

# Big-Bang Simulation for embedding network distances in Euclidean space

Yuval Shavitt    Tomer Tankel  
Dept. of Electrical Engineering,  
Tel-Aviv University, Israel  
Email: {shavitt,tankel}@eng.tau.ac.il

**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 scheme 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 extent 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] will be presented later.

<sup>1</sup>This research was supported by a grant from the United States - Israel Binational Science Foundation (BSF), Jerusalem, 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 [3], [4], [5], [6]. 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 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 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. The simplicity and low complexity of classical metric MDS [7] makes it appealing in these areas. Recently computer graphics researches [8] 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 sequences into similarity families [9].

Theoretical bounds on the **maximal** pair distortion and the dimension of the target space were derived by Linial

et al. [10] for any discrete metric space<sup>2</sup>. Perhaps the first use of graph embedding techniques in networking is due to Ng and Zhang [5] who suggested to estimate Internet host distances by embedding distances among Tracer nodes and other network regions. Their embedding technique sought minimum of 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 [5] is very sensitive to the initial coordinates for vertices. 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  $30 < n < 750$ .

BBS achieves equally good performance for a wide range of system friction coefficient. Moreover, in the case the pairwise field force is given by the difference between the Euclidean and network pair distances (see Eq. 7), the embedding is insensitive to small changes in the input graph. Indeed, even after modifying as many as 20%

<sup>2</sup>Though [10] 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.

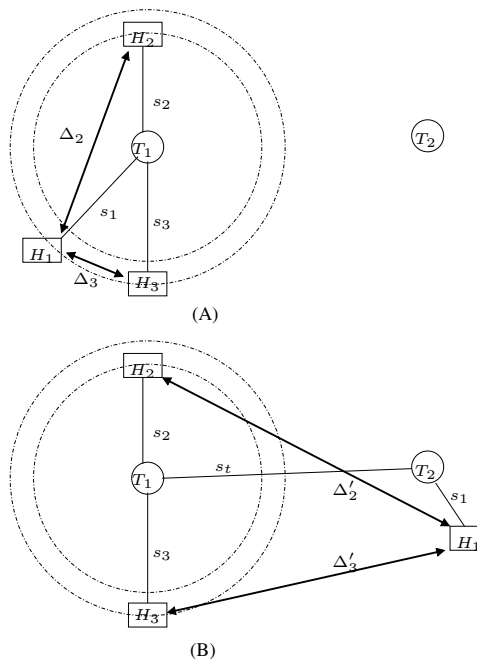


Fig. 1. Problem Statement

of the input edges, and resuming simulation from the previous particles positions, our simulation requires only .5% (for 150 node Waxman topology) and 5% (for 15 node BA topology) of the simulation time starting at the origin.

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, [4] 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  $\Delta_l$  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 [5] 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 dandrils connected to it [11] embedding of distances between nodes located in different dandrils 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.

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 [12] and Barabási-Albert (BA) [13] methods. In Sec. IV we apply BBS in two practical applications, topology aggregation and Internet distance maps, comparing it with previous results from [2] and [4], [5], respectively.

## II. BIG-BANG SIMULATION

### 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

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

Here  $v_k$ ,  $k = 1, \dots, n$  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.

Our method works in four calculation phases. The potential field force in each calculation phase  $p$  is derived from a phase-specific potential energy,  $E_T^{(p)}$ . The phase

pair embedding error function denoted by  $E_{ij}^{(p)}$ , assumes the form

$$E_{ij}^{(p)}(v_i, v_j) = \mathcal{F}(\|v_i - v_j\|, \Delta_{ij}), \quad (2)$$

for  $i \neq j$  and  $v_i \neq v_j$

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  $(i, j)$ . 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.

The potential function of the first phase  $E_T^{(1)}$ , Eq. (6), is equal to the simple squared error used in [5]. 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 the next phases is to reduce the distortion of large relative error edges at the price of slightly increasing the average relative error. The pair error functions in the next phases are increasingly sensitive to the directional relative error Eq. (15). Numerical stability is maintained however, because earlier phases are normally capable of substantially reducing the maximum pair embedding error.

A calculation phase consists of several iterations which moves 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 Newton's movement equations 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 particles position. They include the maximum particles velocity  $\max_i \|\dot{v}_i\|$  and maximum symmetric pair distortion  $d_{ij}$  Eq. (9).

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.

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

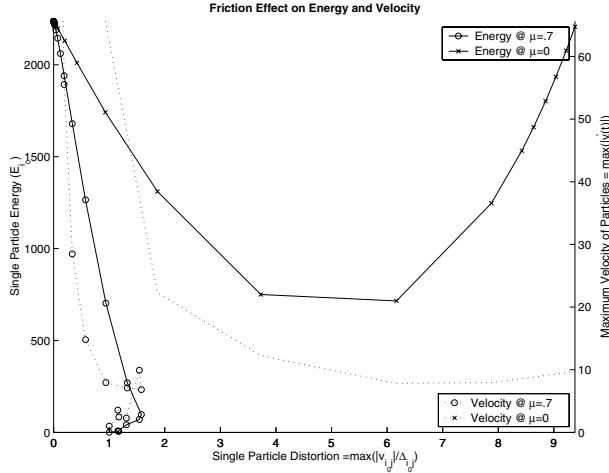


Fig. 2. Friction Effect on Energy and Velocity

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. 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-E) 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  $E_{i_0} = \sum_{j=1, j \neq i_0}^n E_{i_0j}(v_{i_0}, v_j)$  . 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 (x) depict no friction, that is  $\mu_k = 0$ . Each marked point represents 10 milli-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  $\|v_{i_0j}\|/\Delta_{i_0j} > 1$ , the field

force attracts the particles back to each other and they are stopped at equilibrium of  $\|v_{i_0j}\|/\Delta_{i_0j} = 1$ . 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  $\mu_s, \mu_k$ , are accounted for static and moving particles, respectively.

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  $n = 50$  tracers in Waxman and BA topologies. The figure indicates that all three values are almost indifferent to friction in the wide range  $0.003 \leq \mu_k \leq 3$ .

### C. 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  $v_{i_0}$

$$\vec{F}_{i_0} = -\nabla_{v_{i_0}} E_T(v_1, \dots) = - \sum_{i,j=1, i>j}^n \nabla_{v_{i_0}} E_{ij} . \quad (3)$$

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

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

Let  $F_{ij}$  denote the magnitude of the field force element induced on particle  $i_0$  by particle  $j$ , along the course of vector  $v_{j i_0}$ . Then the force affecting particle  $i_0$  is given by

$$\vec{F}_{i_0} = \sum_{\substack{j=1 \\ j \neq i_0}}^n F_{i_0j} \hat{v}_{j i_0} . \quad (4)$$

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. It can easily be shown [14] that

$$F_{i_0j} = \frac{d}{dx} \mathcal{F}(x, \Delta_{i_0j}) \Big|_{x=\|v_{j i_0}\|} . \quad (5)$$

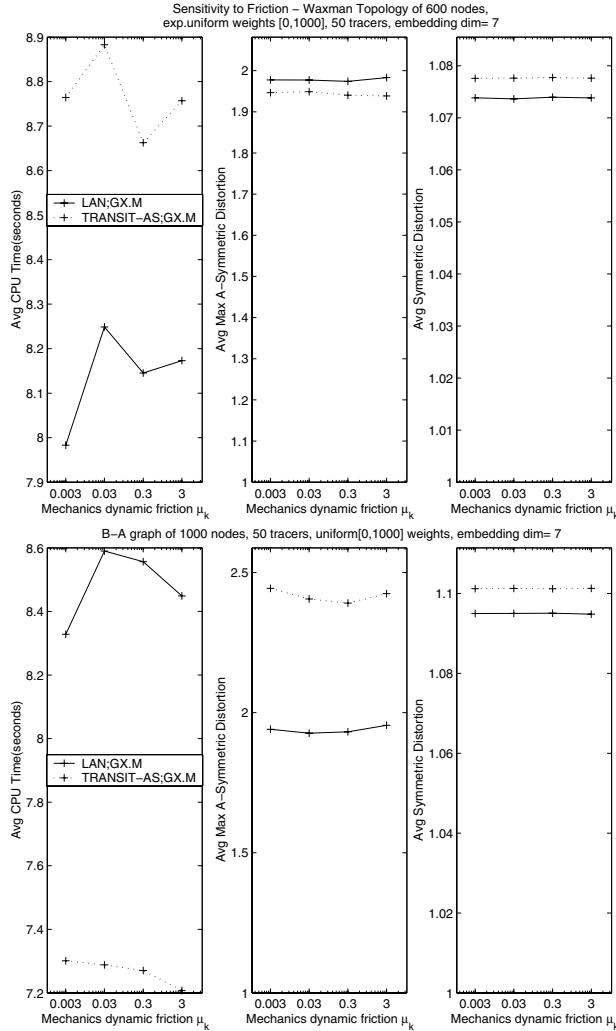


Fig. 3. Sensitivity to Friction Coefficients

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.

#### D. Error Functions of Phases

At the first phase the pair embedding error,  $E_{ij}^{(1)}$  is given by

$$\begin{aligned} E_{ij}^{(1)}(v_i, v_j) &= \mathcal{F}(\|v_i - v_j\|, \Delta_{ij}) \\ &= (\|v_i - v_j\| - \Delta_{ij})^2, \end{aligned} \quad (6)$$

that is the squared pair distance error. The field force induced between two particles is given by (5), that is

$$\begin{aligned} F_{ij}^{(1)} &= \mathcal{F}_x(\|v_i - v_j\|, \Delta_{ij}) \\ &= 2(\|v_i - v_j\| - \Delta_{ij}). \end{aligned} \quad (7)$$

The derivation of the induced forces in the rest of the phases is omitted due to lack of space, see [14] for details.

The second phase pair embedding error function is given by

$$E_{ij}^{(2)} = (d_{ij} - 1)^2 \quad (8)$$

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

$$d_{ij} = \max\left(\frac{\|v_i - v_j\|}{\Delta_{ij}}, \frac{\Delta_{ij}}{\|v_i - v_j\|}\right). \quad (9)$$

Substituting Eq. (6) in the above we find<sup>3</sup>

$$E_{ij}^{(2)} = \frac{E_{ij}^{(1)}}{\min(\|v_i - v_j\|, \Delta_{ij})^2}. \quad (10)$$

The pair embedding error function of the last two phases are functions of the second phase error

$$E_{ij}^{(3)} = \exp(E_{ij}^{(2)})^{\frac{3}{4}} - 1 = \exp(d_{ij} - 1)^{\frac{3}{2}} - 1, \quad (11)$$

$$E_{ij}^{(4)} = \exp(E_{ij}^{(2)}) - 1 = \exp(d_{ij} - 1)^2 - 1. \quad (12)$$

#### E. 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 the triangle inequality holds in a geometric space. 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  $16 \times 16$  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  $\mu_s =$

<sup>3</sup>A similar error function, with  $\Delta_{ij}^2$  replacing the denominator in Eq. (10), was used by Kamada and Kawai [15, eq.(1)] for undirected graph drawing. The sensitivity of 2nd order newton-raphson used there makes it impractical even for small 15 – 20 nodes graph.

.9. Trajectories were matched to the input points using only shift, rotation, and reflection transformation. We've 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. However, except for the trajectory matched by the transformation, no trajectory with zero friction, have even come close to its corresponding input points.

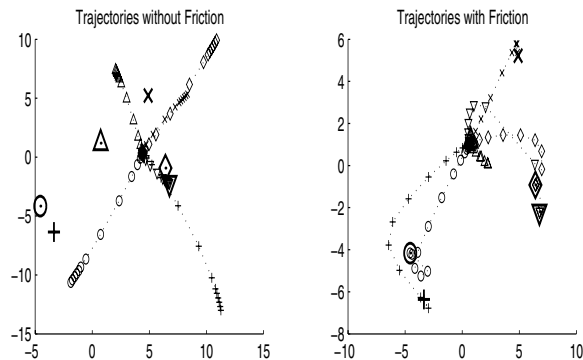


Fig. 4. Friction Effect on simulated metric trajectories

### III. EMBEDDING METHOD COMPARISON

In this section we compare our BBS method to three other Embedding methods, and also to a topology aggregation method [16].

#### A. Other Embedding Methods

We begin by briefly describing the four methods.

1) *Semi-Definite Programming*: (—)<sup>4</sup> SDP is a well known optimization technique that have drawn attention from diverse areas. The use of SDP for graph embedding was introduced in the seminal work of Linial, London, and Rabinovich [10]. The SDP problem in primal standard form is

$$\begin{aligned} \max \quad & \text{tr}(CX) \quad \text{s.t.} \\ & A(X) = \begin{bmatrix} \text{tr}(A_1X) \\ \text{tr}(A_2X) \\ \dots \\ \text{tr}(A_kX) \end{bmatrix} = a, \end{aligned} \quad (13)$$

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  $\Delta$ , and  $X$  is the variable. A Cholesky decomposition of the solution to the SDP problem,  $X =$

<sup>4</sup>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 '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 Johnson and Lindenstrauss [17] random projection from  $\mathcal{R}^n$  into  $\mathcal{R}^d$ .

2) *Multi-Dimensional-Scaling* (—\*—): The classical metric MDS [7] has a simple and low complexity implementation. The Matlab code we used is described in [8, Sec. 4].

3) *Down-Hill Simplex (DHS)* (+): This minimization method is known for a very long time [18], and was just recently used as an embedding method in [5]. For Euclidean embedding, the minimization target function is either equal to the total embedding error from phase 1,  $E_T^{(1)}$ , Eq. (6); or the normalized pair error given by

$$E_{ij}^{(N)} = \left( \frac{\|v_i - v_j\| - \Delta_{ij}}{\Delta_{ij}} \right)^2. \quad (14)$$

4) *MST2RST* (—): The MST2RST topology aggregation procedure, discussed in [2], represent a network by an aggregated graph of at most  $3n$  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 two or more Random spanning trees (RST) of this clique. This method had the lowest distance distortion among the aggregation methods considered in [2], and our comparison was done using the C-code used there.

#### B. Environment Details

The network graphs in our comparison were created according to both Waxman [12] and Barabási-Albert (BA) [13] methods. Only a subset of the graph nodes, was selected to be embedded, according to the characteristics of the two applications. For IDMaps [4] 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 [2], 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<sup>5</sup>. For Waxman networks, the edge weights were taken as  $[10^{exp} + .5]$ , 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  $[1, 1000]$ .

<sup>5</sup>However, for  $n = 150$  tracers, we performed only  $5 \times 5 = 25$  embedding, and for  $n = 750$  only 3 embedding, due to their large space and time requirements

### C. Performance Metrics

The *symmetric pair distortion*,  $d_{ij}$ , is defined in Eq. (9). The *directional relative error*,  $E_{rel}$ , was defined by [5, Eq. 4] and for Euclidean embedding is given by

$$E_{rel} \equiv \frac{\|v_i - v_j\| - \Delta_{ij}}{\min(\|v_i - v_j\|, \Delta_{ij})^2} \quad (15)$$

Comparing with Eq. (10) we find that  $|E_{rel}| = \sqrt{E_{ij}^{(2)}}$ , and substituting Eq. (8) we thus have  $d_{ij} = 1 + |E_{rel}|$ . The *average symmetric pair distortion* is given by  $\bar{d}_{ij} = 1 + \overline{|E_{rel}|}$ .

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

$$\min \left( c_1 c_2 : c_1 \Delta_{ij} \geq \|v_i - v_j\| \geq \frac{1}{c_2} \Delta_{ij} \right), \quad (16)$$

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

An alternative measure, defined by Linial et al. [10], is the *one-sided distortion*

$$\min \left( c : \Delta_{ij} \geq \|v_i - v_j\| \geq \frac{1}{c} \Delta_{ij} \right), \quad (17)$$

over all node pairs  $i, j = 1, \dots, n; i \neq j$ . 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  $\bar{d}_{ij}$ .

### D. Comparison Results

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 DHS, SDP, and BBS 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 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 then MDS, e.g.  $P(d_{ij} > 1.3)$ , is .05 for LAN-BBS and TRANSIT-BBS vs. .6 for LAN-MDS and .98 for TRANSIT-MDS.

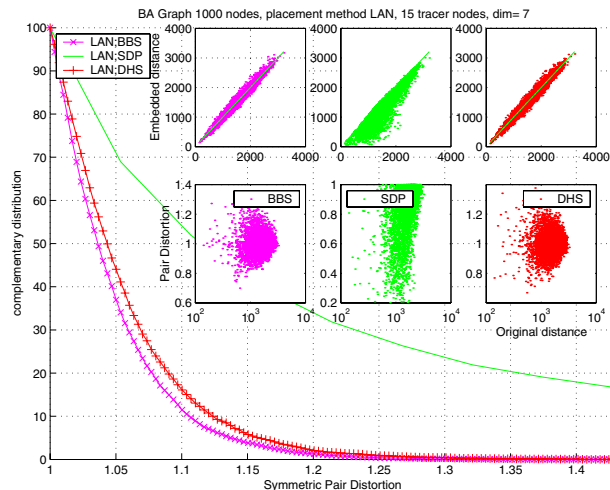


Fig. 5. BA Symmetric Pair Distortion

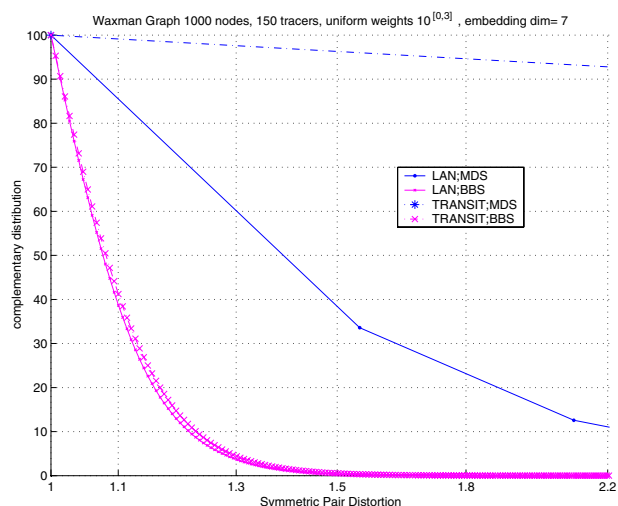


Fig. 6. Waxman Symmetric Pair Distortion

b) *Embedded Graph Size*: Fig. 7 depicts the performance of the different methods as a function of the number of embedded nodes. For  $n > 30$ , the long running times of SDP and high sensitivity of DHS exclude them from the comparison. The linear approximation for  $n \geq 30$ , depicted in dotted lines in the average CPU time graph, indicates that MDS, MST2RST and BBS has complexity  $Cn^2$ . The value of  $C$  calculated on IBM Thinkpad 600X with Pentium-3 450Mhz processor, is  $3 \times 10^{-5}$  for MDS,  $4 \times 10^{-4}$  for MST2RST, and  $3 \times 10^{-3}$  for BBS. The BBS embedding distortion is the lowest for all graph sizes except for  $n = 750$  where MST2RST distortion is smaller in TRANSIT, 6 vs. 24, and in LAN, 7.5 vs. 12.5. BBS has the lowest average symmetric pair

distortion in all graph sizes. The symmetric pair distortion of MST2RST and SDP are not comparable with the rest of the methods, since all MST2RST edges are expanded and most SDP edges are contracted.

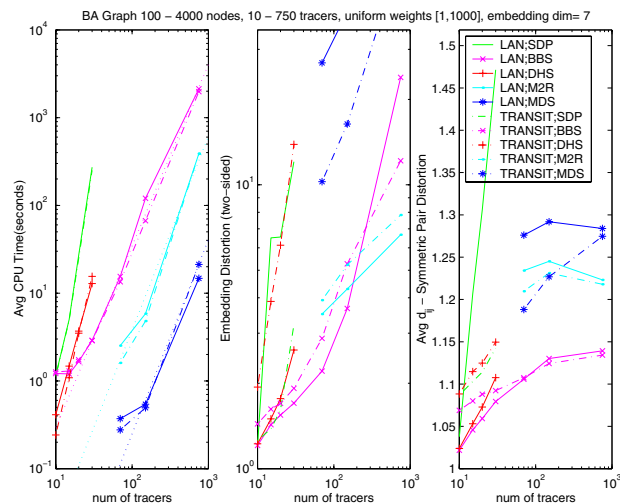


Fig. 7. Performance Metrics vs B-A Graph Size

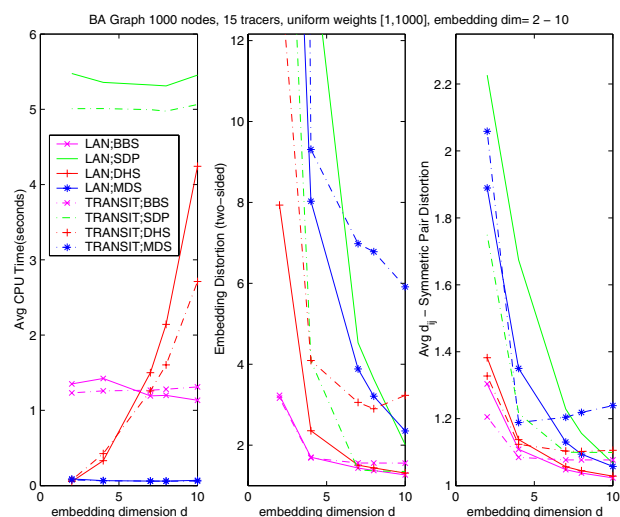


Fig. 8. Performance Metrics vs B-A Embedding Dimension

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 improve when the embedding dimension increases. For the BA topology the knee point, where the improvement diminishes, is at  $d = 7$  as indicated by Ng and Zhang [5].

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.

#### 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 MST2RST 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  $P$ ,  $P \gg 1$ , in the definition of pair embedding error functions

$$\hat{E}_{ij}^{(p)}(v_i, v_j) = \begin{cases} E_{ij}^{(p)}(v_i, v_j) & \text{if } \frac{\|v_i - v_j\|}{\Delta_{ij}} \geq 1 \\ PE_{ij}^{(p)}(v_i, v_j) & \text{otherwise.} \end{cases} \quad (18)$$

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  $P_1$ ;  $P \gg P_1 > 1$ . 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  $P$ . In the rest of the phases, the calculation continues with the field force which directly incorporates  $P$ .

Fig. 9 illustrates the effect of the price factor  $P = 512$  on our embedding method compared to embedding without it, i.e.,  $P = 1$ . 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  $d = 2$  our performance is worth than MST1RST, at  $d \geq 3$  it supersedes MST2RST.

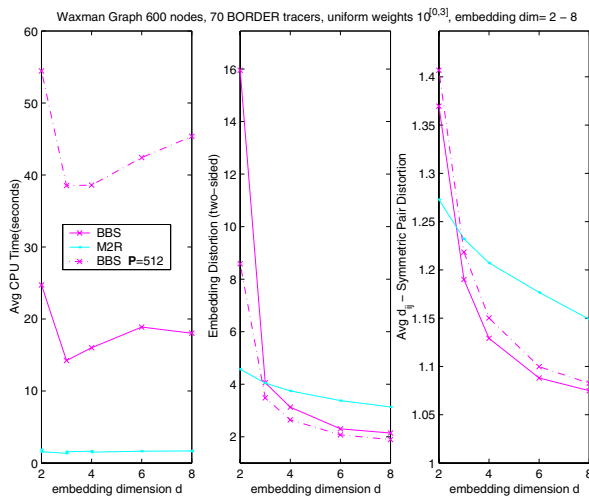


Fig. 9. Waxman Symmetric vs. A-Symmetric Aggregation

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 than 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.

### B. Internet Distance Estimation

IDMaps [4] is a project that aims to build a global architecture for Internet host distance estimation and distribution. The architecture is based on Tracers, which

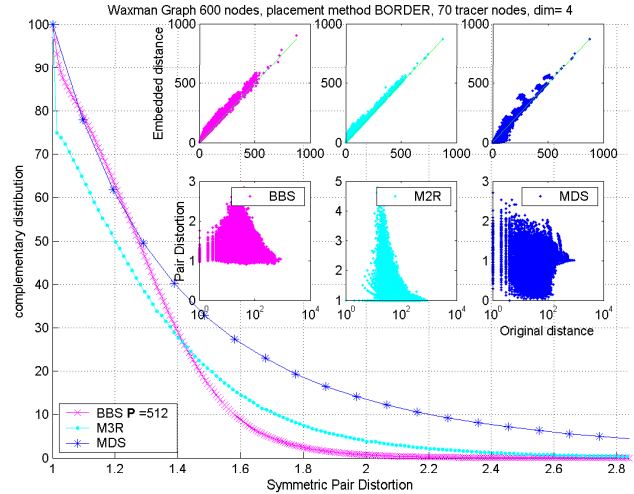


Fig. 10. Aggregation Distortion Histograms

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 [11]. Indeed, due to the triangle inequality, the Euclidean distance between distant nodes is bound to be smaller than 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. (15). 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 [4]. We used 15 Tracers and measured distances from each host to all 15 Tracers, that matches the conditions of the similar figures in [5]. 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.

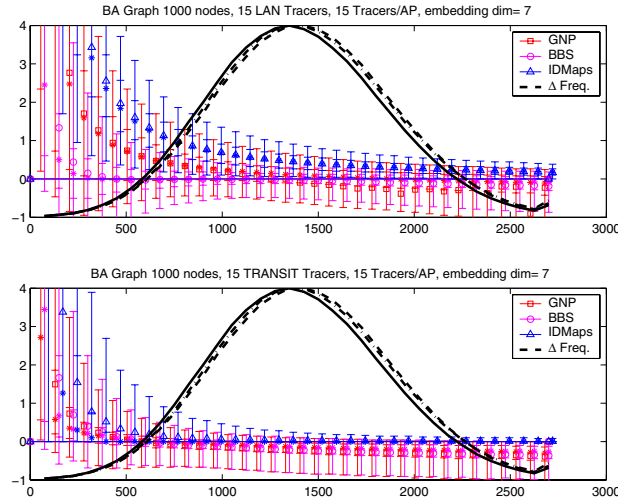


Fig. 11. Relative Error (stand-alone)

IDMaps additive estimation for TRANSIT is less accurate than Euclidean embedding at  $\Delta < 400$ , having larger positive relative errors, but more accurate for  $\Delta > 700$ , having 0 average and median there. Had we known the value of  $\Delta$  for each pair we could chose between IDMaps additive and Euclidean embedding estimations, using a threshold of  $\Delta_{Th} = 550$ . 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  $R$  between the two, given by

$$R = \frac{Euclidean\ distance}{IDMaps\ additive}, \quad (19)$$

and the estimated distance is selected as follows

$$\begin{cases} Euclidean\ distance & \text{if } R < R_{Th} \\ IDMaps\ additive & \text{otherwise.} \end{cases} \quad (20)$$

Fig. 12 illustrates the improvements in accuracy of our estimation compared to IDMaps additive, with threshold

$R_{Th} = 0.45$ . As in the previous figure we used 15 Tracers and measured distances from each host to all 15 Tracers.

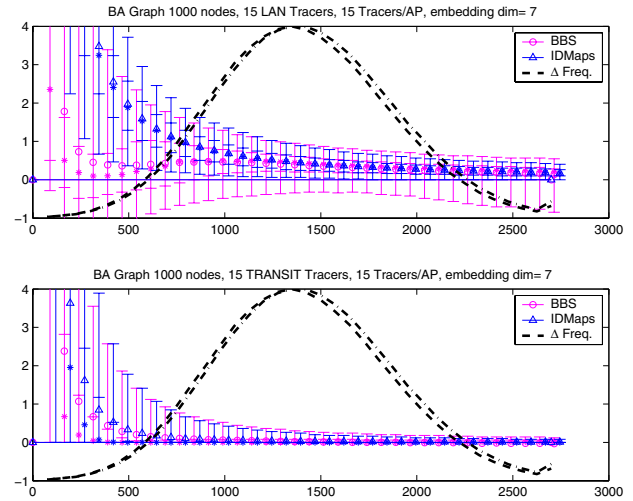


Fig. 12. Relative Error (Thld. selection)

One could have selected the closet mirror as the one with smallest distance estimated by (20). 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

$$R' = \frac{\min_k \{Euclidean\ distance\}}{\min_k \{IDMaps\ additive\}}, \quad (21)$$

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

$$\begin{cases} k_{min}^{(Euclidean)} & \text{if } R' < R'_{Th} \\ k_{min}^{(IDMaps\ additive)} & \text{otherwise.} \end{cases} \quad (22)$$

We compared the mirror selection accuracy of IDMaps additive with BBS using the selection criterion of (22) with  $R'_{Th} = .45$ , i.e., the same threshold value used for distance estimation. Following [4] 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 'x' mark, and is depicted with increasing marker sizes.

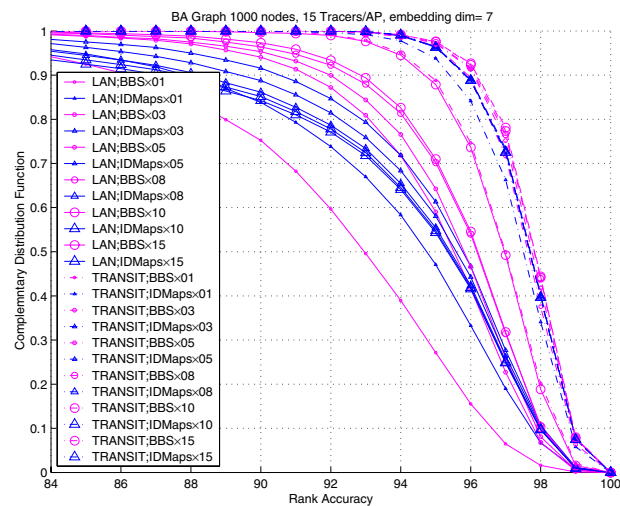


Fig. 13. Mirror Selection Accuracy (Thld. selection)

Additive IDMaps becomes *less accurate* when more than 2 Tracer measurements are used. The accuracy of our threshold selection however, improves with each additional Tracer measurement. With 3 measurements its more accurate than additive IDMaps for both LAN and TRANSIT placement. With 15 measurements for LAN placement its 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.

## 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.

## REFERENCES

- [1] "Private network - network interface specification version 1.0 (PNNI)," The ATM Forum technical committee," Tech. Recommendation, Mar 1996, af-pnni-0055.000.
- [2] B. Awerbuch and Y. Shavitt, "Topology aggregation for directed graphs," *IEEE/ACM Trans. Networking*, Feb 2001.
- [3] W. Theilmann and K. Rothermel, "Dynamic distance maps of the internet," in *Infocom*, 2000, tel Aviv, Israel.
- [4] 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.
- [5] T. Ng and H. Zhang, "Predicting internet network distance with coordinates based approaches," in *Infocom*, 2002.
- [6] 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, pp. 1369–1382, Sept. 2002.
- [7] W. Togerson, "Multidimensional scaling of similarity," *Psychometrika*, vol. 30, pp. 379–393, 1965.
- [8] G. Zigelman, R. Kimmel, and N. Kiryati, "Texture mapping using surface flattening via multidimensional scaling," *IEEE Trans. Visual. Comput. Graphics*, 4-6 2002.
- [9] 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.
- [10] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, pp. 215–245, 1995.
- [11] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz, "Characterizing the internet hierarchy from multiple vantage points," in *Infocom*, 2002.
- [12] B. Waxman, "Routing of multipoint connections," *IEEE J. Select. Areas Commun.*, pp. 1617–1622, 1988.
- [13] R. Albert and A.-L. Barabási, "Topology of evolving networks: local events and universality," *Physical Review Letters*, pp. 5234–5237, 11 Dec. 2000.
- [14] Y. Shavitt and T. Tankel, "Big-Bang simulation for embedding network distances in Euclidean space," E.E.-Systems Department, Tel-Aviv University, Tech. Rep., July 2002.
- [15] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, pp. 7–15, 1989.
- [16] B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt, "Routing through networks with hierarchical topology aggregation," *Journal of High-Speed Net.*, vol. 7, 1998.
- [17] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, pp. 189–206, 1984.
- [18] J. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, pp. 308–313, 1965.