Big-Bang Simulation for Embedding Network Distances in Euclidean Space

Yuval Shavitt and Tomer Tankel

Abstract— **Embedding of a graph metric in Euclidean space efficiently and accurately is an important problem in general with applications in topology aggregation, closest mirror selection, and application level routing. We propose a new graph embedding scheme called Big-Bang Simulation (BBS), which simulates an explosion of particles under force field derived from embedding error. BBS is shown to be significantly more accurate, compared to all other embedding methods including GNP. We report an extensive simulation study of BBS compared with several known embedding schemes and show its advantage for distance estimation (as in the IDMaps project), mirror selection and topology aggregation.**

I. Introduction

Knowledge of the distances between all pairs of a group of nodes can improve the performance of many practical networking problems, such as routing through a subnetwork and selecting the closest mirror server. However, measuring and to a greater extend dissemination of this information becomes impractical even for a few tens of nodes, since the number of node pairs is quadratic in the number of nodes. Thus, researchers sought ways to reduce the all pair distance representation while preserving the distance in the reduced representation as close as possible to the original ones. Next we shortly describe two example networking problems, where all pair distance information is required.

Routing through a subnetwork. When routing through an ATM sub-network, the distances between all pairs of border nodes, i.e., nodes that are connecting the sub-network to other sub-network, are used to compute the shortest (or cheapest) path through the cloud [1]. For this end, each network advertises its distance matrix in a compressed manner, and it is recommended that the matrix representation is smaller than $3b$, where b is the number of border nodes [1]. The best compression technique that was suggested in the past [2], [3] will be presented later.

This research was supported by a grant from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

corresponding author: tankel@eng.tau.ac.il

The authors are with the Dept. of Electrical Engineering, Tel-Aviv University, Israel.

Selecting the closest mirror. Recently, there was a large interest in using distance maps of the Internet to aid in tasks such as closest mirror selection and application layer multicast [4], [5], [6], [7]. In the IDMaps project it was identified that the number of possible nodes which represent the distance map granularity is in the thousands which makes accurate distance dissemination impractical. Due to the practicality of the measurement process and to reduce the representation, IDMaps suggests to use a smaller number t of measurement points, *Tracers*, that measure distances among themselves and then use them as a reference distance map to the other network regions.

A relatively new approach to represent a network distance matrix is to map network nodes into points in a real Euclidean space. Such a mapping is designed to preserve the distance between any pair of network nodes close to the Euclidean distance between their geometric images. Such a mapping is called an embedding and the embedding dimention d is the dimention of the Euclidean space. Ideally graph edge lengths are exactly embedded in the geometric edges. However, it can be easily shown that an exact embedding is not always possible, e.g., in case of a tree, and in-fact, in most cases embedding introduces some distortion. In a 'good' embedding, the average and maximum distance distortion over all pairs of nodes are relatively low. The distance distortion is defined for each pair as the maximum of the ratio between the original and Euclidean distance and its inverse.

Outside the networking community embedding has been used for quite a long time in many diverse research areas. Multi Dimensional Scaling (MDS) is widely used in areas of statistics and vision, in which its simplicity and efficiency is an advantage over more complicated methods. Classical metric MDS [8] develops the metric as a symmetric bilinear form and calculates the leading d eigene values of the corresponding matrix[1]. Recently computer graphics researchers [9] suggested to use MDS for mapping flat textures over curved surfaces with minimum distortion. Embedding is used extensively in bio-informatics, and specifically for classification of protein

[1]Finding the leading eigenvalues of the matrix of a symmetric bilinear form is equivalent to Singular Value Decomposition (SVD).

sequences into similarity families [10].

Embedding of graph edges in the Euclidean plane is also achieved by the *spring embedder* algorithm for graph drawing applications, This algorithm is a heuristic based on a physical model which was described first in [11]. The heuristics which followed this direction, such as [12], [13], [14], are reviewed in a recent survey [15].

Theoretical bounds on the **maximal** pair distortion and the dimension of the target space were derived by Linial et al. [16] for any discrete metric space[2]. Perhaps the first use of graph embedding techniques in networking is due to Ng and Zhang [6] who suggested to estimate Internet host distances by embedding distances among Tracers nodes and other network regions. Their embedding technique sought minimum of the total square embedding errors over all nodes pairs, which is proportional to the **average pair** distortion.

In this paper we present a new scheme for embedding, based on a novel idea, utilizing notions from Newtonian mechanics. We shall compare our scheme, *Big-Bang Simulation (BBS)*, to other embedding methods and show that it produces the best embedding over various parameter choices with reasonable complexity. This is not to say that for special cases, e.g., when the number of embedded nodes is very small, it will outperform all other schemes, but it will be better than any other scheme when all cases are considered.

a) Big-Bang Simulation: BBS models the network nodes as a set of particles, each is the 'geometric image' of a node, that is the position of this particle in Euclidean space. The particles are traveling in that space under the effect of potential force field. The force field reduces the potential energy of the particles, that is related to the total embedding error of all particle pairs. Each pair of particles is pulled or repulsed by the field force induced between them depending on their pair embedding error, that is the embedding error of the distance between them. As a particle accelerates under the effect of the force field it is also attenuated by simulated friction force.

The BBS scheme advantage over conventional gradient minimization schemes, such as steepest decent and downhill simplex (DHS), is that the kinetic energy accumulated by the moving particles enables them to escape the local minima. Moreover, DHS which was used by Ng and Zhang [6] is very sensitive to the initial vertices coordinates. The key idea which inspired the name 'Big-Bang Simulation' is that all particles are **initially placed at the same point**, the origin. Starting with such initial condition BBS obtains 'good' embedding, that is **low average** distortion and **quite low maximal** distortion, in several hundreds simulation iterations for input graph sizes in the range $30 < n < 750$. The above good performance is equally achieved for to a wide range of system friction coefficient.

[2]Though [16] and more recent publications discusses mapping to l_p, we will concentrate only on embedding in l_2 which is the most popular in applications.

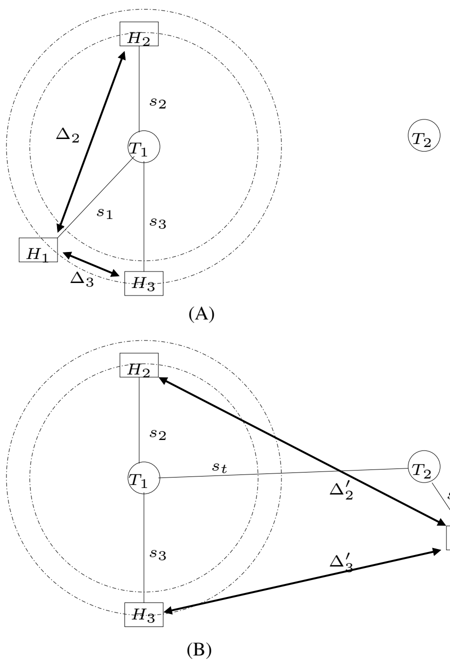


Fig. 1: Problem Statement

Next we discuss in more details one of the applications mentioned above for which we have applied our embedding and compare it to previous results.

b) Internet Distance Maps Application: While it was shown that IDMaps performs quite well, e.g., it correctly points a client to the closest mirror server in about 85% of the cases [5], it is far from being ideal.

Fig. 1 illustrates the main reason for inaccuracies in IDMaps: In the example, client H_1 estimates the network distance Δ_l to mirrors $H_l,\ l = 2, 3$ as a sum of several measured segments, $s_1 + s_t + s_l$. The inaccuracy of IDMaps estimation is larger when a nearest Tracer T_1 is shared by the two nodes. For instance, in the case depicted in Fig. 1A, the shortest network distance is $\Delta_3 \ll \Delta_2$ although the smallest sum of segments is $s_1 + s_2 < s_1 + s_3$. On the other hand, in the case depicted in Fig. 1B, the client is located near the Tracer T_2, and both mirror hosts are located near another Tracer T_1. The distance s_t between T_1 and T_2 is larger than two times the distances from the client and mirror hosts to their nearest Tracer. Thus the distances $\Delta'_l;\ l = 2, 3$ are both dominated by s_t and are approximately equal to the estimated sum $s_1 + s_t + s_l$.

Having realized that IDMaps is blind to position, Ng and Zhang [6] suggested to use coordinate–based mechanisms to predict Internet network distances: Tracers are embedded in an Euclidean space and each network region finds its image coordinates according to distances measured from the region to some Tracers. The estimated distance between two regions is then given by the distance between their images in Euclidean space.

However, due to the Internet AS structure that has a core in the middle and many tendrils connected to it [17] embedding of distances between nodes located in different tendrils will result in large embedding error. Thus, we suggest a **threshold criterion** to select either the Euclidean distance or the additive estimation of IDMaps. Our criterion performs well for multiple types of input graphs and different Tracers selection algorithms, and outperforms both the additive IDMaps and GNP.

c) Dynamic Distance Map: Due to the dynamic character of real networks, the calculated distance map has to be updated to track network distance changes. like in IDMaps [5], we expect only part of the Tracer pair distances to be measured at every measurement cycle. In the following, we discuss the map recalculation due to changes. We distinguish between two scenarios: Slow Variation, where less than 20% of the Tracer pair distances changes, and Large Transient, where more than 20% of the Tracer pair distances are changed between consecutive measurements.

Labovitz et al. [18] studied BGP route changes and concluded, that in reality Internet routing is stable. They found that over 80% of the routes change at a frequency lower than once a day. Similarly, Paxson [19] found that a very high percentage of the routes are stable. Given that our measurements cycle is 15–20 minutes long, we can expect the slow variation scenario to be the common one, where large transient would happen rarely, due to major failures or attacks.

Slow Variations. Such update cycles recur constantly. In this case one does not need to recalculate the coordinates from scratch and can use the previous coordinates as the initial condition for the embedding calculation. We verified that BBS embedding is insensitive to small changes in the input graph, in case the pairwise field force is given by the difference between the Euclidean and the network pair distances (see Eq. 22). Indeed, even after modifying as many as 20% of the input edges, and resuming simulation from the previous particles positions, our simulation CPU time is only .5% (for 150 node Waxman topology) and 5% (for 15 node BA topology) of the CPU time of the full calculation.

Large Transient. When network topology undergoes major change, e.g., failure of major AS, we are required to recalculate all the coordinates. In IDMaps the number of measurements is $t^2 + rN$, where t is the number of Tracers, N the number of Address Prefixes(APs) and r the number of measurement per Adress Prefix. Due to the added accuracy of our embedding we can use fewer Tracers than IDMaps, but we need more measurements per AP. That is, we use smaller t values and pay with higher r values, $r = 5$ compared to $r = 2$ in IDMaps. This gives us a factor of 3 more measurements, which we feel worth the added accuracy.

The rest of the paper is organized as follows. The BBS simulation, initial condition, friction, and an example embedding in the Euclidean plane are discussed in next section. In Sec. III, BBS is compared to four other methods, using simulated graphs created according to both Waxman [20] and Barabási-Albert (BA) [21] methods. In Sec. IV we apply BBS in two practical applications, topology aggregation and Internet distance maps, comparing it with previous results from [22] and [5], [6], respectively.

II. Big–Bang Simulation

In this section we shall discuss the embedding of network distances using Big-Bang Simulation (BBS). We develop the potential field force equation, discuss the initial conditions for them and the effect of friction. Finally an example of simulated metric is given, which is embedded perfectly in the $2d$ Euclidean space, that is the linear plane.

A. The Model

The vertices of the graph, the network nodes, are modeled as a set of particles, traveling in the Euclidean space under the affect of potential force field. Each particle is the geometric image of a vertex. The field force is derived from potential energy which is equal to the total embedding error

$$E_T(v_1, v_2, \ldots, v_n) = \sum_{\substack{i,j=1 \\ i>j}}^{n} E_{ij}(v_i, v_j) \ . \qquad (1)$$

Here $v_k, \ \ k = 1, \ldots, n$ are vectors designating the coordinates of the n network nodes in the target Euclidean space $\mathcal{R}^d$. The distance embedding error of a pair of particles, the 'pair embedding error' is denoted by E_{ij}. The field force induced between the two particles either pulls or repulses the particle pair, aiming to reduce their pair embedding error.

The rational behind our approach is similar to locking an adaptive tracking loop using an increasing feedback constant to increase sensitivity, or decrease the tracking bandwidth. Our method thus comprises of several calculation phases performed sequentially. We start with an insensitive but less optimal potential function, and as we move along to the next phases we use a more and more sensitive and adequate potential function. The sensitivity of the pair error functions in each phase increases with respect to the directional relative error Eq. (33). Numerical stability is maintained, because earlier phases are normally capable of substantially reducing the maximum pair embedding error.

In the first phase, the induced pair field force is equal to the difference between the Euclidean and network pair distances. The resulting field force can be realized by attaching an ideal spring with fixed elastic coefficient to each pair of particles. The rest length of the spring in each pair is equal to the network pair distance. The objective of this phase is to find an approximate minimal energy configuration of the system of particles. The objective of the rest of the phases is to reduce the distortion of large relative error edges at the price of slightly increasing the average relative error. During each calculation phase the particles are traveling in trajectories which tends to reduce the potential energy of the entire system. At the end of each phase the system approximately achieves an equilibrium point where the potential energy is minimized.

At the beginning of the first phase, particles are placed at the origin. The field forces for the first iteration are chosen randomly. The initial position of particles in the next phase is at the point where the potential energy had achieved its global minimum in the previous phase. That point need not be the final position of the previous phase because the particles trajectories are stopped near but not precisely at equilibrium.

The potential field force in each calculation phase p is derived from a phase-specific potential energy, $E_T^{\langle p\rangle}$. The phase pair embedding error function denoted by $E_{ij}^{\langle p\rangle}$, assumes the form

$$\begin{aligned} E_{ij}^{\langle p\rangle}(v_i, v_j) &= \mathcal{F}(\|v_i - v_j\|, \Delta_{ij}), \\ \text{for } i \neq j \quad &\text{and} \quad v_i \neq v_j \end{aligned} \tag{2}$$

where $\|\mathbf{x}\|$ is the Euclidean size of vector $\mathbf{x} \in \mathcal{R}^d$, i.e. $\|\mathbf{x}\| = \sqrt{\sum_{l=1}^{d} x_l^2}$, and, $\Delta = (\Delta_{ij})$ is the distance matrix among network node pairs (i, j). The potential function of the first phase $E_T^{\langle 1\rangle}$, Eq. (21), is equal to the simple squared error used in [6].

A calculation phase consists of several iterations which move the particles in discrete time intervals. The particles positions and velocities calculated in the current iteration are the input to the next iteration. An iteration begins with calculating the field force $\vec{F}_i$ on each particle at current particles positions. In all iterations, except the first iteration of phase 1, the field forces are derived from the potential energy (1). Next, assuming constant field forces in time interval $(t,\, t+\delta t)$, apply Eq. (6-7) to calculate the positions and velocities at time $t+\delta t$. At the end of an iteration the new potential energy (1) and other statistics are calculated at the new particle positions. They include the maximum particle velocity $\max_i \|\dot{v}_i\|$ and maximum symmetric pair distortion d_{ij} Eq. (24).

The phase of calculation is ended if one of the following conditions, concerning the total energy E_T and rest statistics, are met:

1) Particles are almost perfectly embedded; The distortion satisfies either $E_T < \epsilon$ or $max_{i,j}(d_{ij}) < 1+\epsilon\prime$.
2) Particles are almost at halt; The maximum particle velocity decreases below threshold.
3) Particles are near an Equilibrium; The difference between local minimum and maximum of potential energy decreases below threshold.
4) Slow convergence rate of Energy; The reduction pace of potential energy is below threshold.
5) Divergence of Particles; The maximum velocity grows above threshold.

However, except for the first condition, which is always checked, the other conditions are checked only after the physical system's time is greater than phase's minimum period. The minimum period for the first phase equals 15 seconds, and minimum period for the other phases is 10 seconds.

The time step δt for the next iteration is adjusted according to the total energy and rest statistics. In the beginning of each phase the time step is small, to prevent the system from oscillating. The particles acquire velocity and a definite direction, which reduces the potential energy. The time step is gradually increased as the energy continues to decrease. There is a tradeoff between increasing the time step for greater numerical efficiency and keeping it small to detect and attract particles to global minimum points of the potential energy function. This tradeoff is handled by backtracking the particles to their previous position if the calculated statistics indicates minima of potential energy was skipped.

In each iteration we move all n particles, where $n \equiv |V|$ is the number of graph vertices being embedded. The force elements affecting each particle are induced by all other $(n-1)$ particles. A single iteration thus has a complexity of $O(n^2)$. Because the total number of iterations I is *independent* of n, the overall complexity of the method is $O(In^2)$, that is linear in the input size.

B. Movement Equations

The instantaneous acceleration of an object is, according to the second movement law of Newton

$$\vec{a} \equiv \frac{d^2}{dt^2}\vec{x} = \frac{\vec{F_C}}{m} \tag{3}$$

where $\vec{F_C}$ is the combined force affecting the object, $\vec{x}$ is its position at time t[3], and m is its mass. Assuming unit mass for all particles, $m = 1$, and neglecting the variation in the combined force $\vec{F_C}$ within a small time interval $(t,\, t + \delta t)$, we have

$$x(t + \delta t) - x(t) = \dot{x}(t)\delta t + \tfrac{1}{2}\ddot{x}(t)(\delta t)^2\ , \tag{4}$$

and

$$\dot{x}(t + \delta t) - \dot{x}(t) = \ddot{x}(t)\delta t\ ; \tag{5}$$

where $\dot{x}$, and $\ddot{x}$ denotes the first and second t-derivatives, that are the velocity and acceleration of the particle respectively. The combined force affecting particle i at position v_i is given by $\vec{F}_{Ci} = \vec{F_i} - \vec{Fr}_{i}$, where $\vec{F_i}$ is the field force affecting this particle, and $\vec{Fr}$ is its simulated friction force, discussed below. Thus we have[4]

$$\begin{aligned} \dot{v_i}(t+\delta t) - \dot{v_i}(t) &= \left(\vec{F_i} - \vec{Fr}_i\right)\Big|_t \delta t && (6)\\ v_i(t+\delta t) - v_i(t) &= \dot{v_i}(t)\delta t\ + \\ & \tfrac{1}{2}\left(\vec{F_i} - \vec{Fr}_i\right)\Big|_t (\delta t)^2 && (7)\end{aligned}$$

a) Initial Conditions: The initial conditions for all phases except the first phase are:

$$\left\{\begin{matrix} v_i(t)|_0 &=& v_i^* \\ \dot{v_i}(t)|_0 &=& 0 \end{matrix}\right\} \text{ for } i = 1,\, 2\, \ldots,\, n\ , \tag{8}$$

and

$$E_T^{\langle p-1\rangle}(v_1^*,\, v_2^*, \ldots) = \min_t E_T^{\langle p-1\rangle}(v_1,\, v_2, \ldots)\ ; \tag{9}$$

where $p > 1$ is the phase number. Thus at $t = 0$ all particles are at rest at the point where the potential energy achieved its global minimum along the particles trajectories of the previous phase. The initial conditions for first phase are given by

$$\left\{\begin{matrix} v_i(t)|_0 &=& 0 \\ \dot{v_i}(t)|_0 &=& 0 \end{matrix}\right\} \text{ for } i = 1,\, 2\, \ldots,\, n \tag{10}$$

that is all particles are at rest in the origin. At the origin however, its impossible to decompose the field force into induced forces between pairs since $v_{ji} \equiv v_j - v_i = 0$, for all $i,\, j$. At $t = 0$, in first iteration, we thus set the field forces at $t = 0$ to

$$\begin{aligned} \vec{F_i}(t)\Big|_0 &= \sum_{\substack{j=1\\ j\neq i_0}}^{n} \mathcal{F}_x(x, \Delta_{ij})\big|_{x=\|u(i,j)\|}\ \hat{u}(i,j) && (11)\\ u(i,j) &= \begin{pmatrix} u_1^{ij} & u_2^{ij} & \ldots & u_d^{ij}\end{pmatrix}\ ; \end{aligned}$$

where $\hat{x}$ denote the unit vector along the course of vector x. Here u_l^{ij}, that are coordinates of vectors $u(i,j)$, are independent random variables uniformly distributed across the interval $(0,\, 1]$. The pair embedding error function of phase 1 $\mathcal{F}^{\langle 1\rangle}$ and the derived field forces are given in section II-E below.

Substituting (11, 10) in the approximate velocity and position equations (6-7) we find the position and velocity of all particles at time $t = \delta t$.

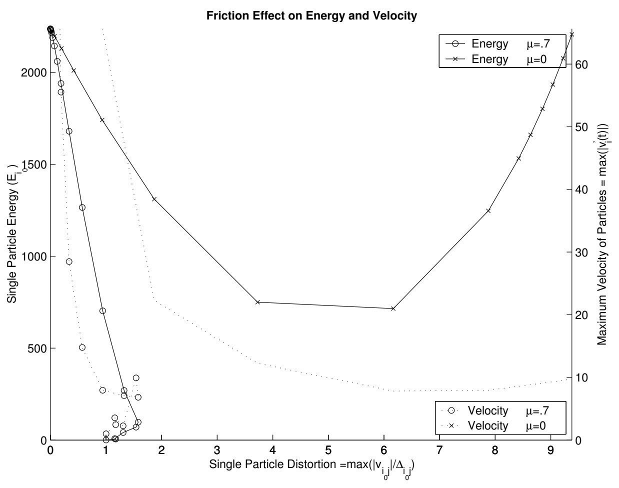


Fig. 2: Friction Effect on Energy and Velocity

C. *Friction* vs. *Kinetic Energy*

As a particle accelerates under the affect of the force field it is also attenuated by simulated friction force. The friction force slows the particles down, so they can be drawn into wells of the potential energy. This effect is illustrated by Fig. 2, comparing two runs with an identical simulated metric input (see paragraph II-F) consisting of 20 nodes.

The figure shows the movement of one of the particles during the first calculation phase of each run. The horizontal axis depicts the maximum ratio between Euclidean distance and original network distance for all node pairs attached to that one particle i_0. As our particle converges to its perfect embedding its horizontal position approaches the distance ratio 1. The solid lines depicts the potential energy of the particle i_0, given by

[3]For notational convenience, we'll omit the vector notation from position x.

[4]Our particle positions are denoted by v, to emphasize they are images of vertices, although this letter normally denotes mechanics velocity.

$E_{i_0} = \sum_{j=1,\, j\neq i_0}^{n} E_{i_0 j}(v_{i_0}, v_j)$. The dashed line depicts the maximum velocity of all particles, $max_{i=1}^{n}(\|\dot{v}_i\|)$, which is closely related to the total kinetic energy of the system. The lines marked with circles (o) depict the case where the dynamic friction coefficient is $\mu_k = 0.7$, whereas the lines marked with cross $(\times)$ depict no friction, that is $\mu_k = 0$. Each marked point represents 10 mili–seconds of CPU time. The total CPU time per phase 1 is thus $200ms$ with friction and $160ms$ without friction. When comparing the solid and dashed lines we should recall that initially our particles were placed at the origin, where the distance ratio equals 0. The effect of friction is to attenuate the particles that are moving away from each other, so that immediately as $\|v_{i_0 j}\| / \Delta_{i_0 j} > 1$, the field force attracts the particles back to each other and they are stopped at equilibrium of $\|v_{i_0 j}\| / \Delta_{i_0 j} = 1$. Without friction however, the particles are moving away too fast and the attracting field force just cause them to oscillate forever with constant velocity.

The friction force attenuating each particle depends on the normal force which is particle dependent, and on the constant friction coefficients of the system. Different friction coefficients, denoted μ_s, μ_k, are accounted for static and moving particles, respectively. More specifically,

$$\vec{Fr}_i = \begin{cases} \mu_s N_i \hat{\vec{F}_i} & \text{if } \dot{v}_i = 0 \\ \mu_k N_i \hat{\dot{v}_i} & \text{otherwise ;} \end{cases} \tag{12}$$

where $\dot{x}$ denote the particle velocity. The system static and dynamic friction coefficients are constant for all particles. When the particle velocity approaches zero the friction force used in movement equation is only the dynamic one. In order to prevent particle to switch its direction during an iteration due to friction, the friction is affecting only at the approximately part of iteration time-interval where the velocity is opposite to it.

The normal force size N_i, represents the relative weight of the particle. Particles with larger weight are less affected by small changes in the positions of the rest of the particles. In the first phase it is given by

$$N_i = \frac{n}{adjustFactor} \underset{\substack{j=1 \\ j\neq i}}{\overset{n}{\boldsymbol{avg}}} \Delta_{ij} \ , \tag{13}$$

whereas at phases $p = 2, 3, \ldots$ all particles have equal relative weight

$$N_i = \frac{n}{normalizedAdjustFactor} \ \forall i = 1, 2, \ldots, n \tag{14}$$

since the pair error functions are normalized by the pair network distance (25) in those phases. The adjust factors are empirical constants set to

$$\begin{cases} adjustFactor = & 7500 \\ normalizedAdjustFactor = & 100 \ . \end{cases} \tag{15}$$

Fig. 2 might suggest that BBS is sensitive to the system friction coefficients. Thus, we checked the sensitivity to system friction coefficients of the CPU time, indicating calculation iterations count, and the maximum and average of symmetric pair distortion d_{ij}. Due to space limitations, we present sensitivity of results for a fixed graph size, that is $n = 50$. Note that in the case that the graph size is fixed, a calculation iteration has constant complexity, so that the CPU time is proportional to total number of iterations. Fig. 3 illustrates results for

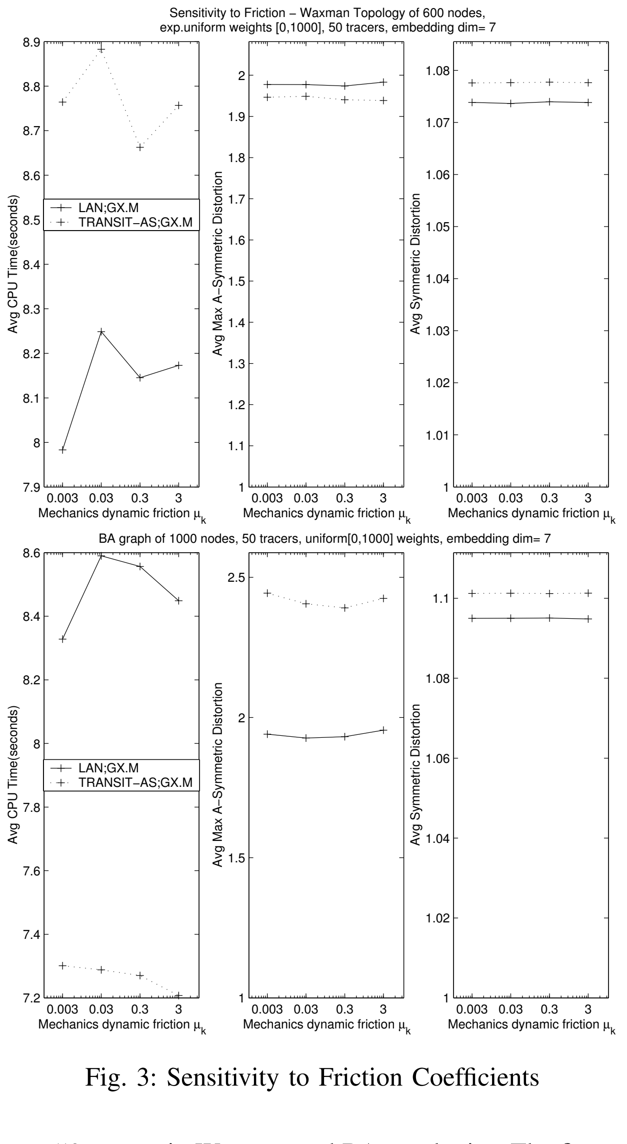


Fig. 3: Sensitivity to Friction Coefficients

$n = 50$ tracers in Waxman and BA topologies. The figure indicates that all three values are almost indifferent to friction in the wide range $0.003 \leq \mu_k \leq 3$. As for larger and smaller graph sizes, in limited tests we've found similar indifference to friction coefficients.

D. Potential Field Force

We shall now calculate the field force $\vec{F}_{i_0}$, derived from the potential energy (1), which affects one particle i_0. This force is given by the opposite of the potential energy gradient with respect to the position of its particle v_{i_0}

$$\vec{F}_{i_0} = -\nabla_{v_{i_0}} E_T(v_1, \dots) = -\sum_{i,j=1, i>j}^{n} \nabla_{v_{i_0}} E_{ij} \ . \quad (16)$$

Here $\nabla_{\mathbf{x}} f(\cdot)$ denotes the gradient with respect to vector $x = (x_1, x_2, \dots, x_d)$ defined as

$$\nabla_x f \equiv \left(\frac{\partial}{\partial x_1} f \quad \frac{\partial}{\partial x_2} f \quad \dots \quad \frac{\partial}{\partial x_d} f \right) \ .$$

In (2) we've assumed that the pair embedding error E_{ij} depends only on pair Euclidean distance, so using the chain derivation rule we get

$$\nabla_{v_{i_0}} E_{ij} = \left. \frac{d}{dx} \mathcal{F}(x, \Delta_{ij}) \right|_{x=\|v_{ij}\|} \cdot \nabla_{v_{i_0}} \|v_{ij}\| \ , \quad (17)$$

where $v_{ij} \equiv v_i - v_j$ is the vector connecting between particle positions v_j and v_i. Taking the derivative of its Euclidean distance we find

$$\nabla_{v_{i_0}} \|v_{ij}\| = \begin{cases} \frac{v_{ij}}{\|v_{ij}\|} = -\hat{v_{ji}}, & \text{if } i = i_0 \text{ and } j \neq i_0 \\ 0, & \text{otherwise.} \end{cases} \quad (18)$$

So for $i \neq i_0$ all derivatives are zero, and the other derivatives are the opposite of the unit vectors along the course of the vector v_{ji_0} connecting between particle positions v_{i_0} and v_j. Let F_{ij} denote the magnitude of the field force element induced on particle i_0 by particle j, along the course of vector v_{ji_0}. Then the force affecting particle i_0 equals the sum of induced forces by the other particles

$$\vec{F}_{i_0} = \sum_{\substack{j=1 \\ j \neq i_0}}^{n} F_{i_0 j} \hat{v_{ji_0}} \ . \quad (19)$$

The sign of F_{ij} determines whether the induced force pulls or repulse the two particles i, j. If $F_{ij} > 0$, the induced force pulls the two particles together, whereas if $F_{ij} < 0$ the induced force repulse them apart. Substituting (17), (18) in (16) and comparing with the force expression (19), we thus have

$$F_{i_0 j} = \left. \frac{d}{dx} \mathcal{F}(x, \Delta_{i_0 j}) \right|_{x=\|v_{ji_0}\|} \ . \quad (20)$$

Namely the field force induced on a particle by another particle is given by the derivative of the pair embedding error with respect to the Euclidean distance between the particles. One can easily check that signs are correct in the above relation. If $F_{ij} > 0$ then the derivative of pair embedding error is positive, so in order to decrease it the Euclidean distance should be decreased. The pair Euclidean distance is decreased by the induced field force because in case $F_{ii} > 0$, the two particles are pulled by the field force. The case of $F_{ij} < 0$ follows immediately from symmetry.

E. Error Functions of Phases

At the first phase the pair embedding error, $E_{ij}^{\langle 1 \rangle}$ is given by

$$\begin{aligned} E_{ij}^{\langle 1 \rangle}(v_i, v_j) &= \mathcal{F}(\|v_i - v_j\|, \Delta_{ij}) \\ &= (\|v_i - v_j\| - \Delta_{ij})^2 \ , \end{aligned} \quad (21)$$

that is the squared pair distance error. The field force induced between two particles is given by (20), that is

$$\begin{aligned} F_{ij}^{\langle 1 \rangle} &= \mathcal{F}_x(\|v_i - v_j\|, \Delta_{ij}) \\ &= 2(\|v_i - v_j\| - \Delta_{ij}) \ . \end{aligned} \quad (22)$$

Namely in the first phase the force induced between each particle pair is equal to 2 times the distance error of the two particles. Substituting in expression (11) for the field force at $t = 0$ we find

$$\vec{F}_i(t)|_{t=0} = 2 \sum_{\substack{j=1 \\ j \neq i_0}}^{n} (\|u(i,j)\| - \Delta_{ij}) \hat{u}(i,j) \ . \quad (23)$$

The error functions we chose below for the rest calculation phases are increasingly sensitive to directional relative error Eq. (33). An equally good output should result from choosing any similar series of increasingly sensitive error functions.

The symmetric pair distortion d_{ij} is defined as the maximum between the expansion and the contraction ratios

$$d_{ij} = \max\left(\frac{\|v_i - v_j\|}{\Delta_{ij}}, \frac{\Delta_{ij}}{\|v_i - v_j\|} \right) \ . \quad (24)$$

The pair embedding error function of the second phase is selected as follows

$$E_{ij}^{\langle 2 \rangle} = (d_{ij} - 1)^2 \quad (25)$$

Substituting Eq. (21) in the above we find

$$E_{ij}^{\langle 2 \rangle} = \frac{E_{ij}^{\langle 1 \rangle}}{\min\left(\|v_i - v_j\|, \Delta_{ij}\right)^2} \ ; \quad (26)$$

So the second phase function is the square of pair distance error, divided by the square of minimum between embedding and network pair distance.

For last two phases we selected the following 2 functions,

$$E_{ij}^{\langle 3\rangle} = \exp^{(E_{ij}^{\langle 2\rangle})^{\frac{3}{4}}} - 1 = \exp^{(d_{ij}-1)^{\frac{3}{2}}} - 1 \,, \quad (27)$$

$$E_{ij}^{\langle 4\rangle} = \exp^{E_{ij}^{\langle 2\rangle}} - 1 = \exp^{(d_{ij}-1)^{2}} - 1 \,. \quad (28)$$

The derivation of the induced forces in the other phases following the first phase is given in the Appendix.

F. Example of Perfect Embedding

We take as a simple example the metric representing the distances among fixed points in an Euclidean space. Note that the Euclidean distances between any set of fixed points is a metric because they satisfy the triangle inequality. Moreover, this metric can be embedded perfectly, i.e., with distortion 1 for all pairs, by choosing the coordinates of those fixed points as geometric images for its vertices. For ease of presentation we stick to the simplest case of 2 dimensional space, that is the XY plane. Next we calculate the input metric, that is the $\frac{n}{2}(n-1)$ Euclidean distances among these n points, and run our BBS procedure with $d = 2$ on this metric.

If the embedding is accurate we would expect that Euclidean distances would match exactly, up to the rounding error, to the simulated metric distances. We experimented with BBS embedding with and without friction. The results are illustrated in Fig. 4. We scattered $n = 20$ random points uniformly in a 16×16 rectangle, $(x_i, y_i) \in (-8, +8) \times (-8, +8)$. For presentation clarity we have drawn only the trajectories of the first 6 particles out of $n = 20$. The trajectories on the right side are from a run with zero friction, $\mu_k = \mu_s = 0$, whereas on the left side they are with friction coefficient of $\mu_k = .7$ and $\mu_s = .9$. Trajectories were matched to the input points using only the shift, rotation, and reflection transformations. We placed an enlarged marker at each of the input points, and marked the trajectory of the corresponding particle with a smaller marker of the same type. Indeed all trajectories on the right, with friction, converged to their corresponding input points. Especially interesting is one of the right trajectories, marked with triangle pointing up, which left the origin and then returned back home to its point at the origin. To the contrary, none of the trajectories with zero friction, except for the trajectory matched by the transformation, have even come close to its corresponding input points.

III. Embedding Method Comparison

In this section we compare our BBS method to three other Embedding methods, and also to a topology aggregation method [3] and force directed graph drawing method [12].

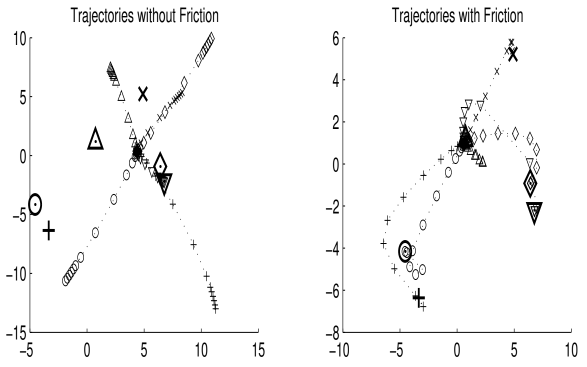


Fig. 4: Friction Effect on simulated metric trajectories

A. Other Embedding Methods

We begin by briefly describing the four methods.

1) Semi–Definite Programming: (——)[5] SDP is a well known optimization technique that has drawn attention from diverse areas. The use of SDP for graph embedding was introduced in the seminal work of Linial, London, and Rabinovich [16]. The SDP problem in primal standard form is

$$\max \; tr(CX) \qquad \text{s.t.}$$
$$A(X) = \begin{bmatrix} tr(A_1X) \\ tr(A_2X) \\ \cdots\cdots\cdots \\ tr(A_kX) \end{bmatrix} = a \,, \qquad (29)$$

where all the matrices A_i, X, and C are symmetric, and a is a vector. The matrices A_i, C and the vector a depend on the metric Δ, and X is the variable. A Cholesky decomposition of the solution to the SDP problem, $X = MM^t$, yields the embedding coordinates matrix $M_{n\times n}$. The embedding dimension, that is the number of columns of matrix M, is reduced by Johonson and Lindenstrauss [23] random projection from $\mathcal{R}^n$ into $\mathcal{R}^d$. The Matlab code we used is due to [24]. It executes the Brian Borchers SDP solver [25].

2) Multi–Dimensional–Scaling ($-\!\ast\!-$)*:* The classical metric MDS [8] has a simple and low complexity implementation. The Matlab code we used is described in [9, Sec. 4].

3) Down-Hill Simplex (DHS) ($-\!\boxminus\!-$)*:* This minimization method is known for a very long time [26], and was recently used as an embedding method in [6]. We elaborate on our implementation, which follows the description in

[5]The graphs in this section use a unique marker for each method which is given in the heading of the paragraph describing each method. Our embedding method, Big-Bang Simulation (BBS), is marked with '$\times$'.

[6], while trying to achieve the best embedding in this way.

The method searches the minimum of an arbitrary function h of m variables. For this end it uses a simplex with $m+1$ vertices in $\mathcal{R}^m$. In each iteration the simplex vertex with the largest h value is either reflected through or contracted towards the opposing simplex face. In case both h values, in reflected and contracted points, are larger than previous h value of this vertex, the simplex is contracted towards the opposing simplex face of the vertex with the lowest h value. For Euclidean embedding, the function h of $m = nd$ variables, is either equal to the total embedding error from phase 1, $E_T^{\langle 1 \rangle}$, Eq. (21); or the normalized pair error used in GNP, [6]

$$E_{ij}^{\langle GNP \rangle} = \left(\frac{\|v_i - v_j\| - \Delta_{ij}}{\Delta_{ij}} \right)^2 . \tag{30}$$

We experimented with both unnormalized and normalized minimization target functions and both yielded similar results. The results for DHS throughout this paper were calculated with the normalized target function Eq. (30). The initial condition of this method are the positions of the $m+1$ simplex vertices in $\mathcal{R}^m$. Following [27], we take m simplex vertices with equal length along the m unit vectors, and the origin as $m+1$'th vertex

$$\begin{aligned} \vec{s}_{m+1} &= 0 \\ \vec{s}_k &= (L\delta_{ik})_{i=1}^m \;; k = 1, \ldots m \,; \end{aligned} \tag{31}$$

where $\delta_{ik} = 1$ if $i = k$, or 0 otherwise. The length L depends on the average and standard deviation of the embedded distances $(\Delta_{ij})_{i,j=1}^n$

$$L = 2 * \overline{\Delta} + 6 * \sigma_\Delta \tag{32}$$

The **C**-code for our comparison is taken from [27, ch. 10 sect. 4] which also describes the DHS method in more details.

4) MSTxRST (—●—): The MSTxRST topology aggregation procedure, discussed in [22], represent a network by an aggregated graph of at most $(x+1)n$ edges. The edges in the aggregated graph are the union of a minimum spanning tree (MST) of a clique that is built from the shortest paths of the original graph, and x or more Random spanning trees (RST) of this clique. For instance in MST2RST the aggregated graph has $3n$ edges. This method had the lowest distance distortion among the aggregation methods considered in [22], and our comparison was done using the **C**-code used there.

5) KK89 Force Directed Graph Drawing (—○—): This algorithm was used for a different purpose, that is to create a straight-line drawing of a graph with as much symmetry as possible. The mechanical model of this algorithm, called *spring embedder*, resembles our model. The drawing process simulate a mechanical systems, where vertices are replaced by rings, and edges are replaced by springs. Many attempts were made to draw graphs utilizing the spring model. However, in most of these attempts spring connect only between neighbor vertices, and the spring original length is set to the corresponding edge length. We did limited testing of BBS in which forces were considered only between neighboring nodes, and verified it *did not converge well* for simulated metric graph of medium sizes, that is 50-100 nodes. Thus, all the embedding methods compared here take as their input the clique built from the shortest paths of the input graph.

The first attempt to model such clique with springs, is due to Kamada and Kawai [12]. We decided to compare Kamada and Kawai, because their KK89 algorithm is simple, and its embedding error function is same as ours in phase 2, except that Δ_{ij}^2 replaces the denominator in Eq. (26). We set the algorithm constants as follows:

$$\begin{aligned} L_0 &= 10 \,, \\ K &= 1 \,, \\ \epsilon &= 0.00001 \,. \end{aligned}$$

The 2nd order Newton-Raphson (NR) method of KK89 is very sensitive even with a simulated metric graph of small size, that is $15-20$ nodes. Thus the following changes were made to the original algorithm

outer The outer loop of the KK89 algorithm selects a particle with maximal gradient size $\|\nabla_{v_m} E_T\|$, and executes the inner loop with that particle. Originally this loop continues as long as the maximal gradient size is larger than ϵ. However, we stop after 1000 times, leaving particles at their last position.

inner The inner loop of the KK89 algorithm finds a local minimum of $E_T(\ldots, v_m, \ldots)$ as a function of the position v_m, using 2nd order NR. Originally this loop continues until $\|\nabla_{v_m} E_T\| \le \epsilon$. However we stop after 500 iterations and check if the gradient size $\|\nabla_{v_m} E_T\|$ is smaller than its size at the beginning of the loop. If yes the outer loop continues and selects the next particle to be moved, but otherwise we abort the outer loop.

region We check after each iteration of 2nd order NR, that its particle remains inside the ball with radius L_0, that is twice the size of the drawing. In case the result point is outside the ball, we place the particle at random point in a $.001-$neighborhood of the origin.

B. Environment Details

The network graphs in our comparison were created according to both Waxman [20] and Barabási-Albert (BA) [21] methods. Only a subset of the graph nodes was selected to be embedded according to the characteristics of the two applications. For IDMaps [5] the subset was selected using two Tracer placement methods, TRANSIT or LAN (stub), which select the Tracers among the highest or lowest degree nodes, respectively. For topology aggregation [22], the subset includes all the border nodes which are located on the edges of the Waxman topology rectangle area.

To increase the confidence each experiment was conducted on 10 networks using 5 sets of random weights per network. Namely each point in the comparison graph results from 50 embedding experiments[6]. For Waxman networks, the edge weights were taken as $\lfloor 10^{exp} + .5 \rfloor$, where exp are $i.i.d.$ uniformly distributed in the interval $(0, 3]$. For BA networks the edge weights are $i.i.d.$ uniformly distributed in the interval $[1, 1000]$.

C. Performance Metrics

The *symmetric pair distortion*, d_{ij}, is defined in Eq. (24). The *directional relative error*, E_{rel}, was defined by [6, Eq. 4] as

$$\frac{predicted\, distance \;-\; measured\, distance}{\min\left(measured\, distance;\; predicted\, distance\right)}$$

and for Euclidean embedding is given by

$$E_{rel} \;\equiv\; \frac{\left\|v_i - v_j\right\| - \Delta_{ij}}{\min\left(\left\|v_i - v_j\right\|,\, \Delta_{ij}\right)^2} \tag{33}$$

Comparing with Eq. (26) we find that $|E_{rel}| = \sqrt{E_{ij}^{\langle 2 \rangle}}$, and substituting Eq. (25) we thus have $d_{ij} = 1 + |E_{rel}|$. The <u>*average symmetric pair distortion*</u> is given by $\overline{d}_{ij} = 1 + \overline{|E_{rel}|}$.

As a measure of the worst-case distortion we use the *two-sided embedding distortion* defined as

$$\min\left(c_1 c_2 \;:\; c_1 \Delta_{ij} \geq \|v_i - v_j\| \geq \frac{1}{c_2}\Delta_{ij}\right), \tag{34}$$

over all node pairs $i, j = 1, \ldots, n;\; i \neq j$.

An alternative measure, defined by Linial et al. [16], is the *one-sided distortion*

$$\min\left(c \;:\; \Delta_{ij} \geq \|v_i - v_j\| \geq \frac{1}{c}\Delta_{ij}\right), \tag{35}$$

over all node pairs $i, j = 1, \ldots, n;\; i \neq j$. The above measures are comparable since from linear contraction of our coordinates we get $c = c_1 c_2$. However this contraction increases the other metric $\overline{d}_{ij}$.

[6]However, for $n = 450$ tracers, we performed only 5×1 embeddings, due to their large space and time requirements

D. Comparison Results

A different marker depicts each of the embedding methods compared in following figures. Lines with no marker depict SDP method, lines marked with '$*$' depict MDS method, '$+$' depict DHS method, '$\bullet$' depict MSTxRST method, and '$\circ$' depict KK89 method. Some figures uses different line styles to distinguish between the several tracer placement methods shown on the same picture.

a) Symmetric Pair Distortion: We compare the accuracy of the five methods using the complementary symmetric distortion distribution over all pairs of the 50 embedding experiments. Fig. 5 compares all methods, except from topology aggregation(MSTxRST), for the LAN placement methods of $n = 15$ Tracers in the BA topology which is typical to the IDMaps application discussed here-on. BBS is more accurate than DHS, having smaller complementary distribution along the entire range of the distortion. For example the probability $P(d_{ij} > 1.1)$, is .11 for BBS vs. .15 for DHS, i.e., a 35% increase. SDP is much worse than both. The insets in the top right depicts the non-symmetric pair distortion in the lower graphs and the embedded distance in the upper graphs, both vs. original pair distances. The SDP embedding contract nearly all edges whereas DHS and BBS has similar mass of contracted and expanded edges. Fig. 6 compare MDS, KK89 and BBS for the LAN and TRANSIT placement methods of $n = 150$ tracers in the Waxman topology. We rule out SDP and DHS as practical embedding methods for large n due to thier long running time and sensitivity, respectively. The accuracy of BBS is much better than MDS, e.g., $P(d_{ij} > 1.3)$, is .05 for LAN-BBS and TRANSIT-BBS vs. .6 for LAN-MDS and .999995 for TRANSIT-MDS.

b) Embedded Graph Size: Fig. 7 depicts the performance of the different methods as a function of the number of embedded nodes. For $n > 30$ or 70 the long running times of SDP or KK89 and high sensitivity of DHS exclude them from the comparison. For BBS we depicted in dotted lines, a linear fit of the average CPU time graph which indicates that BBS has complexity $Cn^2,\ \forall n \in [10, 450]$. The value of C calculated for Pentium-IV 2.0Ghz processor is 6×10^{-4}. The BBS embedding distortion is the lowest for all graph sizes except for $n \geq 150$ where MST6RST distortion is smaller e.g. for $n = 450$, the distortions are 4 vs. 15.5 respectively. BBS has the lowest average symmetric pair distortion in all graph sizes. The symmetric pair distortion

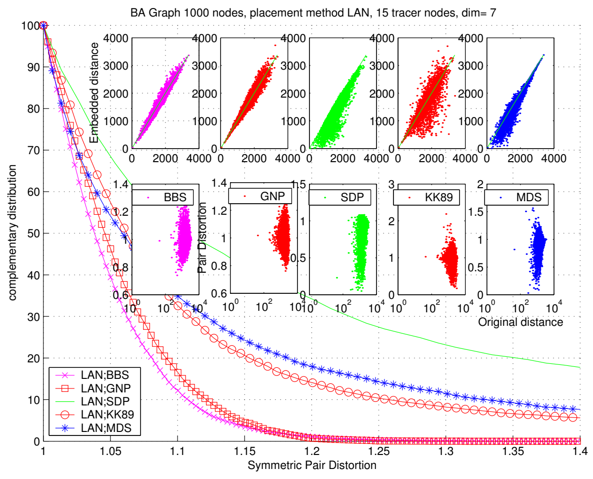


Fig. 5: BA Symmetric Pair Distortion

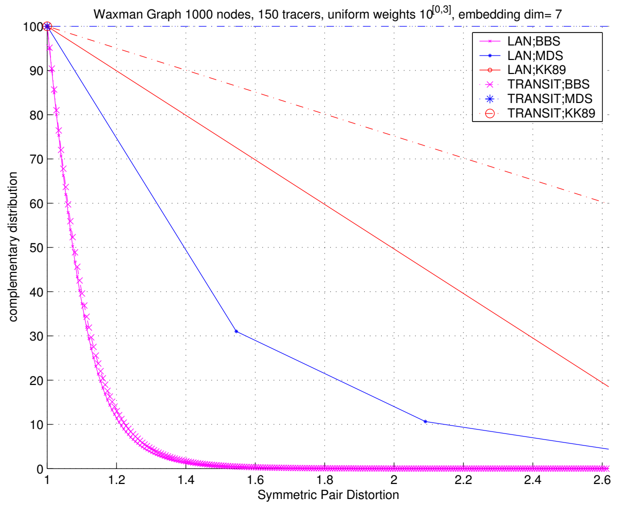


Fig. 6: Waxman Symmetric Pair Distortion

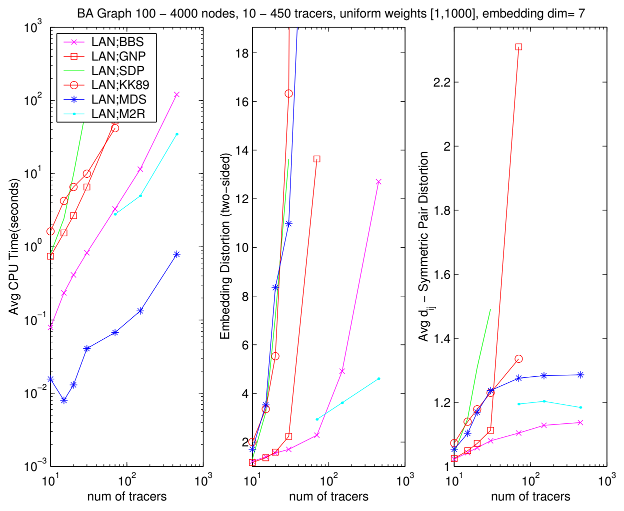


Fig. 7: Performance Metrics vs B-A Graph Size

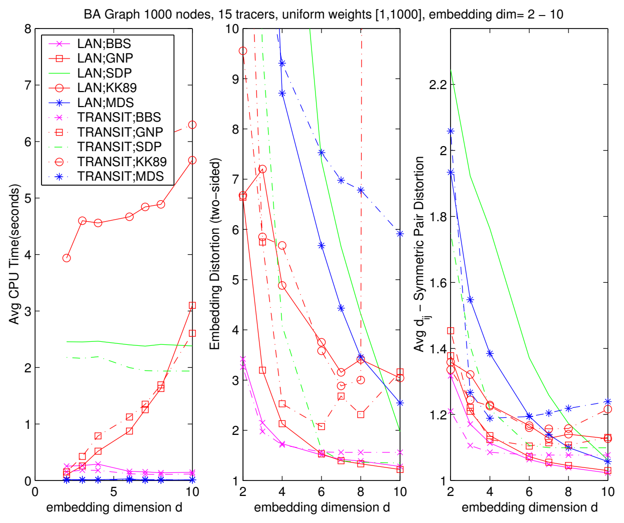


Fig. 8: Performance Metrics vs B-A Embedding Dimension

of MST6RST and SDP are not comparable with the rest of the methods, since all MST6RST edges are expanded and most SDP edges are contracted.

c) Embedding Dimension: Fig. 8 illustrates the effect of the embedding dimension on the BA topology with $n = 15$ Tracers. Naturally, the performance of all methods improves when the embedding dimension increases. For the BA topology the knee point, where the improvement diminishes, is at $d = 7$ as was found by Ng and Zhang [6].

For $d < 7$, the performance gap between all other methods and ours is significant, as can be seen by the two right graphs of Fig. 8. The difference is larger for the two-sided embedding distortion. The larger performance gap in embedding distortion is explained by the improvement of large distortion pairs in the last two phases of our calculation. Although the main objective of SDP is to reduce the embedding distortion, with LAN placement it has a larger embedding distortion compared to both DHS and BBS for all dimensions. However, with TRANSIT placement for $d \geq 7$, the SDP distortion is the smallest. The results for the Waxman topology with $n = 70$ and 150 Tracers are similar, and are thus omitted. However, for $n > 30$ the sensitivity of DHS rules it out as a viable method.

d) Input Graph Sensitivity: We tested embedding of the BA and Waxman topologies with 15 TRANSIT and 150 BORDER nodes respectively. We either increased or decreased the weights of 15% of the input edges by 10%, and disconnect or reconnect an additional 5% of the edges. The input graph is embedded first with regular initial conditions, and the modified graph is embedded with initial positions of the unmodified graph embedding. The test is repeated 3 times where the initial positions in each embedding are the output positions of previous embedding. We compare the accuracy and complexity of the 3 embedding by embedding the 3 modified graphs again with regular initial conditions. With unnormalized square embedding error, Eq. (21), the sensitivity of BBS is **low**. The calculations from previous graph position achieves comparable performance metrics with 13 iterations and $.01 - .15$ CPU seconds for BA or Waxman topology respectively, compared to $200 - 1000$ iterations and $.15 - 10$ CPU seconds, with regular initial conditions.

IV. Applications

A. *Topology Aggregation*

Topology aggregation is used in hierarchical networks to compactly represent the cost of traversing a network between every possible entrance and exit points. If the aggregation decreases the cost of an edge it means that a routing application that is using the aggregated view to find a route in the network may select a path with a higher cost than it expects. It is generally considered undesirable to receive such bad news, since, e.g., if the traversal cost means delay we may choose a route which violates the application bound. Thus, for topology aggregation we seek an embedding that favors length increase over length decrease.

The MSTxRST aggregation procedure (Sec. III-A.4) insures that an edge length in the aggregation is never smaller than the original. However, in the embedding methods we discussed so far some distances are contracted while other are expanded. A transformed embedding in which no edges are contracted (or expanded) was discussed in Sec. III-C. The problem with the transformed embedding is that it increases the average pair distortion which is, of course, important for aggregation performance.

An alternative way to favor expansion in our embedding is an introduction of a **price factor** denoted $\boldsymbol{P}$, $\boldsymbol{P} \gg 1$, in the definition of pair embedding error functions

$$\hat{E}_{ij}^{\langle p\rangle}(v_i, v_j) = \begin{cases} E_{ij}^{\langle p\rangle}(v_i, v_j) & \text{if } \frac{\|v_i - v_j\|}{\Delta_{ij}} \geq 1 \\ \boldsymbol{P} E_{ij}^{\langle p\rangle}(v_i, v_j) & \text{otherwise.} \end{cases} \tag{36}$$

Thus, the weight of a contracting pair in the total embedding error will be larger than the weight of an expanding pair. Such a price factor can be directly incorporated into the DHS method, but should pose inherent difficulty for MDS since it is not linear. We introduce the price factor into our calculation at the end of the first calculation phase. Particles are placed at the best position of the first phase, and then moved by a modified field force incorporating the price factor $\boldsymbol{P}_1$; $\boldsymbol{P} \gg \boldsymbol{P}_1 > 1$. As particles reach near an equilibrium of the modified field force the price factor is increased again, and particles continue from the previous equilibrium point to the next equilibrium. This procedure is repeated until the price factor is increased to the final value $\boldsymbol{P}$. In the rest of the phases, the calculation continues with the field force which directly incorporates $\boldsymbol{P}$.

Fig. 9 illustrates the effect of the price factor $\boldsymbol{P} = 512$ on our embedding method compared to embedding without it, i.e., $\boldsymbol{P} = 1$. The middle graph shows that the price factor decreases our two sided embedding distortion by approximately half at $d = 2$ and for $d > 2$ its effect is only modest. However, with the price factor, the average symmetric pair distortion increases approximately by 2%, which translates to 10% increase of the absolute relative error. Here the knee point where the improvement diminishes is at $d = 6$. Although at $d = 2$ (or $x = 1$) our performance is worth than MSTxRST, at $d = x + 1 \geq 3$ it supersedes MSTxRST.

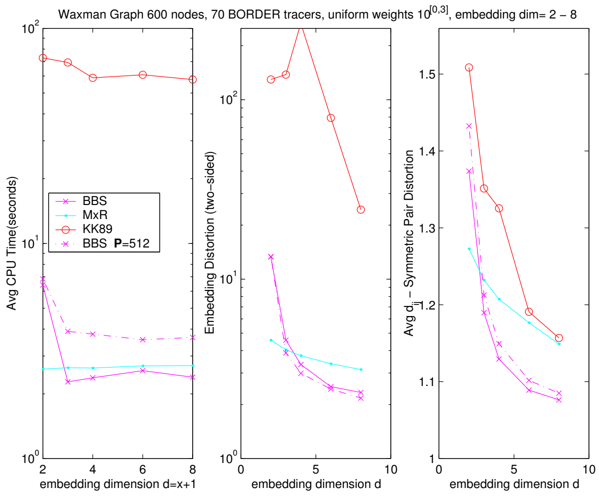


Fig. 9: Waxman Symmetric vs. A-Symmetric Aggregation

Fig. 10 compares the symmetric pair distortion histograms of MST3RST and BBS with $d = 4$ and $P = 512$. The insets in the top right illustrates nicely the effect of the price factor which is to force all pairs to expand rather then contract. Almost all the pairs of BBS are above the $y = x$ line in the upper inset and the $y = 1$ line in the lower inset.

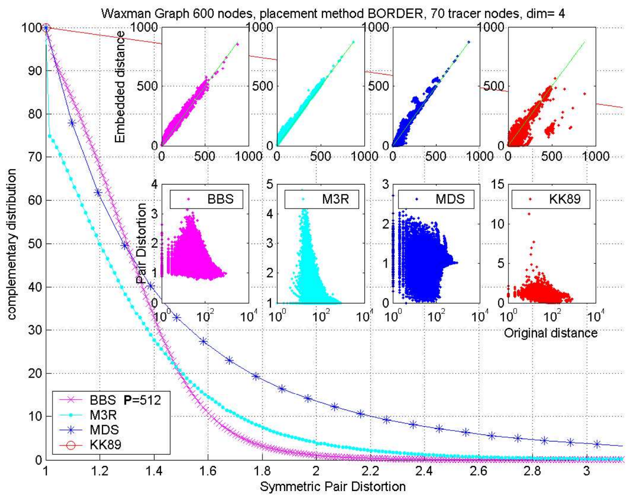


Fig. 10: Aggregation Distortion Histograms

B. *Internet Distance Estimation*

IDMaps [5] is a project that aims to build a global architecture for Internet host distance estimation and distribution. The architecture is based on Tracers, which are instrumentation boxes, that are placed in the Internet. Each Tracer measures distances to other Tracers and to address prefixes (AP) that are close to it. These measurements are multicast to topology servers which combine them to an estimated map of the Internet.

Euclidean embedding yields smaller relative distance errors compared to IDMaps, especially when both nodes are sharing a single nearest Tracer (see Fig. 1A). The main problem with Euclidean distance estimation is underestimation of large measured distances. That is an inherent problem when embedding the BA topology, as all shortest path between distant nodes must go through a small number of core nodes [17]. Indeed, due to the triangle inequality, the Euclidean distance between distant nodes is bound to be smaller then their shortest path which goes through the core nodes. On the other hand, IDMaps sum of segments accurately estimate the longer paths going through the core nodes, as in Fig. 1B.

Throughout this section we've estimated distances in BA graphs, using 3 sets of random weights per each of 5 simulated BA graphs. Fig. 11 compares the additive estimation of IDMaps with Euclidean embedding by GNP and BBS methods using the *directional relative error*, Eq. (33). We only experimented with LAN and TRANSIT Tracer placement methods, illustrated on the top and bottom pictures respectively, because LAN placement is the most easy to deploy, and TRANSIT placement yields the best mirror selection performance [5]. We used 15 Tracers and measured distances from each host to all 15 Tracers, that matches the conditions of the similar figures in [6]. The groups of vertical lines 75ms apart depicts the distribution of relative errors for measured distances belonging to the 75ms interval. The lines marked with square, upright triangle, and circle markers depicts GNP, IDMaps, and BBS, respectively. The method marker is placed at the average relative error point, and the star marker depicts the median. Each line has whiskers at the 5, 25, 75, and 95 percentiles.

For each placement and calculation method we randomly picked 150 nodes out of the 1000 graph nodes in each of the 15 simulated BA graphs. We estimated the distance from all other graph nodes to each picked node, that is a total of $2,247,750$ distance pairs per method.. The thick lines depict the overall count of measured pair distances per interval.

With both placement methods, IDMaps has longer percentile lines compared to GNP and BBS, for $\Delta < 400$. The cause for such large errors is explained in Fig. 1 where the distance Δ is much smaller than IDMaps estimation. The median and average of IDMaps TRANSIT placement are much lower than its LAN placement, and at interval $225 < \Delta < 3000ms$ its median is 0 and average is below 0.5 and decreases rapidly to 0 as distance increases. BBS's median and average approaches 0 for both LAN and TRANSIT placement at $\Delta > 225ms$. For $\Delta > 700ms$ in TRANSIT or $1500ms$ in LAN however our median and average becomes **negative** down to -0.3 at $\Delta \geq 2500$. IDMaps additive estimation has larger positive relative errors than Euclidean estimation for short distances, but is more accurate for longer distances. Therefore best distance estimate is by selecting Euclidean estimation for short distance, and IDMaps additive estimation for longer distances, based on a threshold of the Euclidean distance. Unfortunately, the optimum threshold point changes for different placement methods, graph topologies, and ranges of random edge weights.

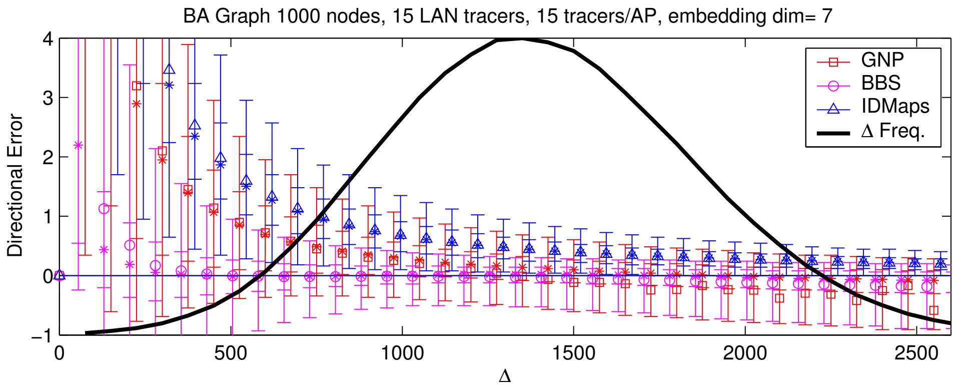
An alternative for selecting between Euclidean and IDMaps additive estimations is using the ratio $\boldsymbol{R}$ between the two, given by

$$\boldsymbol{R} = \frac{Euclidean\ distance}{IDMaps\ additive}\ , \tag{37}$$

and the estimated distance is selected as follows

$$\begin{cases} Euclidean\ distance & \text{if } \boldsymbol{R} < \boldsymbol{R}_{Th} \\ IDMaps\ additive & \text{otherwise.} \end{cases} \tag{38}$$

Fig. 12 illustrates the improvements in accuracy of our estimation compared to IDMaps additive, with threshold $\boldsymbol{R}_{Th} = 0.45$. As in the previous figure we used 15 Tracers and measured distances from each host to all 15 Tracers.



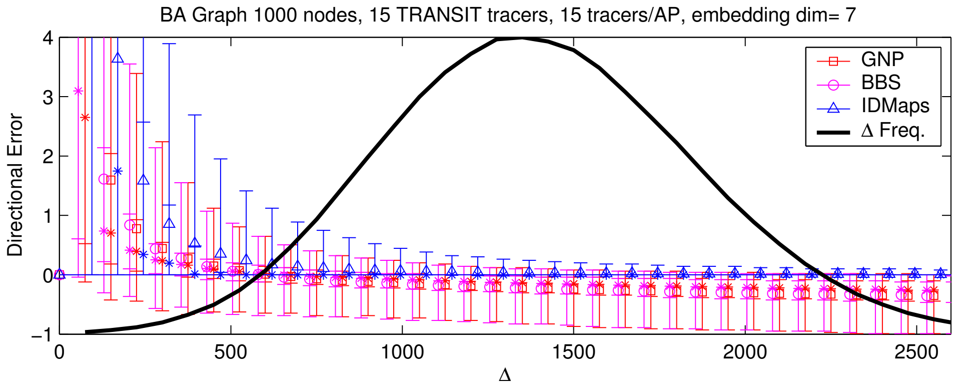


Fig. 11: Directional Relative Error (stand-alone)

One could have selected the closet mirror as the one with smallest distance estimated by (38). Such naive approach however doesn't maintain the ordering among estimated distances, as some were estimated by additive IDMaps and some as Euclidean distances. We calculate the ratio

$$R' = \frac{\min_k \{Euclidean\ distance\}}{\min_k \{IDMaps\ additive\}}\ , \tag{39}$$

where the minimum among mirrors for either method is achieved by the closet mirror denoted by $k_{min}^{\langle\cdot\rangle}$. The closet mirror is then selected as

$$\begin{cases} k_{min}^{\langle Euclidean\rangle} & \text{if } \boldsymbol{R' < R'_{Th}} \\ k_{min}^{\langle IDMaps\ additive\rangle} & \text{otherwise.} \end{cases} \tag{40}$$

We compared the mirror selection accuracy of IDMaps additive with BBS using the selection criterion of (40) with $R'_{Th} = .45$, i.e., the same threshold value used for distance estimation. Following [5] we randomly selected 10 mirror servers and estimated the closet mirror to each of the rest of the graph nodes acting as clients. The client decision is considered correct if it selects the mirror whose client-mirror distance is at most twice the optimal distance. For each mirror group rank accuracy is defined as the percentage of correct client decisions. Fig. 13 illustrates the average cumulative distribution function (CDF) of rank accuracy. Each mark is the average of the CDFs from the 15 simulated graphs, where each CDF consists of 300 mirror group experiments performed on a single graph. The number of Tracer distance measurements per AP, is specified in the legend after the $'\times'$ mark, and is depicted with increasing marker sizes.

The accuracy of additive IDMaps improves with 3 Tracer measurements compared to 1. Using TRANSIT placement, IDMaps doesn't improve with more than 3 Tracer measurements, and using LAN Tracer placement it becomes even less accurate with the additional measurements. The accuracy of our threshold selection however, improves with each additional Tracer measurement. With 3 measurements it is more accurate then additive IDMaps for both LAN and TRANSIT placement. With 15 measurements for LAN placement it is nearly as accurate as IDMaps for TRANSIT placement, pointing clients to the closest mirror server with confidence 0.95 in 94% of the cases, compared to 88% of the cases of additive IDMaps for LAN placement.

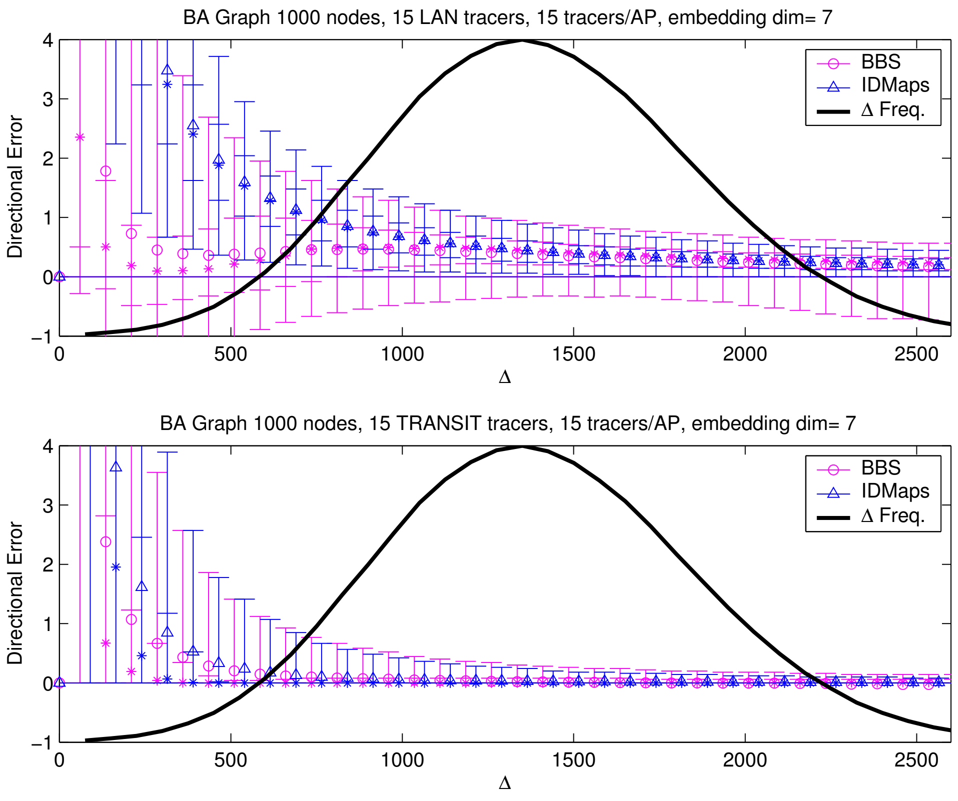


Fig. 12: Directional Relative Error (Thld. selection)

V. Concluding Remarks

We presented a novel scheme for embedding a graph metric in a d-dimensional Euclidean space, and showed that with one exception (SDP for very small networks with LAN placement and high dimension) BBS was always the most accurate embedding scheme. In addition, BBS execution time is second only to MDS, but MDS has a stability problem in large graphs and works well only with high d. In addition, MDS is not applicable to topology aggregation as we stated before. Finally, BBS is insensitive to its only arbitrary parameters, the dynamic and static friction coefficients, as demonstrated in Fig. 3, thus no fine tuning is required.

We demonstrated the efficiency of our scheme for important networking problems: topology aggregation, closest mirror selection, and distance estimation. We believe our method can be applied to other problems as well, such as routing in ad-hoc networks, and efficient building of peer-to-peer networks and application layer multicast.

Acknowledgement

We would like to thank Udi Ashkenazi for implementing the SDP embedding algorithm and for his insight of SDP. We also thank Cheng Jin for providing the IDMaps simulation code, and for helping us integrating new features into it.

Appendix

We derive the expressions of field forces from error function of rest calculation phases. Following (20) we derivate the error function of phase 2 (25), and using (22) we get

$$\begin{aligned} F^{\langle 2\rangle}_{ij} &= \frac{dE^{\langle 2\rangle}_{ij}}{d\|v_{ji}\|} \\ &= F^{\langle 1\rangle}_{ij} \begin{cases} \Delta_{ij}^{-2} & \text{if } \|v_{ji}\| > \Delta_{ij} \\ \|v_{ji}\|^{-2} & \text{if } \|v_{ji}\| < \Delta_{ij} \end{cases} \\ = 2(\|v_{ji}\| - &\Delta_{ij}) \begin{cases} \frac{1}{\Delta_{ij}} & \text{if } \|v_{ji}\| > \Delta_{ij} \\ \frac{\Delta_{ij}}{\|v_{ji}\|^3} & \text{if } \|v_{ji}\| < \Delta_{ij} \end{cases} . \end{aligned} \tag{41}$$

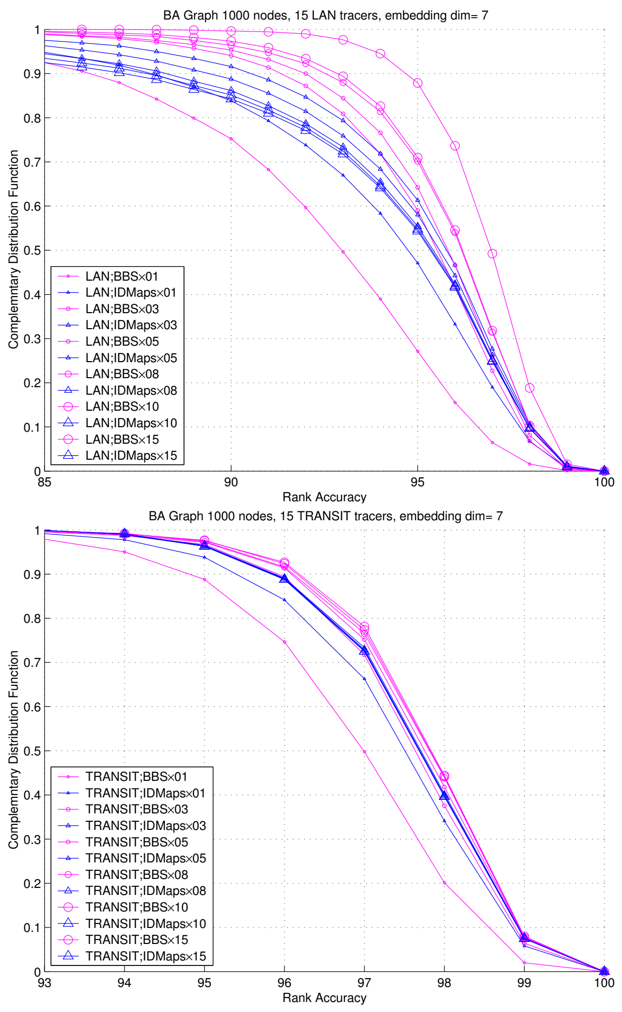


Fig. 13: Mirror Selection Accuracy (Thld. selection)

Similarly for phase 3 and 4 whose error functions are defined in (27-28), and substituting (41) we get

$$\begin{aligned} F_{ij}^{\langle 3\rangle} &= \frac{3}{4}\frac{F_{ij}^{\langle 2\rangle}}{\sqrt{d_{ij}-1}} \qquad E_{ij}^{\langle 3\rangle} && (42) \\ &= \frac{3}{2}\sqrt{d_{ij}-1} \begin{cases} \frac{1}{\Delta_{ij}} & \text{if } \|v_{ji}\| > \Delta_{ij} \\ \frac{\Delta_{ij}}{\|v_ij\|^2} & \text{if } \|v_{ji}\| < \Delta_{ij} \end{cases} \\ F_{ij}^{\langle 4\rangle} &= \qquad F_{ij}^{\langle 2\rangle} \qquad E_{ij}^{\langle 4\rangle}\ . && (43) \end{aligned}$$

References

[1] "Private network – network interface specification version 1.0 (PNNI)," Tech. Recomendation, The ATM Forum technical committee, Mar 1996, AF-PNNI-0055.000.

[2] Whay Chiou Lee, "Topology aggregation for hierarchical routing in ATM networks," *Computer Communication Review*, vol. 25, no. 2, pp. 82 – 92, Apr. 1995.

[3] B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt, "Routing through networks with hierarchical topology aggregation," *Journal of High-Speed Net.*, vol. 7, 1998.

[4] W. Theilmann and K. Rothermel, "Dynamic distance maps of the internet," in *Infocom*, 2000, Tel Aviv, Israel.

[5] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A global internet host distance estimation service," *IEEE/ACM Trans. Networking*, Oct. 2001.

[6] T. Ng and H. Zhang, "Predicting internet network distance with coordinates based approaches," in *Infocom*, 2002.

[7] E. Cronin, S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," *IEEE J. Select. Areas Commun.*, vol. 20, no. 7, pp. 1369–1382, Sept. 2002.

[8] W.S. Togerson, "Multidimensional scaling of similarity.," *Psychometrika*, vol. 30, pp. 379–393, 1965.

[9] G. Zigelman, R. Kimmel, and N. Kiryati, "Texture mapping using surface flattening via multidemnsional scaling," *IEEE Trans. Visual. Comput. Graphics*, 4-6 2002.

[10] G. Yona, N. Linial, and M. Linial, "ProtoMap: automatic classification of protein sequences and hierarchy of protein families," *Nucleic Acids Research*, vol. 28, pp. 49–55, 2000.

[11] P. Eades, "A heuristic for graph drawing," *Congressus Numerantium*, vol. 42, pp. 149–160, 1984.

[12] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7–15, 1989.

[13] Thomas M. J. Fruchterman and Edward M. Reingold, "Graph drawing by force-directed placement," *Software - Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[14] R. Davidson and D. Harel, "Drawing graphs nicely using simulated annealing," *ACM Transactions on Graphics*, vol. 15, no. 4, pp. 301–331, 1996.

[15] I. Herman, G. Melancon, and M.S. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Trans. Visual. Comput. Graphics*, vol. 6, no. 1, pp. 24–43, 2000.

[16] N. Linial, E. London, and Yu. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, pp. 215–245, 1995.

[17] L. Subramanian, S. Agarwal, Jennifer Rexford, and R. Katz, "Characterizing the internet hierarchy from multiple vantage points," in *IEEE Infocom 2002*, June 2002.

[18] Craig Labovitz, G. Robert Malan, and Farnam Jahanian, "Internet routing instability," in *ACM SIGCOMM'97*, Aug. 1997.

[19] Vern Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM Trans. Networking*, vol. 5, no. 5, pp. 601 – 615, Oct. 1997.

[20] B.M. Waxman, "Routing of multipoint connections," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1617–1622, 1988.

[21] Réka Albert and Albert-László Barabási, "Topology of evolving networks: local events and universality," *Physical Review Letters*, pp. 5234–5237, 11 Dec. 2000.

[22] B. Awerbuch and Y. Shavitt, "Topology aggregation for directed graphs," *IEEE/ACM Trans. Networking*, Feb. 2001.

[23] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, vol. 26, pp. 189–206, 1984.

[24] E. Ashkenazi, "Experiments with low-distortion low-dimensional embeddings of graphs into normed spaces," Graduate project, E.E.-Systems Department, Tel-Aviv University, Apr. 2002.

[25] B. Borchers, *CDSP 3.2, A library for semidefinite programming*, Dec. 2000, http://www.nmt.edu/∼borchers/csdp.html.

[26] J.A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.

[27] Numerical Recipes Software, *Numerical Recipes in C: The art of scientific computing*, Cambridge University Press, 1992.