A Layout Advisor for Timing-Critical Bus Routing

Wei Huang and Andrew B. Kahng
Computer Science Dept., UCLA

Problem Statement

Given:

- Block floorplan with *B* blocks and *n* terminals
- Delay constraints for each source-sink terminal pair
- User parameter *p,* optimization objective

Output:

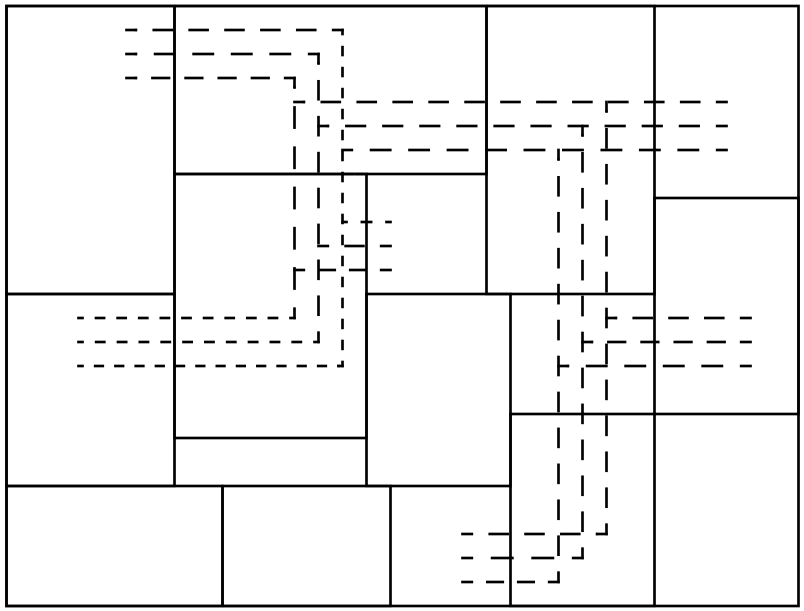
- *p* best routing solutions

Example Objective:

- min treelength s.t. delay constraints
- max delay slack over all constraints

Motivations

- Bus routing
  - wide buses dominate global interconnect
  - top-down ASIC methodology: topology, layer assignment, interconnect optimization all done in chip planning



- Objectives
  - optimize timing
  - optimize routing resource usage
- Multi-source buses
  - limited resource forces time-sharing of global interconnects

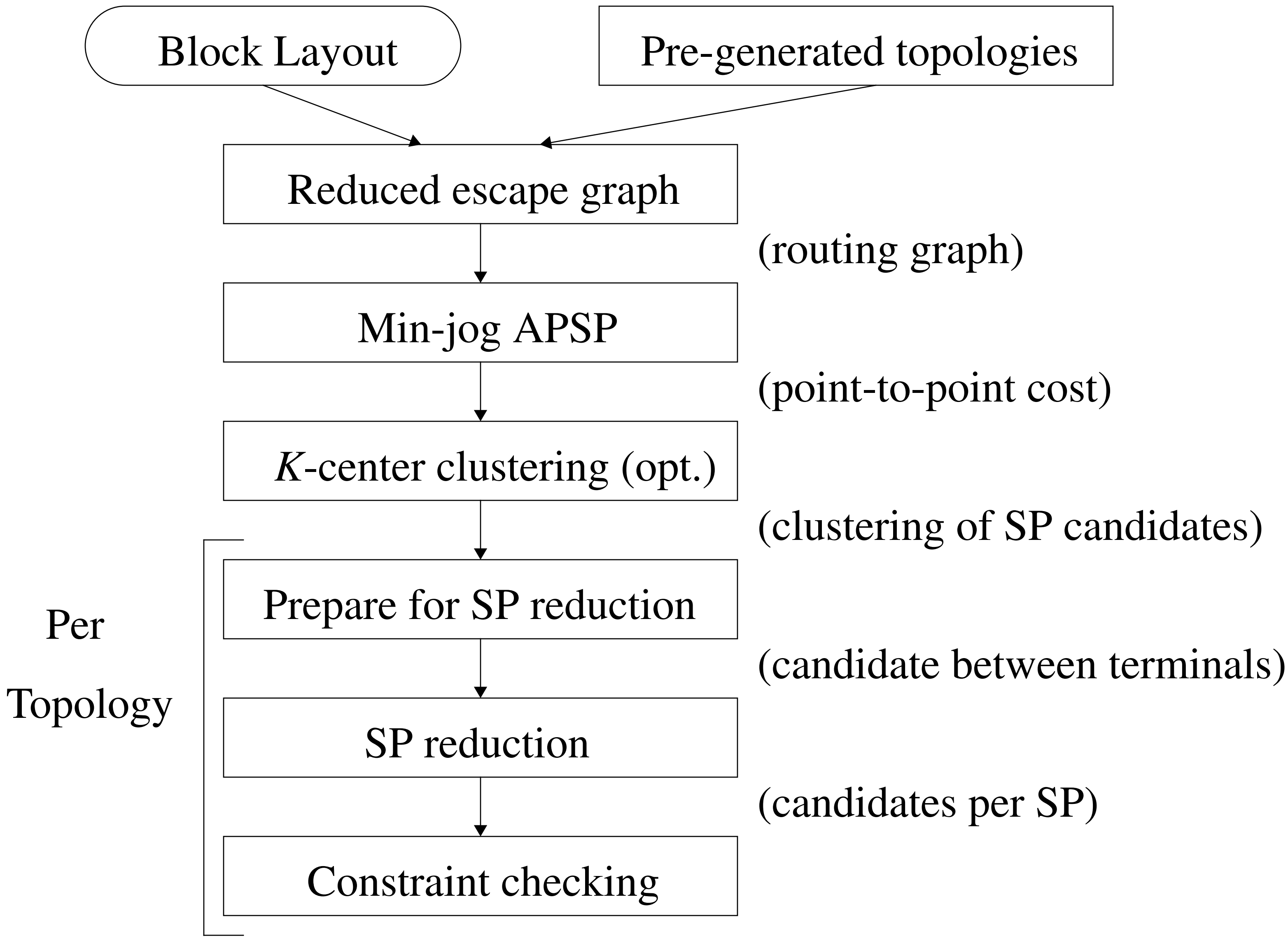
Previous Work

- Performance-driven routing tree approaches (1987-)
- LP with pathlength lower, upper bounds (OPP96)
- Multi-source formulation is NP-complete (FPP95)
- Min-diameter Steiner arborescence (CM95)
- Multi-terminal (exhaustive, heuristic) w/ obstacles (GC94)

Our Work

- Philosophy:
  - spend time on good solutions for global buses
- Contributions
  - framework for upper delay bounds, alternate delay models, bus splitting
  - exhaustive search over all solutions (topology and embedding)
  - general per-layer per-region routing cost model
  - terminals allowed inside block regions
  - heuristics to maintain reasonable runtimes

General Procedure



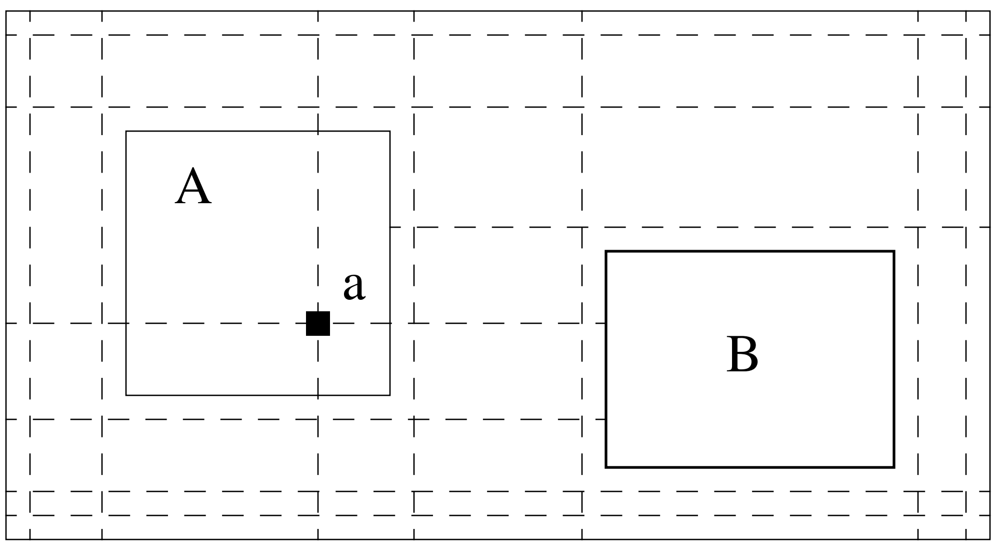
Escape Graph Generation (CR88)

- Line-sweep based, O(max(n,$mlogm$))

  m = number of escape segments

  n = number of segment intersections, bounded by m^2
- Allow terminals inside of obstacles
- Reduce escape graph, O(nk^2)

  k = number of terminals



Min-Jog All-Pairs Shortest Paths

1. Costs: *H-cost* = $(1+\varepsilon_1)$ * *cost*, *V-cost* = $(1+\varepsilon_2)$ * *cost*

   jog cost = ε_3

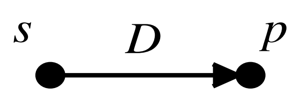
2. Do a BFS from each source *s:*

   For each $t \neq s$ encountered, update:

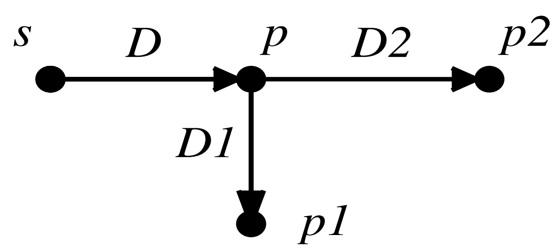
   t.[s] (* = from left, right, top, bottom)*

   apsp[s][t] = min (t.[s])*

3. Trace back using *apsp[s][t]* and *t.*[s]*



i) Update *p.left[s]* **with** *D*



ii) Update *p1.top[s]* **with** *D+D1+jog_cost*

Update *p2.left[s]* **with** *D+D2*

Steiner Candidate Reduction I

- Need for Reduction:

20 blocks ---> 100 candidate Steiner points
6-terminal net ---> 105 topologies, 4 Steiner points

100^4 possibilities for each topology

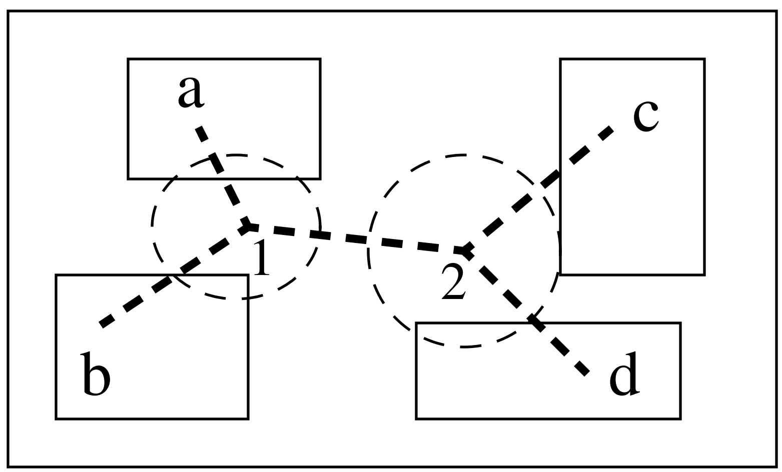
$105 \times 10^8 \approx 10^{10}$ possible trees

- For each pair of terminals

x is a candidate Steiner point w.r.t. (*a,b*) if it can be on *a-b* path

$delay(a, cx) + delay(x, b) \leq constraints(a, b)$ (Linear)

delay(a,b) with *x* as only $Steiner\ point \leq constraints(a, b)$ (Elmore)



Steiner Candidate Reduction II

- For each topology
    - For each Steiner point in topology
        - Computer terminal pairs whose paths contain SP
        - Compute Steiner candidate list
- Implementation: precompute lists of candidate SP's
    - feasible for a terminal and all other terminals,
    - feasible for a terminal and all except one other terminals,
    - feasible for a specific terminal pair
- For Elmore delay, consider subtree capacity

  ---> further reduce the candidate list

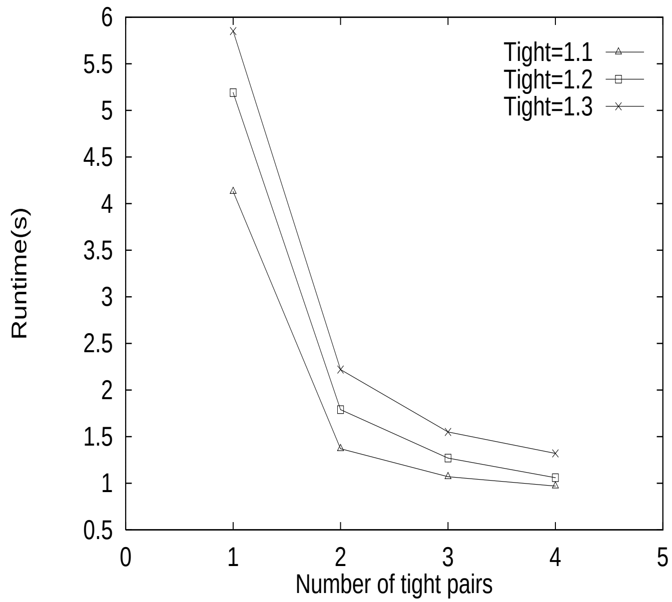
Experimental Results I

- Different terminal pairs with tight constraints

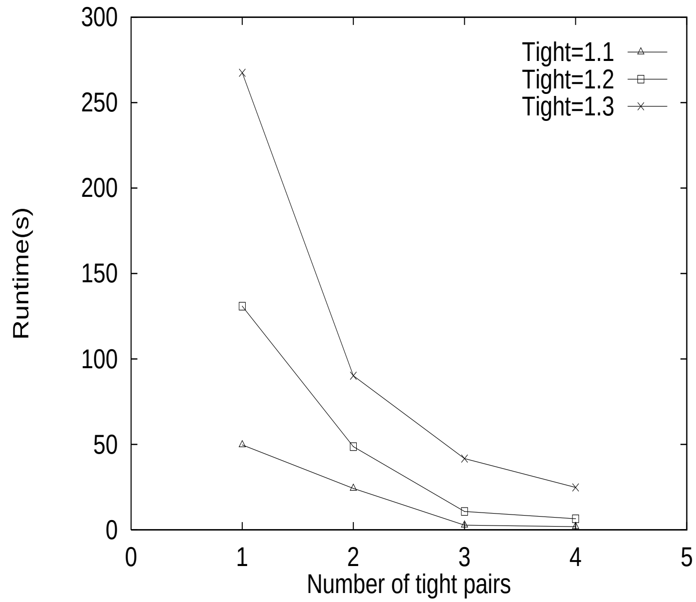
  ---> different minimum wire length routing trees are found

Experimental Results II

- Increased # obstacles ---> increased runtime
- Increased # terminals ---> increased runtime
- Increased # tight pairs ---> decreased runtime, increased min treelength
- Increased tightness ---> decreased runtime, increased min treelength



(a) 5-terminal nets



(b) 6-terminal nets

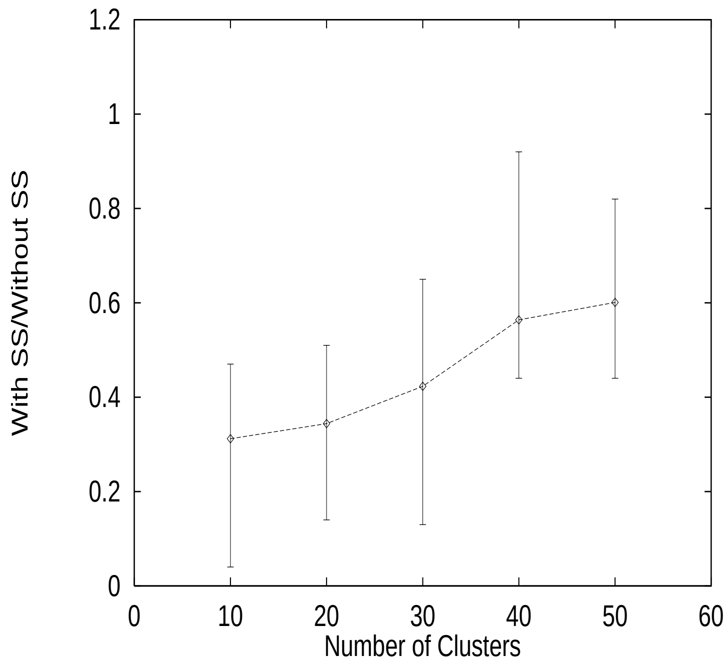
Speedup by Steiner Point Sampling

- Large net, loose constraints ---> Many candidate SP's
- Apply *K*-center clustering (G85)
- In topology checking:

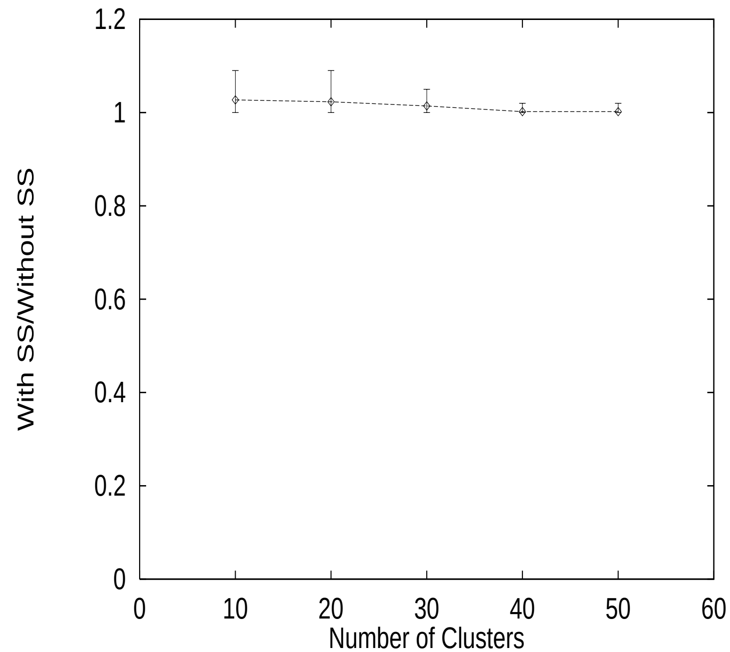
  - if too many candidates for an SP, use one representative for each cluster
  - if representative passes constraint checking, check all points of cluster
  - alternative: local search through combinations of candidate points

Speedup by Steiner Point Sampling

Runtime and min treelength comparison for varying number of clusters in *K*-center clustering



(a) Runtime



(b) Min treelength

Observations

- More speedup for large nets
- More speedup for loose constraints; local search further improves runtime
- Little or no speedup for tight constraints
- Average treelength 1-10% more than optimal
- Best case: up to 90% runtime improvement
- Worst case: up to 21% more than optimal treelength

Conclusions

- Efficient implementation of exhaustive, and heuristic constructions for timing-critical bus routing
- Tested with synthetic data: small (5- or 6- terminal) multi-source or single-source nets
- Speedup methods for loosely constrained small nets

Future Work

- Bus splitting
- Heuristic grouping of terminals in large nets
- Incorporate delay lower bounds
- Detailed geometric embedding advice (layer assignment, via pattern, etc.)