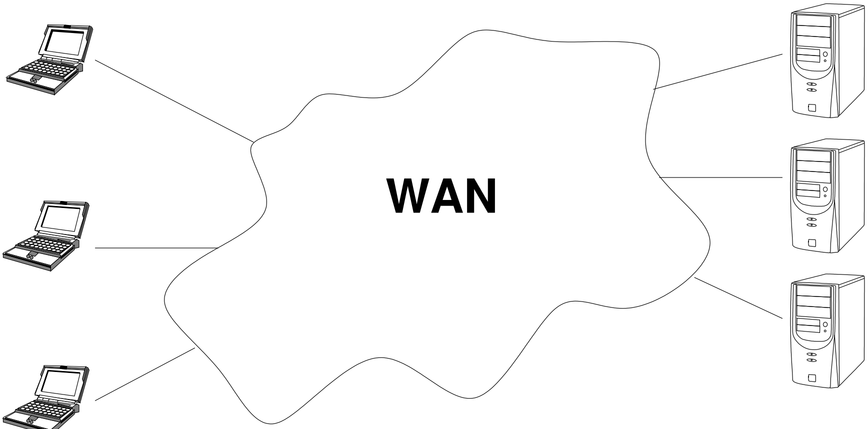
Matrix Factorizations for Computer Network Tomography

David Bindel

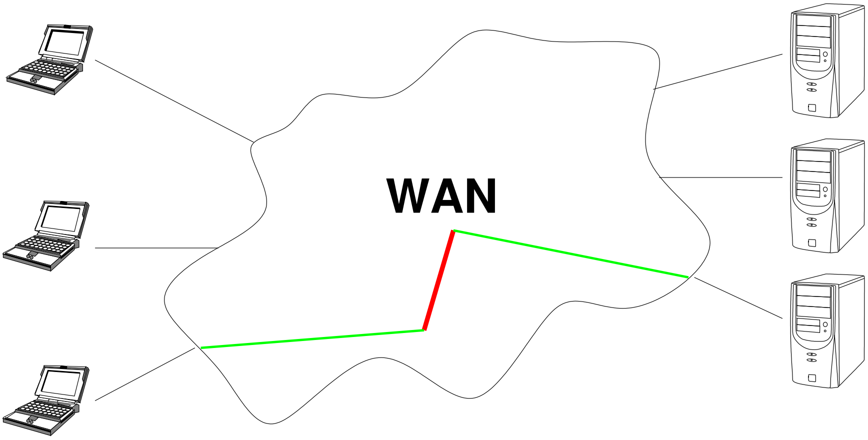
Department of Computer Science
Cornell University

8 Apr 2011

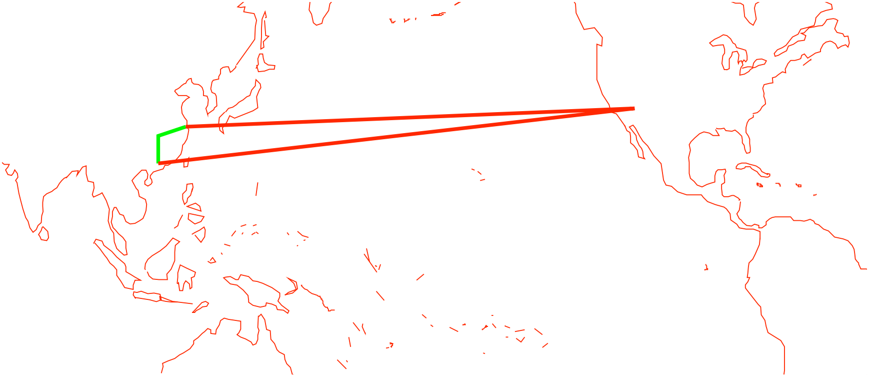
A fuzzy picture



A problem case



More problems



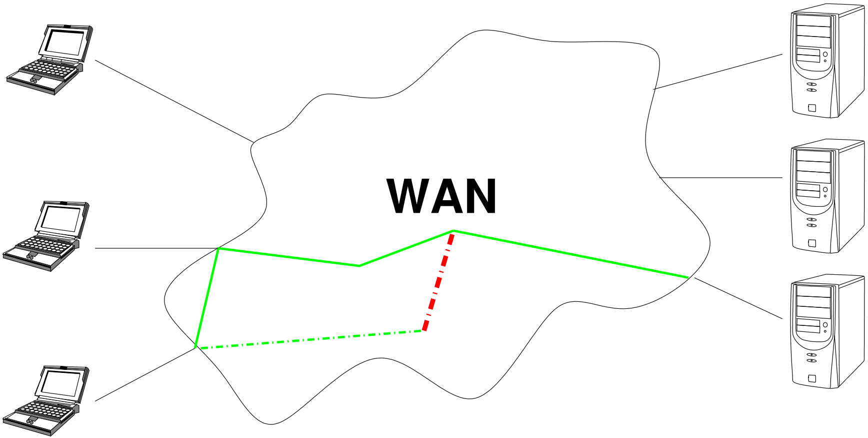
Network “ossification”

Hard:

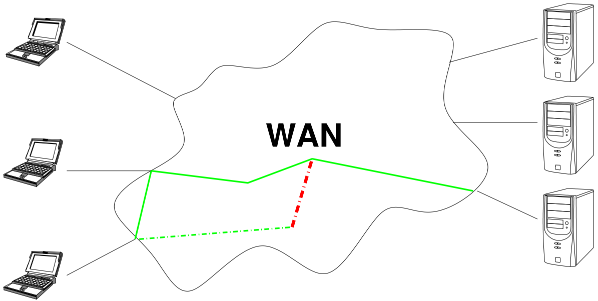
- Get my ISP to change routing tables
- Change BGP to be smarter

Easier: an *overlay* network that I control

Overlays to the rescue?



Overlays and measurement



Would like to figure out:

- Properties of every *end-to-end* routing path (latency, packet loss rates, jitter, ...)
- Location of problem spots in network

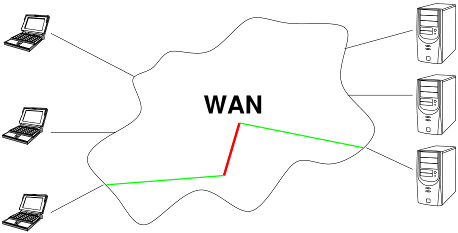
Goal: infer network properties from a few path measurements.

The beginning

"I typed in the SVD from Numerical Recipes; it ran for a few days, then told me it wouldn't converge. What can I do?"

(Y. Chen, 2003)

Additive metrics



For latency, $-\log P(\text{successful transmission})$, jitter

$$\text{path property} = \sum_{\text{link } l \text{ on path}} \text{property of } l$$

Discrete analog to the *Radon transform*

$$Rf(L) = \int_L f(x)\,|dx|$$

Additive metrics and path matrices

Write additive property in matrix form as

$$Gx = b$$

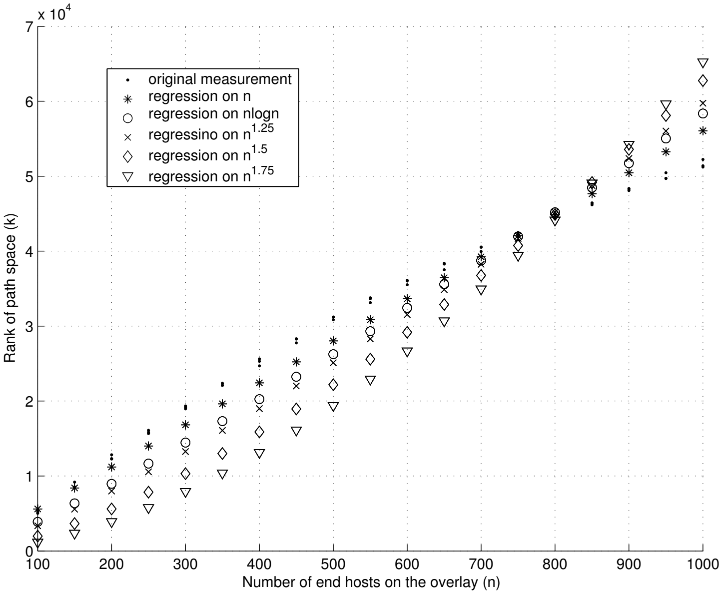
where

- b_i = property of ith end-to-end path
- x_j = property of link j
- $G_{ij} = \begin{cases} 1 & \text{if path } i \text{ uses link } j \\ 0 & \text{otherwise} \end{cases}$

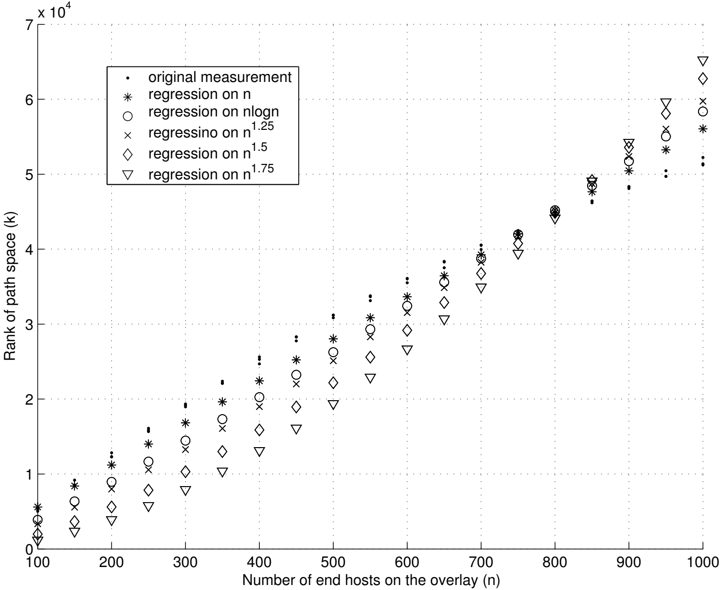
Notes on G

- Short paths $\implies$ G sparse
- Paths mostly stable $\implies$ occasional low-rank updates to G
- $k = \mathrm{rank}(G) <$ # links $\ll$ # paths (for n sufficiently large).

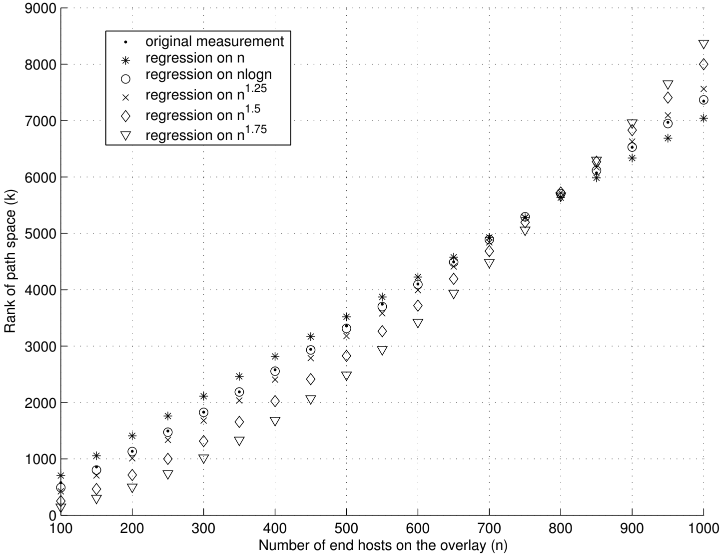
Rank of G: Lucent scan (upper bound)



Rank of G: AS-level Albert-Barabasi



Rank of G: AS-level Albert-Barabasi + RT Waxman



The big questions

Given the model $Gx = b$, $k = \mathrm{rank}(G) \ll$ number of paths:

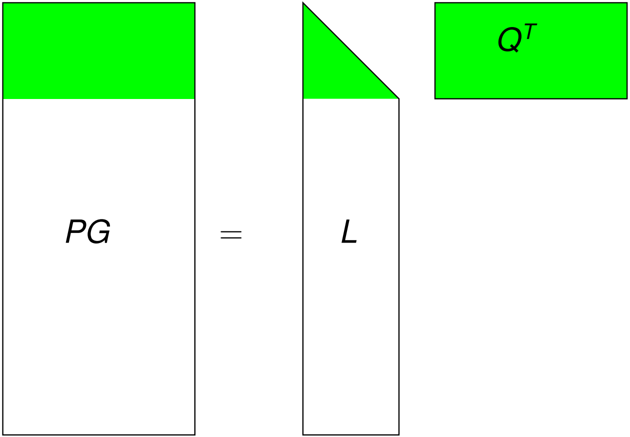
- What can we infer?
- How do we do it fast?
- How does the low rank arise?

Path property inference

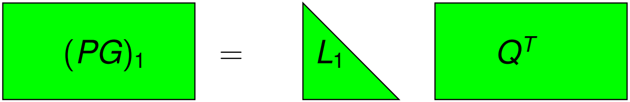
Suppose $Gx = b$ and G known. Measure k paths, infer others?

(Chen, B., Song, Chavez, Katz —
ToN 2007, SIGCOMM 2004, IMC 2003)

Rank-revealing decomposition of G



Path inference via LQ^T factorization



Problem: Knowing $Gx = b$, infer b from partial measurement:

1. Factor $PG = LQ^T$
2. Solve $L_1 y = (Pb)_1$
3. Compute remainder of b via $Pb = Ly$
   - Or use $b = G(Qy)$ – don't need to save all of L

Other considerations for $PG = LQ^T$

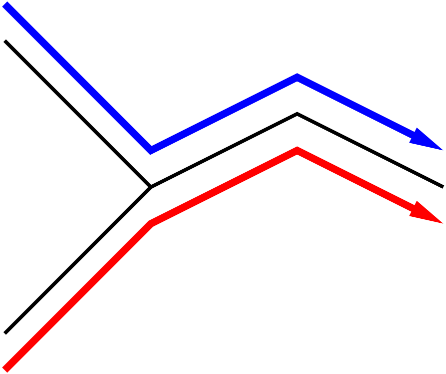
- Factorization becomes dense!
  - Pre-processing cuts cost (topology virtualization)
  - Single precision + tricks helps, too
  - Doesn't scale well beyond 200–300 hosts
- Want to balance measurement load
  - Random initial permutation works well
  - Trade between numerical stability and load balance
- Can update factorization for low-rank changes to G:
  - When nodes enter/exit the overlay
  - When routing paths change

Link property inference

Suppose $Gx = b$ and G known. Measure k paths, estimate x?

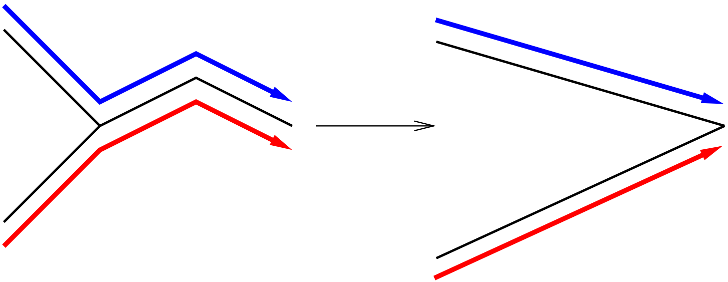
(Zhao, Chen, B. — ToN 2009, SIGCOMM 2006)

Identifiability issues



If both paths are flaky, what is to blame?

Network virtualization and matrix factorization



Factor out a zero-one "virtualization matrix":

$$G(:, \text{fan}) = \begin{bmatrix} c_1 & c_2 & c_1 + c_2 & c_1 + c_2 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Even virtual links may be "unidentifiable."

Dealing with ill-posedness

Inferring link properties is ill-posed! Possible approaches:

- Add statistical assumptions on link properties
- Compute bounds using positivity of x, b
- Infer properties of path *segments*

Subpath inference

Subpath with indicator p is identifiable if

$$p^T = z^T G$$

If $G = LQ^T$, identifiable iff $\|Q^T p\|_2 = \|p\|_2$.

But in a directed graph only end-to-end paths are identifiable!

Subpath inference and good paths

Observation:

- Most paths are “good” ($0 \leq b_i < \epsilon$)
- Any link on a good path is good ($0 \leq b_i < \epsilon$)

Subpath inference and the good path algorithm

Given $Gx = b$, $x \geq 0$ and $b \geq 0$ are sparse. Then

$$G(:, J)x(J) = b$$

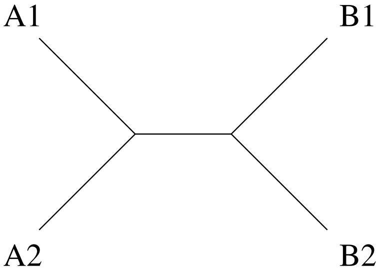
where $J^c = \{\text{links on good paths}\}$. Infer subpaths in reduced problem.

Mean infered subpath length: 3.9 real links (2.3 virtual).

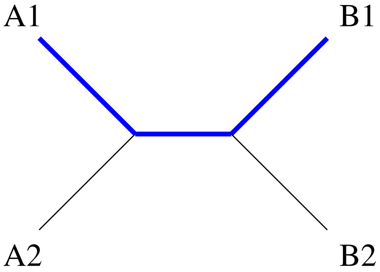
“Waiter, there’s a fly in my soup!”

- Why is G *really* low rank?
- Can’t I get a cheaper, sparser factorization?
- I want to use structure!

The junction pattern

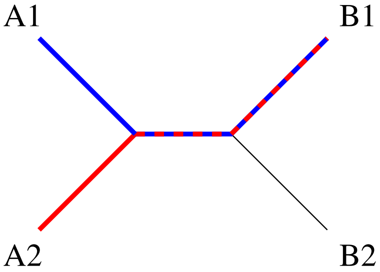


The junction pattern



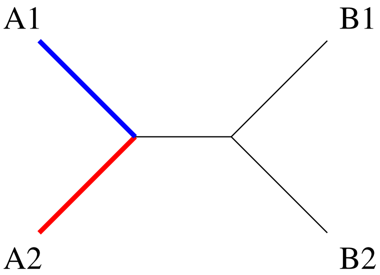
$A_1 B_1$

The junction pattern



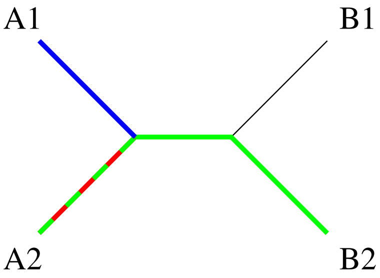
$$A_1 B_1 - A_2 B_1$$

The junction pattern



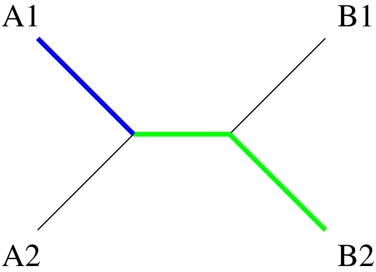
$A_1 B_1 - A_2 B_1$

The junction pattern



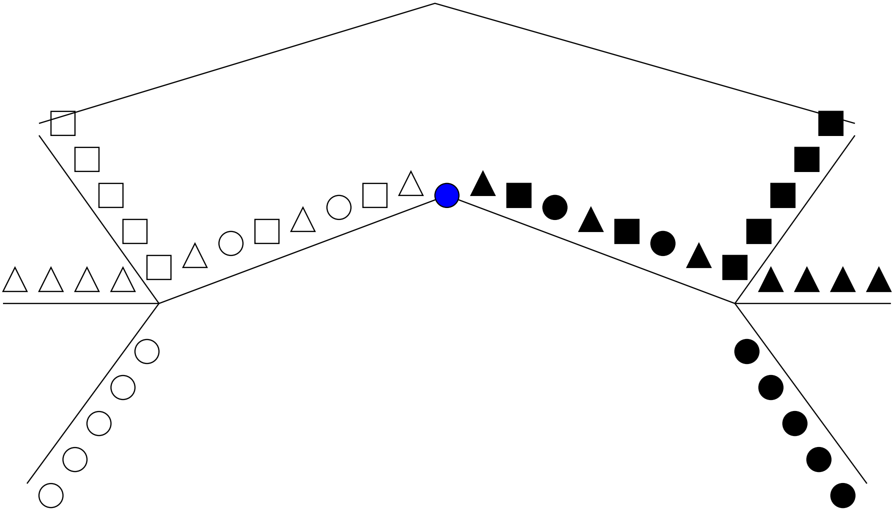
$$A_1 B_1 - A_2 B_1 + A_2 B_2$$

The junction pattern

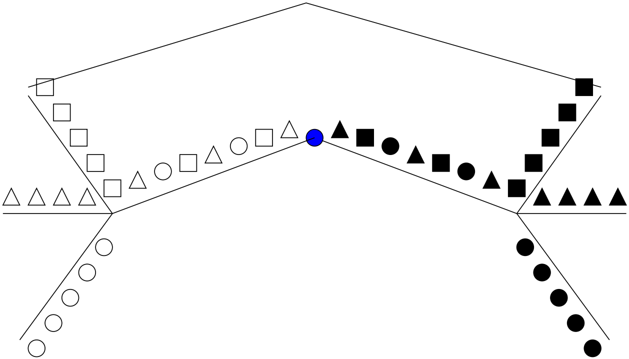


$$A_1 B_1 - A_2 B_1 + A_2 B_2 = A_1 B_2$$

More complicated junctions?



More complicated junctions?

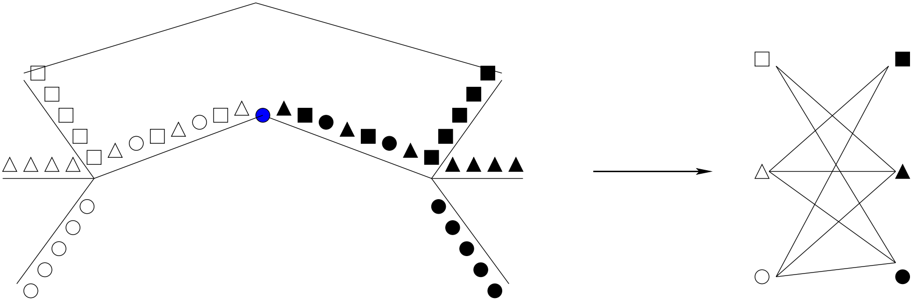


Linear dependencies

  □ + ▲
− △ + ▲
+ △ + ●

= □ + ●

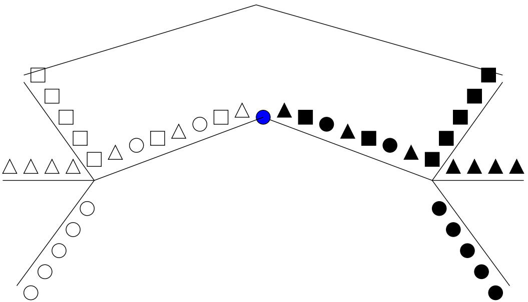
Junctions and bipartite graphs

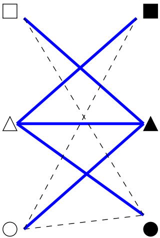


Define a bipartite *router graph* at r:

- Path segments from sources to r are nodes on the left
- Path segments from r to destinations are nodes in the right
- Edges indicate complete paths traversing r

Junctions and bipartite graphs



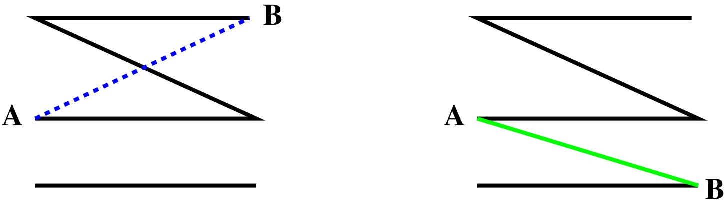


Spanning trees in the router graph

$$\Longrightarrow$$

spanning sets among path vectors

Algorithm basics



Process each path in turn, build router forests incrementally.
Processing a path from A to B through r, have either

1. $A - r$ and $r - B$ in same component $\implies$ could infer path at r from existing paths
2. $A - r$ and $r - B$ in different components $\implies$ might make new inferences via this path

Elimination algorithm

To process path from *A* to *B*:

```
for each router r on path
  update hash h of route up to r
  if (A,h)-(r,B) in router graph
    mark that path can be inferred at r
  else
    add (A,h)-(r,B) to router graph
    for each edge e from source in [(A,h)]
      if e goes to component for (A,h) and
         edge is not already marked then
        put edge on top of list to be processed

if path was not inferred, mark as measured
```

Storage

Sufficient to store:

- Each router graph ($\approx \text{nnz}(G)$ edges)
- Union-find structures for tracking components
- Markers for which paths are measured
- Router used for inference for each inferred path

Choice of representative paths

Spanning trees are not unique! Want representative paths such that

- There are few redundant measurements
- No host (or router) is overloaded with measurement traffic
- Most inferences involve few paths

Don't know how to do this yet...

Junction elimination and factorization

G_r = matrix of path vectors for paths crossing r:

$$G_r = \begin{bmatrix} E_r^S & E_r^D \end{bmatrix} \begin{bmatrix} P_r^S \\ P_r^D \end{bmatrix} = \begin{bmatrix} I \\ T_r \end{bmatrix} \begin{bmatrix} \bar{E}_r^S & \bar{E}_r^D \end{bmatrix} \begin{bmatrix} P_r^S \\ P_r^D \end{bmatrix} = \begin{bmatrix} I \\ T_r \end{bmatrix} \begin{bmatrix} \bar{G}_r \end{bmatrix}$$

where

- Rows of P_r are path segments to/from the router
- Rows of E_r indicate how path segments sum to form paths
- $\bar{E}_r$ corresponds to spanning tree edges
- $\bar{G}_r$ is the corresponding subset of paths
- Rows of T_r consist of ± 1 entries (and zeros) corresponding to paths through the spanning tree

Matrix factorization perspective

Junction elimination yields

$$PG = \begin{bmatrix} I \\ T \end{bmatrix} \bar{G},$$

where the first factor is a product of matrices of the form

$$\begin{bmatrix} I \\ \tilde{T}_k \end{bmatrix}$$

with rows of $\tilde{T}_k$ representing paths through router graphs.

Matrix factorization perspective

Can combine with topology virtualization:

$$PG = \begin{bmatrix} I \\ T \end{bmatrix} \hat{G} S$$

where S is a zero-one virtualization matrix.

Summary

Path matrix G useful for network inference:

- Measure a few paths, infer rest
- Measure a few paths, estimate link behaviors

Cost of factoring G limited scalability.

New factorization for the path matrix G is

- Compact (proportional to G in storage)
- Easy to compute
- Nearly rank-revealing
- Faithful to the underlying problem structure

Questions?

I have lots more questions:

- How should we process paths for load balancing, etc?
- Can these methods be distributed?
- What else can we infer about networks with this machinery?
- Are there other places where this factorization applies?

Questions from you?