Linear Constraint Graph for Floorplan Optimization with Soft Blocks

Jia Wang
Dept. of ECE
Illinois Institute of Technology
Chicago, Illinois, United States

Hai Zhou
Dept. of EECS
Northwestern University
Evanston, Illinois, United States

November, 2008
Outline

Overview of Floorplanning

Linear Constraint Graph

The LCG Floorplanner

Experimental Results

Conclusions
Floorplanning

- Determine the locations and shapes of modules
  - Various objectives: area, interconnect, voltage island, etc.
  - Various constraints: soft blocks, abutment, etc.

- Constructive approaches: fast
  - Usually limited to area and interconnect optimization

- Simulated annealing (SA): flexible
  - Use a floorplan representation to explore different floorplans
  - Manage objectives through a cost function
Floorplanning

- Determine the locations and shapes of modules
  - Various objectives: area, interconnect, voltage island, etc.
  - Various constraints: soft blocks, abutment, etc.
- Constructive approaches: fast
  - Usually limited to area and interconnect optimization
- Simulated annealing (SA): flexible
  - Use a floorplan representation to explore different floorplans
  - Manage objectives through a cost function
Floorplanning

- Determine the locations and shapes of modules
  - Various objectives: area, interconnect, voltage island, etc.
  - Various constraints: soft blocks, abutment, etc.

- Constructive approaches: fast
  - Usually limited to area and interconnect optimization

- Simulated annealing (SA): flexible
  - Use a floorplan representation to explore different floorplans
  - Manage objectives through a cost function
Floorplan Exploration

To explore all non-overlapping floorplans. How?

- Generate a floorplan topology (in some representation) in SA
- Map from the floorplan topology to physical floorplans

Packing: area minimized physical floorplans only, when module sizes are known

Constraint graph + mathematical programming: all possible physical floorplans

- Soft blocks [Young et al. 2001], [Lin et al. 2006], [Lee et al. 2007]
- Placement constraints [Young et al. 2004]
- Wire length [Tang et al. 2006]
- Complexity of constraint graphs matters
Floorplan Exploration

- To explore all non-overlapping floorplans. How?
  - Generate a floorplan topology (in some representation) in SA
  - Map from the floorplan topology to physical floorplans
- Packing: area minimized physical floorplans only, when module sizes are known
  - Constraint graph + mathematical programming: all possible physical floorplans
    - Soft blocks [Young et al. 2001], [Lin et al. 2006], [Lee et al. 2007]
    - Placement constraints [Young et al. 2004]
    - Wire length [Tang et al. 2006]
    - Complexity of constraint graphs matters
Floorplan Exploration

- To explore all non-overlapping floorplans. How?
  - Generate a floorplan topology (in some representation) in SA
  - Map from the floorplan topology to physical floorplans

- Packing: area minimized physical floorplans only, when module sizes are known

- Constraint graph + mathematical programming: all possible physical floorplans
  - Soft blocks [Young et al. 2001], [Lin et al. 2006], [Lee et al. 2007]
  - Placement constraints [Young et al. 2004]
  - Wire length [Tang et al. 2006]
  - Complexity of constraint graphs matters
To explore all non-overlapping floorplans. How?
- Generate a floorplan topology (in some representation) in SA
- Map from the floorplan topology to physical floorplans

Packing: area minimized physical floorplans only, when module sizes are known

Constraint graph + mathematical programming: all possible physical floorplans
- Soft blocks [Young et al. 2001], [Lin et al. 2006], [Lee et al. 2007]
- Placement constraints [Young et al. 2004]
- Wire length [Tang et al. 2006]
- Complexity of constraint graphs matters
Constraint Graph Basics

- Horizontal graph: *left-to* relations
- Vertical graph: *below* relations
- At least one relation between any pair of modules
Constraint Graph Basics

- Horizontal graph: *left-to* relations
- Vertical graph: *below* relations
- At least one relation between any pair of modules
Constraint Graph Basics

- Horizontal graph: *left-to* relations
- Vertical graph: *below* relations
- At least one relation between any pair of modules
Redundancy in Constraint Graph

- Transitive edges: relations implied by others
- Over-specification: more than one relation between two modules
Redundancy in Constraint Graph

- Transitive edges: relations implied by others
- Over-specification: more than one relation between two modules
Ad-Hoc Approaches for Constraint Graph Generation

- From polar graphs ([Ohtsuki et al. 1970])
  - There is no method to explore them in SA
  - Apply to mosaic floorplans only
- From sequence-pairs ([Murata et al. 1996])
  - Relatively straightforward and widely used in previous works
  - No over-specification. Transitive edges can be removed
  - Worst-case complexity: $\Theta(n^2)$ edges
  - $O(n \log n)$ edges on average [Lin 2002]
Ad-Hoc Approaches for Constraint Graph Generation

- From polar graphs ([Ohtsuki et al. 1970])
  - There is no method to explore them in SA
  - Apply to mosaic floorplans only

- From sequence-pairs ([Murata et al. 1996])
  - Relatively straightforward and widely used in previous works
  - No over-specification. Transitive edges can be removed
  - Worst-case complexity: $\Theta(n^2)$ edges
  - $O(n \log n)$ edges on average [Lin 2002]
Constraint Graphs as Floorplan Representation

- **TCG – Transitive Closure Graph ([Lin et al. 2001])**
  - Keep pair-wise relations including transitive edges
  - No over-specification
  - Always $\Theta(n^2)$ edges
- **ACG – Adjacent Constraint Graph ([Zhou et al. 2004])**
  - Intentionally reduce complexity
  - Forbid transitive edges, over-specification, and “crosses”
    - Cross: a structure that may result in $\Theta(n^2)$ edges
  - Complexity: at most $O(n^2)$ edges [Wang 2005]
Constraint Graphs as Floorplan Representation

- **TCG – Transitive Closure Graph ([Lin et al. 2001])**
  - Keep pair-wise relations including transitive edges
  - No over-specification
  - Always $\Theta(n^2)$ edges

- **ACG – Adjacent Constraint Graph ([Zhou et al. 2004])**
  - Intentionally reduce complexity
  - Forbid transitive edges, over-specification, and “crosses”
    - Cross: a structure that may result in $\Theta(n^2)$ edges
  - Complexity: at most $O(n^3)$ edges [Wang 2005]
Our Contribution: *Linear Constraint Graph*

- A general floorplan representation based on constraint graphs
- At most $2n + 3$ vertices and $6n + 2$ edges for $n$ modules
- Intuitively combine the ideas of polar graphs and ACGs
- One application: floorplan optimization with soft blocks
Outline

Overview of Floorplanning

Linear Constraint Graph

The LCG Floorplanner

Experimental Results

Conclusions
Cross Avoidance

- Crosses may result in $\Theta(n^2)$ edges
- Use alternative relations as proposed by ACG
  - However, still have complicated patterns/relations
- Use a “bar” similar to polar graphs
  - Require a dummy vertex in the graph
  - Need a systematic approach!
Cross Avoidance

- Crosses may result in $\Theta(n^2)$ edges
- Use alternative relations as proposed by ACG
  - However, still have complicated patterns/relations
- Use a “bar” similar to polar graphs
  - Require a dummy vertex in the graph
  - Need a systematic approach!
Cross Avoidance

- Crosses may result in $\Theta(n^2)$ edges
- Use alternative relations as proposed by ACG
  - However, still have complicated patterns/relations
- Use a “bar” similar to polar graphs
  - Require a dummy vertex in the graph
  - Need a systematic approach!
Intuitions for Linear Constraint Graph (LCG)

- Avoid horizontal crosses: use alternative relations as ACGs
- Avoid vertical crosses: use horizontal bars as polar graphs
- Introduce dummy vertices to constraint graphs to reduce number of edges
From Floorplan to LCG

- A floorplan with non-overlapping modules
- Construct LCG by adding modules from bottom to top
  - Horizontal graph: planar, w/o transitive edge
  - Vertical graph: separate modules not separated horizontally
Boundary: $s_h$, $t_h$, $s_v$, $t_v$

Top modules: $a$

- Only need to check modules on the top for insertion since modules are inserted from bottom to top
From Floorplan to LCG

- Insert $e$ between $a$ and $t_h$
  - Break $a \rightarrow t_h$ into $a \rightarrow e$ and $e \rightarrow t_h$
- Top modules: $a \rightarrow e$
Insert $g$ between $e$ and $t_h$
  
  Break $e \rightarrow t_h$ into $e \rightarrow g$ and $g \rightarrow t_h$

Top modules: $a \rightarrow e \rightarrow g$
From Floorplan to LCG

- Insert $d$ between $s_h$ and $e$
  - Add one bar $w$ on top of $a$
  - Insert $s_h \rightarrow d$ and $d \rightarrow e$
- Top modules: $d \rightarrow e \rightarrow g$
From Floorplan to LCG

- Insert $b$ between $s_h$ and $d$
  - Break $s_h \rightarrow d$ into $s_h \rightarrow b$ and $b \rightarrow d$
- Top modules: $b \rightarrow d \rightarrow e \rightarrow g$
From Floorplan to LCG

- Insert $i$ between $e$ and $t_h$
  - Add one bar $x$ on top of $g$
  - Insert $e \rightarrow i$ and $i \rightarrow t_h$

- Top modules: $b \rightarrow d \rightarrow e \rightarrow i$
From Floorplan to LCG

- Insert c between $s_h$ and d
  - Add one bar y on top of b
  - Insert $s_h \rightarrow c$ and $c \rightarrow d$
- Top modules: $c \rightarrow d \rightarrow e \rightarrow i$
From Floorplan to LCG

- Insert $f$ between $c$ and $i$
  - Add one bar $z$ on top of $d$, $e$, $x$, $y$
  - Insert $c \rightarrow f$ and $f \rightarrow i$

- Top modules: $c \rightarrow f \rightarrow i$
Insert $h$ between $f$ and $i$

- Break $f \rightarrow i$ into $f \rightarrow h$ and $h \rightarrow i$

Top modules: $c \rightarrow f \rightarrow h \rightarrow i$
Observation: for each new module, we either
- Break a horizontal edge into two
- Insert two horizontal edges and a horizontal bar

Horizontal Adjacency Graph (HAG)
- Each edge connects two modules adjacent to each other
- $n + 2$ vertices, at most $2n$ edges, at most $n - 1$ bars
- Planar – faces correspond to horizontal bars
Observation: for each new module, we either
- Break a horizontal edge into two
- Insert two horizontal edges and a horizontal bar

Horizontal Adjacency Graph (HAG)
- Each edge connects two modules adjacent to each other
- $n + 2$ vertices, at most $2n$ edges, at most $n - 1$ bars
- Planar – faces correspond to horizontal bars
Above and Below Paths

Each face of HAG is surrounded by two paths: the above path and the below path.

- The left/right-most modules are the same
- Other modules on the above path are above the bar
- Other modules on the below path are below the bar

- The length of each path is at least 2
- Otherwise there is a transitive edge
Above and Below Paths

Each face of HAG is surrounded by two paths: the above path and the below path.

- The left/right-most modules are the same
- Other modules on the above path are above the bar
- Other modules on the below path are below the bar

The length of each path is at least 2
- Otherwise there is a transitive edge
Vertical Relations in LCG

- **Implied by HAG**
  - Separate modules not separated horizontally
  - From a bar to a module, a module to a bar, or a bar to a bar
  - Each module connects to 2 bars: above and below
  - Each bar connects to at most 4 bars

- **Vertical Companion Graph (VOG)**
  - At most $n - 1$ bars
  - At most $2n + 3$ vertices, at most $4n + 2$ edges
Vertical Relations in LCG

- Implied by HAG
  - Separate modules not separated horizontally
  - From a bar to a module, a module to a bar, or a bar to a bar
  - Each module connects to 2 bars: above and below
  - Each bar connects to at most 4 bars

- Vertical cOmpanion Graph (VOG)
  - At most $n - 1$ bars
  - At most $2n + 3$ vertices, at most $4n + 2$ edges
Combine HAG and VOG into a constraint graph
At most $2n + 3$ vertices and $6n + 2$ edges
Can represent any non-overlapping floorplan
Outline

Overview of Floorplanning

Linear Constraint Graph

The LCG Floorplanner

Experimental Results

Conclusions
Perturbations of LCG

- The planar HAG allows relatively easy perturbations
  - Update VOG accordingly
- Three perturbations with $O(n)$ complexity
  - Exchange two modules: no change in topology
  - insertH: change vertical relation to horizontal
  - removeH: change horizontal relation to vertical
- The perturbations are complete
  - Any LCG can be converted to any other LCG by at most $3n$ perturbations
Perturbations of LCG

- The planar HAG allows relatively easy perturbations
  - Update VOG accordingly
- Three perturbations with $O(n)$ complexity
  - Exchange two modules: no change in topology
  - insertH: change vertical relation to horizontal
  - removeH: change horizontal relation to vertical
- The perturbations are complete
  - Any LCG can be converted to any other LCG by at most $3n$ perturbations
The planar HAG allows relatively easy perturbations
  - Update VOG accordingly
Three perturbations with $O(n)$ complexity
  - Exchange two modules: no change in topology
  - insertH: change vertical relation to horizontal
  - removeH: change horizontal relation to vertical
The perturbations are complete
  - Any LCG can be converted to any other LCG by at most $3n$ perturbations
The insertH Operation

- Insert $b \rightarrow a$
- Remove transitive edges
  - Remove $c \rightarrow a$ if $c$ starts the above path
  - Remove $b \rightarrow d$ if $d$ ends the below path
The removeH Operation

- Remove $b \rightarrow a$
- Insert $c \rightarrow a$ and $b \rightarrow d$
  - $c \rightarrow a$ is optional iff $a$ has at least 2 incoming edges
  - $b \rightarrow d$ is optional iff $b$ has at least 2 outgoing edges
Floorplan Optimization w/ Soft Blocks

[Young et al. 2001], [Lin et al. 2006]

- The area of each soft block is known.
- The decision variables are the widths of the soft blocks and the positions of all the modules.
- Derive non-overlapping condition for the modules from LCG as a system of difference equations.
- Apply Lagrangian relaxation to minimize the perimeter of the floorplan.
Outline

Overview of Floorplanning

Linear Constraint Graph

The LCG Floorplanner

Experimental Results

Conclusions
Experimental Results for General Floorplans

- 3 GSRC benchmarks with hard blocks
- Compare to Parquet [Adya et al. 2003] and ACG [Zhou et al. 2004]
- Wire length optimization (wire length + chip area)

<table>
<thead>
<tr>
<th>name</th>
<th>Parquet</th>
<th></th>
<th>ACG</th>
<th></th>
<th>LCG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ds(%)</td>
<td>wl</td>
<td>t(s)</td>
<td>ds(%)</td>
<td>wl</td>
</tr>
<tr>
<td>n100</td>
<td>7.95</td>
<td>324k</td>
<td>30</td>
<td>8.33</td>
<td>308k</td>
</tr>
<tr>
<td>n200</td>
<td>10.99</td>
<td>581k</td>
<td>150</td>
<td>10.26</td>
<td>540k</td>
</tr>
<tr>
<td>n300</td>
<td>12.05</td>
<td>709k</td>
<td>288</td>
<td>13.00</td>
<td>644k</td>
</tr>
</tbody>
</table>
Area Optimization w/ Soft Blocks

- 5 modified MCNC benchmarks with soft blocks
- Aspect ratio bound: $[0.5, 2]$
- Compare to [Lin et al. 2006] (SP+TR)

<table>
<thead>
<tr>
<th>name</th>
<th>n</th>
<th>SP+TR</th>
<th>LCG+TR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ds(%)</td>
<td>t(s)</td>
</tr>
<tr>
<td>apte</td>
<td>9</td>
<td>0.04</td>
<td>34</td>
</tr>
<tr>
<td>xerox</td>
<td>10</td>
<td>0.08</td>
<td>43</td>
</tr>
<tr>
<td>hp</td>
<td>11</td>
<td>0.09</td>
<td>41</td>
</tr>
<tr>
<td>ami33</td>
<td>33</td>
<td>0.28</td>
<td>383</td>
</tr>
<tr>
<td>ami49</td>
<td>49</td>
<td>0.24</td>
<td>694</td>
</tr>
</tbody>
</table>
Wire Length Optimization w/ Soft Blocks

- 5 modified MCNC benchmarks with soft blocks
- Aspect ratio bound: [0.5, 2]
- Compare to [Lin et al. 2006] (SP+TR)

<table>
<thead>
<tr>
<th>name</th>
<th>SP+TR</th>
<th></th>
<th>LCG+TR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ds(%)</td>
<td>wl(mm)</td>
<td>t(s)</td>
</tr>
<tr>
<td>apte</td>
<td>0.09</td>
<td>125.29</td>
<td>35</td>
</tr>
<tr>
<td>xerox</td>
<td>0.21</td>
<td>145.16</td>
<td>46</td>
</tr>
<tr>
<td>hp</td>
<td>0.26</td>
<td>43.42</td>
<td>37</td>
</tr>
<tr>
<td>ami33</td>
<td>0.50</td>
<td>57.89</td>
<td>349</td>
</tr>
<tr>
<td>ami49</td>
<td>1.17</td>
<td>290.89</td>
<td>615</td>
</tr>
</tbody>
</table>
Overview of Floorplanning

Linear Constraint Graph

The LCG Floorplanner

Experimental Results

Conclusions
Conclusions

- **Linear Constraint Graphs** (LCG) are proposed as a general floorplan representation based on constraint graphs
  - For $n$ modules, each LCG has at most $2n + 3$ vertices and at most $6n + 2$ edges
  - LCGs can represent any non-overlapping floorplans
- Simulated annealing based floorplanner is presented
- The advantages of LCGs is confirmed by the experimental results.
Q & A
Thank you!