Multi-Commodity Flow with In-Network Processing

Moses Charikar
Yonatan Naamad
Jennifer Rexford
X. Kelvin Zou

Outline

In-Network Processing

- Computer Networks are now dual-purpose.
  1. Route traffic
  2. Perform services

In-Network Processing

- Computer Networks are now dual-purpose.
  1. Route traffic
  2. Perform services
     - Firewalls
     - Load balancers
     - Video transcoders
     - Traffic encryption/compression
     - Etc.

In-Network Processing

- Computer Networks are now dual-purpose.
  1. Route traffic
  2. Perform services
     - Firewalls
     - Load balancers
     - Video transcoders
     - Traffic encryption/compression
     - Etc.
- Novel uses require novel algorithms.

Middlebox Processing

- Historically, middleboxes were single purpose.
- Network Function Virtualization (NFV) allows for greater flexibility.

- Question: how do we best utilize this newfound flexibility?

Our Model

- **Given**
  1. A graph G modeling our network.
  2. Edge capacities B_e on links.
  3. Processing capacities C_v on vertices.
  4. A collection of flow demand (s_i, t_i) pairs.
- **Find**
  - A way to route **and** process as much flow as possible.

Our Model

- **Given**
  1. A graph G modeling our network.
  2. Edge capacities B_e on links.
  3. Processing capacities C_v on vertices.
  4. A collection of flow demand (s_i, t_i) pairs.
- **Find**
  - A way to route **and** process as much flow as possible.

- Assumption: one unit of flow requires one unit of processing.

Routing **vs.** Steering

Our problem has two components:

- Routing routes traffic between endpoints.
  - Equivalent to Maximum Multicommodity Flow
- Steering steers traffic to processing nodes.
  - Equivalent to Multi-source/sink Maximum flow
- We attempt to solve the *joint* routing and steering problem.

An (exponential) LP

- Problem admits an obvious walk-based LP

$$
\begin{array}{lll}
\textsc{Maximize} & \displaystyle\sum_{i=1}^{D}\sum_{\pi\in P} p_{i,\pi} & \\
\textsc{Subject to} & \displaystyle p_{i,\pi} = \sum_{v\in\pi} p^{v}_{i,\pi} & \forall i \in 1..D, \pi \in P \\
& \displaystyle\sum_{i=1}^{D}\sum_{\pi\ni e} p_{i,\pi} \le B_e & \forall e \in E \\
& \displaystyle\sum_{i=1}^{D}\sum_{\pi\in P} p^{v}_{i,\pi} \le C_v & \forall v \in V \\
& p^{v}_{i,\pi} \ge 0 & \forall p^{v}_{i,\pi}
\end{array}
$$

- Problem: program size may be exponential.

Equivalent LP

- We can also write an edge-based LP

Equivalent LP

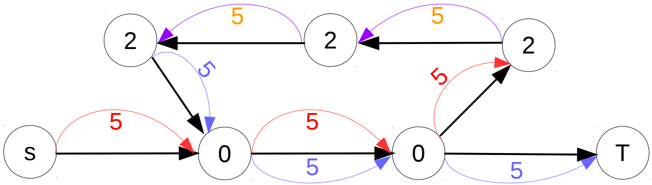
- We can also write an edge-based LP
    - Too big to fit on this slide.

Equivalent LP

- We can also write an edge-based LP
  - Too big to fit on this slide.
- Key ideas:
  1. Each flow demand gets two sets of variables: f_i and w_i.
     - $w_i =$ unprocessed flow being routed
     - $f_i =$ total flow being routed
  2. w_i absorbed at middleboxes, f_i absorbed at terminals
  3. w_i is bounded by f_i
  4. Other constraints are standard extensions of the multicommodity flow LP.

Proof of Equivalence (outline)

- Equivalence of the two LPs is nontrivial
- OPT may use edges more than once. Care is required!



- Proof outline:
  - Cancel redundant w_i and f_i flows as much as possible.
  - Argue that cycles must have some e where $w_i(e)$ drops.
  - Peel off this edge and proceed
- Conclusion: $O(|V| \cdot |E| \cdot |D|)$ algorithm for converting between the two LPs.

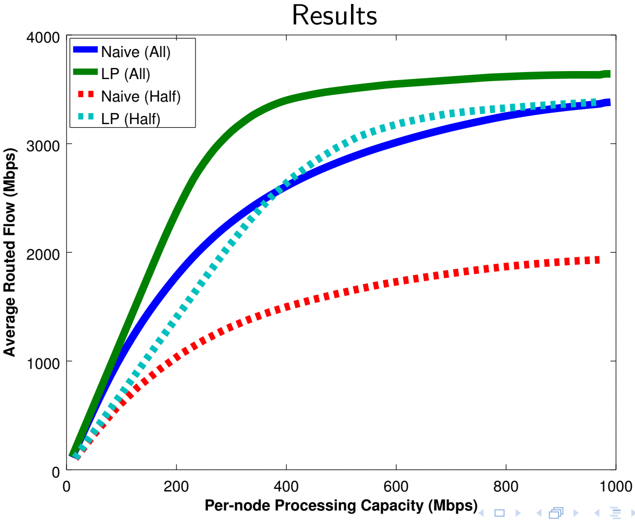
Multiplicative Weights

- Edge-based LP can get unwieldy.
- $(1-\epsilon)$ MW-based approximation in $\tilde{O}(dm^2/\epsilon^2)$ time.
- Similar to the Garg-Könemann algorithm with a more elaborate update step.

Experiments

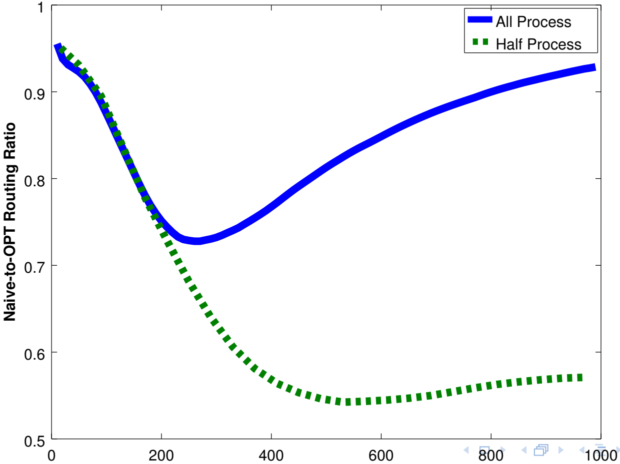
- Ran experiments on Abilene network traces
- Baseline: route flow first, worry about processing later
- Two sets of processing power distributions:
  - All nodes get equal processing capacity
  - A random subset of $n/2$ nodes get processing capcity.

Experiments (absolute)



Experiments (ratio)

Results



Outline

Network Design

- We now know how to utilize middleboxes.
- How do we optimally place them in the first place?
- Middleboxes are indivisible $\rightarrow$ combinatorial problem.

Network Design

- **Given**
  1. An edge-capacitated graph G.
  2. The set of flow demands
  3. For each middlebox v, proposals and costs of installing various amounts of processing capacity
- **Find**
  1. The optimal purchase plan of middlebox processing power in G.

Four Key Problems

	Directed	Undirected
Budgeted Maximization		
Minimization		

Four Key Problems

	Directed	Undirected
Budgeted Maximization	NP-Hard	NP-Hard
Minimization	NP-Hard	NP-Hard

Four Key Problems

		Directed	Undirected
Maximization	Approximation	$O(\log n)$	$O(1)^{(\dagger)}$
	Hardness	$O(1)$	$O(1)$
Minimization	Approximation	$O(\log n)^{(*)}$	$O(\log n)^{(*)}$
	Hardness	$O(\log n)$	$O(\log n)$

End

Questions?