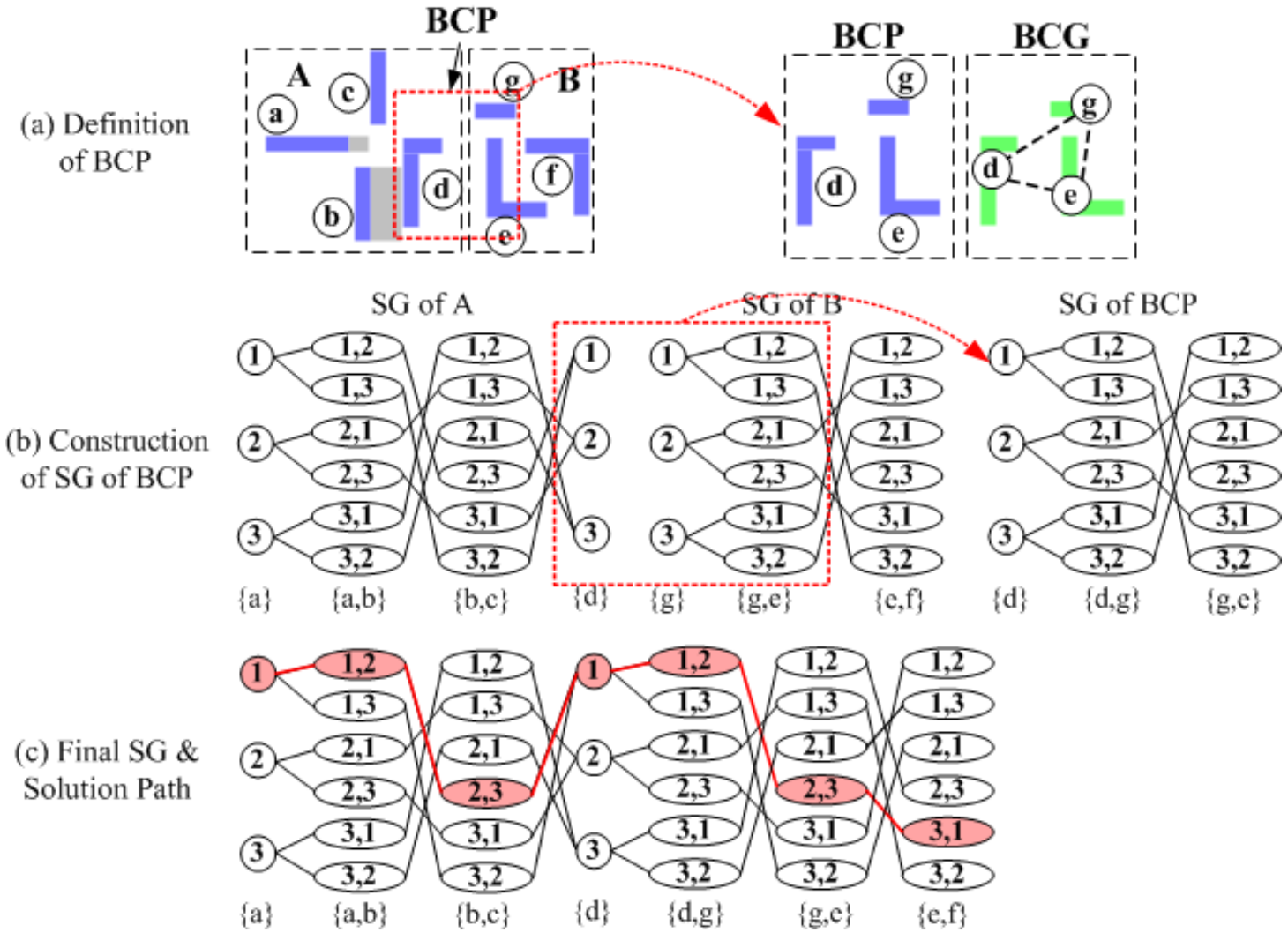


Millions of features
vs
Hundreds of cells in library

Solution graph of cells can be pre-computed

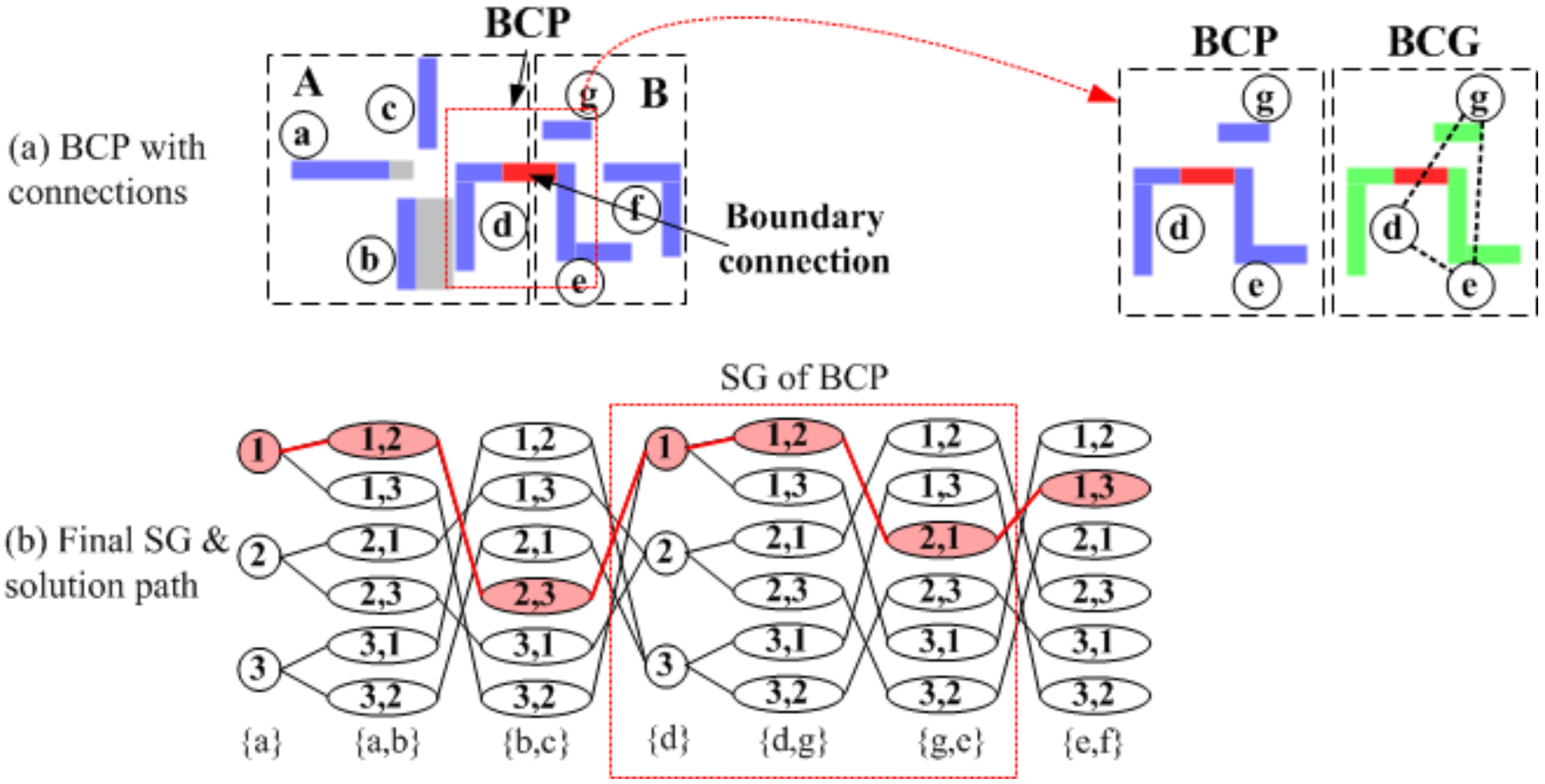
Reuse the solution graphs of the cells

Boundary Polygons



- Boundary constraint polygons (BCP): polygons within a distance of d_{min} on the boundaries of two cells

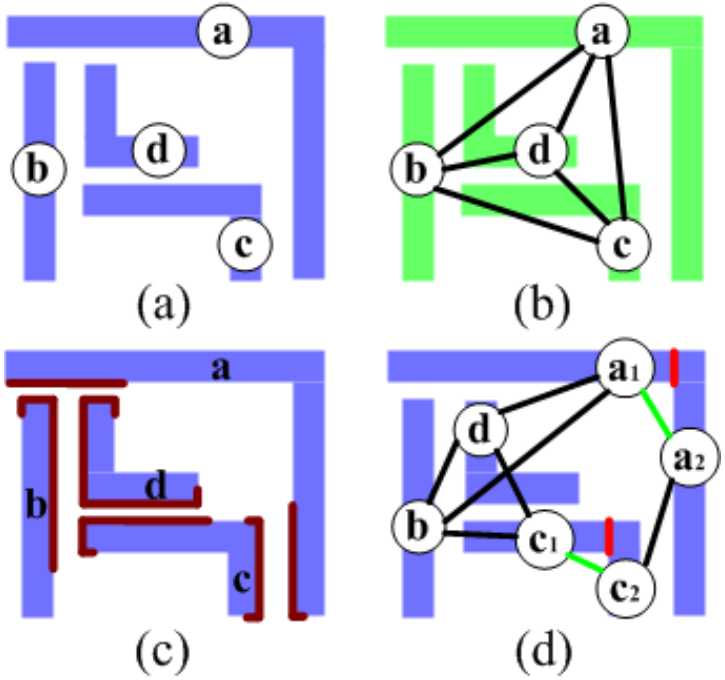
Boundary Connections

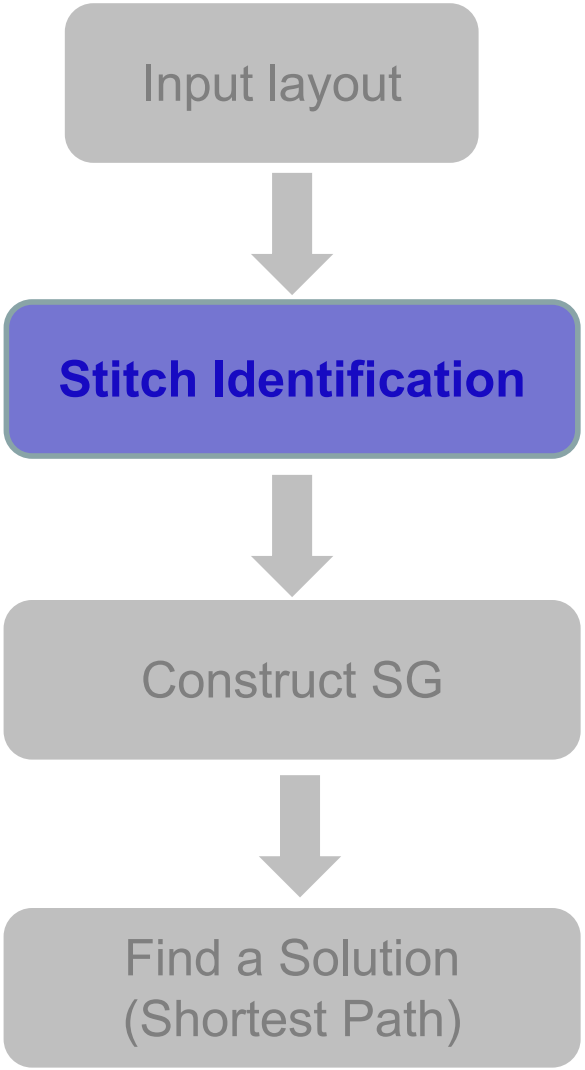


- BCP with connections:
  - Connecting polygons are assigned the same color
  - Combine SGs of A, B, and BCP

TPL with Stitches

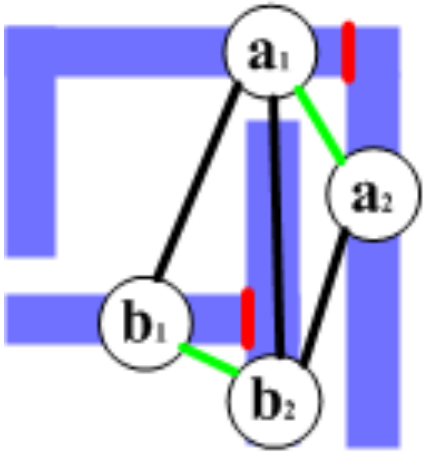
- Stitch identification
  - Conflict graph
  - Node projection
  - Compute legal stitches

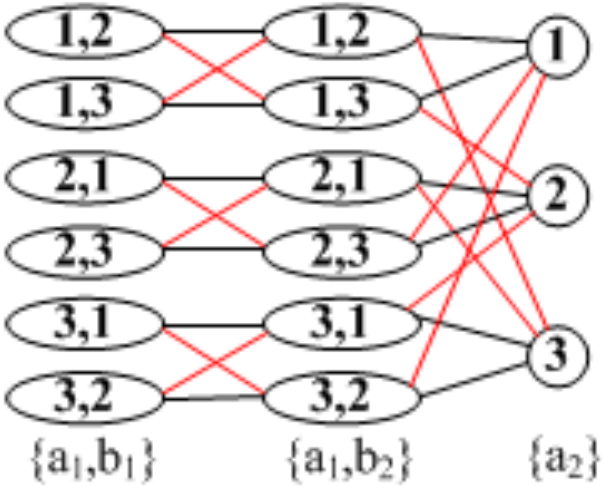


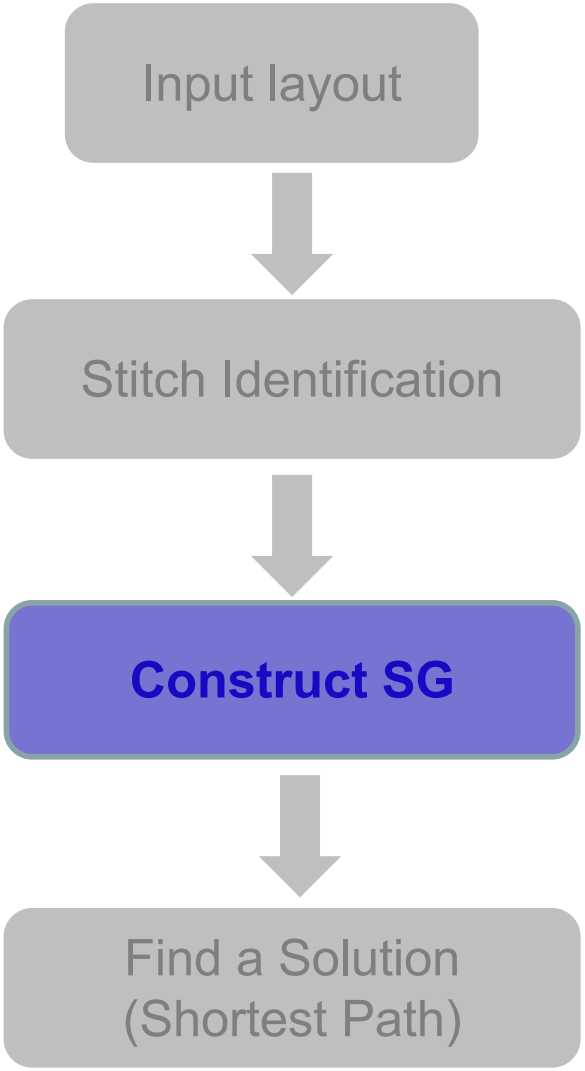


TPL with Stitches

- Similar to TPL without stitches
  - Scan cutting lines from left to right
  - Edges are added between compatible solutions
  - Weight are assigned to edges



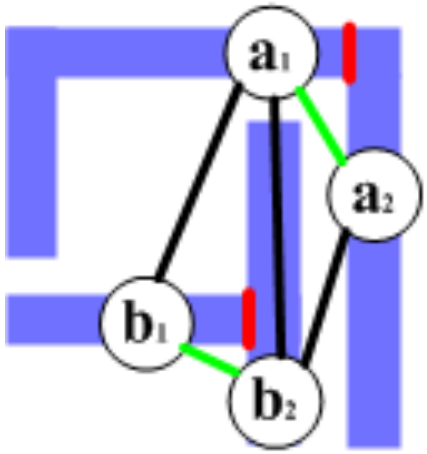


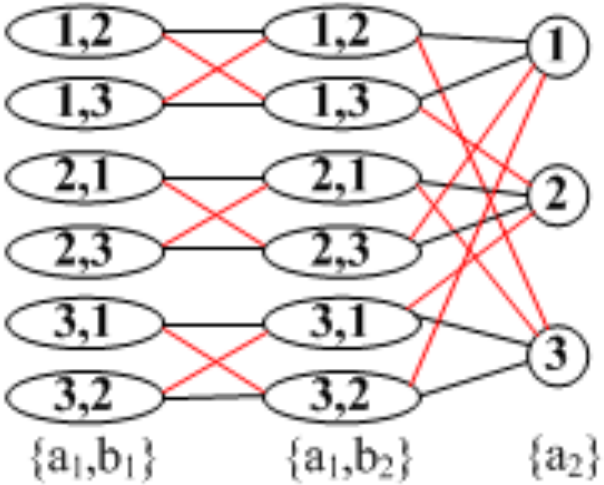


- **The red edge is of weight 1**
- **The black edge is of weight 0**

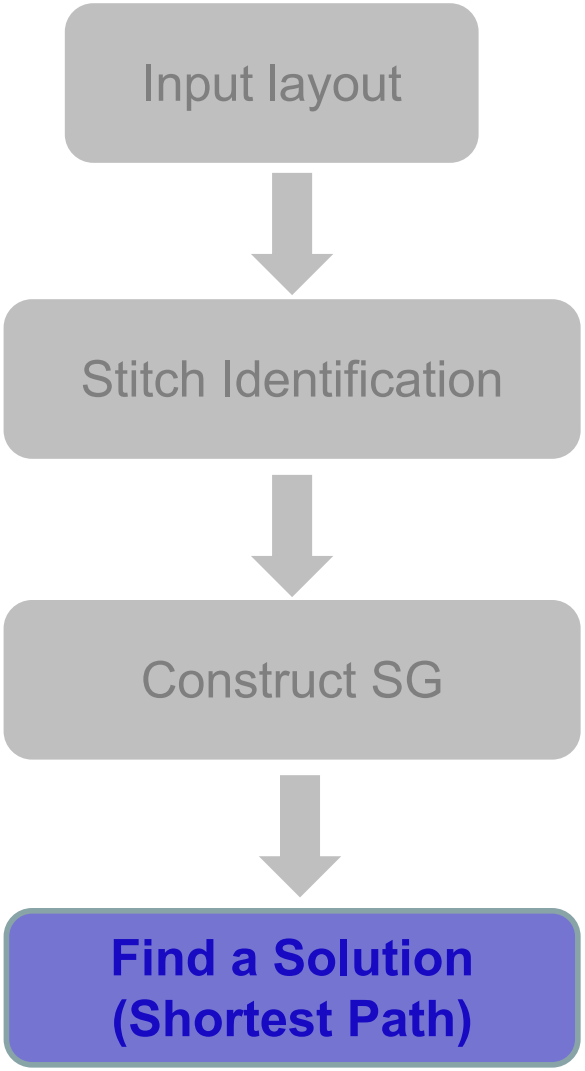
TPL with Stitches

- Solution graph
  - Weighted graph
  - Shortest path formulation
  - Guarantees to find an optimal solution if one exists



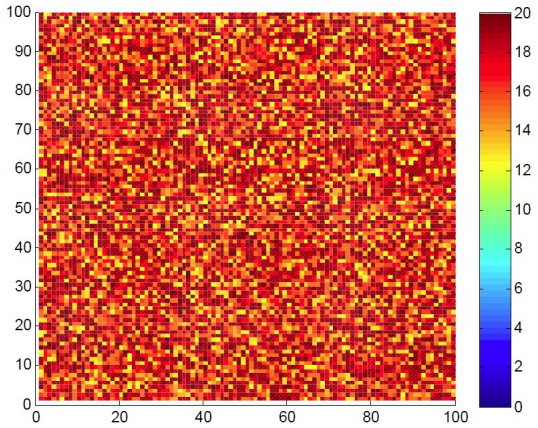


- **The red edge is of weight 1**
- **The black edge is of weight 0**

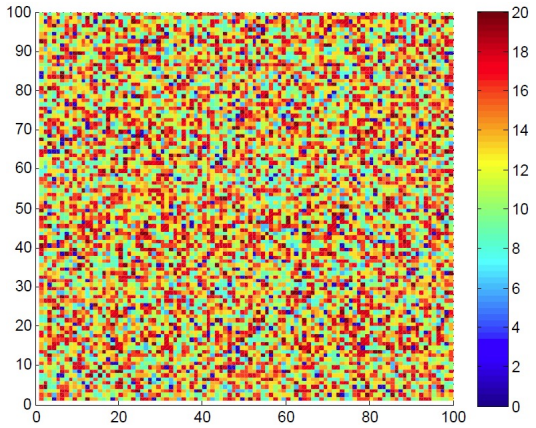


TPL with Color Balancing

- Evenly distribute the features among different masks
- Less color balancing means
  - Unbalanced usage of mask resources
  - Decreased size of process window



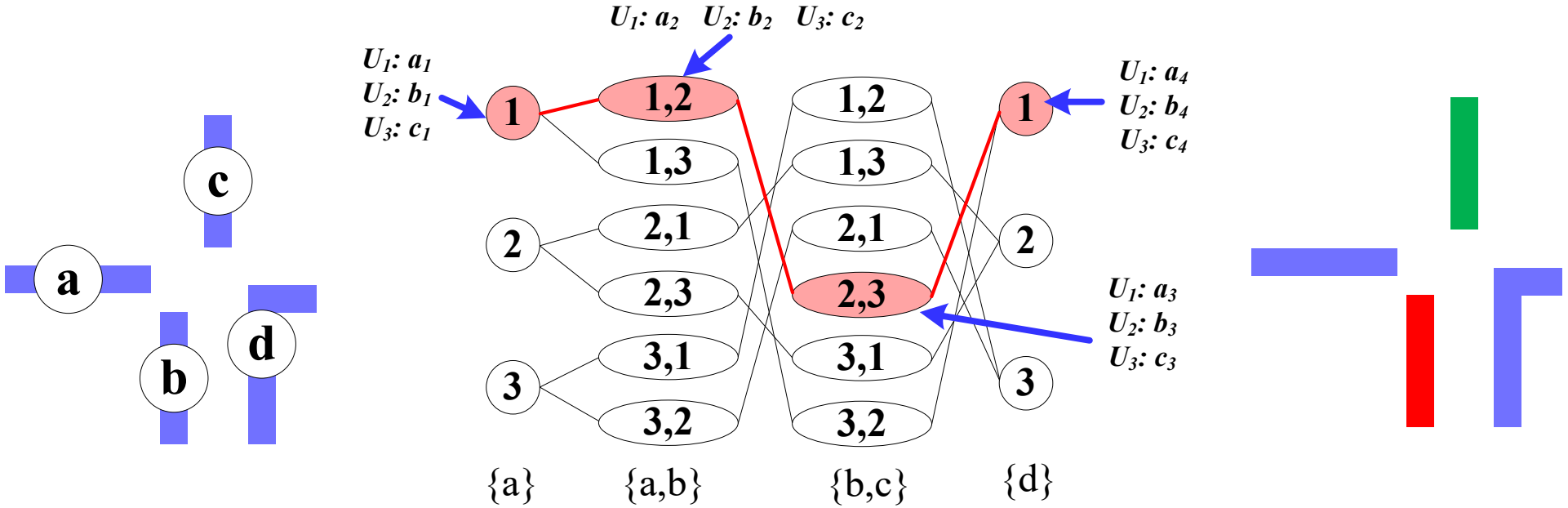
- Balancing Map without color balancing



- Balancing Map with color balancing

Color Balancing

- Color-balancing aware path search
  - Recording utilizations of three masks in every node
  - Modified shortest path algorithm



U_1: Utilization of mask 1 U_2: Utilization of mask 2 U_3: Utilization of mask 3

Experimental Results

Benchmarks	n	Tracks	Balanced area ratio	Random area ratio	Time (s)
C1	106690	143	1:1:1	1:0.27:0.23	10
C2	674841	358	1:1:1	1:0.26:0.24	66
C3	2695803	715	1:1:1	1:0.25:0.24	264
C4	10782073	1429	1:1:1	1:0.25:0.24	1062
C5	26949406	715	1:1:1	1:0.26:0.24	2655

- Five benchmarks
  - Generated using NanGate FreePDK45 Generic Open Cell Library
  - Much balanced solution
  - Reasonable run time

Experimental Results

Benchmarks	n	Tracks	Time1 (s)	Time2 (s)	Improve (%)
C1	106690	143	10	16	35.6
C2	674841	358	66	101	34.2
C3	2695803	715	264	403	34.4
C4	10782073	1429	1062	1610	34.0
C5	26949406	715	2655	4028	34.1
Ave.	8241763	672	812	1232	34.5

- Run time comparisons
  - Hierarchical approach: 34.5 % improvement on average
  - Without losing optimality of the algorithm

Experimental Results

Benchmarks	n	Tracks	Stitch candidates	Stitches	Time (s)
C6	179201	143	78102	3420	80
C7	904292	322	394349	17146	388
C8	4449681	715	1940587	83916	1900
C9	10031115	1072	4382524	188854	4277
C10	17813611	1429	7778321	334642	7613

TPL with stitches

- ✓All solutions computed are guaranteed to have the minimum number of stitches
- ✓Reasonable run time

Experimental Results

Bench marks	SDP Based [Yu+ ICCAD11]			[Fang+ DAC12]			Proposed [Tian+ ICCAD12]		
	C	S		C	S		C	S	
C432	3	1	×	0	6	✓	--	--	×
C499	0	0	✓	0	0	✓	0	0	✓
C880	1	6	×	1	15	×	0	7	✓
C1355	1	6	×	1	7	×	0	3	✓
C1908	0	1	✓	1	0	×	0	1	✓
C2670	2	4	×	2	14	×	0	6	✓
C3540	5	6	×	2	15	×	--	--	×
C5315	7	7	×	3	11	×	--	--	×
C6288	82	131	×	19	341	×	--	--	×
C7552	12	15	×	3	46	×	--	--	×

Experimental Results

Bench marks	SDP Based [Yu+ ICCAD11]			[Fang+ DAC12]			Proposed [Tian+ ICCAD12]		
	C	S		C	S		C	S	
S1488	1	1	×	0	6	✓	0	2	✓
S38417	44	55	×	20	122	×	--	--	×
S35932	93	18	×	46	103	×	--	--	×
S38548	63	122	×	36	280	×	--	--	×
S15850	73	91	×	36	201	×	--	--	×

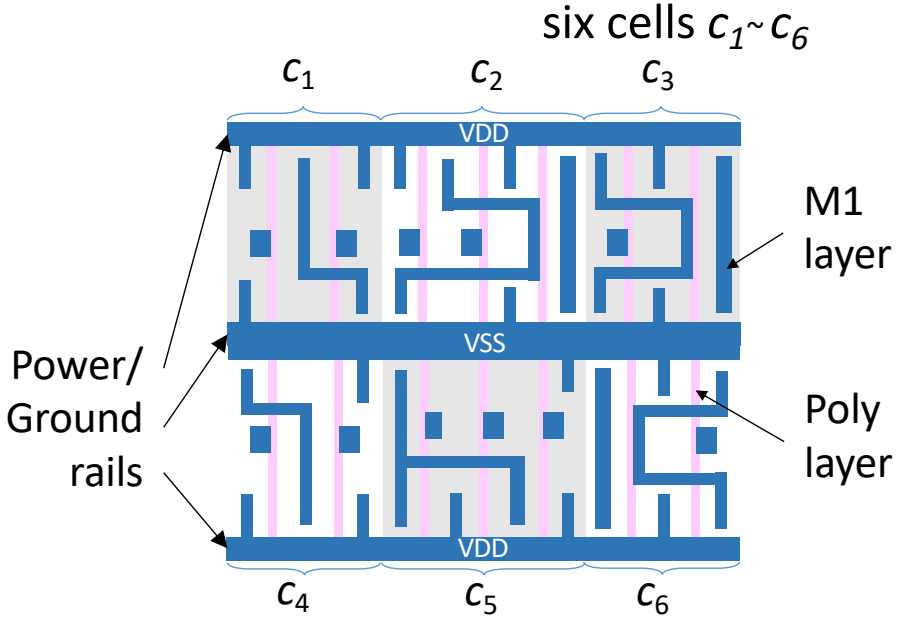
References

- [Yu+ ICCAD11] Layout decomposition for triple patterning lithography
- [Fang+ DAC12] A novel layout decomposition algorithm for triple patterning lithography
- [Tian+ ICCAD12] A polynomial time triple patterning algorithm for cell based row-structure layout

A Cell-Based Row-Structure Layout Decomposer for Triple Patterning Lithography

Hsi-An Chien, Szu-Yuan Han, Ye-Hong Chen,
and Ting-Chi Wang

Motivation



- TPL layout Decomposition
  - General layout: *NP*-hard
  - Row-structure layout (M1 layer)
    - For decomposable layout
      - Polynomial-time solvable with minimum stitches [Tian+ ICCAD12]
    - Non-decomposable layout
      - No decomposition solution
      - Hard to help designers further modify the layout
      - Desired: Decomposition solution with minimum total cost (coloring conflicts & stitches)

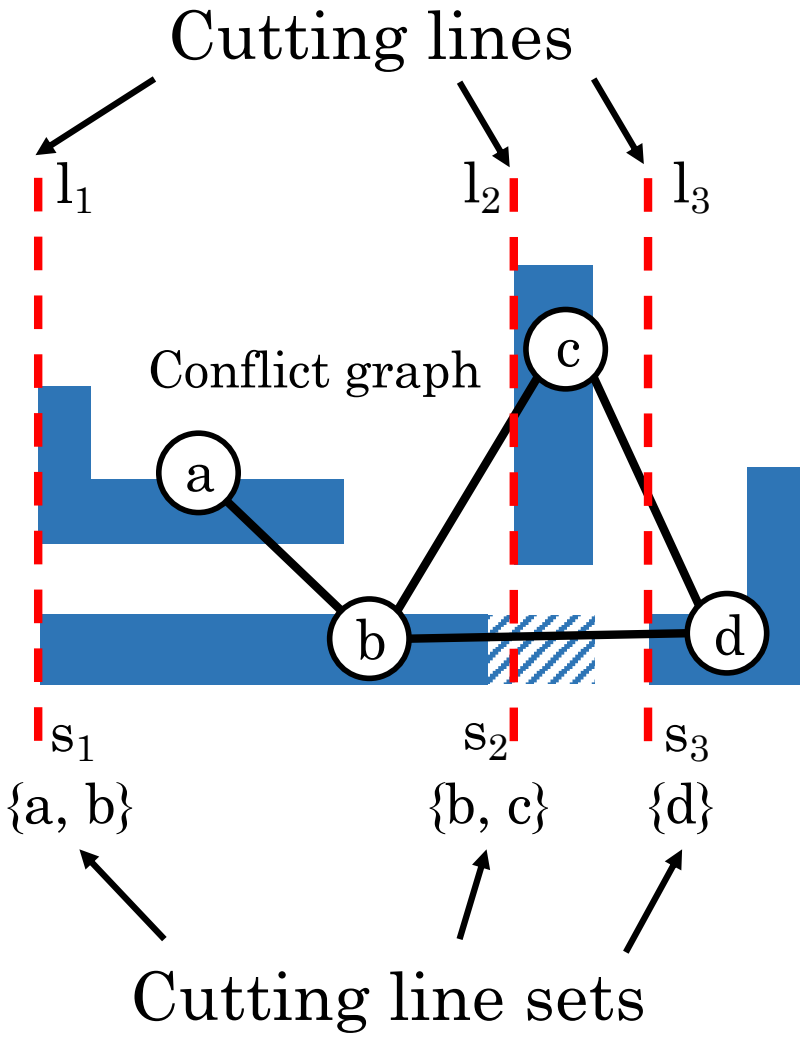
A Modified Problem Formulation

- **Given**
  - A cell-based row-structure M1 layout
  - A minimum coloring spacing d_{min}
- **Objective**
  - Finding a TPL decomposition with a minimal weighted sum of coloring conflicts and stitches.

$$\text{weighted sum} = \alpha \,\#\text{conflicts} + \beta \,\#\text{stitches}$$

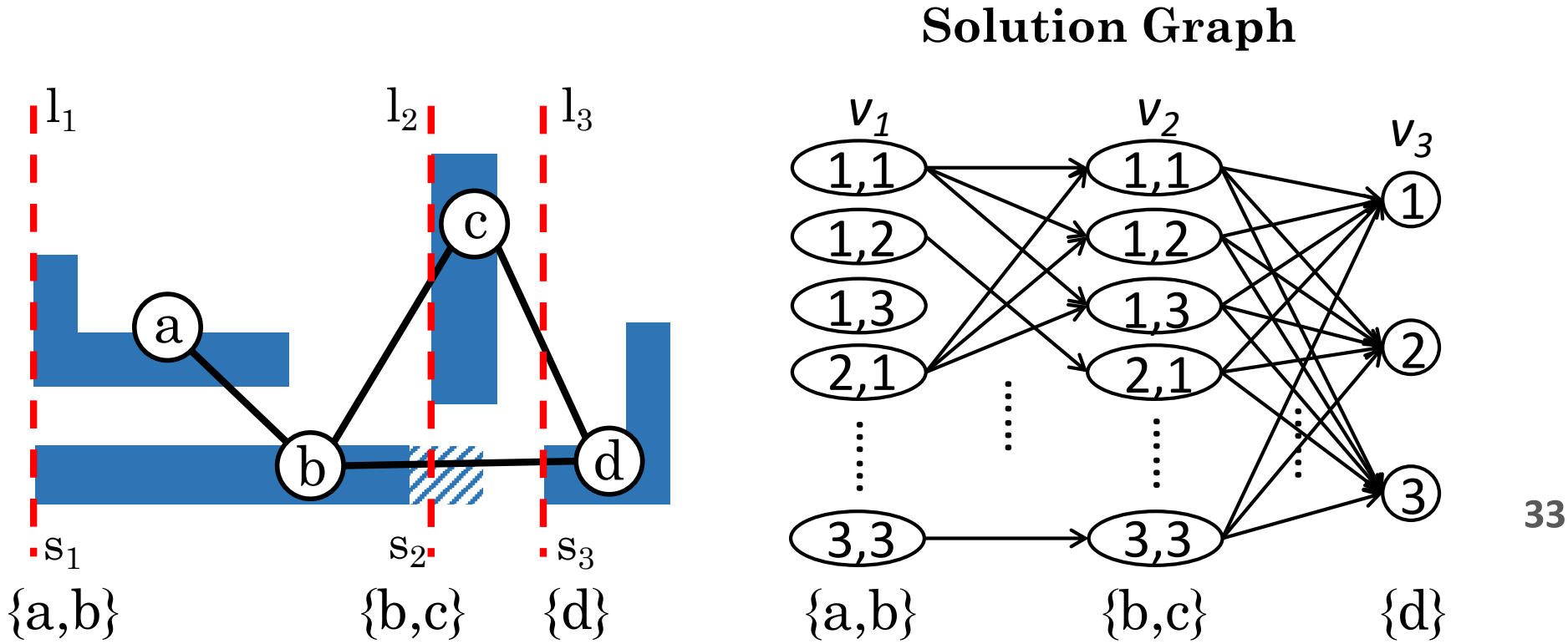
Review of [Tian+ ICCAD12]

- Conflict graph
  - An undirected graph
  - Node: polygon in the given layout
  - Edge: the distance between two corresponding polygons is less than d_{min}
- Cutting line
  - A vertical line aligned with the left boundary of at least one polygon
- Cutting line set
  - A set of polygons intersecting with the corresponding cutting line



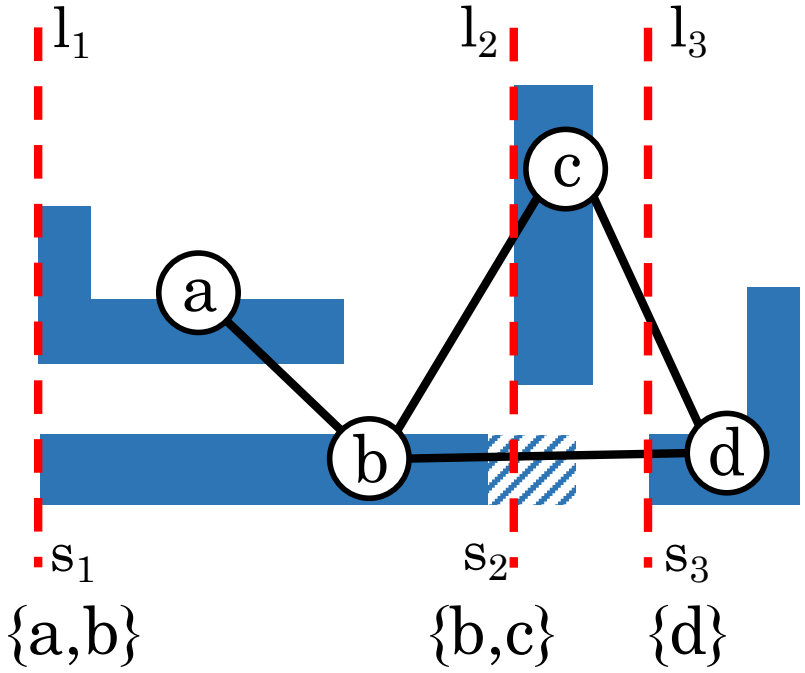
New Graph Model (1/4)

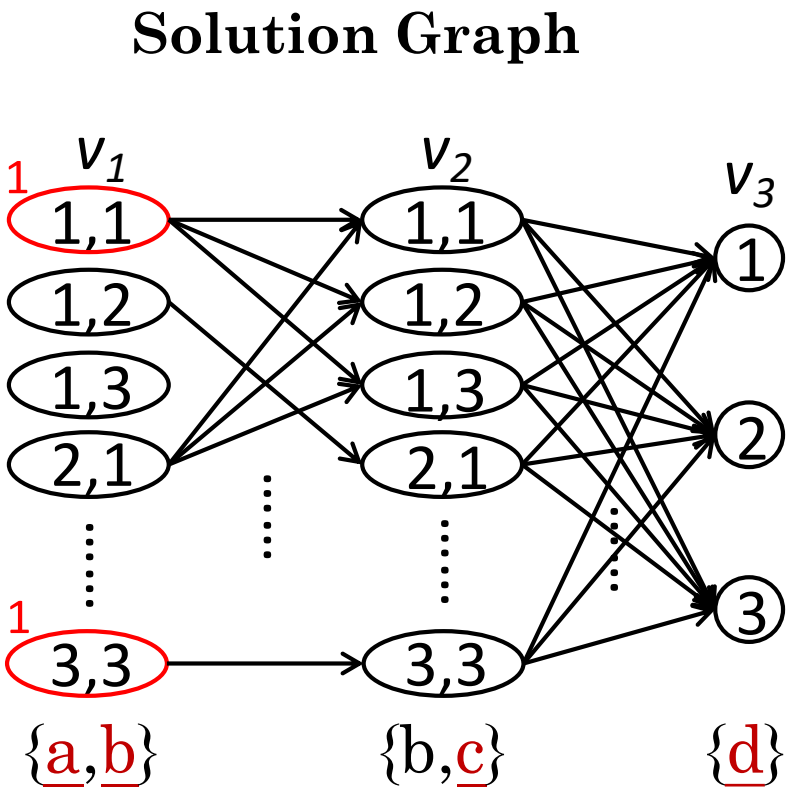
- Solution Graph (SG): directed graph
- Each node records a coloring solution of all the polygons in a cutting line set
- All possible coloring solutions are enumerated



New Graph Model (2/4)

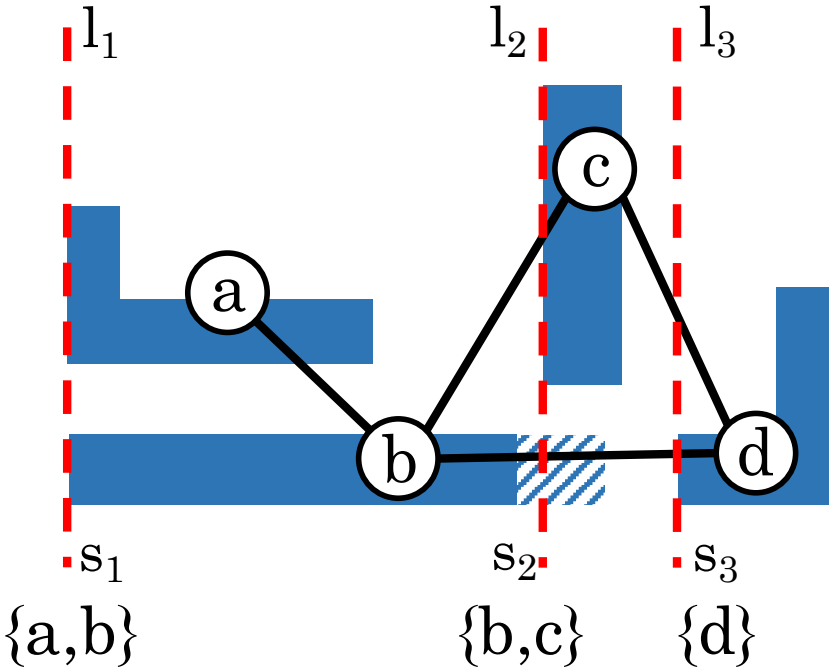
- Let s_i* be the subset of polygons in s_i that create the cutting line for set s_i
- Weight on a node
  - #conflicts between polygons within s_i*

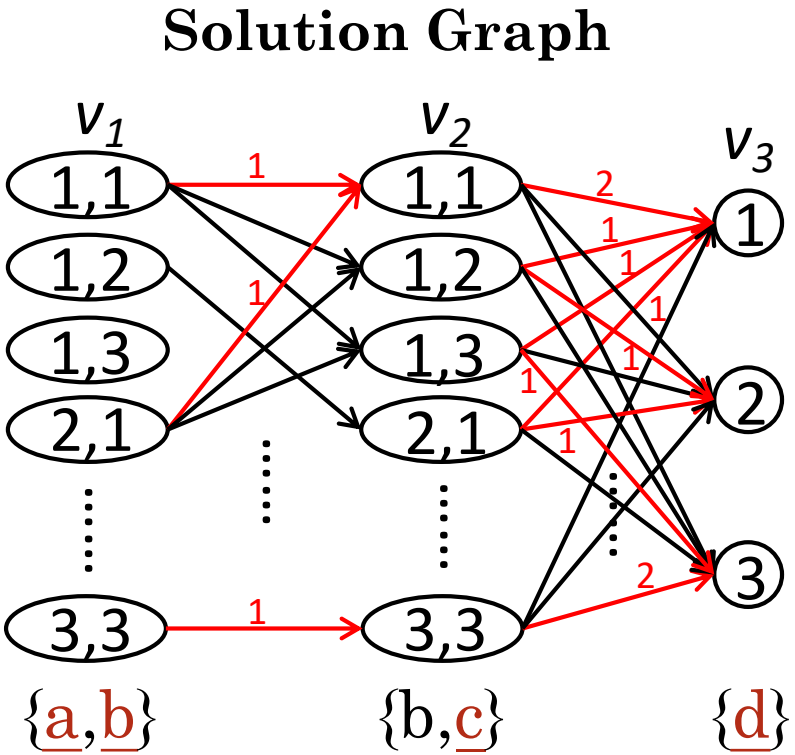




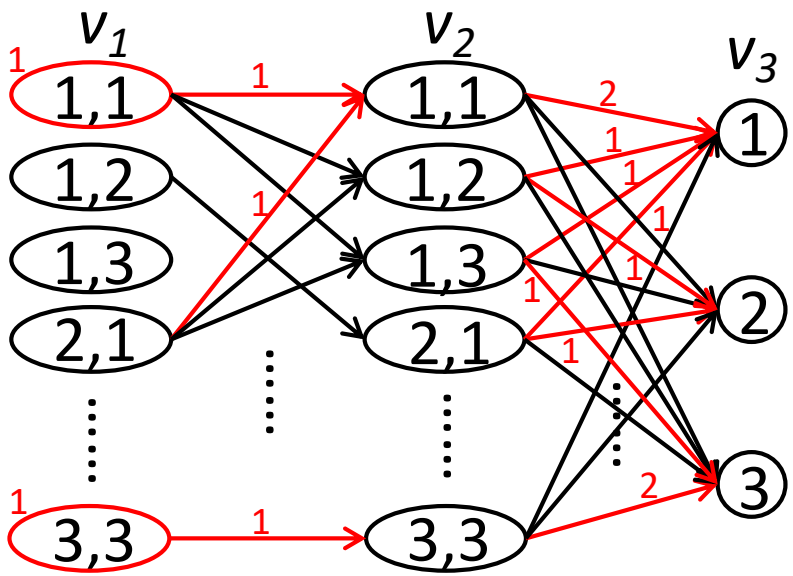
New Graph Model (3/4)

- Weight on a node
  - #conflicts between polygons within s_i*
- Weight on an edge
  - #conflicts between polygons p and q s.t. $p \in s_{i-1}$ and $q \in s_i$*





New Graph Model (4/4)



- **Lemma 1**

  Each possible decomposition of the layout *P* without stitch insertion corresponds to a path in the SG of *P*

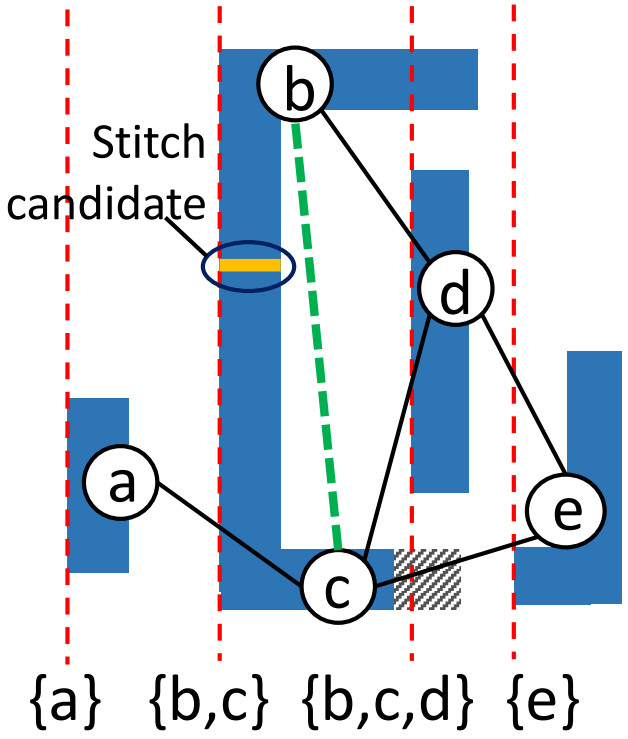
- **Lemma 2**

  The TPL layout decomposition problem without stitch insertion for *P* can be optimally solved by finding a least-cost path in the SG of *P*

Stitch Insertion

- Given a layout with a set of stitch candidates
  - Stitch edge in conflict graph
- **Lemma 3**

  The TPL layout decomposition problem for *P* with a given set of stitch candidates can be optimally solved by finding a least-cost path in the SG of the fractured polygons of *P*



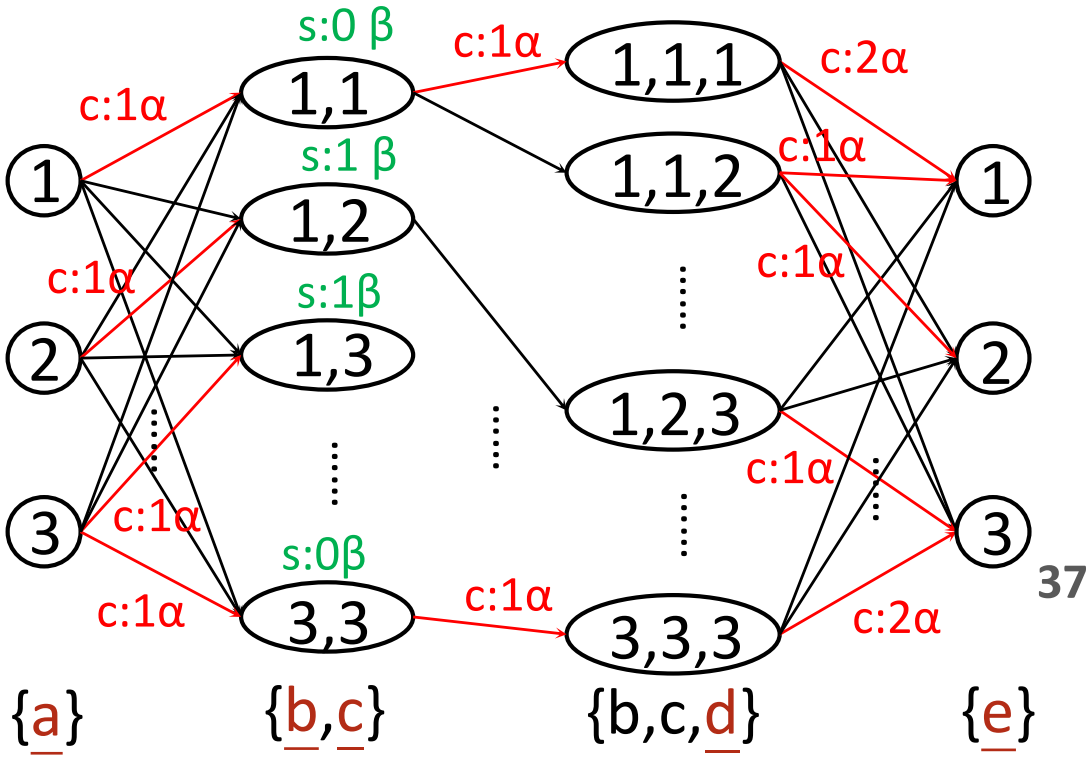
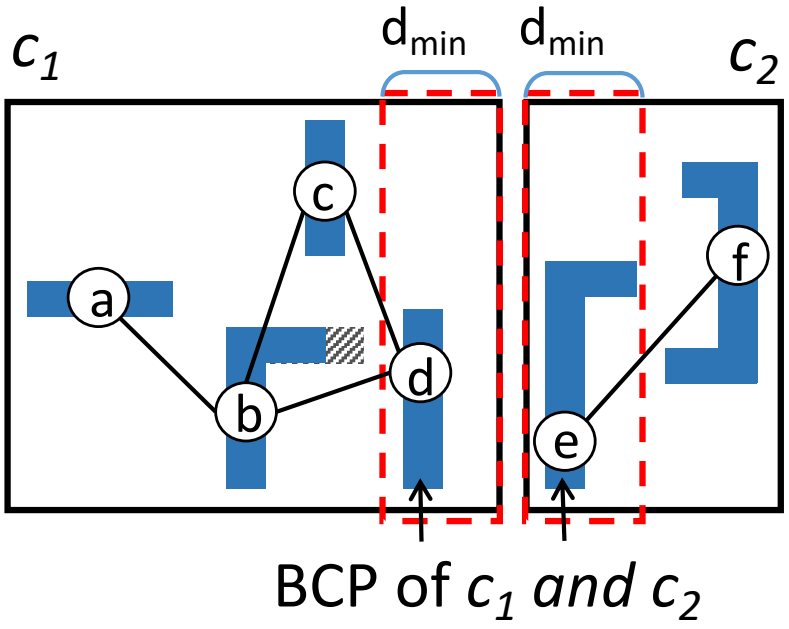
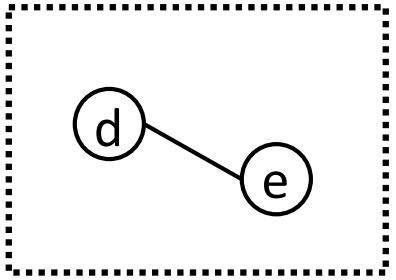


Table Look-up (1/2)

- Speed-up
  - Store the SG for each cell in the cell library
  - Copy the SG from the table for each cell instance
  - Conflicts between two adjacent cells
    - Pre-compute Boundary Conflicting Polygon set (BCP)



BCP of c_1 *and* c_2

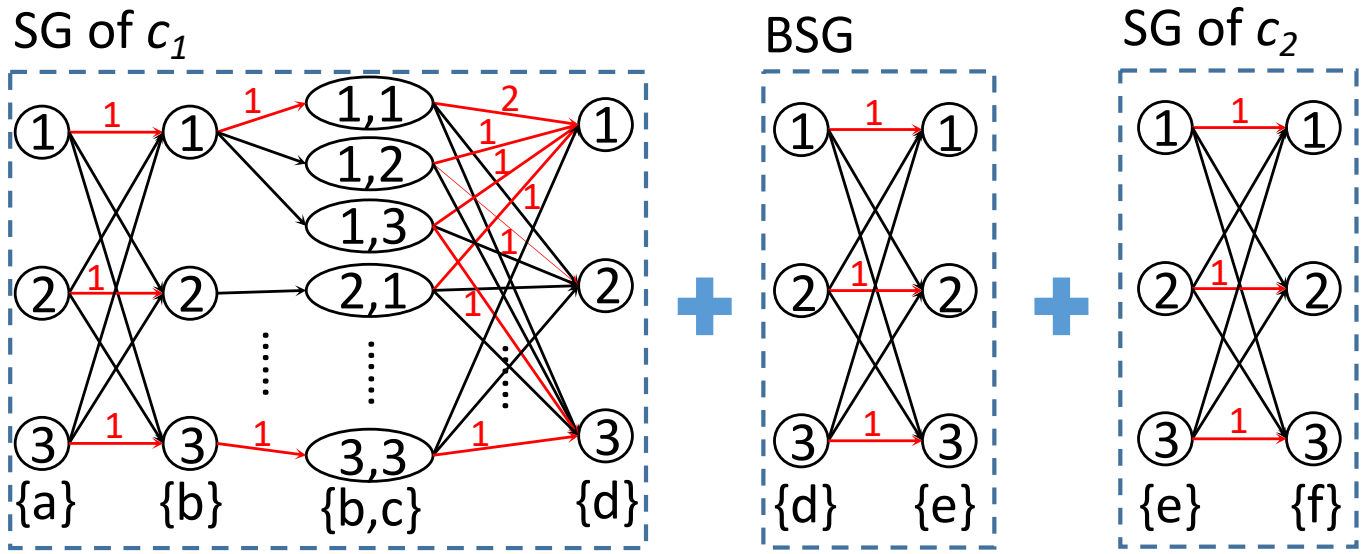


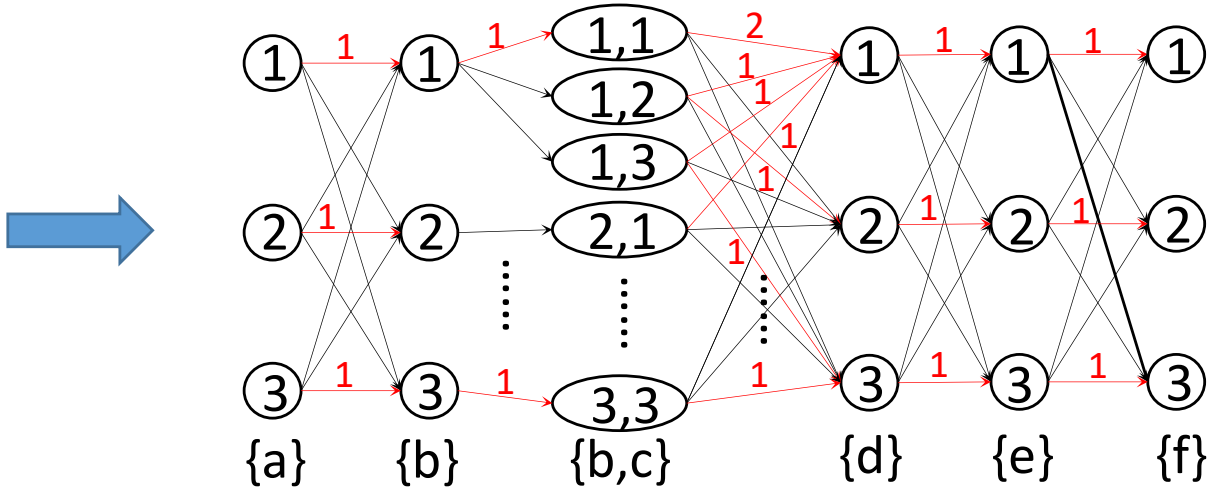
Conflict graph of the BCP

Table Look-up (2/2)

- Final solution graph

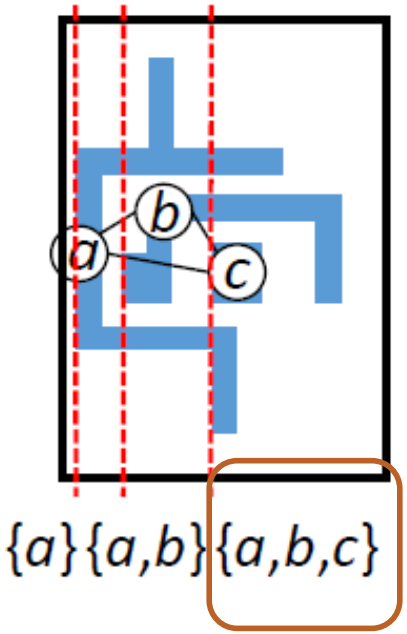
  Combine the SGs of the two adjacent cells and the SG of their BCP





Drawback of Solution Graph

- Too many nodes and edges
- Graph size ⬆ Performance ⬇



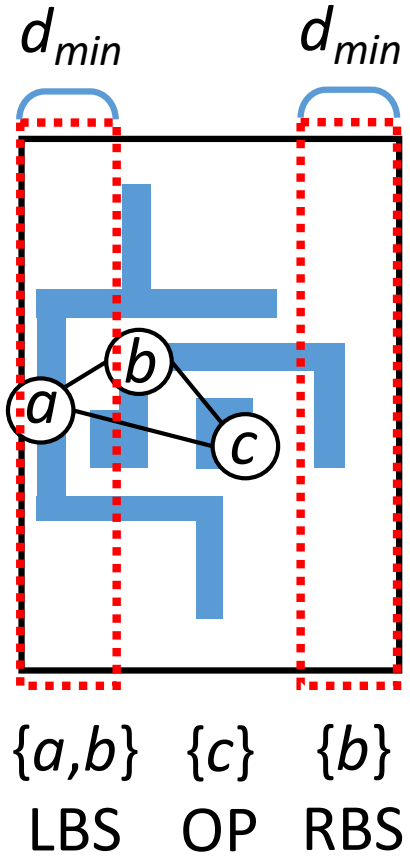
#nodes = 27
#edges = 0

Speed-up

- Several techniques to reduce the size of solution graph
  - Simple Solution Graph
  - Reduced Simple Solution Graph
  - Reduced Simple Solution Graph for BCP
- Acceleration without loss of decomposition quality

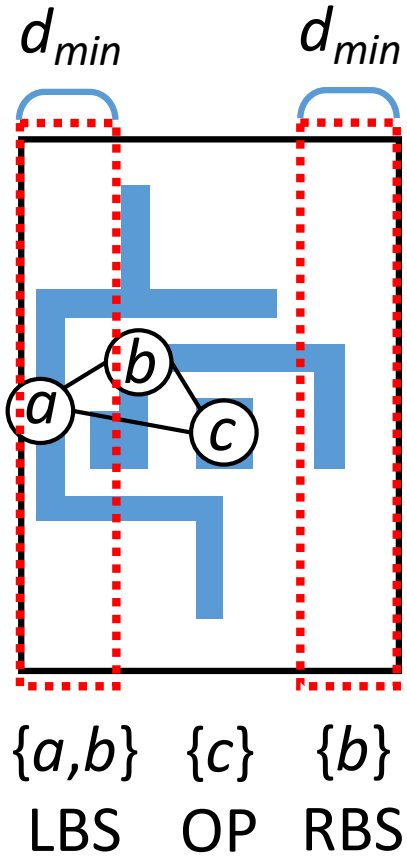
Observation

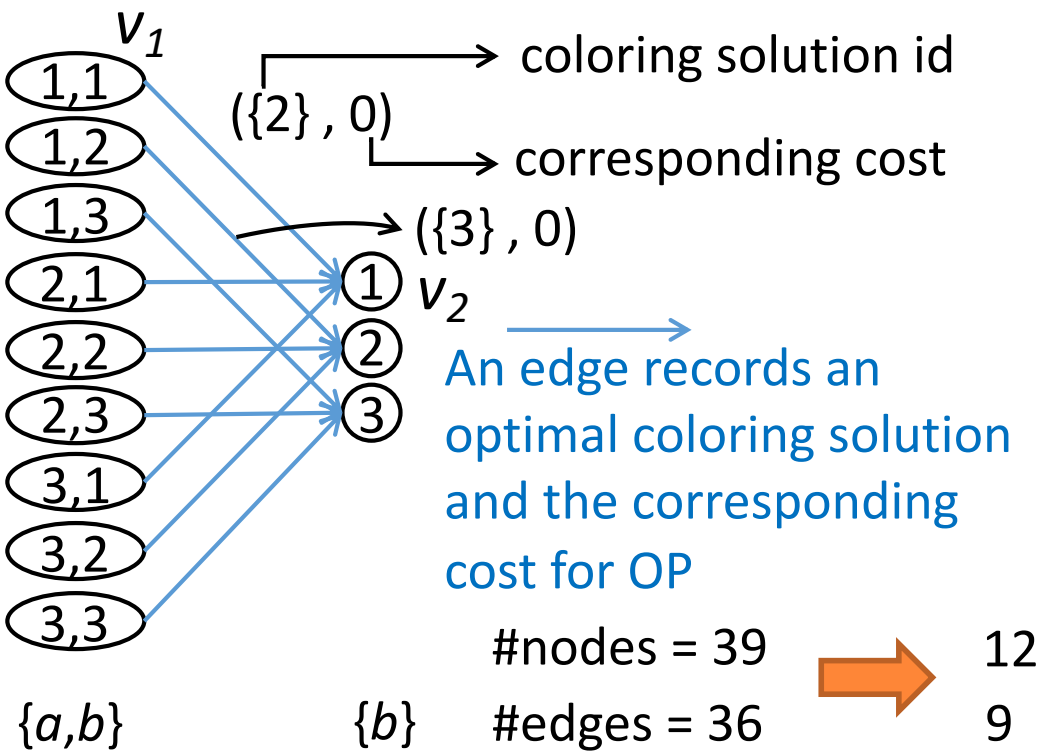
- Let P_{c_i} be the set of polygons in the M1 layer of a cell c_i
- The left boundary set (LBS)
  - LBS_{c_i}
- Right boundary set (RBS)
  - RBS_{c_i}
- Other polygons (OP)
  - $OP_{c_i} = P_{c_i} \setminus (LBS_{c_i} \cup RBS_{c_i})$



Simple Solution Graph

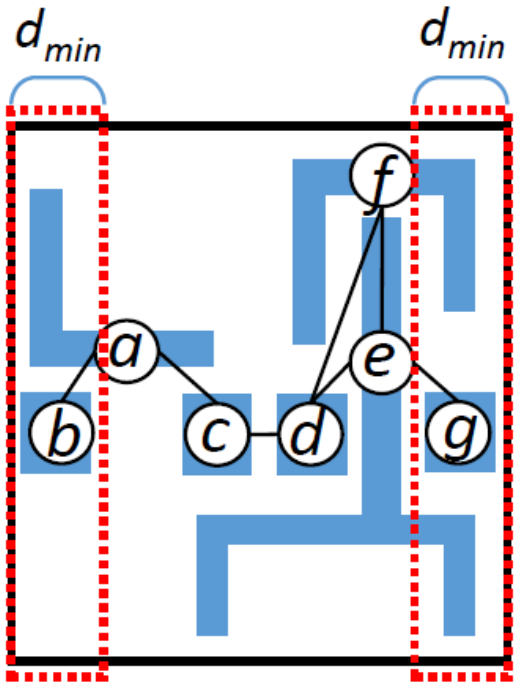
- The simple solution graph of P_{c_i} is defined as a solution graph for $LBS_{c_i} \cup RBS_{c_i}$
  - An edge connecting from a node of LBS_{c_i} to a node of RBS_{c_i}
  - An optimal coloring solution of OP_{c_i} and the corresponding cost



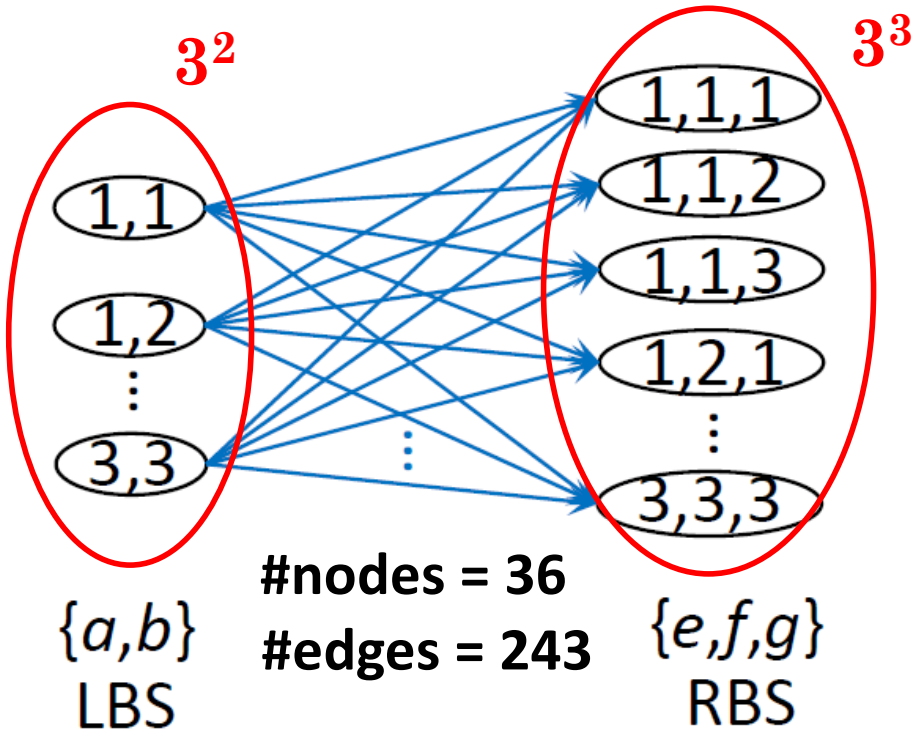


Reduced Simple Solution Graph (1/5)

- If the LBS has *n* polygons and the RBS has *m* polygons, the number of the edges between them could be up to 3^{m+n}



Conflict graph



Simple solution graph

Reduced Simple Solution Graph (2/5)

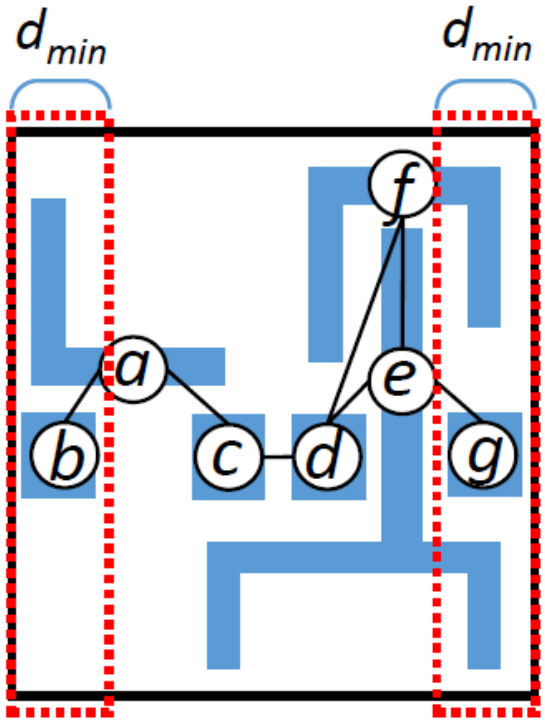
- To reduce the number of the edges
  - *Dummy LBS* (DLBS) for the LBS
  - *Dummy RBS* (DRBS) for the RBS

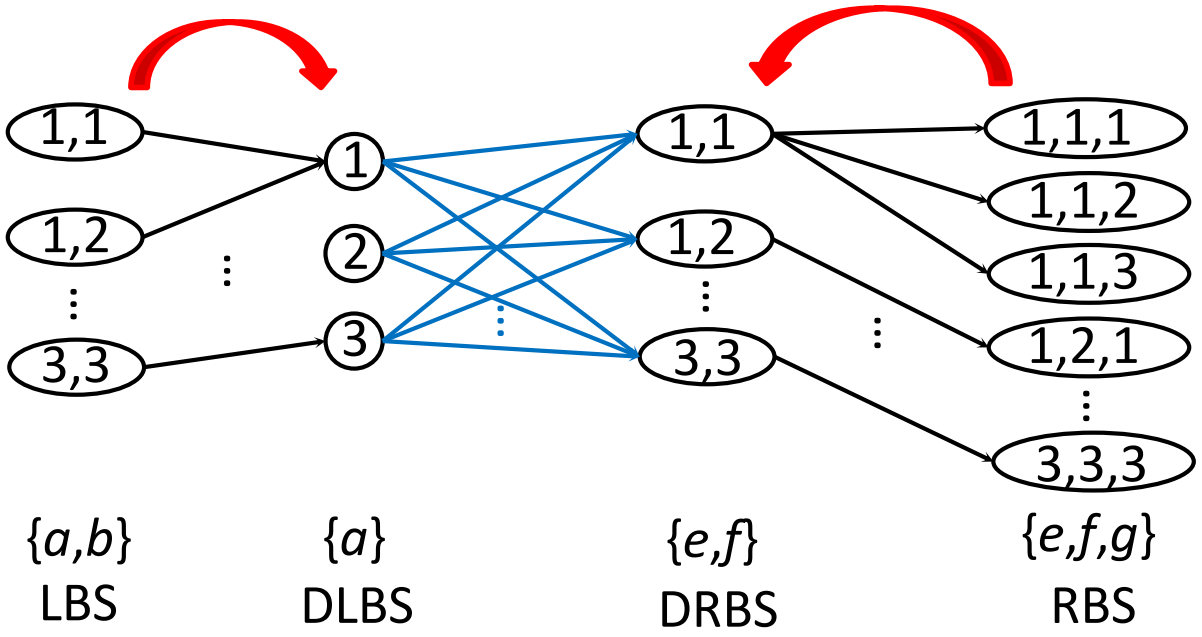
- The DLBS and DRBS are two subsets of LBS and RBS, respectively
- Each polygon in the DLBS (DRBS) might cause a coloring conflict with at least a polygon in R*BS* (LBS) or OP

Reduced Simple Solution Graph (3/5)

Scenario 1: DLBS ⊂ LBS and DRBS ⊂ RBS, where both the DLBS and DRBS are not empty sets

➡ Both the DLBS and the DRBS are added into the graph





#nodes = 36
#edges = 243

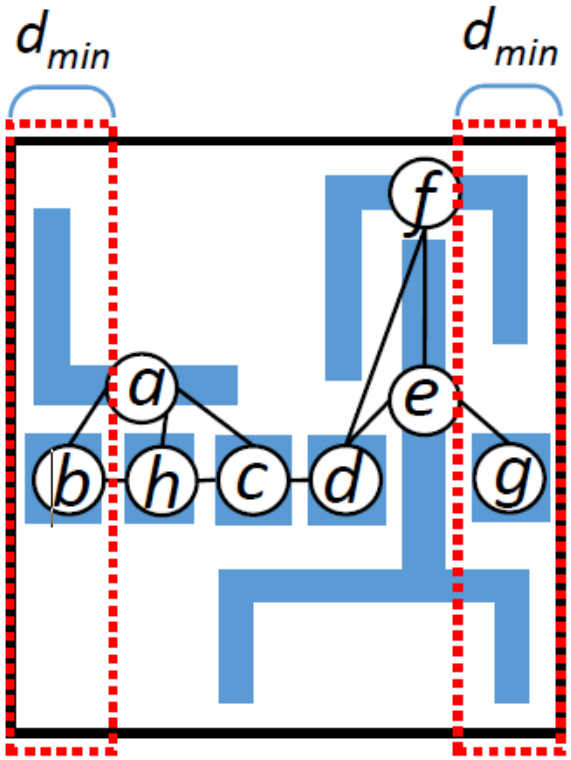
➡

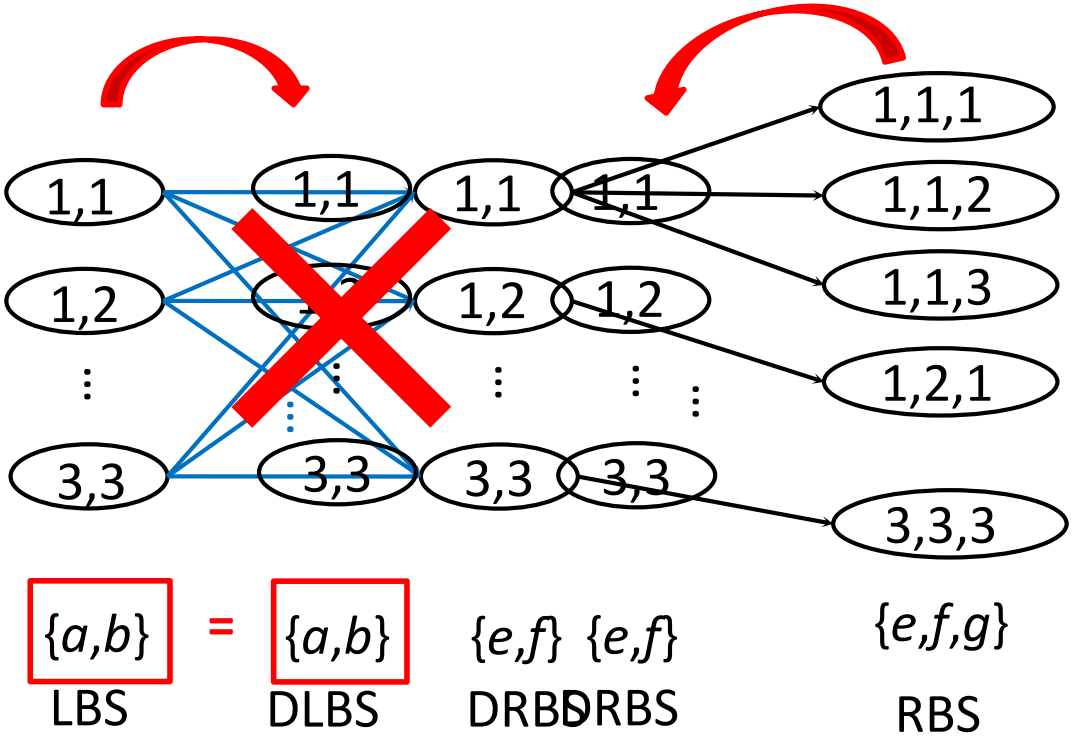
#nodes = 48
#edges = 63

Reduced Simple Solution Graph (4/5)

Scenario 2: DLBS = LBS or DRBS = RBS, where both the DLBS and DRBS are not empty sets

➡ Do not create the DLBS or the DRBS





#nodes = 36
#edges = 243

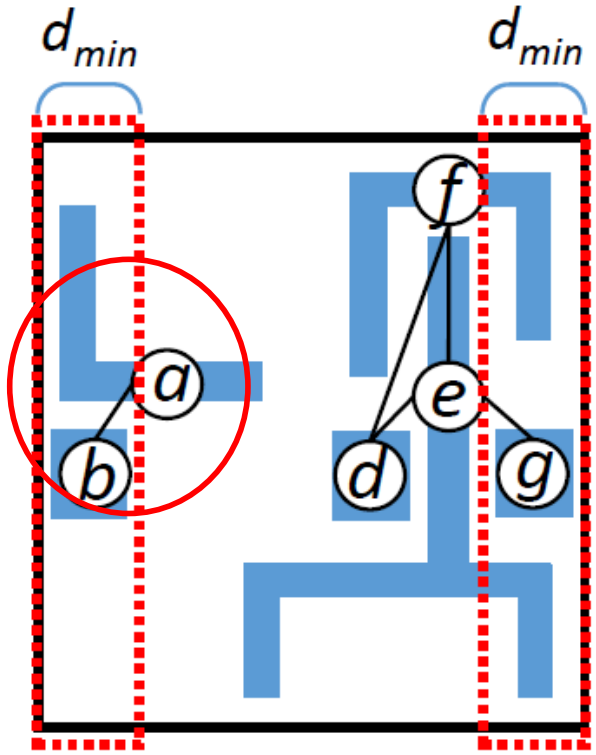
➡

#nodes = 45
#edges = 108

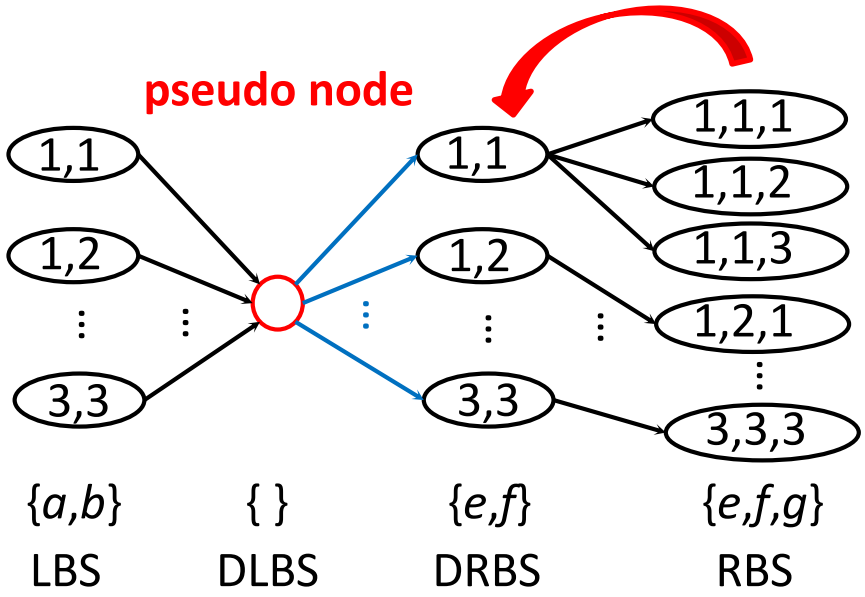
Reduced Simple Solution Graph (5/5)

Scenario 3: DLBS and/or DRBS is an empty set

➡ A pseudo node with weight of 0 is created for the DLBS (DRBS), if DLBS (DRBS) is an empty set



No conflict with OP or RBS



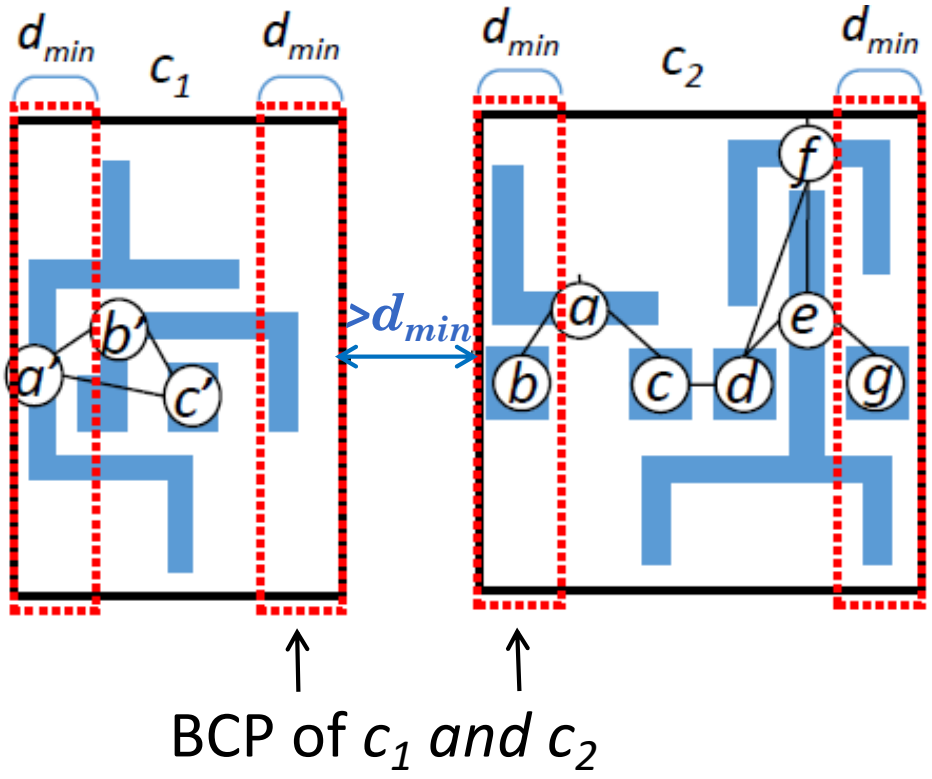
#nodes = 36
#edges = 243

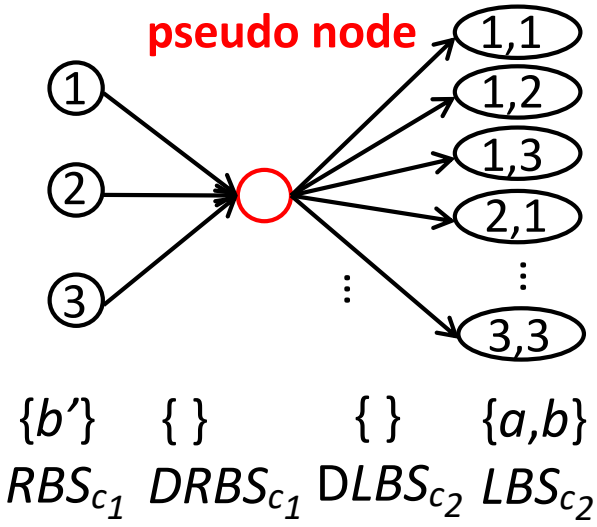
➡

#nodes = 46
#edges = 45

Reduced Simple Solution Graph for BCP

- As we replace the solution graph for a cell by its reduced simple solution graph, we also replace the solution graph for a BCP with its reduced simple solution graph





#nodes = 12
#edges = 27

#nodes = 13
#edges = 12

Overall Approach

- Given a cell library and a set of stitch candidates for each cell
  - Off-line build a look-up table
    - Reduced simple solution graph
    - Each cell type
    - The BCP of each cell pair
  - For each row
    - Construct solution graph
    - Find a least-cost path
  - Simultaneously solve the TPL layout decomposition problem for each row

Experimental Results

- Workstation with 2.0 GHz Intel Xeon CPU and 96 GB memory
- Compare with two state-of-art TPL decomposers
  - Decomposer-A

    B. Yu, Y.-H. Lin, G. Luk-Pat, D. Ding, K. Lucas, and D. Z. Pan, "A high-performance triple patterning layout decomposer with balanced density", in *ICCAD*, 2013.
  - Decomposer-B

    H. Tian, H. Zhang, Q. Ma, Z. Xiao, and M. D. F. Wong, "A polynomial time triple patterning algorithm for cell based row-structure layout", in *ICCAD*, 2012.
- The stitch candidates*

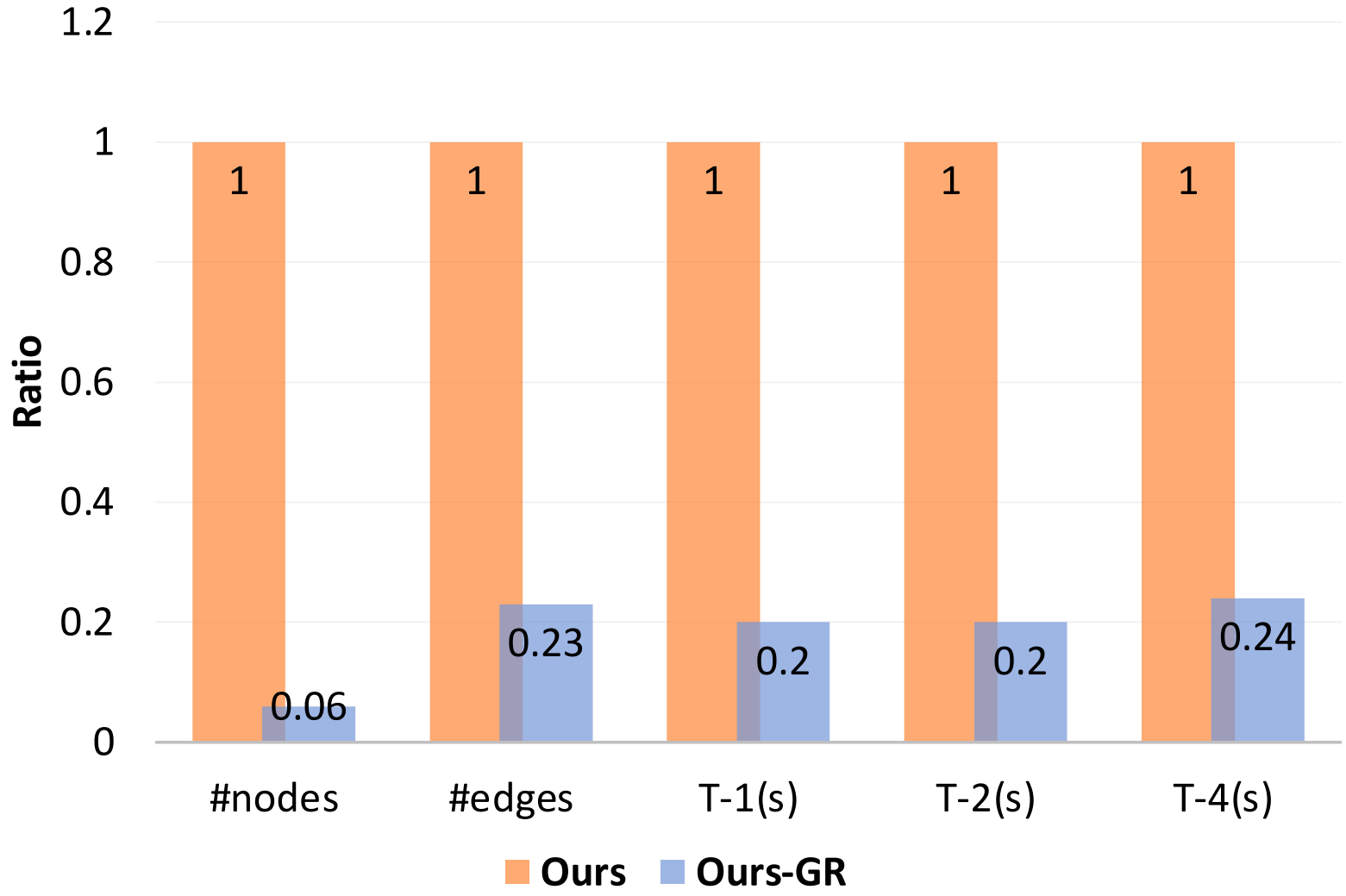
*J. Kuang and E. F. Y. Young, "An efficient layout decomposition approach for triple patterning lithography," in *DAC*, 2013.

Comparison with Decomposer-A

Test Case	Decomposer-A				Ours-GR			
	Cost	#C	#S	T-1	Cost	#C	#S	T-4
alu-70	408.1	336	721	213	286.5	185	1015	4
alu-80	N/A	N/A	N/A	>18000	474.4	353	1214	8
alu-90	N/A	N/A	N/A	>18000	520.6	398	1226	5
byp-70	656.9	502	1549	830	636.0	437	1990	10
byp-80	1076.5	897	1795	2544	831.2	612	2192	14
byp-90	N/A	N/A	N/A	>18000	1012.4	780	2324	22
div-70	726.2	568	1582	913	620.7	424	1967	12
div-80	703.7	545	1587	1182	578.4	383	1954	11
div-90	656.9	502	1549	830	528.9	338	1909	11
ecc-70	N/A	N/A	N/A	>18000	134.6	89	456	3
ecc-80	207.3	159	483	485	197.5	140	575	4
ecc-90	N/A	N/A	N/A	>18000	275.4	214	614	6
efc-70	N/A	N/A	N/A	>18000	218.4	143	754	3
efc-80	N/A	N/A	N/A	>18000	247.1	172	751	4
efc-90	302.8	241	618	202	209.0	134	750	3
ctl-70	248.5	215	335	970	209.3	170	393	3
ctl-80	N/A	N/A	N/A	>18000	212.6	167	456	4
ctl-90	187.5	150	375	537	152.4	111	414	4
top-70	N/A	N/A	N/A	>18000	2320.3	1754	5663	15
top-80	3221.1	2774	4471	3807	2702.0	2118	5840	31
top-90	3506.0	3002	5040	9305	3233.9	2618	6159	36

Test case: OpenSPARC T1 Designs

Benefits of Graph Reduction



Test case: OpenSPARC T1 Designs **T-i: runtime with i threads**

Comparisons with Decomposer-A&-B

Test Case	Decomposer-B			Decomposer-A			Ours		
	Cost	#C	#S	Cost	#C	#S	Cost	#C	#S
C432	N/A	N/A	N/A	0.4	0	4	0.4	0	4
C499	0	0	0	0	0	0	0	0	0
C880	0.7	0	7	0.7	0	7	0.7	0	7
C1355	0.3	0	3	0.3	0	3	0.3	0	3
C1908	0.1	0	1	0.1	0	1	0.1	0	1
C2670	0.6	0	6	0.6	0	6	0.6	0	6
C3540	N/A	N/A	N/A	1.9	1	9	1.8	1	8
C5315	N/A	N/A	N/A	0.9	0	9	0.9	0	9
C6288	N/A	N/A	N/A	22.2	1	212	20.5	0	206
C7552	N/A	N/A	N/A	2.6	0	26	2.3	0	22
S1488	0.2	0	2	0.3	0	3	0.2	0	2
S38417	N/A	N/A	N/A	25.7	20	57	24.4	19	54
S35932	N/A	N/A	N/A	52	46	60	48	44	40
S38584	N/A	N/A	N/A	49.1	36	131	47.6	36	116
S15850	N/A	N/A	N/A	45.9	34	119	44.1	34	98
Ratio				1.06	1.03	1.12	1	1	1

Test case: ISCAS-85&89 circuits

References

- [Tian+ ICCAD12] A polynomial time triple patterning algorithm for cell-based row-structure layout
- [Chien+ ISPD15] A Cell-Based Row-Structure Layout Decomposer for Triple Patterning Lithography