Routing in VLSI Design and Communication Networks

Tamás Terlaky — McMaster University

Anthony Vannelli — University of Waterloo

Hu Zhang — McMaster University

MOPTA, Windsor, July 25, 2005

Overview

- global routing in VLSI design and multicast routing in communication networks;
- integer linear programming formulation;
- approximation algorithms for min-max resource-sharing problems;
- block problems and virtual layer method;
- results and more applications;
- future work.

Global Routing in VLSI Design

Given:

- a lattice graph $G = (V, E)$ (rectangular holes allowed), $|V| = n$ and $|E| = m$;
- K nets $S_1, \ldots, S_K \subseteq V$.

Solution: a set of trees $T_1, \ldots, T_K$, where T_k spans the vertex set S_k, for $k \in \{1, \ldots, K\}$.

Goal: **minimize** a certain objective function.

Objectives

- overall wirelength (sum of the tree lengths);
- maximum edge congestion (number of trees that use an edge);
- total number of bends of trees (number of vias in physical layer);
- combination of some or all of above objectives.

Multicast Routing in Communication Networks

Given:

- an arbitrary connected graph $G = (V, E)$, $|V| = n$ and $|E| = m$;
- K requests $S_1, \ldots, S_K \subseteq V$, and all $|S_k| \geq 2$.

Solution: a set of trees $T_1, \ldots, T_K$, where T_k spans the vertex set S_k, for $k \in \{1, \ldots, K\}$.

Goal: **minimize** a certain objective function.

Objectives

- maximum edge congestion;
- overall link cost (sum of the tree lengths);
- combination of above objectives.

Previous Results for Global Routing I

- $\mathcal{NP}$-hard **(Lengauer 90)**;
- sequential routing (ordered nets routed in sequential way):
  - maze runner (heuristic) **(Lee 61)**;
  - enhancements of maze runner **(Hadlock 75)**;
  - variants of maze runner **(Kuh, Marek-Sadowska 85), (Sherwani 99)**;

Previous Results for Global Routing II

- integer programming methodologies:
  - two terminals per net by multicommodity flow ($|S_k| = 2$ for all $k = 1, \ldots, K$) **(Shragowitz, Keel 87)**;
  - minimizing overall wirelength w.r.t. capacity **(Vannelli 91)**;
  - minimizing the maximum tree length **(Lengauer, Lungering 00)**;
  - minimizing the maximum congestion **(Raghavan, Thompson 87)**;
  - minimizing a linear combination of all objectives **(Behjat 02), (Behjat, Vannelli, Rosehart 05)**.

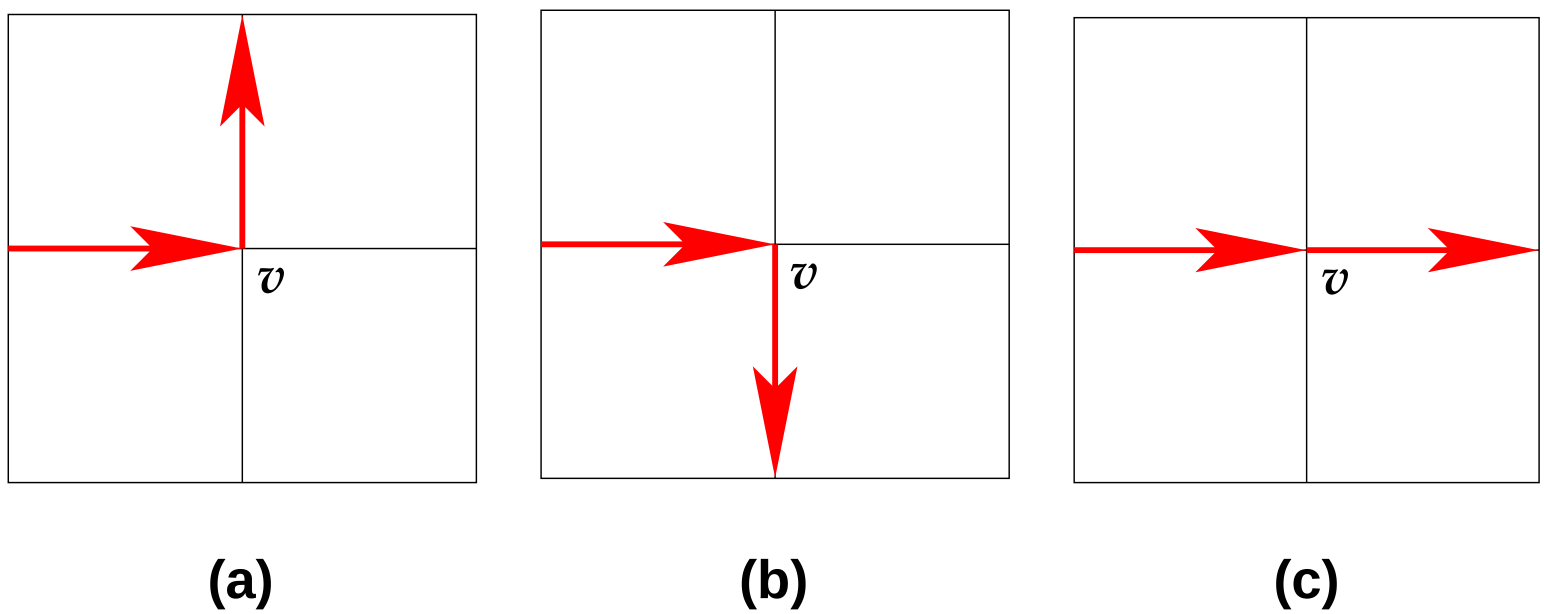
Previous Results for Multicast Routing

- multicast congestion/packing problem (minimizing the maximum congestion) **(Jansen, Z. 02), (Baltz, Srivastav 03), (Lu, Z. 05)**;
- minimizing overall link cost:
  - group multicast routing (identical requests in directed graphs with edge capacities) **(Jia, Wang 97), (Cai, Deng, Wang 04)**;
  - undirected version can not be approximated with $\exp(\text{poly}(n))$ **(Ma, Wang 00)**;
  - bounded degree for one request **(Hu et al. 04), (Lin 05), (Cai, Lin, Xue 05)**.

Our Contributions

- generalized models for both problems;
- approximation algorithms with theoretical performance bounds;
- virtual layer method: a combinatorial technique for lattice graphs with *bend-dependent vertex cost*.

Bend-Dependent Vertex Cost



- positive vertex costs occur with bends: cost(a)=cost(b);
- no vertex cost without bend: cost(c)$= 0$.

Integer Linear Program I

$$\begin{aligned}
\min \quad & \alpha \sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k} \ell(T) x_k(T) + \beta \sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k} v(T) x_k(T) & \\
\text{s.t.} \quad & \sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k \& e_i \in T} x_k(T) \leq c_i, & \text{for all } e_i \in E; \\
& \sum_{T \in \mathcal{T}_k} x_k(T) = 1, & \text{for } k = 1, \ldots, K; \\
& x_k(T) \in \{0, 1\}, & \text{for all } T \text{ and } k = 1, \ldots, K.
\end{aligned}$$

- $\mathcal{T}_k$: set of all trees spanning net S_k;
- $x_k(T)$: indicator variable representing whether tree T is selected for S_k;
- c_i: individual capacity of edge e_i controlling the congestion;
- $\ell(T)$ and $v(T)$: length and number of bends of tree T;
- α and β: artificial weights of overall wirelength and total number of bends ($\alpha = 1$ and $\beta = 0$ for communication networks).

Integer Linear Program II

$$\min \quad g$$

$$\text{s.t.} \quad \alpha \sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k} \ell(T) x_k(T) + \beta \sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k} v(T) x_k(T) \le g,$$

$$\sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k \& e_i \in T} x_k(T) \le c_i, \qquad \text{for all } e_i \in E;$$

$$\sum_{T \in \mathcal{T}_k} x_k(T) = 1, \qquad \text{for } k = 1, \ldots, K;$$

$$x_k(T) \in \{0, 1\}, \qquad \text{for all } T \text{ and } k = 1, \ldots, K$$

Linear Programming Relaxation

With binary search strategy and a guessed objective value g, the linear relaxation can be formulated as following packing problem:

$$
\begin{aligned}
\min \quad & \lambda \\
\text{s.t.} \quad & \sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k \& e_i \in T} \frac{x_k(T)}{c_i} \leq \lambda, && \text{for all } e_i \in E; \\
& \alpha \sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k} \frac{\ell(T) x_k(T)}{g} + \beta \sum_{k=1}^{K} \sum_{T \in \mathcal{T}_k} \frac{v(T) x_k(T)}{g} \leq \lambda, \\
& \sum_{T \in \mathcal{T}_k} x_k(T) = 1, && \text{for } k = 1, \ldots, K; \\
& x_k(T) \in \{0, 1\}, && \text{for all } T \text{ and } k = 1, \ldots,
\end{aligned}
$$

Strategy and Difficulties

Strategy:

- solve the LP-relaxation;
- apply the randomized rounding.

Difficulties:

- there are exponentially many variables (so exact algorithms such as standard interior point methods do not work);
- the block problem is the minimum Steiner tree problem, that is $\mathcal{APX}$-hard (so many approximation algorithms for packing problems have running times depending on the input data or on the approximation ratio – only pseudo polynomial time).

Min-Max Resource-Sharing Problems

$$\min\{\lambda | f(x) \leq \lambda \cdot \vec{1}, x \in B\},$$

where $f : B \longrightarrow \mathbb{R}_{+}^{M}$ is a vector of M non-negative continuous convex functions defined on a non-empty convex compact set $B \in \mathbb{R}^{N}$, and e is the vector of all ones.

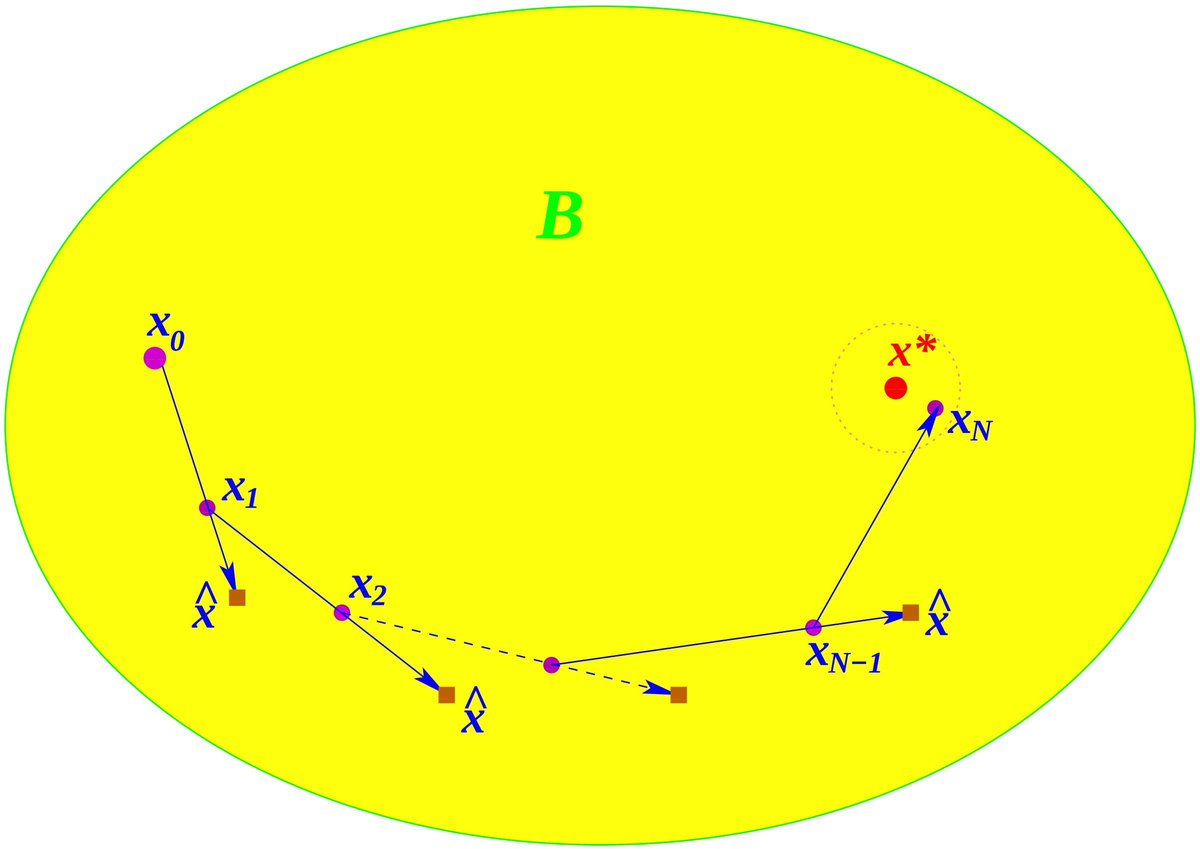
The linear version is called the *packing problem*.

Approximation Algorithm: The algorithm in **(Jansen, Z. 02)** can deliver an approximate solution to the min-max resource-sharing problem within a polynomial number of iterations, where each iteration calls a given approximate block solver once.

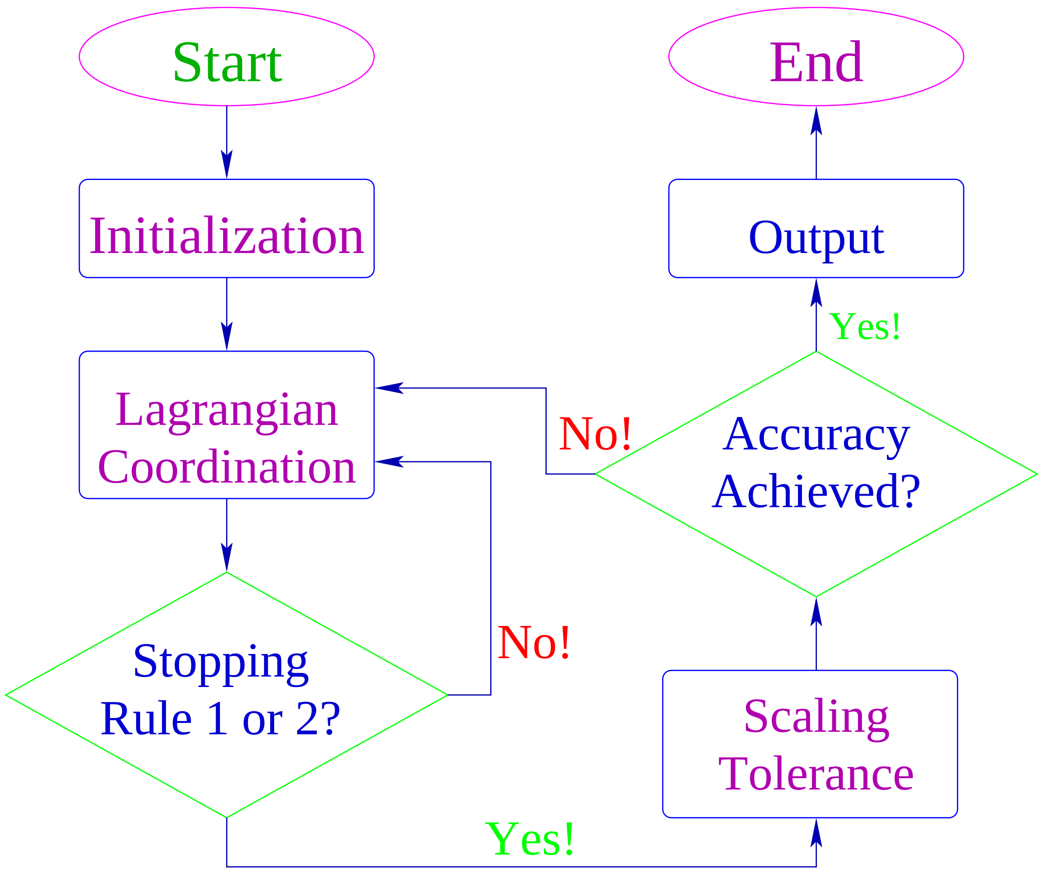
Approximation Algorithms

- main ideas:
  - Lagrangian or price-directive decomposition;
  - computation in scaling phase;
  - price vector (dual vector) easy for computation;
  - an approximate block solver called once per iteration.
- advantages:
  - column generation-like technique to obtain only a polynomial number of non-zero variables;
  - complexity independent of input data or approximation ratio;
  - derivative-free, no computation of matrix parameters.

Lagrangian Coordination



Flowchart



Start
Initialization
Lagrangian Coordination
Stopping Rule 1 or 2?
No!
Yes!
Scaling Tolerance
Accuracy Achieved?
No!
Yes!
Output
End

Block Problem

For a price vector $p \in P = \{p \in \mathbb{R}^{m+1} | \sum_{i=1}^{m+1} p_i = 1, p_i \geq 0\}$

(computed by the algorithm automatically),

the block problem is for net S_k, $k = 1, \ldots, K$, finding

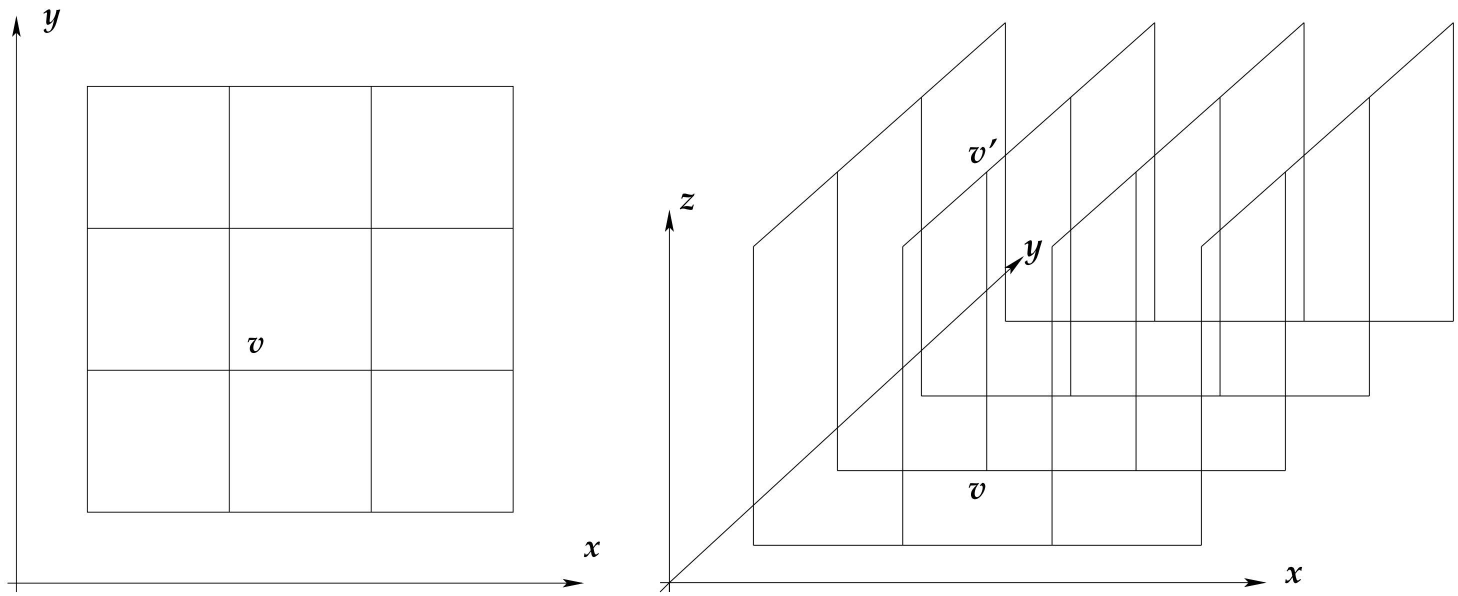
$$\min_{T \in \mathcal{T}_k} \left[\sum_{e_i \in T} \left(\frac{p_i}{c_i} + \frac{\alpha p_{m+1} \ell_i}{g} \right) + \frac{p_{m+1} \beta v(T)}{g} \right]$$

- ℓ_i is the length of edge e_i;
- the first term corresponds to edge weight of the tree T;
- the second term corresponds to bend-dependent vertex cost of T.

Virtual Layer Method

- two layers with identical vertices (same as the original lattice graph);
- only vertical edges of the original graph in the upper layer while only horizontal edges in the lower layer;
- additional edges connecting "mirrors";
- a bend of a tree/path in the original graph equivalent to using the corresponding additional edge once;
- $p_i/c_i + \alpha p_{m+1}\ell_i/g$ assigned as the weight of e_i corresponding to the edge in the original graph;
- $p_{m+1}\beta/g$ assigned as the weight of any additional edge.

Virtual Layer Method



The block problem in the original lattice graph is equivalent to finding minimum Steiner trees for the K nets in the two-layer weighted graph.

Minimum Steiner Tree Problem

- $\mathcal{APX}$-hard **(Bern, Plassmann 89), (Arora et al. 98)**;
- lower bound $\bar{c} = 96/95 \approx 1.0105$ **(Chlebik, Chlebikova 02)**;
- $c = 2$, $\gamma = O(m + n \ln n)$ **(Moore, before 68)**;
- $c = 11/6$, $\gamma = O(mn + s^4)$ **(Zelikovsky 90)**;
- $c = 1 + \ln 3/2 \approx 1.550$, γ polynomial **(Robins, Zelikovsky 00)**,

where c and γ are the approximation ratio and the running time, respectively.

Rounding

With the randomized rounding in **(Raghavan, Thompson 87), (Raghavan 88)**, the following result holds:

Theorem: There exists an asymptotic approximation algorithm for the global routing problem in VLSI design and the multicast routing problem in communication networks that in $O(m(\log m + \varepsilon^{-2} \log \varepsilon^{-1})(k\gamma + m \log \log(m\varepsilon^{-1})))$ time delivers a solution with the objective value at most $c(1+\varepsilon)OPT + O(\max\{\sqrt{OPT \ln m}, \ln m\})$ for a given accuracy $\varepsilon \in (0, 1)$, provided a c-approximate minimum Steiner tree solver, where OPT is the optimum objective value and γ is the running time of the approximate minimum Steiner tree solver.

More Applications

Theorem: There is a polynomial time algorithm for the shortest path problem in lattice graphs, where the total cost is the sum of edge costs and the bend-dependent vertex costs.

Theorem: There is a polynomial time algorithm for the splitable min-cost flow problem in lattice graphs, where the total cost is the sum of edge costs and the bend-dependent vertex costs.

Theorem: There is a polynomial time algorithm for the splitable min-cost multicommodity flow problem in lattice graphs, where the total cost is the sum of edge costs and the bend-dependent vertex costs.

Future Work

- feasibility version: maximizing the number of routed nets/requests;
- multi-layer model: physical structure in VLSI design and manufacturing;
- optical network model: wavelength assignment;
- implementation;
- better and/or faster algorithms.