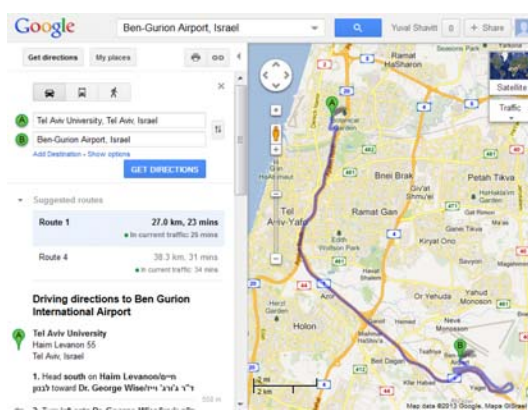
Network Oracles
Distances and Shortest Paths

Yuval Shavitt
School of Electrical Engineering

TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

Motivation

- Large Graphs server
  - Shortest route queries



Motivation (cont.)

- Large Graphs server
  - Distance queries

GCC mirror sites

Our releases are available on the GNU FTP server and its mirrors. The following sites mirror the gcc.g
(Phoenix, Arizona, USA) directly:

- Austria: gd.tuwien.ac.at, thanks to Antonin.Sprinzl at tuwien.ac.at
- Bulgaria: gcc.igor.onlinedirect.bg, thanks to igor at onlinedirect.bg
- Canada: http://gcc.parentingamerica.com, thanks to James Miller (jmiller at parentingamerica.co
- Canada: http://gcc.skazkaforyou.com, thanks to Sergey Ivanov (mirrors at skazkaforyou.com)
- France (no snapshots): ftp.lip6.fr, thanks to ftpmaint at lip6.fr
- France, Brittany: ftp.irisa.fr, thanks to ftpmaint at irisa.fr
- France, Versailles: ftp.uvsq.fr, thanks to ftpmaint at uvsq.fr
- Germany, Berlin: ftp.fu-berlin.de, thanks to ftp at fu-berlin.de
- Germany: ftp.gwdg.de, thanks to emoenke at gwdg.de
- Germany: ftp.mpi-sb.mpg.de, thanks to ftpadmin at mpi-sb.mpg.de
- Germany: http://gcc.cybermirror.org, thanks to Sascha Schwarz (cm at cybermirror.org)
- Greece: ftp.ntua.gr, thanks to ftpadm at ntua.gr
- Hungary, Budapest: robotlab.itk.ppke.hu, thanks to Adam Rak (neurhlp at gmail.com)
- Japan: ftp.dti.ad.jp, thanks to IWAIZAKO Takahiro (ftp-admin at dti.ad.jp)
- Japan: ftp.tsukuba.wide.ad.jp, thanks to Kohei Takahashi (tsukuba-ftp-servers at tsukuba.wide.ad.jp)
- Latvia, Riga: mirrors.webhostinggeeks.com/gcc/, thanks to Igor (whg.igp at gmail.com)
- The Netherlands, Nijmegen: ftp.nluug.nl, thanks to Jan Cristiaan van Winkel (jc at ATComputing.nl)
- Slovakia, Bratislava: gcc.fyxm.net, thanks to Jan Teluch (admin at 2600.sk)
- UK: ftp://ftp.mirrorservice.org/sites/sourceware.org/pub/gcc/, thanks to mirror at mirrorservice.org
- UK, London: http://gcc-uk.internet.bs, thanks to Internet.bs (info at internet.bs)
- US, Saint Louis: http://gcc.petsads.us, thanks to Sergey Kutserey (s.kutserey at gmail.com)
- US, San Jose: http://www.netgull.com, thanks to admin at netgull.com

arXiv mirror sites

- cn.arXiv.org (China)
- fr.arXiv.org (France)
- de.arXiv.org (Germany)
- in.arXiv.org (India)
- jp.arXiv.org (Japan)
- es.arXiv.org (Spain)
- uk.arXiv.org (U.K.)
- lanl.arXiv.org (née xxx.lanl.gov, U.S. mirror at Los Alamos)
- arXiv.org (U.S. primary site at Cornell University)

Answering a Query

- Graphs are large, but not too much
  - Can run SP algorithm in seconds or minutes
  - Too slow for answering queries
  - Too much CPU to answer many queries
- We want to answer many queries fast

Distance Query Usage

Closest mirror selection:
- Fairly gross estimation is sufficient.

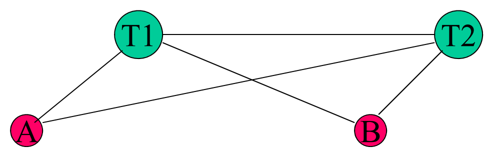
Application layer construction:
- Application level multicast trees
- Optimizing overlay routing

Building a Distance Query

- Trade-off CPU with storage: pre-calculate all pair shortest paths
- Memory requirement is $O(n^2)$ – too large
  - n=100k nodes $\Rightarrow$ table size 10^{10} entries
  - Internet AP (500k) $\Rightarrow$ table size 2.5×10^{11} entries
- Not practical

The IDMaps Approach

- Select $t \ll n$ points (tracers) in the graph
- Calculate and store the t^2 distances among them
- Calculate and store the $n \cdot t$ distances between each tracer and the rest of the graph
- Distance: min{A-T1-B, A-T2-B, A-T1-T2-B}



[Trans. on Net. 2001, Francis *et al.*]

Questions and Challenges

- How many Tracers do we need?
- Where Tracers should be located?
- Do we need to calculate all the t^2 distances?
  - What is the tradeoff between overhead and accuracy?
- Do we need to calculate all $n \cdot t$ distances ?

Tracer Placement

Number of Centers:

Given a network G with *n* nodes, a bound *d*, find a smallest set of centers S_C such that the distance between any node *i* and its closest center $C_i \in S_C$ is bounded by *d*.

minimize N

s.t., $S_C \subseteq V$, $| S_C | = N$, and

$\forall\ v \in V$: $d(v, C_v) \leq d$

Tracer Placement

Given a network G with *n* nodes place K Tracers where it minimize the maximum distance between a node and the nearest Tracer.

This problem is known as the minimum K-center problem.

The distance should satisfy the triangle inequality.

k-HST
k hierarchical well-seperated trees

- An attempt to solve both problems together
- An adaptation of an algorithm that was designed for a different problem

[Bartal, FOCS 1996]
[Awerbuch & Shavitt, Trans. on Net., 2001]

k-HST

Recursively partition the graph:

- Select an arbitrary node from current (parent) partition
  - All the node within a random radius form a new (child) partition
  - The radius is a factor of *k* smaller than parent partition radius
- Recurse until all partitions and singletons
- Build a virtual tree of partitions using child-parents
- Embed the tree

k-HST

The randomization of the partition radius is done so that the probability that a short link is cut by the partition decreases exponentially as one climb the tree.

⇒nodes close together are more likely to be partitioned down the tree

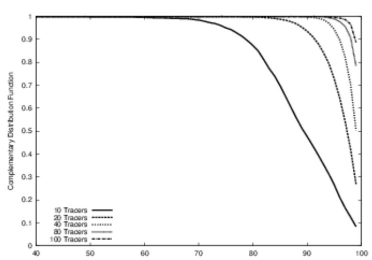
Using the *k*-HST

- Starting from the tree root – push the tracer location down until the diameter constraint is reached.
  - Place the Tracers in the corresponding partition centers.
- Given a budget of centers
  - Push the Tracers down from the largest diameter until meeting the budget

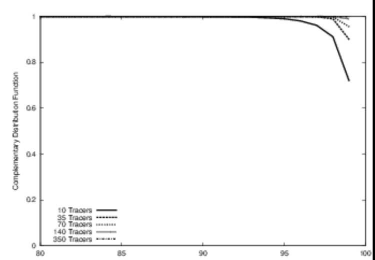
Minimum *k* centers

- Known to be an NP complete problem
- A factor 2 approximation is solvable in O(N|E|)

Effect of Tracer Number



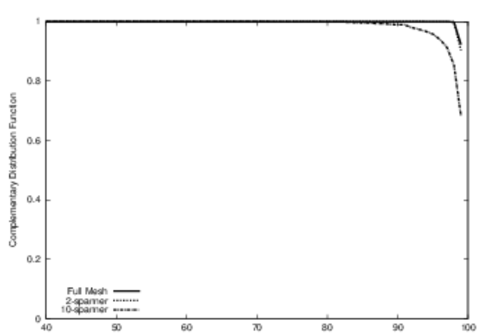
a. 1,000-node Tiers network



b. 4,200-node Inet network

Tracer-to-Tracer distance

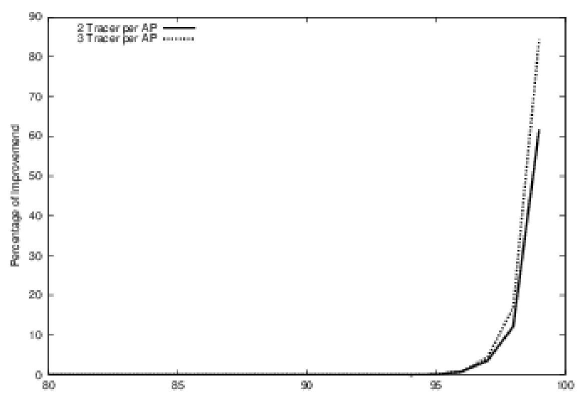
- Storing all t^2 links may be too large.
- A graph with 500,000 nodes (Internet APs)
  - Say t=5,000 require us to hold 25M entries
  - Important if links need to be ‘maintained’
- A simple reduction
  - Using a Spanner



Effect of t-spanner on 1,000-node Inet network with 100 Tracers.

Tracer to Node Table Size

- Maintain for each node distances to some closest Tracers
  - Fixed number
  - Based on the partition diameter



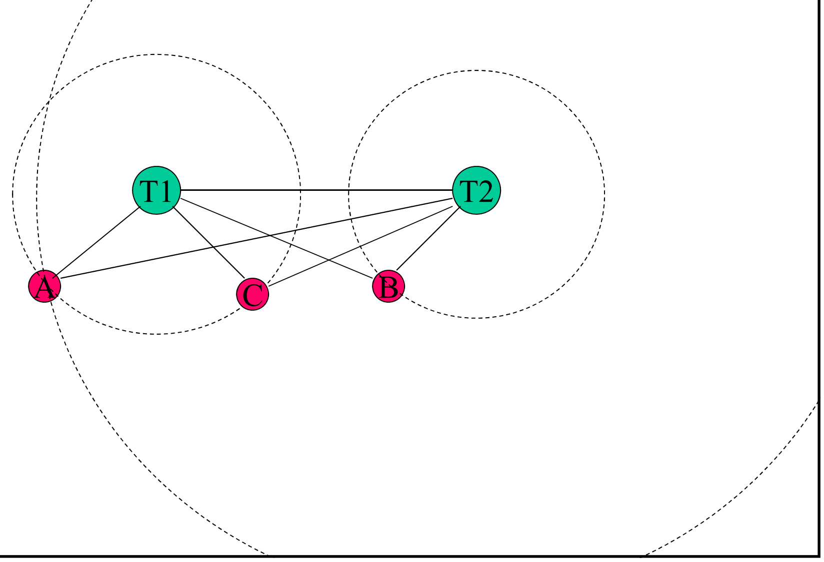
Mirror selection on 1,000-node Waxman network, 10 Tracers,

IDMaps

- Advantage:
  - Can be easily distributed (for the Internet)
- Main ideas:
  - Spread enough Tracers in the Internet
  - Each tracer measure the distance to all (or closest) AP
  - Tracers measure the full clique among them

[Trans. on Net. 2001, Francis *et al.*]

No Sense of Geometry



Pro & Cons

- Excellent results for mirror selection
  - With relaxed requirement:
    $\forall i \; 1 \leq i \leq K \; d(s_j, c) \leq \alpha d(s_i, c) + \beta$
- Bad estimation of short distances

Embedding

Given a weighted graph, embed the nodes in some metric space, such that:

- The distance in the embedding space is close to the distance in the graph
- Hope: multi-link path distances will be well estimated, as well.

Embedding Solutions for Networking

Distortion = Max{ Real dist. /computed dist., computed dist. / Real dist.}

- GNP [Ng and Zhang, Infocom'02]
  - Euclidean Embedding in R^d, down-hill-simplex
  - Not accurate, high max/var symmetric distortion
- BBS [Tankel and Shavitt, Infocom'03]
  - Accurate and Scalable Euclidean Embedding in R^d
  - Under estimation errors for long distances
- Hyperbolic Embedding [Tankel and Shavitt, Infocom'04]
  - Improves embedding in some cases

Other Embedding Methods

- Semi-Definite Programming (SDP)

  Best known theoretical result- **[Linial *et al*. 95]**
- Multi-Dimensional Scaling (MDS)

  Simple and low complexity implementation
- Down-Hill Simplex (DHS)

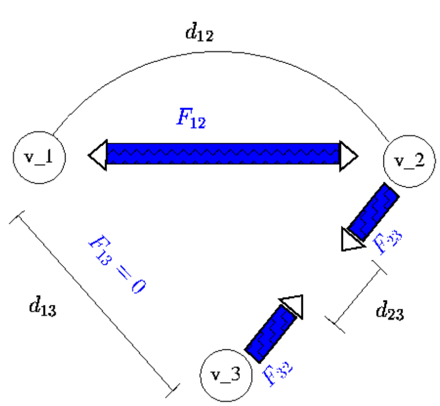
  Used in GNP **[Ng Zhang 02]**

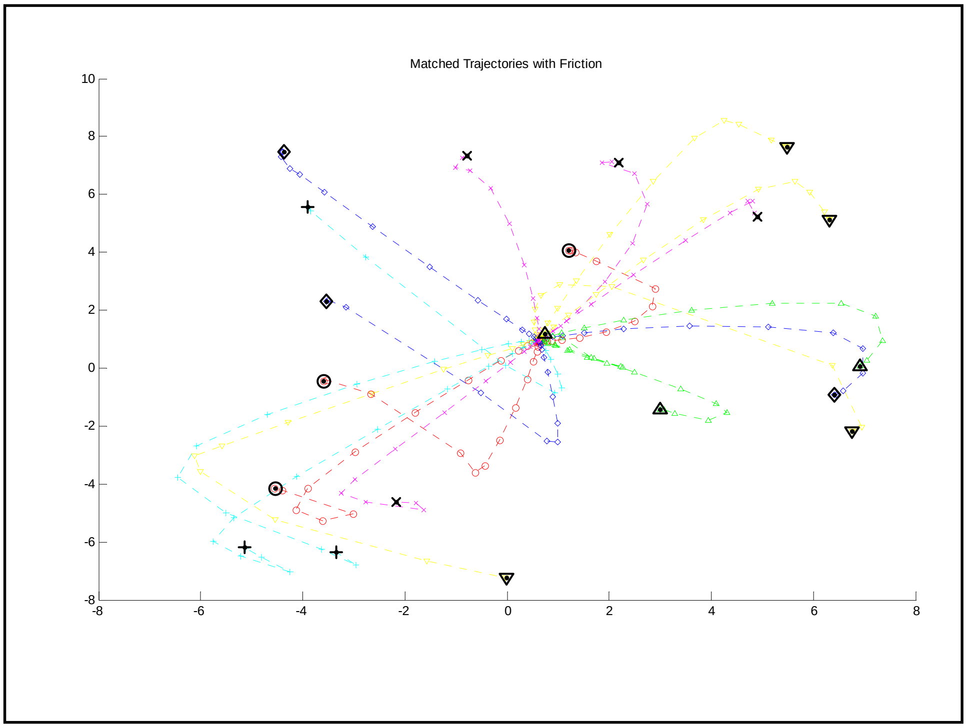
BBS - Basic Idea

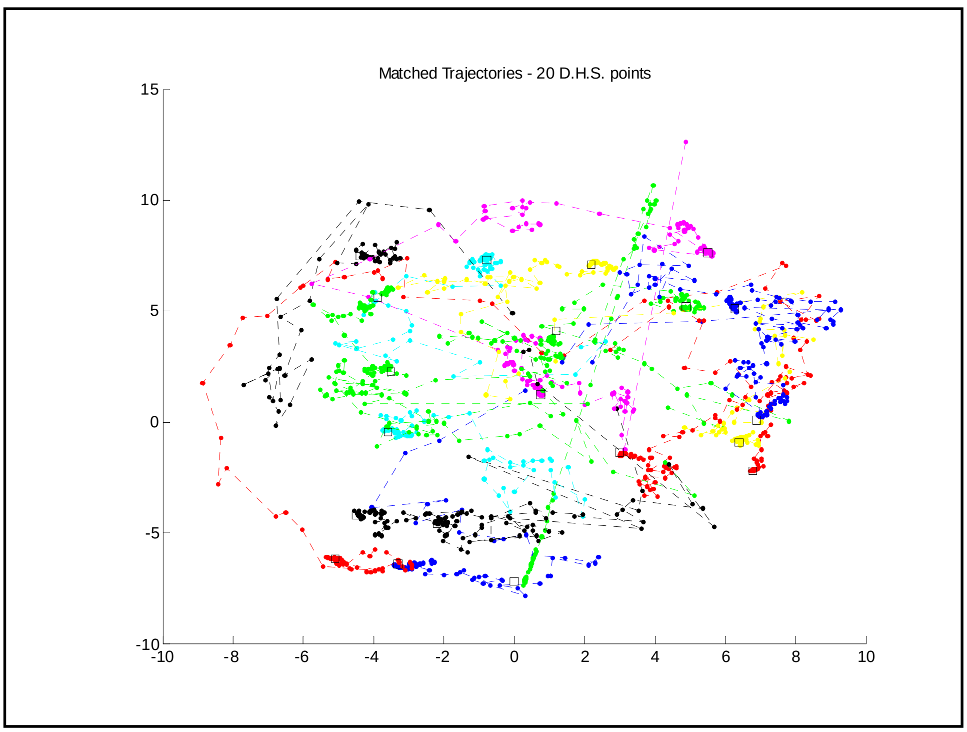
A physical model:

- particles = network nodes (Tracers, clients)
- inter-particle force & friction = difference between measured and embedded distances
- Kinetic energy = drive particles out of local minima of the error function

Inter Particle Forces

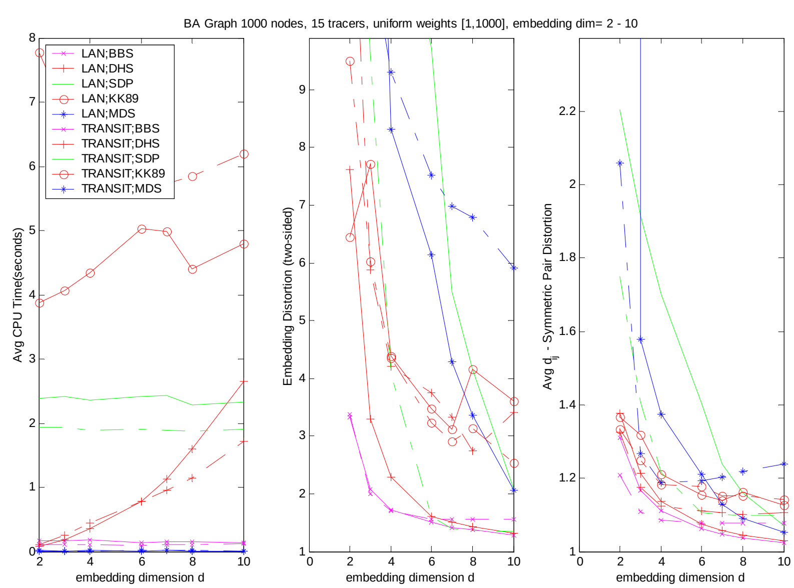



Matched Trajectories with Friction
10
8
6
4
2
0
-2
-4
-6
-8
-8
-6
-4
-2
0
2
4
6
8


Matched Trajectories - 20 D.H.S. points
15
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10

BBS Features

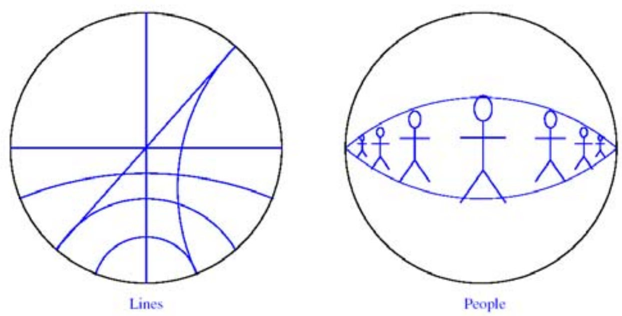
- Particles with larger estimation errors move faster
- Equilibrium points of the potential function are points where the field force, F_i, is zero for all particles V_i
- Friction slows down particles so they can slip into potential wells.



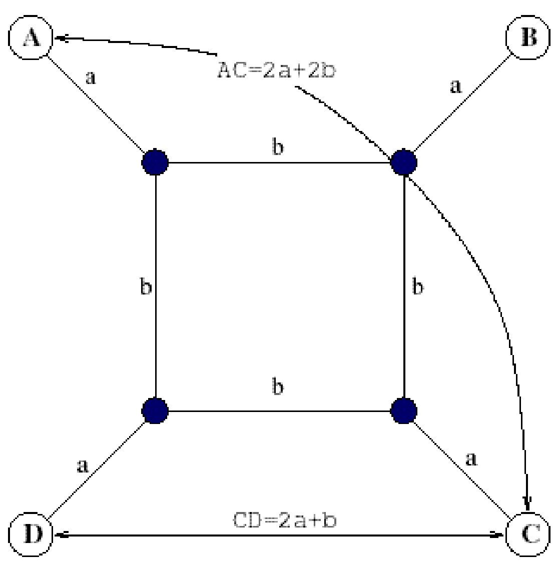
Hyperbolic Embedding

Due to Internet economics, routes tend to pass through the center

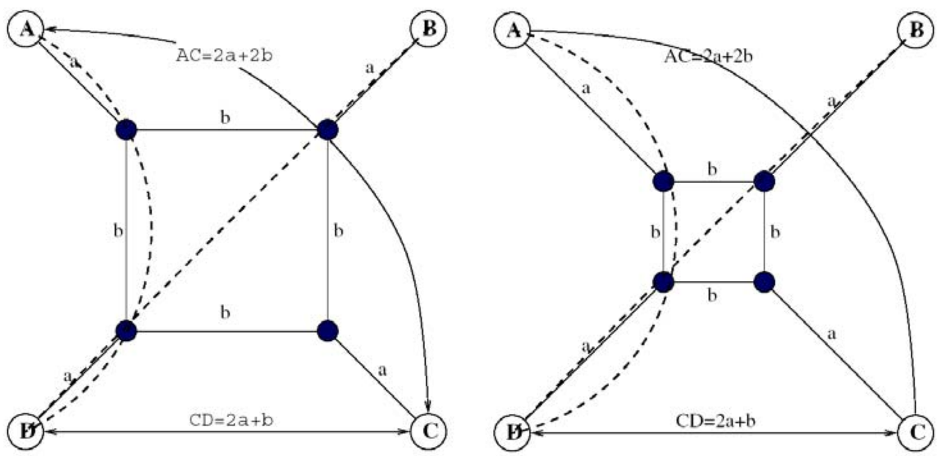
Example: Poincare disk D^2



Embedding Example in D^2



Curvature and Distances Ratio



Embedding Methods

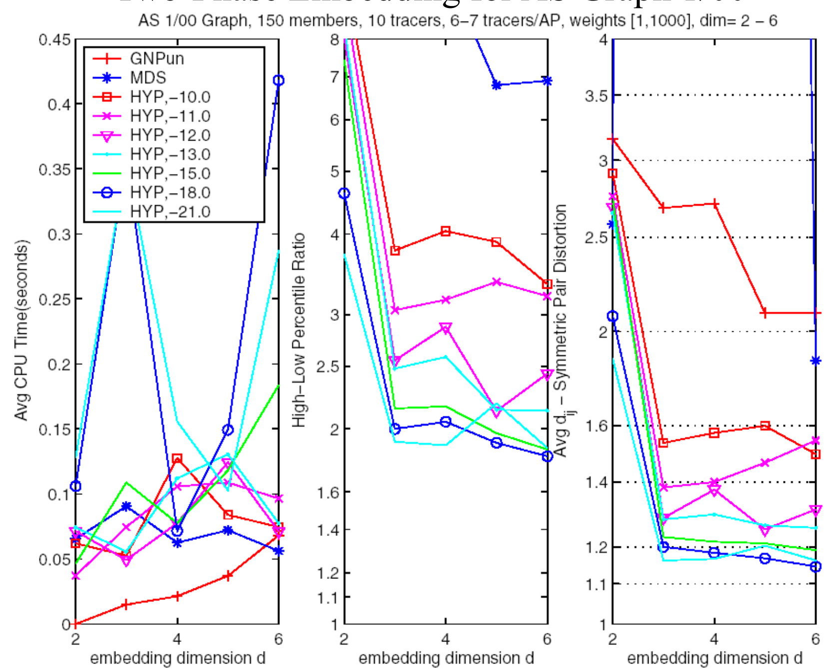
- ~~All pair (AP)~~ Not scalable
  - ~~Embed *n*-nodes metric, *n*(*n*-1)/2 distance pairs, at once~~.
- Two phase (TP)
  - Embed Small subset of *t* *Tracers*, *t*(*t*-1)/2 distance pairs.
  - For each of the other nodes, embed its distances to several *nearest* Tracers.
- Random + Neighbors (RN)
  - Embed with distances to
    - The 1-neighborhood
    - Order of **`log(n)`** peer nodes, selected uniformly at random.
  - No fixed tracers

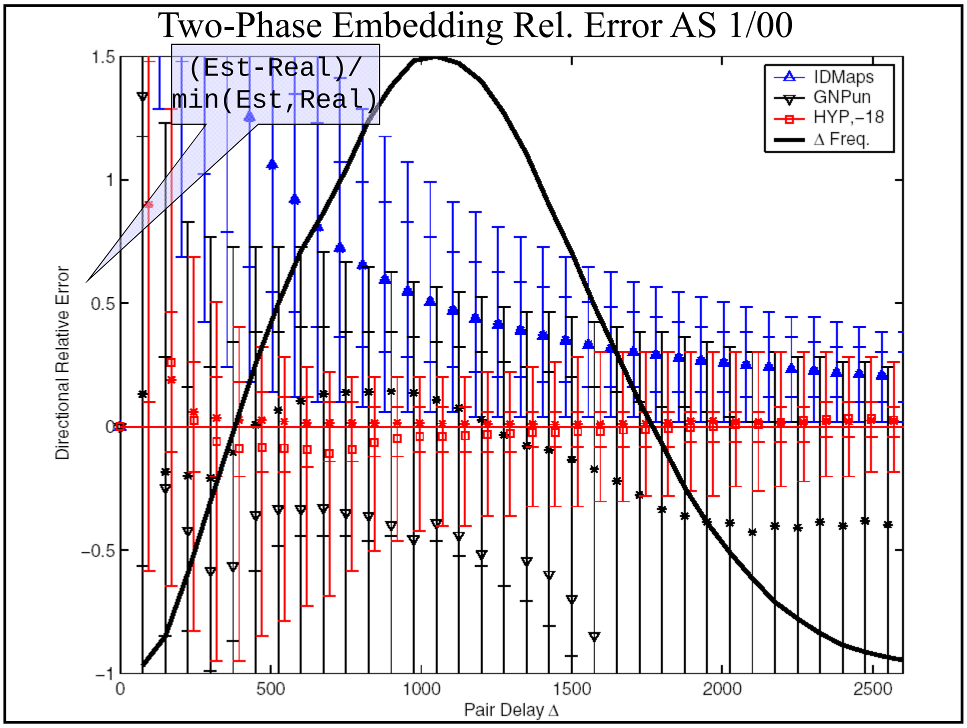
Rand. Neigh. vs. Two Phase

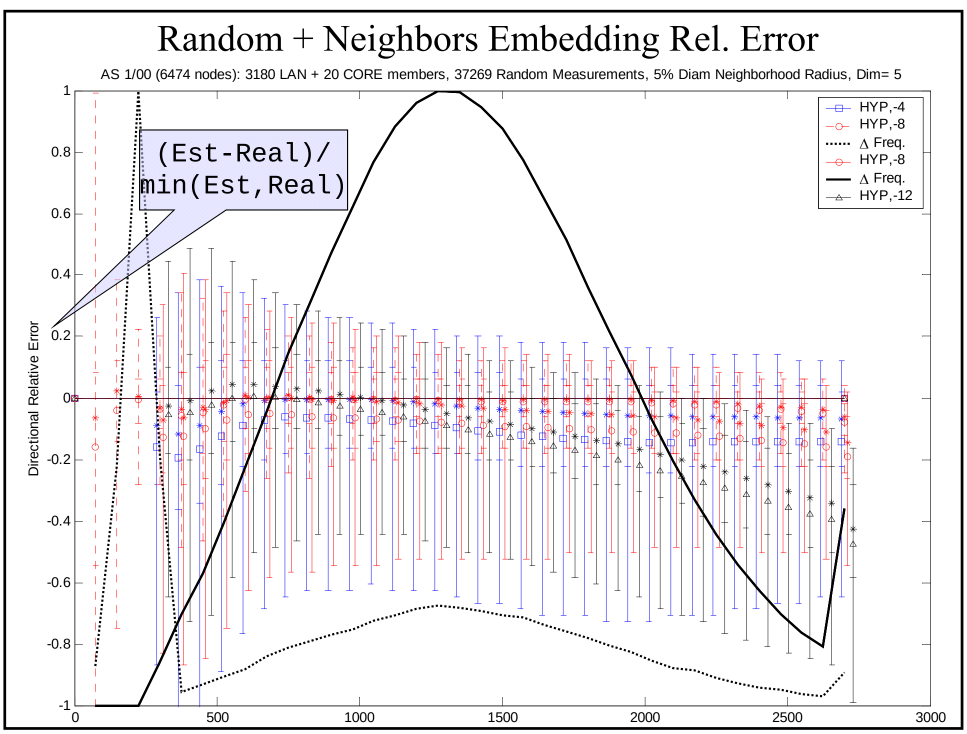
Two Phase
- Non-symmetric
- Distributed
- Over-estimation of short distances
- Sensitive to Tracer failure

Random + Neighbors
- Symmetric
- Central calculation
- Equally accurate for all distances.

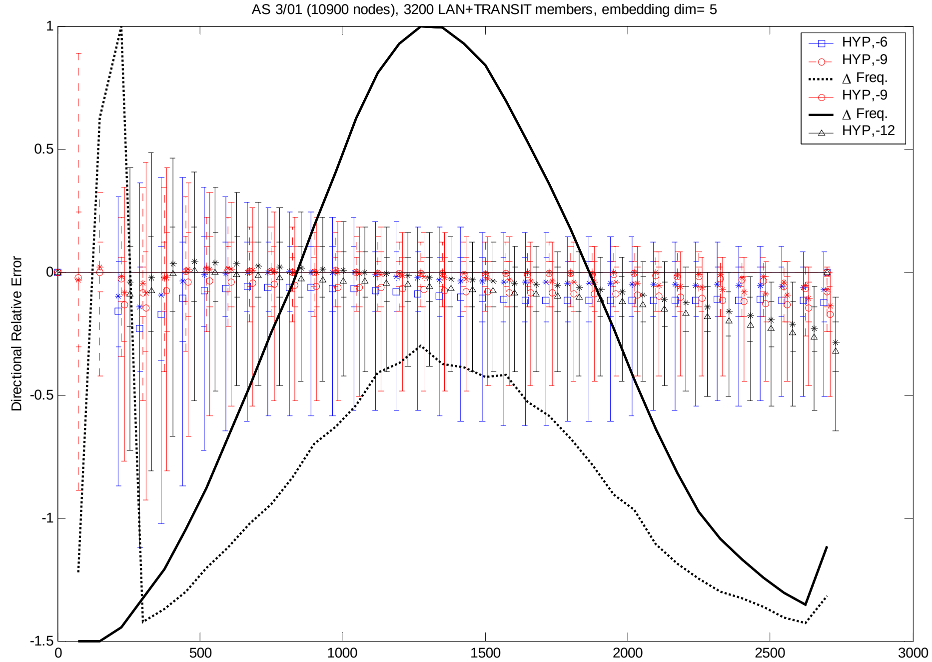
Two-Phase Embedding for AS Graph 1/00



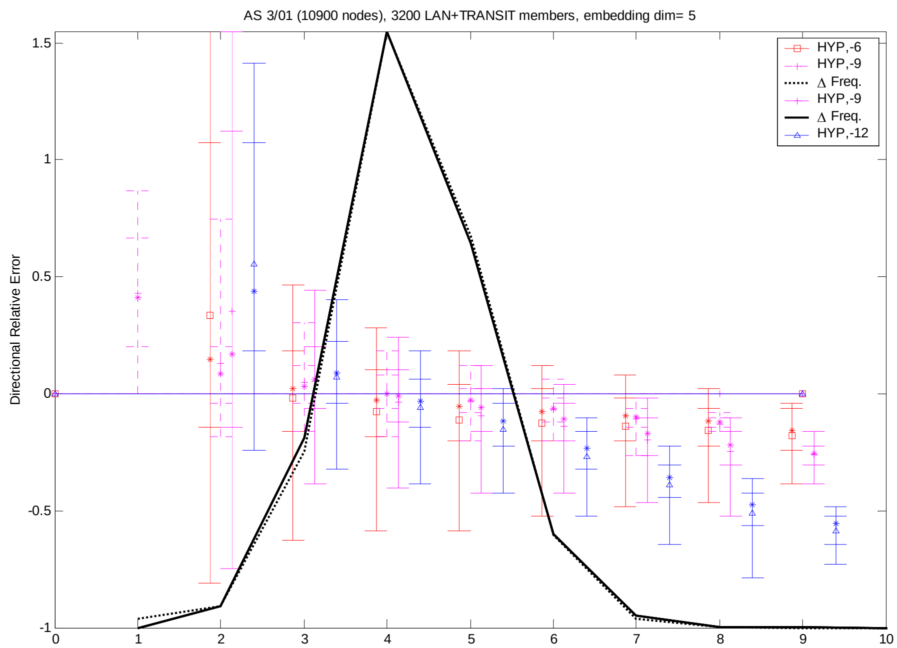

Two-Phase Embedding Rel. Error AS 1/00
(Est-Real)/
min(Est,Real)
IDMaps
GNPun
HYP,−18
Δ Freq.
Directional Relative Error
Pair Delay Δ
1.5
1
0.5
0
−0.5
−1
0
500
1000
1500
2000
2500


Random + Neighbors Embedding Rel. Error
AS 1/00 (6474 nodes): 3180 LAN + 20 CORE members, 37269 Random Measurements, 5% Diam Neighborhood Radius, Dim= 5
(Est-Real)/
min(Est,Real)
HYP,-4
HYP,-8
Δ Freq.
HYP,-8
Δ Freq.
HYP,-12
Directional Relative Error
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000

Random + Neighbors Embedding Rel. Error AS 3/01



Random + Neighbors Embedding Rel. Error AS 3/01



Nearly Tight Low Stretch Spanning Trees

Any graph G with n points has a distribution $\mathcal{T}$ over spanning trees such that for any edge (u, v) the expected stretch $\mathrm{E}_{\mathrm{T}\sim\mathcal{T}}\,[\mathrm{d_T}(u, v)/\mathrm{d_G}(u, v)]$ is bounded by $\tilde{O}(\log n)$.

Can be extended for weighted graphs.

[Abraham, Bartal, Neiman, FOCS 2008]

Shortest Path Oracle with PreProcessing

- The stretch of a tree is not practical
- Build a DAG that captures the distances in the graph (pre-processing)
  - Hierarchical $\Rightarrow$ Logarithmic calculation time
  - Linear size
- Use the DAG to calculate shortest path for a point to point query
  - Logarithmic time

Multi-level Proximity Routing (MPR)

- An hierarchical soft clustering structure for a weighted graph G = (V,A).
- A query algorithm is answering pair distance queries by searching the paths of the source and the destination in the hierarchy.
- The result is an approximation of the shortest path.
- Can be approximated

MPR Hierarchical Construction

- The input graph is the level 1 graph
- Building level l+1 graph (aggregation)
  - Select:
    - each l-level node scores its neighbors
    - Scores are used to decide which nodes are selected to the higher level
  - Interpolate
    - Connect the l+1 level nodes using 1-, 2-, or 3-hop paths
  - Post filter
    - Remove redundant links

Score Stage

- Sub graph G_i, contains node i and its 2-neighborhood, and the links from i and its 1-neighborhood to its 2-neighborhood
- The coverage set of neighbor j of node i

$$S_j^i = \{x | i \rightsquigarrow j \rightsquigarrow \ldots x \text{ is a shortest path in } G_i\} ,$$

→Select score of neighbor j of node i

$$s_j^i = \frac{\left|S_j^i\right|}{A(i,j)}$$

Select Stage

- Each cluster head i which is not selected iteratively select neighbors $j_1, j_2 \ldots j_k$ with maximum select score until

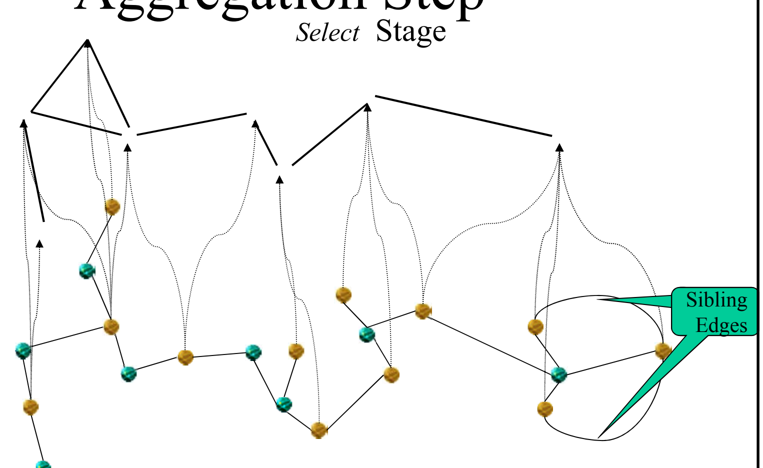
$$\sum_{k \le p_i} s_{j_k}^i > \gamma_p \sum_{j \in N_i^{\langle l \rangle}} s_j^i$$

- Here $\gamma < 1$ is the aggregation factor.
- Increasing it yields more optimal but denser MPR, with larger memory and run time complexities.

Interpolate Stage

- l-level path types among parents $\{p_1 \rightsquigarrow p_2\}$, $\{p_1 \rightsquigarrow u_1 \rightsquigarrow p_2\}$ and $\{p_1 \rightsquigarrow u_1 \rightsquigarrow u_2 \rightsquigarrow p_2\}$ where p_1 and p_2 are selected parents of their unselected child u_1 and u_2 respectively.
- The least cost computed path between v_i and v_j is the corresponding edge weight $w_{ij}^{<l+1>}$.
- In order to reduce the aggregation complexity, the edge $e_{ij}^{<l+1>}$ is **filtered** if $w_{ij}^{<l+1>}$ is not less than $\min_{k\neq i,j} w_{ik}^{\langle l+1\rangle} + w_{kj}^{\langle l+1\rangle}$.

Aggregation Step



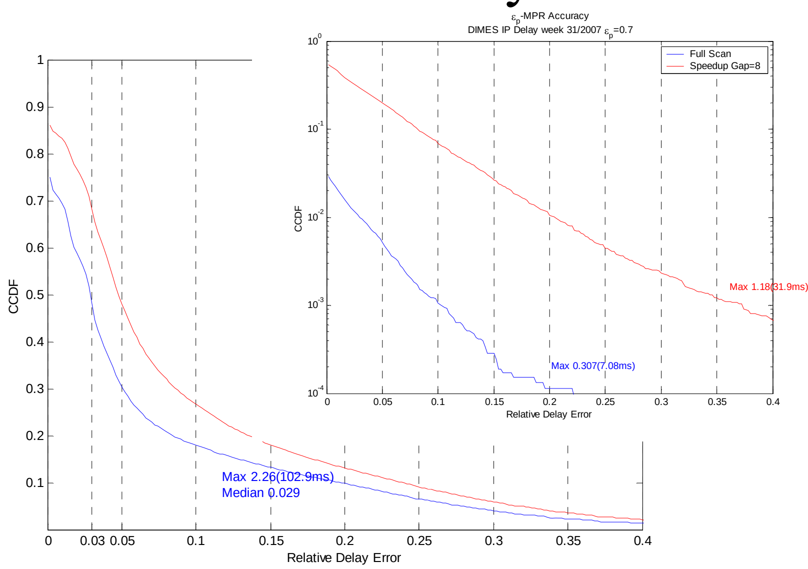
MPR Experiments

Graph	Nodes	Arcs/ Edges	Basic MPR CPU*	
			Build sec	Query ms
DIMES IP Delay w16/08	138721	602970 (directed)	47.7	0.15
DIMACS 9'th Euro-Road	18010173	42188664 (directed)	1681.3	0.42
Simulate Ad-Hoc	1281966	8554957 (undirected)	688.1	1.9

ε-MPR Aggregation

- Basic (Heuristic) Aggregation is accurate enough (tunable threshold) for most pairs
- No tight worst-case analysis
- Select enough parents until the tractability conditions are satisfied:
  - ε_P-stretched paths among **parents**
  - ε_C-stretched arcs among adjacent children
- If ε_P=0 →(1+ε_C)-stretched query

DIMES IP Delay



Euro-Road Network