Exploring Adjacency in Floorplanning

Jia Wang
Illinois Institute of Technology
Chicago, Illinois, USA

Hai Zhou
Fudan University, China
Northwestern University, USA

January, 2009

Outline

Floorplanning

- Determine the locations and shapes of modules
  - Various objectives and constraints
- Realize adjacency relations
  - Modules communicating with each other should be close to each other.
  - Relevant algorithms are quite complicated.
- Place modules without overlap
  - Usually optimize floorplans through simulated annealing (SA)
  - Time consuming

Floorplanning

- Determine the locations and shapes of modules
  - Various objectives and constraints
- Realize adjacency relations
  - Modules communicating with each other should be close to each other.
  - Relevant algorithms are quite complicated.
- Place modules without overlap
  - Usually optimize floorplans through simulated annealing (SA)
  - Time consuming

Floorplanning

- Determine the locations and shapes of modules
  - Various objectives and constraints
- Realize adjacency relations
  - Modules communicating with each other should be close to each other.
  - Relevant algorithms are quite complicated.
- Place modules without overlap
  - Usually optimize floorplans through simulated annealing (SA)
  - Time consuming

Floorplan with Adjacency Graph

[Kozminski et al. 1984], [Bhasker et al. 1988], [Lai et al. 1988]

- Common floorplan flow with adjacency graphs
  - Start with structure graph, most likely derived from interconnects
  - Planarize and properly triangulate in order to transform the graph into an adjacency graph, i.e. one with a rectangular dual
  - Construct the rectangular dual, i.e. a floorplan of rooms, each contains a module
- Not widely used today
  - Although complexities of algorithms are usually $O(n)$, practical implementations are quite complicated (e.g. compared to sequence-pairs)
  - Large whitespace when the shapes of modules are not flexible, e.g. hard modules

Floorplan with Adjacency Graph

[Kozminski et al. 1984], [Bhasker et al. 1988], [Lai et al. 1988]

- Common floorplan flow with adjacency graphs
  - Start with structure graph, most likely derived from interconnects
  - Planarize and properly triangulate in order to transform the graph into an adjacency graph, i.e. one with a rectangular dual
  - Construct the rectangular dual, i.e. a floorplan of rooms, each contains a module
- Not widely used today
  - Although complexities of algorithms are usually $O(n)$, practical implementations are quite complicated (e.g. compared to sequence-pairs)
  - Large whitespace when the shapes of modules are not flexible, e.g. hard modules

Floorplan with Adjacent Relations

- Many floorplan approaches capture adjacent relations partially.
- Mosaic floorplans, e.g. CBL [Hong et al. 2000] and TBS [Young et al. 2003]
  - A floorplan of rooms
  - Allow T-junction to slide – not all adjacent relations are captured
- Adjacent Constraint Graph (ACG) [Zhou et al. 2004] Linear Constraint Graph (LCG) [Wang et al. 2008]
  - Capture adjacent relations by removing redundancies in constraint graphs
  - They are still constraint graphs.

Floorplan with Adjacent Relations

- Many floorplan approaches capture adjacent relations partially.
- Mosaic floorplans, e.g. CBL [Hong et al. 2000] and TBS [Young et al. 2003]
  - A floorplan of rooms
  - Allow T-junction to slide – not all adjacent relations are captured
- Adjacent Constraint Graph (ACG) [Zhou et al. 2004] Linear Constraint Graph (LCG) [Wang et al. 2008]
  - Capture adjacent relations by removing redundancies in constraint graphs
  - They are still constraint graphs.

Floorplan with Adjacent Relations

- Many floorplan approaches capture adjacent relations partially.
- Mosaic floorplans, e.g. CBL [Hong et al. 2000] and TBS [Young et al. 2003]
  - A floorplan of rooms
  - Allow T-junction to slide – not all adjacent relations are captured
- Adjacent Constraint Graph (ACG) [Zhou et al. 2004] Linear Constraint Graph (LCG) [Wang et al. 2008]
  - Capture adjacent relations by removing redundancies in constraint graphs
  - They are still constraint graphs.

Our Contribution: Constrained Adjacency Graph (CAG)

- Propose Constrained Adjacency Graph (CAG) to extend previous adjacency graph approaches by introducing explicit adjacency constraints
- Derive sufficient and necessary conditions for CAG
- Present a linear complexity algorithm to construct floorplans from CAG
- Propose to use CAG for general floorplans through packing
- Present an iterative algorithm for CAG to improve general floorplans in area without changing the adjacency relations dramatically

Outline

Overview of Floorplanning

Constrained Adjacency Graph

Whitespace Reduction

Experiments

Conclusions

Dissected Floorplan and Adjacency Graph

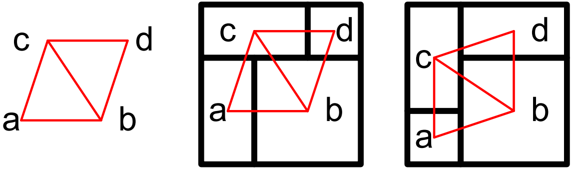
- Dissected floorplan
  - The floorplan area is dissected into rectangular rooms.
  - Each room accommodates a module.
  - No four rooms share a common point.
- Adjacency graph
  - Planar graph. Each face is a triangle.
  - No simple definition – algorithmically defined [Kozminski et al. 1984], [Bhasker et al. 1988]

Dissected Floorplan and Adjacency Graph

- Dissected floorplan
  - The floorplan area is dissected into rectangular rooms.
  - Each room accommodates a module.
  - No four rooms share a common point.
- Adjacency graph
  - Planar graph. Each face is a triangle.
  - No simple definition – algorithmically defined [Kozminski et al. 1984], [Bhasker et al. 1988]

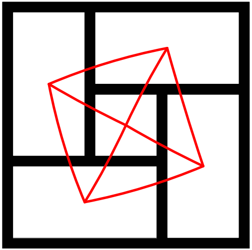
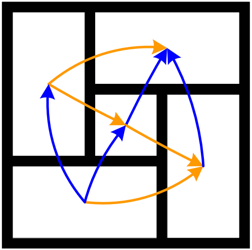
Dissected Floorplan and Adjacency Graph

- Dissected floorplan
  - The floorplan area is dissected into rectangular rooms.
  - Each room accommodates a module.
  - No four rooms share a common point.
- Adjacency graph
  - Planar graph. Each face is a triangle.
  - No simple definition – algorithmically defined [Kozminski et al. 1984], [Bhasker et al. 1988]
- One adjacency graph may correspond to multiple dissected floorplans.

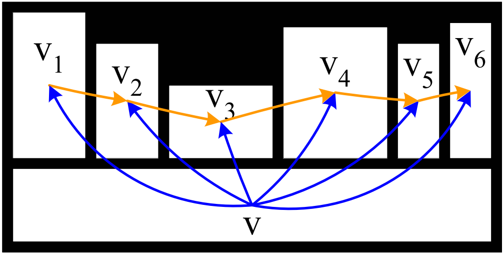


Constrained Adjacency Graph (CAG)

- Constrained Adjacency Graph
  - Introduce explicit adjacency constraints to edges
- Directed vertical edge
  - Between two rooms sharing horizontal border
  - From bottom to top
- Directed horizontal edge
  - Between two rooms sharing vertical border
  - From left to right

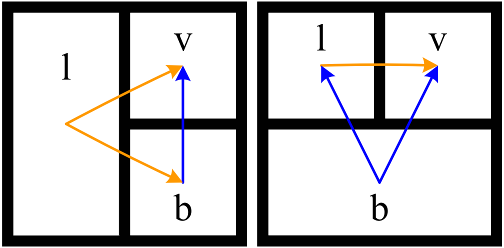



Neighbor Condition for CAG



- v_i's are the rooms to the top of v.
- v_i is the bottom-most left neighbor of v_{i+1}.
- v_{i+1} is the bottom-most right neighbor of v_i.
- Apply to other borders of v.

Corner Condition for CAG



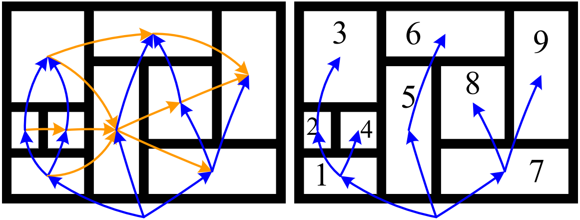
- v's bottom-left corner is not on the boundary of the floorplan iff it has both bottom and left neighbors.
  - b: left-most bottom neighbor of v
  - t: bottom-most left neighbor of v
- l is the top-most left neighbor of b, or
- b is the right-most bottom neighbor of l.
- Apply to other corners.

Sufficient and Necessary Conditions for CAG

- Necessary conditions for a graph to be a CAG
  - Neighbor and Corner conditions must hold for all vertices.
  - They are necessary conditions.
- Sufficient condition for a graph to be a CAG
  - Neighbor and Corner conditions are sufficient.
  - Prove by constructing a dissected floorplan from a graph when the conditions hold

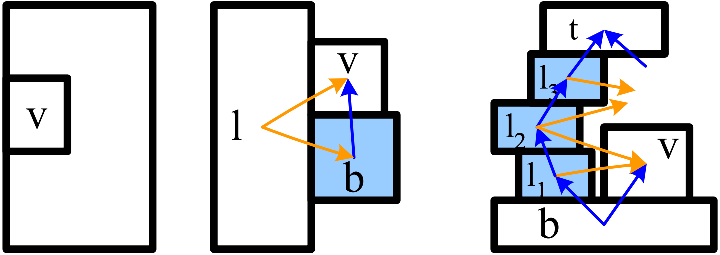
Construct Dissected Floorplan from CAG

- Compute horizontal and vertical positions for rooms separately
- For example, compute horizontal positions
  - Intuition: when deciding the horizontal position for a room, the horizontal positions of all its left neighbors should be computed already
  - Perform depth-first-search (DFS) on vertical edges, visiting neighbors from left to right
  - Sort the vertices according to their DFS discovery time



Construct Dissected Floorplan from CAG (Cont.)

- Compute horizontal position for each room following the order



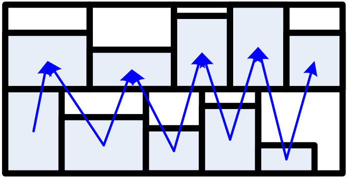
v.left = 0   v.left = b.left   v.left = max l_i.left+l_i.width

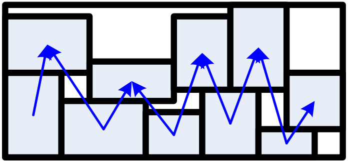
- Overall time complexity and space complexity: both $O(n)$

Outline

Packing of Dissected Floorplans

- Rooms could be much larger than the modules because of the required adjacencies.
- If two rooms occupy a same horizontal interval, there is a vertical path in CAG separating them.





  - Fix horizontal positions
  - Use vertical edges to pack, i.e. as constraint relations
  - Adjacent relations are kept.
  - Called V-packing, similarly there is H-packing
- CAG are extended to handle general floorplans.

Packed Dissected Floorplan

- The floorplan after V-packing may have large whitespace along the horizontal direction
- Intuition: a dissected floorplan has small whitespace if H-packing won't change it
  - Therefore, after V-packing, the floorplan has little whitespace
- H-packed dissected floorplan
  - Will not change after H-packing
  - Similarly there are V-packed dissected floorplans.
  - How to obtain?

Packed Dissected Floorplan

- The floorplan after V-packing may have large whitespace along the horizontal direction
- Intuition: a dissected floorplan has small whitespace if H-packing won't change it
  - Therefore, after V-packing, the floorplan has little whitespace
- H-packed dissected floorplan
  - Will not change after H-packing
  - Similarly there are V-packed dissected floorplans.
  - How to obtain?

Packed Dissected Floorplan

- The floorplan after V-packing may have large whitespace along the horizontal direction
- Intuition: a dissected floorplan has small whitespace if H-packing won't change it
  - Therefore, after V-packing, the floorplan has little whitespace
- H-packed dissected floorplan
  - Will not change after H-packing
  - Similarly there are V-packed dissected floorplans.
  - How to obtain?

The Tree-Weaving Algorithm

- H-packing will result in a longest path tree that determines horizontal positions of the modules
- H-Tree-Weaving algorithm
  - Given the tree, construct a CAG
  - In the corresponding dissected floorplan, rooms have the same horizontal positions as modules in the H-packing
  - Similarly there is the V-Tree-Weaving algorithm.
- Maintain neighbor and corner conditions
  - See paper for details

The Tree-Weaving Algorithm

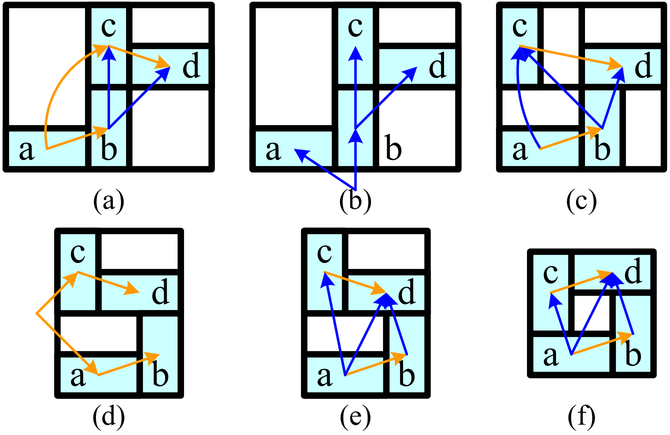
- H-packing will result in a longest path tree that determines horizontal positions of the modules
- H-Tree-Weaving algorithm
  - Given the tree, construct a CAG
  - In the corresponding dissected floorplan, rooms have the same horizontal positions as modules in the H-packing
  - Similarly there is the V-Tree-Weaving algorithm.
- Maintain neighbor and corner conditions
  - See paper for details

The Tree-Weaving Algorithm

- H-packing will result in a longest path tree that determines horizontal positions of the modules
- H-Tree-Weaving algorithm
  - Given the tree, construct a CAG
  - In the corresponding dissected floorplan, rooms have the same horizontal positions as modules in the H-packing
  - Similarly there is the V-Tree-Weaving algorithm.
- Maintain neighbor and corner conditions
  - See paper for details

Iterative Packing

- Apply H-Tree-Weaving and V-Tree-Weaving alternatively
- Reduce whitespace without changing adjacent relations dramatically



Outline

CAG Floorplanning for Interconnects

- Greedy iterative floorplanning
- Initial floorplan
  - Order modules horizontally by solving one quadratic programming problem similar to analytical placement
  - Place modules column by column from left to right
  - Ignore vertical order
- Each iteration
  - Randomly pick up two modules for swapping
  - Perform iterative packing
  - Accept the swapping if cost function is improved
- Stop after a predefined number of iterations or rejections

Experimental Setup

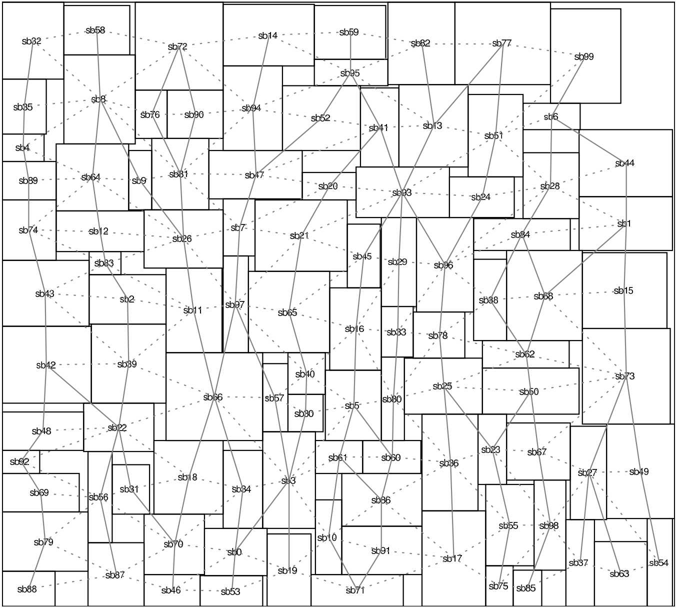
- GSRC benchmarks: n100, n200, n300
- Compare to Parquet [Adya et al. 2003]
  - Sequence-pair based SA floorplan optimization
  - Free-outline mode
  - Start with a quadratic programming solution
  - No module rotations
- Cost function: HPWL+area

Experimental Results

name	method	area	HPWL	time(s)	#moves
n100	CAG	195.9K/204.5K	302.1K/312.8K	15.30	31.3K
	Parquet A	196.8K/206.0K	320.2K/342.9K	14.90	96.5K
	Parquet B	195.0K/203.5K	313.6K/338.5K	29.80	175.8K
n200	CAG	197.0K/205.4K	540.9K/553.3K	30.86	26.0K
	Parquet A	207.4K/218.2K	613.8K/647.9K	29.40	54.0K
	Parquet B	197.4K/202.5K	578.9K/624.5K	149.2	256.6K
n300	CAG	304.4K/315.6K	649.0K/665.8K	61.61	33.8K
	Parquet A	335.2K/351.0K	750.6K/800.0K	58.89	62.5K
	Parquet B	306.9K/314.6K	709.2K/757.3K	290.6	325.7K

- Minimum/Maximum among 10 runs
- Parquet A: similar running time as CAG
- Parquet B: similar solution quality as recent Parquet results [Chan et al. 2005]

Floorplan Result for n100



Outline

Conclusions

- *Constrained Adjacency Graph*s (CAG) are proposed to extend previous adjacency graph approaches.
  - By introducing explicit adjacency constraints
  - Sufficient and necessary conditions are derived.
  - Linear complexity algorithm to construct floorplans from CAG is presented
- Handle general floorplans through packing.
  - Improve area without changing the adjacency relations dramatically through iterative packing
- Possible to combine with Linear Constraint Graph [Wang et al. 2008] to obtain a versatile floorplan representation that would provide both adjacent and constraint relations.

Q & A

Thank you!