Journal of Interconnection Networks
Vol. 15, Nos. 1 & 2 (2015) 1550001 (27 pages)

DOI: 10.1142/S0219265915500012

World Scientific
www.worldscientific.com

Mutual Embeddings

ILKER NADI BOZKURT
Duke University, Department of Computer Science
Levine Science Research Center, 308 Research Drive, Durham, NC 27708
ilker@cs.duke.edu

HAI HUANG
Intel Corporation, Chandler, AZ 85226
hai.huang2@intel.com

BRUCE MAGGS
Duke University, Department of Computer Science - Akamai Technologies
Levine Science Research Center, 308 Research Drive, Durham, NC 27708
bmm@cs.duke.edu

ANDRÉA RICHA
Arizona State University
School of Computing, Informatics, and Decision Systems Engineering
Box 878809, Tempe, AZ 85287
andrea.richa@asu.edu

MAVERICK WOO
Carnegie Mellon University, CyLab
Robert Mehrabian Collaborative Innovation Center,
4720 Forbes Avenue, Pittsburgh, PA 15213
pooh@cmu.edu

Received 11 February 2015
Accepted 15 September 2015

This paper introduces a type of graph embedding called a *mutual embedding*. A mutual embedding between two n-node graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is an identification of the vertices of V_1 and V_2, i.e., a bijection $\pi : V_1 \to V_2$, together with an embedding of G_1 into G_2 and an embedding of G_2 into G_1 where in the embedding of G_1 into G_2, each node u of G_1 is mapped to $\pi(u)$ in G_2 and in the embedding of G_2 into G_1 each node v of G_2 is mapped to $\pi^{-1}(v)$ in G_1. The identification of vertices in G_1 and G_2 constrains the two embeddings so that it is not always possible for both to exhibit small congestion and dilation, even if there are traditional one-way embeddings in both directions with small congestion and dilation. Mutual embeddings arise in the context of finding preconditioners for accelerating the convergence of iterative methods for solving systems of linear equations. We present mutual embeddings between several types of graphs such as linear arrays, cycles, trees, and meshes, prove lower bounds on mutual embeddings

between several classes of graphs, and present some open problems related to optimal mutual embeddings.

Keywords: Graph embedding; mutual embedding; support tree preconditioners.

1. Introduction

1.1. *Graph embedding*

Graph embeddings have proven useful in many contexts, such as mapping a graph describing a computation onto a network of processors, minimizing area in a VLSI layout, or enabling one network of processors to emulate another with a different topology.[15,17,20] Graph embedding has also been used in the design and analysis of preconditioned iterative solvers for symmetric, diagonally dominant (SDD) systems of linear equations. There has been a long line of work in this area in the last two decades, culminating in Refs. 25 and 31. We refer the reader to Refs. 2, 3, 6, 11, 22, 25 and 31 for the history and references.

Graph embedding can be defined for weighted and unweighted graphs, and the embedding can map an edge to a single simple path or to a set of simple paths each with a weight (fraction). We give the definition for the general case for completeness, i.e., fractional embedding for weighted graphs, even though except for Section 2 our results in this paper deal with the simpler case, where we have unweighted graphs and edges are mapped to single non-fractional paths.

Let $G = (V_G, E_G)$ denote the guest graph and $w_G(e)$ denote the weight of an edge e of G. Similarly, let $H = (V_H, E_H)$ denote the host graph and $w_H(e)$ denote the weight of an edge e of H. The embedding of G into H is specified by a pair of mappings. The first mapping maps each vertex in G to a vertex in H. The second mapping maps each edge e in G to a set of simple paths, denoted $P_H(e)$, between the images of its endpoints in H, where each path is designated with a non-negative fraction and the fractions over all paths in the set sum to 1.

The quality of an embedding can be measured by several quantities such as congestion, dilation, expansion and load. *Expansion* is defined as $|V_H|/|V_G|$ and *load* is the maximum number of vertices of G mapped to a single vertex of H. The *dilation* of an edge e in G is defined as $\max_{p \in P_H(e)} |p|$, i.e., the length of the longest path in the set of paths e is mapped to in H; the dilation of the embedding is the maximum dilation over all edges in G. The *congestion* of an edge h in H is $\sum_{\{e \in E(G) | h \in p, p \in P_H(e)\}} \frac{f_p(e) w_G(e)}{w_H(h)}$ where p denotes a path in $P_H(e)$ and $f_p(e)$ its fraction respectively. The congestion of the embedding is the maximum congestion over all edges in H. In this work, we focus on load-1 embeddings between graphs of the same size, so the congestion and dilation of the embedding are the quantities of interest.

When we have an unweighted graph and each edge is mapped to a single path, dilation is again the length of the longest path used in the embedding. The congestion of an edge counts the appearance of that edge in the paths used in the embedding.

The congestion of the embedding is again the maximum congestion over all edges in H.

Before we go on, let us present two small and well-known lemmas concerning the congestion and dilation of an embedding for the simpler case described in the above paragraph.

The first lemma relates the congestion of an embedding to the bisection width of the guest and host graphs. Bisection width of a graph is the minimum number of edges that has to be removed to obtain two disconnected subgraphs with the same number of vertices. The second lemma relates the dilation of an embedding to the diameter of the guest and host graphs.

In the following two lemmas, G_1 and G_2 are two arbitrary connected graphs and let c_1 and d_1 denote the congestion and dilation of an embedding of G_1 into G_2 respectively.

Lemma 1.1. *Let $BW(G_1)$ and $BW(G_2)$ denote the bisection widths of G_1 and G_2 respectively. Suppose G_1 is embedded into G_2 with load 1, then $c_1 \geq \frac{BW(G_1)}{BW(G_2)}$.*

Lemma 1.2. *Let $D(G_1)$ and $D(G_2)$ denote the diameters of G_1 and G_2 respectively. Suppose G_1 is embedded into G_2 with load 1, then $d_1 \geq \frac{D(G_2)}{D(G_1)}$.*

1.2. *Mutual embedding*

This paper introduces the notion of *mutual embeddings*, which is defined on two graphs with the same number of vertices. Consider the graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = |V_2|$. A mutual embedding between G_1 and G_2 is an embedding of G_1 into G_2 and another embedding of G_2 into G_1, with the additional constraint that the vertex mappings in the two embeddings are functional inverses of each other. Thus, it makes sense to simplify our notation by consolidating the two mappings into a bijection $\pi : V_1 \to V_2$.

Problem Statement. Given graphs G_1 and G_2, the mutual embedding problem is to find the bijection π, and the mappings of edges of G_1 to paths in G_2, and vice versa, while minimizing some function of the congestions and the dilations of the two embeddings, denoted c_1, c_2, d_1, and d_2. For example, we may wish to minimize the product $c_1c_2d_1d_2$.

Having said above, finding the optimal mutual embedding (in the desired sense) between two graphs is a difficult problem. In fact, even finding optimal one way embeddings when we have two arbitrary graphs is difficult. Bandwidth minimization problem, in which a minimum dilation embedding of an arbitrary graph to a line is sought, is NP-hard.[23] So, finding an embedding between two arbitrary graphs with minimum dilation is NP-hard as well, as we can easily reduce bandwidth minimization to finding an embedding with optimal dilation between two arbitrary graphs. Finding an optimal embedding based on congestion is not easier either. This is because the cutwidth minimization problem, which asks for an embedding of an

arbitrary graph into a line with minimum congestion, is NP-hard for general graphs as well.[9] Therefore we will limit ourselves to proving lower bounds for mutual embeddings in this paper. It is possible to find optimal embeddings between special types of graphs and embeddings between different types of graphs have been studied extensively.[1,7,10,13–15,17,24,27–30] Some of the mutual embeddings we present in this paper are based on previously discovered one-way embeddings between different types of graphs.

There can be many different mutual embeddings between two graphs G_1 and G_2. The next two examples show two different mutual embeddings between a linear array[a] and a cycle. Let $G_1 = (V_1, E_1)$ denote a cycle and $G_2 = (V_2, E_2)$ denote a linear array, both with n nodes. Suppose the nodes in the linear array are numbered 1 through n in consecutive order, and that the nodes in the cycle are numbered in a similar fashion with node n being connected to node 1. Throughout, we assume that all graphs are undirected.

Example 1.1. Let the map $\pi : V_1 \to V_2$ be such that $\pi(i) = i$. Each edge of the linear array is mapped to the identical edge in the cycle. Each edge of the cycle is mapped to the unique path between the images of its endpoints in the linear array. This defines a mutual embedding which is shown in Figure 1.

For this mutual embedding, we have $c_2 = 1$ and $d_2 = 1$ since the linear array is a subgraph of the cycle. We also have $c_1 = 2$ and $d_1 = n - 1$ because the edge connecting nodes 1 and n in the cycle has dilation $n - 1$ in the linear array, and adds congestion 1 to every edge in the linear array.

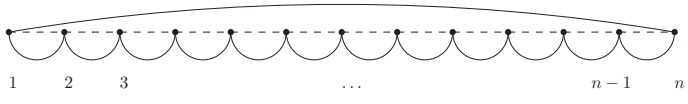

Fig. 1. An example mutual embedding between a linear array (dashed edges) and a cycle.

Example 1.2. Let the map $\pi : V_1 \to V_2$ be such that

$$\pi(i) = \begin{cases} 2i - 1, & \text{if } 1 \leq i \leq \lceil \frac{n}{2} \rceil \\ 2(n - i + 1), & \text{otherwise.} \end{cases}$$

Each edge of the linear array is mapped to the shortest of the two paths between the images of its endpoints. Each edge of the cycle is mapped to the unique path between the images of its endpoints in the linear array. This defines a mutual embedding. Figure 2 shows how the cycle is embedded into the linear array in this example.

[a]We prefer the term linear array to line, which is a topology where processors are arranged in a line.

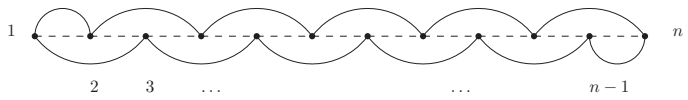

Fig. 2. Another example mutual embedding between a linear array (dashed edges) and a cycle.

In this mutual embedding, the cycle is embedded into the linear array by using jumps of size 2 starting from the first node. When we reach to the end of the linear array, we make a turn and make jumps of size 2 again (1 where necessary). So, we have $c_1 = 2$ and $d_1 = 2$. For the embedding of the linear array into the cycle, consider the edge in the middle of the linear array. The path that this edge is mapped to has to traverse half of the cycle either from left or right, so $d_2 = \lceil \frac{n}{2} \rceil$. Now consider all the edges of the linear array in the left half. All these edges will traverse the cycle from the left, so cycle edges incident on vertex 1 of the linear array have $c_2 = \lceil \frac{n}{2} \rceil$.

It is important to note that two embeddings taken together do not necessarily form a mutual embedding. The embedding of the linear array into the cycle in Example 1.1 (which has $c_2 = 1$ and $d_2 = 1$) and the embedding of the cycle into the linear array in Example 1.2 (which has $c_1 = 2$ and $d_1 = 2$) do not form a mutual embedding together. This is because the vertex mappings are not inverses of each other. Moreover, the two embeddings of a mutual embedding constrain each other and it is often not possible to find mutual embeddings with small congestion/dilation in both directions. In examples 1.1 and 1.2, we have small congestion/dilation in one direction; whereas, at least one of these quantities is $\Omega(n)$ in the other direction. If we wanted to minimize maximum dilation/congestion instead, we could have another mutual embedding in which we take jumps of size $\sqrt{n}$ while embedding the cycle to the linear array. Note that, we have $d_1 d_2 = \Omega(n)$ for both examples. Later we will show that for any mutual embedding between the linear array and the cycle, we have $d_1 d_2 = \Omega(n)$.

After these introductory examples, we will finish this section with some simple observations. When we have a mutual embedding between graphs A and B, and a mutual embedding between graphs B and C, we can use these mutual embeddings to get a mutual embedding between A and C. Moreover, we will also get trivial upper bounds on dilation and congestion from these two mutual embeddings. We will state this obvious result as a theorem without proof.

Theorem 1.1. Suppose we have graphs A, B, and C and we have a mutual embedding $\tau : A \to B$ and another mutual embedding $\sigma : B \to C$. Then, $\sigma \circ \tau$ defines a mutual embedding between A and C. Moreover, we have $d_{A\to C} \leq d_{A\to B} d_{B\to C}$, $c_{A\to C} \leq c_{A\to B} c_{B\to C}$, $d_{C\to A} \leq d_{C\to B} d_{B\to A}$, and $c_{C\to A} \leq c_{C\to B} c_{B\to A}$.

As a final observation, note that any lower bounds that apply to traditional one-way embeddings carry over to mutual embeddings. For example, Koch *et al.* proved lower bounds on congestion and dilation of embedding complete binary trees into complete ternary trees[15]; a similar result can also be proved in the opposite

direction. In these lower bounds either the congestion or the dilation is $\omega(1)$ but not both. Hence, in the mutual embedding, these lower bounds hold in both directions.

1.3. *Notation and summary of results*

In the rest of the paper, we continue using $BW(G)$ and $D(G)$ notation as in Lemmas 1.1 and 1.2 to refer to the bisection width and the diameter of a graph G respectively. In a mutual embedding of graphs G_1 and G_2, c_1 and d_1 refer to the congestion and dilation of embedding G_1 into G_2 respectively. Similarly, c_2 and d_2 refer to the congestion and dilation of embedding G_2 into G_1 respectively. We also use the notation $d_{G\to H}$ to denote the dilation of embedding G into H, when the the direction of embedding is not immediately clear from the context, such as the embeddings in Section 4 where we have embeddings between three different graphs. Unless otherwise stated, each graph in a mutual embedding is assumed to have n vertices.

The rest of the paper is organized as follows. In Section 2, we describe the relationship between mutual embeddings and support tree preconditioners. The embeddings in this section make use of fractional paths (flows). The contributions in this section include a mutual embedding between a graph G and its Räcke complement (described later) and a different version of the (so called) congestion-dilation lemma, which is a well-known result in support theory. Section 3 deals with mutual embeddings between linear arrays and arbitrary graphs, and includes a result to obtain lower bounds on products of dilations. This section includes an interesting result showing that the product of dilations in a mutual embedding between a linear array and cycle is $\Omega(n)$, even though the two graphs are very similar to each other, and, simple and optimal embeddings exist in either direction. Section 4 generalizes the technique given in the previous one to mutual embeddings between two arbitrary graphs. The relationship of product of dilations in our mutual embeddings and distortion in metric space embeddings is examined in Section 5. The rest of the paper examines mutual embeddings between meshes of different dimensions and also between fat-trees and meshes. We discuss possible extensions and future work in the conclusion.

2. Mutual Embeddings and Support Tree Preconditioners

The reader may ask the question, why are mutual embeddings important? One motivation arises in analyzing the running time of a class of iterative solvers for systems of linear equations $Ax = b$, where A is a Laplacian matrix, and, where another Laplacian matrix B is used as a preconditioner. The support-tree conjugate gradient (STCG) algorithm of Gremban *et al.* is an example of such an iterative solver for Laplacian systems,[12] where the performance of the preconditioner is determined by the support of B for A, denoted as $\sigma(A/B)$. Following,[11] the support of matrix B for A is defined as

$$\sigma(A/B) = \min\{\tau : \tau B - A \text{ is positive semi-definite}\}.$$

Gremban experimentally showed that STCG works well on meshes[12] and also bounded the number of iterations to converge for certain classes of graphs in his thesis.[11] In general, the iteration count of such solvers can be bounded by the generalized condition number of A and B, $\kappa(A, B)$, which, for Laplacians, is equal to $\sigma(B/A)\sigma(A/B)$.[22] Graph embedding comes in to the picture for bounding the support, as shown by the following well-known so called congestion-dilation lemma.

Lemma 2.1. *Given an embedding of G into H, $\sigma(L(G)/L(H)) \leq c_{G\to H} d_{G\to H}$, where $L(G)$ and $L(H)$ denote the Laplacians of G and H respectively.*

The proof and slight variations of the lemma can be found in Refs. 3 and 5. The next lemma emphasizes that if we have a mutual embedding between two arbitrary weighted graphs G_1 and G_2, we can bound the condition number (and, hence the iteration count) for using $L(G_1)$ as a preconditioner for $L(G_2)$, and vice versa. The quality of the mutual embedding, based on the quantity $c_1 d_1 c_2 d_2$, is a direct indicator of how well the graphs G_1 and G_2 support each other.

Lemma 2.2. *Given a mutual embedding between G_1 and G_2,*

$$\kappa(L(G_1), L(G_2)) \leq c_1 d_1 c_2 d_2.$$

Proof. The result follows from Lemma 4.8 of Gremban[11] and the congestion-dilation lemma. Note that, we cannot bound $\kappa(A, B)$ using arbitrary embeddings in two directions; a mutual embedding is necessary as implied by Lemma 4.8 of Gremban. □

Maggs *et al.* presented an algorithm for finding a preconditioner for arbitrary graphs for the support tree approach[22]; in particular, they showed that the decomposition trees of Bienkowski *et al.*, proposed for constructing oblivious routing schemes,[4] can be used as preconditioners for STCG. Originally, Räcke showed that by constructing a decomposition tree T corresponding to an arbitrary graph G, a routing method which minimizes congestion can be obtained by routing requests on T instead of G.[26] The leaves of T correspond to vertices of G and each internal node u_t of T correspond to a cluster of nodes S_{u_t} of G. The root corresponds to the entire vertex set. The decomposition trees proposed by Bienkowski *et al.* are similar, however the authors also presented a polynomial time algorithm to construct T. In either case, during the construction of T, a concurrent multi-commodity flow problem (CMCFP) is solved for each cluster. The intermediate locations in a route between nodes u and v of G are obtained from the leaf-to-leaf path between u and v on T and the solutions of the CMCFPs in each cluster are used to find the set of paths between the internal nodes of T.

When T is used as a preconditioner for G, an upper bound on $\kappa(G, T)$ has to be obtained to bound the number of iterations until convergence is achieved. Maggs *et al.* obtains this upper bound in several steps. We briefly summarize their approach

below with the additional purpose of giving an example of a mutual embedding with fractional flows (paths).

First, support is defined for matrices of different size and $\sigma(G/T)$ is bounded by embedding G into leaves of T, resulting in congestion ≤ 1 and dilation $O(\log n)$.

The Räcke complement is defined for bounding $\sigma(T/G)$. As mentioned above, Bienkowski *et al.* define a CMCFP for each cluster S_{u_t} (corresponding to a node u_t in T) during initialization. This CMCFP can be represented with a complete graph K_{u_t} on the vertices of S_{u_t}, with edge weights representing the demands. The overlapping of these complete graphs is called the Räcke complement of T, denoted by $RC(T)$. $RC(T)$ is a complete graph on the vertex set V and the weight of an edge (u, v) is the sum of its weights (demands) in each K_{u_t} it appears in. Using a transitivity property of the support, a bound on $\sigma(T/G)$ is obtained by examining $\sigma(T/RC(T))$ and $\sigma(RC(T)/G)$ separately. $\sigma(T/RC(T))$ is bounded using an electrical argument by viewing the weighted graph as a resistive network. Finally, $\sigma(RC(T)/G)$ is bounded by embedding $RC(T)$ into G.

In the embedding of $RC(T)$ into G, an edge (u, v) is mapped to the overlapping of the flow paths between u and v in the solution of the CMCFPs which are created during the construction of T. To bound the congestion and dilation of this embedding, the embedding of each K_{u_t} is analyzed individually. The congestion in each component is bounded by $O(\log^3 n)$ and since there are $O(\log n)$ levels in the decomposition tree, $O(\log^4 n)$ is obtained as an upper bound on the congestion. An upper bound on the dilation is obtained by making use of the concept of flow number introduced by Kolman and Scheideler.[16] $O(\alpha^{-1}(G)\Delta(G)\log^3 n)$ is the obtained upper bound, where $\alpha = \min_{U\subset V, |U|\leq |V|/2} c(U, V\backslash U)$ is the minimum edge expansion of the graph and $\Delta = \max_{v\in V} c(v)$ is the maximum total incident weight on any vertex.

Next, we will show that it is possible to find a good mutual embedding between G and the corresponding Räcke complement $RC(T)$. We describe the mutual embedding for Räcke's decomposition trees. For the decomposition trees of Bienkowski *et al.*, the mutual embedding is obtained in a similar fashion and we will highlight the differences as we go along. The decomposition trees of Räcke are easier to explain, as there are nodes with different colors and differences in routing based on color in the former.

Lemma 2.3. *Let $G = (V, E)$ be an arbitrary, weighted, connected graph and let $T = (V_T, E_T)$ be the corresponding decomposition tree as originally proposed by Räcke. There is a mutual embedding between G and $RC(T)$ such that $d_{G\to RC(T)} = O(\log n)$, $c_{G\to RC(T)} = 1$, $d_{RC(T)\to G} = O(\alpha^{-1}(G)\Delta(G)\log^2 n)$ and $c_{RC(T)\to G} = O(\log^3 n)$.*

Proof. Since the graphs G and $RC(T)$ are defined on the same set of vertices, we naturally map each node v of G to the same node v in $RC(T)$.

The embedding of $RC(T)$ into G is as given above. $RC(T)$ is the union of complete graphs K_{v_t} corresponding to the flow problems in each cluster S_{v_t}. An edge

(u, v) is mapped to the overlapping of the flow paths in the solution of these CMCFPs in which they appear in. Räcke showed that all the $|V_T|$ sets of flows in the CMCFP solutions can be overlapped and then embedded in G while inducing $O(\log^3 n)$ congestion, given the decomposition tree constructed by his algorithm. For the dilation of this embedding, n is a simple upper bound. However, as described above, we can use the concept of flow number of Kolman and Scheideler and their flow shortening lemma. In this case, we get the tighter bound, $O(\alpha^{-1}(G)\Delta(G)\log^2 n)$; where $\alpha = \min_{U\subset V, |U|\leq |V|/2} c(U, V\backslash U)$ is the minimum edge expansion of the graph and $\Delta = \max_{v\in V} c(v)$ is the maximum total incident weight on any vertex. For the details, we refer the reader to Section 4.6 of Maggs *et al.*[22]

To embed G into $RC(T)$, we will use the decomposition tree T as a bridge, by first embedding G into T, and then embedding T into $RC(T)$. T has more vertices than G and $RC(T)$, but we will only use it for finding an embedding between G and $RC(T)$. While embedding G into T, we map each vertex of G to the corresponding leaf node in T. The dilation is $O(\log n)$ since this is the height of T. The congestion of this embedding is 1, because the weight assigned to each edge of T is enough to support the traffic between its cluster and the parent cluster. The weight of an edge between u_t and its child v_t is defined as $\text{out}(S_{v_t})$ which is basically the sum of all edge weights leaving the cluster S_{v_t}. For details, we again refer the reader to Section 4.2 of Maggs *et al.*,[22] where an embedding of G into T is described when the decomposition trees of Bienkowski *et al.* is used. However, the edge weights are assigned in the same way in both cases.

Now, we have to embed T into $RC(T)$. Each leaf of T is mapped to the corresponding (same) node in $RC(T)$. The internal nodes of T don't have corresponding nodes in $RC(T)$ (or G). Räcke suggests that whenever we have to simulate an algorithm on T, we map each internal node u_t of T randomly to one node u in its cluster S_{u_t} with probability $\frac{w_{l+1}(u)}{w_{l+1}(S_{u_t})}$, assuming u_t is at level l. This is sufficient for routing purposes, but we need an embedding between $RC(T)$ and G, and hence we need to preserve the connectivity between the clusters corresponding to internal nodes of T. So, instead we map each internal node u_t of T, fractionally to the nodes in its corresponding cluster S_{u_t} using the probabilities given by Räcke. This leads to a natural mapping of the edges as well. Each edge between a pair of internal nodes u_t and v_t in T is fractionally embedded in $RC(T)$ to connect all of the fractional pieces of u_t and v_t, i.e. we have a set of fractional flows between clusters S_{u_t} and S_{v_t}. Without loss of generality, assume u_t is at level $l-1$ and v_t is at level l and let $u \in S_{u_t}$ and let $v \in S_{v_t}$. The weight induced by $(u_t, v_t) \in E_T$ to the edge $(u, v) \in RC(T)$ as a result of this fractional mapping will be $\frac{w_l(u)}{w_l(S_{u_t})}\text{out}(S_{v_t})\frac{w_{l+1}(v)}{w_{l+1}(S_{v_t})}$. This exactly matches the demand between u and v in the CMCFP for S_{u_t}. Since $RC(T)$ is a union of complete graphs K_{u_t} corresponding to these demands in the CMCFPs, this embedding gives congestion 1 on $RC(T)$. This embedding also has dilation 1, because each edge in T is embedded as a fractional flow using a set of paths of length 1 between nodes in $RC(T)$.

Although, our embedding of T into $RC(T)$ is not really a traditional embedding, we can still compose the embedding of G into T and T into $RC(T)$, because the embedding of G into T does not map any nodes of G to internal nodes of T, only to the leaves. This is why we are able to split them fractionally and just follow the flow paths when we embed each edge of G into T and then each edge of T into $RC(T)$. Thus, the embedding of G into $RC(T)$ has congestion 1 and dilation $O(\log n)$. □

If we use the decomposition trees of Bienkowski *et al.* instead, we will have $c_{RC(T)\to G} = O(\log^4 n)$ instead. This is because, in this case the flows can be overlapped while inducing congestion $O(\log^3 n)$ only at each individual level. Since there are $O(\log n)$ levels of nodes, we get an additional logarithmic factor. For the dilation of embedding G into $RC(T)$, we will have an additional logarithmic factor as well, i.e. we have $d_{RC(T)\to G} = O(\alpha^{-1}(G)\Delta(G)\log^3 n)$. These differences result from the way sparsest cuts are computed in Bienkowski *et al.*'s construction. Congestion and dilation bounds in the other direction remain the same, even though the probabilities used in the fractional node mappings will be different based on the color of the node. This is because the distribution of the flow in this case depends not only a node's level but also its color. However, any edge of T can still be embedded in $RC(T)$ with congestion 1 when we assign the weights (probabilities) according to the color of the nodes. For the details, we refer the interested reader to proofs of Theorem 1 and Claim 2 in Ref. 4.

In passing, we want to mention that with an alternative definition of the dilation in the fractional embedding case, the congestion-dilation lemma (hence Lemma 2.2) will still hold. Suppose the dilation of an edge in the guest graph is defined to be the (weighted) average path length between the images of its endpoints and let *avg_dil(e)* denote the (weighted) average dilation of an edge e in the embedding.

Lemma 2.4. *In a fractional embedding P of G into H,*

$$\sigma(L(G)/L(H)) \leq \left(\max_{h\in E(H)} cong(h)\right)\left(\max_{g\in E(G)} avg\_dil(g)\right).$$

Proof. To prove this result, it is enough to examine the embedding of a single edge of G into H. As showed by Gremban, $L(G)$ and $L(H)$ can be decomposed into m positive semi-definite pieces such that $L(G) = \sum_{i=1}^{m} A_i$ and $L(H) = \sum_{i=1}^{m} B_i$, where $m = |E(G)|$.[11] Each A_i corresponds to an edge e_i of G and each B_i corresponds to a path (between vertices corresponding to the endpoints of e_i) in H in the embedding. The same decomposition can be used in the fractional setting as well with the appropriate weights applied to the entries in each B_i based on the weights of the paths. Suppose we obtained a set of values $\{\tau_1, \tau_2, \ldots, \tau_m\}$ such that each $\tau_i B_i - A_i$ is positive semi-definite. Then $\tau^* B - A$ is positive semi-definite where $\tau^* = \max_i \tau_i$.

Now, we can proceed to bound the support for the embedding of a single edge of G. Let e_j be an edge of G, let A_j and B_j be the corresponding Laplacians for edge e_j and the fractional paths it is mapped to respectively. Let $\{p_i, \forall\ 1 \leq i \leq k\}$ be the

set of paths e_j is mapped to in H, d_i be the length of p_i, and let f_i be the fraction of path p_i such that $\sum_{i=1}^{k} f_i = 1$. Note that for an edge h belonging to a path p_i, its weight in B_j will be $\frac{f_i w(e_j)}{cong(h)}$.

To bound $\sigma(A_j/B_j)$ we will use an electrical argument. Note that when a weighted graph is viewed as a resistive network, the weight of an edge corresponds to the conductance between its endpoints. Hence, the conductance between the ends of e is $w(e)$. The total conductance of the paths in B_j is at least $\sum_{i=1}^{k} \frac{w(e)f_i}{d_i\left(\max_{h\in E(B_j)} cong(h)\right)}$. As shown by Maggs *et al.* $\sigma(A_j/B_j)$ is the minimum number such that for all $\tau \geq \sigma(A_j/B_j)$, τ copies of circuit B_j consumes as much power as circuit A_j under any voltage settings on the nodes. To prove the lemma, we have to show that

$$\left(\max_{h\in E(B_j)} cong(h)\right) * avg\_dil(e) * \sum_{i=1}^{k} \frac{w(e)f_i}{d_i\left(\max_{h\in E(B_j)} cong(h)\right)} \geq w(e),$$

meaning that with $\sigma(A_j/B_j)$ copies of host B_j, the total conductance of the paths will be at least the conductance between the endpoints of e. Note that $avg\_dil(e) = \sum_{i=1}^{k} f_i d_i$. Hence, all we have to show is $\sum_{i=1}^{k} f_i d_i \sum_{i=1}^{k} \frac{f_i}{d_i} \geq 1$. We will prove this using induction. Suppose there are only two paths.

$$\begin{aligned}(f_1d_1 + f_2d_2)(f_1/d_1 + f_2/d_2) &= f_1^2 + f_2^2 + f_1f_2d_1/d_2 + f_1f_2d_2/d_1 \\ &= f_1^2 + f_2^2 + f_1f_2(d_1/d_2 + d_2/d_1)\end{aligned}$$

It is enough to show $(d_1/d_2 + d_2/d_1) = \frac{d_1^2+d_2^2}{d_1d_2} \geq 2$ because $(f_1 + f_2)^2 = 1$. This holds because $(d_1 - d_2)^2 \geq 0$.

Suppose we know $(\sum_{i=1}^{k-1} f_i d_i)(\sum_{i=1}^{k-1} \frac{f_i}{d_i}) \geq 1$ holds where $\sum_{i=1}^{k-1} f_i = 1$ and $d_i \geq 1, \forall\, 1 \leq i \leq k-1$. We want to show $\sum_{i=1}^{k} f_i d_i \sum_{i=1}^{k} \frac{f_i}{d_i} \geq 1$. Let $g_i = f_i, \forall\, 1 \leq i \leq k-2$ and let $g_{k-1} = f_{k-1} + f_k$. We have $\sum_{i=1}^{k-1} g_i = 1$. Similarly, let $d_i^{'} = d_i, \forall\, 1 \leq i \leq k-2$ and let $d_{k-1}^{'} = \frac{f_{k-1}d_{k-1}+f_kd_k}{f_{k-1}+f_k}$. We also have $d_i^{'} \geq 1, \forall\ 1 \leq i \leq k-1$. Then, by the inductive hypothesis, $(\sum_{i=1}^{k-1} g_i d_i^{'})(\sum_{i=1}^{k-1} \frac{g_i}{d_i^{'}}) \geq 1$ holds. Note that $\sum_{i=1}^{k-1} g_i d_i^{'} = \sum_{i=1}^{k} f_i d_i$. So, if we can show $\sum_{i=1}^{k-1} \frac{g_i}{d_i^{'}} \leq \sum_{i=1}^{k} \frac{f_i}{d_i}$ we are done. The last inequality is equivalent to $\frac{g_{k-1}}{d_{k-1}^{'}} \leq \frac{f_{k-1}}{d_{k-1}} + \frac{f_k}{d_k}$. Making the substitutions we get

$$\frac{(f_{k-1} + f_k)^2}{f_{k-1}d_{k-1} + f_kd_k} \leq \frac{f_{k-1}d_k + f_kd_{k-1}}{d_kd_{k-1}},$$

and, after some manipulation we obtain

$$0 \leq d_{k-1}^2 + d_k^2 - 2d_{k-1}d_k.$$

The last inequality holds because the right hand side is equal to $(d_{k-1} - d_k)^2$. This concludes the proof, since

$$\tau^* = \max_j \left(\text{avg\_dil}(e_j) \max_{h \in B_j} \text{cong}(h) \right) \leq \left(\max_{h \in E(H)} \text{cong}(h) \right) \left(\max_{g \in E(G)} \text{avg\_dil}(g) \right). \quad \square$$

Lemma 2.4 implies that a tighter bound can be obtained for the generalized condition number in an application of Lemma 2.2 with the alternative definition of dilation. However, we must stress that while enhancing preconditioned iterative solvers based on support trees is our original motivation for studying mutual embeddings, recent progress in the area[25,31] suggests that mutual embeddings should be best seen as the beginning of an independent intellectual development instead of a pursuit in accelerating iterative solvers. In the rest of the paper, we make the simplifying assumption that the two edge mappings return only singleton sets, i.e., each edge in one graph is always mapped to a single path in the other.

3. Mutual Embeddings Between Linear Arrays And Arbitrary Graphs

Graph embedding problems using linear arrays were considered by many researchers before. Sekanina showed how to embed linear arrays into trees (hence to any connected graph) with dilation 2.[29] In the bandwidth minimization problem (which is a graph layout problem, where dilation is called bandwidth), the vertices of a graph are laid out in a line and the aim is to find the layout with minimal bandwidth. This problem is NP-complete for general graphs[23] and for some other simpler graphs such as trees with maximum degree 3.[8]

While introducing mutual embeddings in Section 1, we presented several example mutual embeddings between linear arrays and cycles. In all the examples, we had $d_1 d_2 = \Omega(n)$. We next show that this is true for any mutual embedding between a linear array and a cycle.

Theorem 3.1. Let $G_1 = C$ be a cycle and let $G_2 = L$ be a linear array. In any mutual embedding between L and C, $d_1 d_2 = \Omega(n)$.

Proof. Let the nodes of the linear array be numbered from 1 to n consecutively from left to right. Let's also color node 1 blue and node n red. Again w.l.o.g. let's assume the cycle's node 1 is mapped to the linear array's node 1.

Let's start following the cycle edges in one direction from node 1 until we hit the line's node n. In the process, we color each intermediate node and each cycle edge blue. After that, we finish tracing the cycle edges from the line's node n back to node 1. In the process, we color each intermediate node and each cycle edge red. The lengths of both the blue cycle path and the red cycle path are at least one.

Now, let's examine the middle one third of the line. The nodes in this section consist of either nodes of one color, or both blue and red nodes.

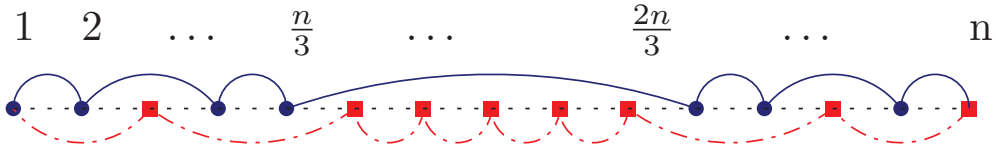

Fig. 3. Case 1 : All nodes in the middle have the same color. (Color online.)

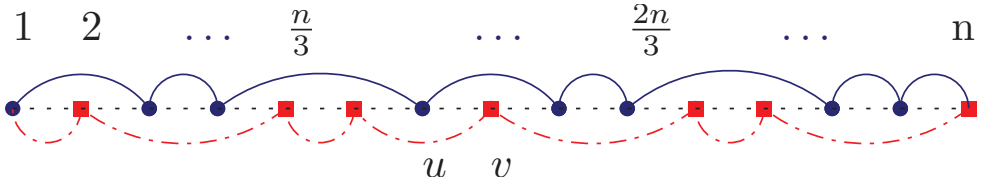

Fig. 4. Case 2 : The middle one third of the linear array have both blue (disks) and red nodes (squares). (Color online.)

If there is just one color in this region, then we are done because one of the cycle edges in the other color must span over this section, which has a line-distance of $n/3$, implying $d_1 = \Omega(n)$. Figure 3 shows an example illustration of this case.

If we have both red and blue nodes in this region, then this section must contain at least one linear array edge (u, v) such that u and v have different colors. In the cycle, there are two paths between nodes corresponding to u and v. W.l.o.g. let us assume u is colored blue, v is colored *red*, and (u, v) is mapped to the path which includes linear array's node n. This path starts from u, hops towards the line's node n through blue cycle edges, and then hops back to v through red cycle edges. Figure 4 shows an example of this case.

Length of this cycle path is a lower bound on the dilation from linear array to cycle. If we use $d_C(u, v)$ to denote the length of this path, then we have $d_C(u, v) \leq d_2$. Moreover, the average distance that is hopped in the line on this cycle path is a lower bound on the dilation of cycle edges. Since the the total hopping distance covered is at least $n/3$, we have $d_1 \geq \frac{n/3}{d_C(u,v)} \geq \frac{n/3}{d_2}$. Therefore, we have $d_1 d_2 \geq \frac{n}{3} = \Omega(n)$. □

Even though we have many examples with same or similar lower bounds on the product of dilations for mutual embeddings between different types of graphs, we find this one particularly interesting. The linear array and cycle are very similar to each other, yet the addition of one more edge to the linear array and completing the cycle results in an $\Omega(n)$ bound.

Next, we examine mutual embeddings between complete binary trees and linear arrays and show that a similar result holds. Embedding complete binary trees into lines was first considered by Paterson *et al.*,[24] however the presented embeddings required constant expansion (increased number of nodes in the host graph). Later, Heckmann *et al.* showed how to embed complete binary trees into lines and grids with optimal dilation.[13] Below, we show that for any mutual embedding between a complete binary tree and a linear array, the the product of dilations is $\Omega(n)$.

Then, we will generalize this idea for mutual embeddings between linear arrays and arbitrary graphs.

Theorem 3.2. Let $G_1 = T = (V_T, E_T)$ be a complete binary tree, let $G_2 = L = (V_L, E_L)$ be a linear array such that $|V_T| = |V_L| = n$. In any mutual embedding of T and L, $d_1 d_2 = \Omega(n)$.

Proof. Let $\pi : T \to L$ be an arbitrary mutual embedding between T and L. Label the nodes in the linear array from left to right as $1, \ldots, n$. Let r be the root of the complete binary tree and let T_s be a subtree rooted at a grand-child of r which contains neither $\pi^{-1}(1)$ nor $\pi^{-1}(n)$. Note that there exists at least one such node among the four grand-children of r and $|T_s| = \Omega(n)$. Let $p, q \in T_s$ be the nodes such that $\pi(p)$ and $\pi(q)$ is minimum and maximum among all nodes of T_s respectively. Then $\pi^{-1}(\pi(p) - 1) \notin T_s$ and $\pi^{-1}(\pi(q) + 1) \notin T_s$.

Since any path on the tree from a node outside T_s to a node inside T_s must pass through s, we have $d_T(p, s) \leq d_T(p, \pi^{-1}(\pi(p) - 1))$, where $d_T(\bullet, \bullet)$ is the distance on the tree. Hence, $d_T(p, s) \leq d_2$ and by a similar argument for q we get $d_T(q, s) \leq d_2$. Combining these two inequalities, we get $d_T(p, q) \leq 2d_2$. On the other hand, we can relate d_1 with $d_T(p, q)$. If $d_T(p, q) = 1$, then obviously $d_1 \geq |T_s|$. If $d_T(p, q) > 1$, there is a set of nodes S on the paths between p and q, we can minimize d_1 by equally spacing the counterparts of the nodes in S in the linear array between $\pi(p)$ and $\pi(q)$. Hence, we have $d_1 \geq |T_s|/d_T(p, q)$, since π mapped every node in T_s to the linear array and it is 1-1. Therefore $d_1 d_2 \geq |T_s|/2 = \Omega(n)$. □

The above proof can be generalized to prove lower bounds on product of dilations for mutual embeddings between a linear array and an arbitrary graph G. The key is to find a substructure of G, which the linear array must enter and leave at least once. Suppose the distance between the entry point and the exit point is b, the size of the substructure is s, then the product of the dilations is at least $\frac{s}{b}$. In most cases, we want to select a substructure with $\Omega(n)$ nodes. Below, we formalize this idea into a theorem for the general case after defining the boundary of a subgraph and the diameter of the boundary.

Definition 3.1. The boundary of a subgraph H of a graph G, denoted $\partial_G(H)$, is the set of vertices in H, which has at least one neighboring vertex outside H. More precisely, $\partial_G(H) = \{u : u \in H \subset G \text{ — } \exists v \notin H \text{ s.t. } (u, v) \in E(G)\}$.

Definition 3.2. The diameter of the boundary $\partial_G(H)$ is the maximum distance in G between any two vertices of $\partial_G(H)$ plus 1, i.e., $D(\partial_G(H)) = \max_{u,v \in \partial_G(H)} d_G(u, v) + 1$, where $D(\partial_G(H))$ denotes the diameter of the boundary, and, $d_G(\bullet, \bullet)$ is the distance between two nodes in G.

Theorem 3.3. Let $G_1 = L = (V_L, E_L)$ be a linear array and $G_2 = G = (V_G, E_G)$ be an arbitrary connected graph such that $|V_L| = |V_G| = n$. If there exists disjoint

subgraphs $H_i \subset G, i = 1, 2, 3$ such that (i) $|H_i| \geq \frac{n}{c}$, where c is a fixed constant, and (ii) $D(\partial_G(H_i)) \leq b$, then $d_1 d_2 = \Omega(\frac{n}{b})$.

Proof. The proof is very similar to the proof of Theorem 3.2. First we label the nodes of L from one end to the other as $1, 2, \ldots, n$. Consider the embedding of L into G, let σ denote the vertex mapping. There exists an H_i such that it contains neither $\sigma(1)$ nor $\sigma(n)$. Let $p, q \in H_i$ such that $\sigma^{-1}(p)$ and $\sigma^{-1}(q)$ have the smallest and the largest labels in the linear array respectively. There exists a node $r \in \partial_G(H_i)$ (respectively $s \in \partial_G(H_i)$) such that the dilation d_1 of embedding the linear array into G satisfies $d_1 \geq d_G(p, r)$ (respectively $d_1 \geq d_G(q, s)$). Hence, $d_G(p, q) \leq 2d_1 + d_G(r, s) \leq 2d_1 + b$ and consequently $d_G(p, q) \leq 2d_1 + d_G(r, s) \leq 2d_1 b$. This holds even when $b = 1$, because in this case the boundary contains just a single node, i.e., $r = s$ and $d_G(r, s) = 0$. On the other hand H_i contains $\frac{n}{c}$ nodes, which implies that the dilation d_2 of embedding G into the linear array satisfies $d_2 \geq |H_i|/d_G(p, q) \geq \frac{n}{c}/d_G(p, q) \geq \frac{n}{c}/(2d_1 b)$. Hence, we have $d_1 d_2 \geq \frac{n}{2cb} = \Omega(\frac{n}{b})$. □

Next we show some example applications of Theorem 3.3. The key to applying the theorem is finding the subgraphs H_i.

Example 3.1. Mutual embedding between a linear array and a complete binary tree.

This example serves as a sanity check, as we know $d_1 d_2 = \Omega(n)$ from Theorem 3.2. Each subtree in the proof of Theorem 3.2 has $\frac{n}{4}$ nodes and the diameter of the boundary is 1. Hence the $\Omega(n)$ lower bound follows.

This bound is tight. As we mentioned before, Heckmann *et al.* showed how to embed complete binary trees into lines with optimal dilation.[13] Their algorithm embeds the left subtree of the root into h blocks of size $l = \lceil (2^h - 1)/h \rceil$, where h is the height of the tree. If there is an edge between two nodes, they are placed either in the same block or in two adjacent blocks. The right subtree of the root is embedded by mirroring the left, and the root is placed in-between. The dilation of this embedding is $\Theta(n/\log n)$. In the embedding of the linear array into the complete binary tree, each edge is mapped into the unique path in the tree. Since the height of the tree is $h = \log n$, the product of dilations is $\Omega(n)$.

Example 3.2. Mutual embedding between a linear array and a two-dimensional square mesh.

Consider a $\sqrt{n} \times \sqrt{n}$ mesh M. Each node of M can be identified by an ordered pair (x, y), where $1 \leq x, y \leq \sqrt{n}$. Define $H_i, i = 1, 2, 3$ such that each H_i is a rectangular slice of M containing one third of the nodes. More precisely, each H_i contains nodes (x, y), where $\frac{(i-1)}{3}\sqrt{n} + 1 \leq x \leq \frac{i}{3}\sqrt{n}$ and $1 \leq y \leq \sqrt{n}$. Then $D(\partial_G(H_i)) \leq 2\sqrt{n}$ for each H_i and hence we have the lower bound of the mutual embedding between a linear array and a mesh as $\Omega(\frac{n}{2\sqrt{n}}) = \Omega(\sqrt{n})$.

The $\sqrt{n}$ lower bound is tight (within a constant factor). We can embed the linear array into the mesh as follows. First embed node 1 in the linear array into the bottom-left corner of the mesh and then embed the consecutive $\sqrt{n}-1$ nodes along the bottom of the mesh. This way node $\sqrt{n}$ is embedded into the bottom-right corner. After that, embed node $\sqrt{n}+1$ to the right-most node in the second layer of the mesh, then embed the consecutive $\sqrt{n}-1$ nodes from right back to the left along second layer and so on. The embedding from the linear array to mesh has dilation 1, while the embedding from the mesh to the linear array has dilation $2\sqrt{n}$. Hence, this mutual embedding reaches the lower bound $\Omega(\sqrt{n})$.

The same technique can be used to obtain optimal mutual embeddings between linear arrays and two-dimensional meshes of any aspect ratio. We will again use the snake pattern to embed the linear array into the mesh. Starting from a corner of the mesh, we can embed the nodes of the linear array consecutively along the shorter side of the mesh and following the same pattern described above afterwards. For example, if our two-dimensional mesh is $n^{1/3} \times n^{2/3}$ then we have $c_1 = d_1 = 1$ and $c_2 = d_2 = \Omega(n^{1/3})$.

Note that, there can be cases where Theorem 3.3 results in a trivial lower bound. For example, we know that $d_1 d_2 = \Omega(n)$ for linear arrays and cycles. However, the only way to find disjoint subgraphs in the cycle is defining $n/3$ consecutive vertices as a subgraph H_i, and this results in a trivial lower bound 1.

We conclude this section with a small example showing a trade-off between congestion in one direction and dilation in the other direction for mutual embeddings between linear arrays and cycles.

Claim 1. Let G_1 be a cycle and G_2 be a linear array. In any mutual embedding of G_1 and G_2 in which adjacent nodes of the linear array are not mapped to adjacent nodes of the cycle has congestion at least 2, i.e. does not have optimal congestion.

Proof. Suppose we started drawing the linear array on the cycle from left to right, i.e. mapping each edge of the linear array to to paths (edges) in the cycle. Let u and v be two vertices of the linear array which are mapped to non-adjacent vertices in the cycle. When (u, v) is mapped to a path, the node right after $\pi(u)$ will be skipped in the cycle. But at some point the mapping needs to visit this node, either in the clockwise or in the counter-clockwise direction. In either case, a cycle edge has to be used again, causing congestion at least 2. □

Corollary 3.1. *If $c_2 < 2$, then $d_1 = n - 1$.*

Proof. The only possible way for c_2 to be less than 2 is for every pair of adjacent nodes in the linear array to be embedded in adjacent nodes in the cycle. But this implies that two adjacent nodes of the cycle must be mapped to the two opposite ends of the linear array. □

4. Mutual Embeddings Between Arbitrary Graphs

In Section 3 we studied the problem of finding mutual embeddings between linear arrays and arbitrary graphs. Using linear arrays as a bridge, we will introduce a technique to find mutual embeddings between arbitrary graphs in this section. We name the technique tri-embedding as our technique studies the embedding among three graphs: one source, one target, and, one relay graph.

Let G_1 and G_2 be two arbitrary graphs and let L be a linear array. From Theorem 1.1, we know that a mutual embedding $\sigma : G_1 \to L$ and a mutual embedding $\tau : L \to G_2$ will give us a mutual embedding $\pi = \tau \circ \sigma : G_1 \to G_2$. In fact, fixing σ still allows us to find a τ for each arbitrary π. We also know that there is always a "good" mutual embedding between a linear array L and an arbitrary graph G_1 such that $d_{L\to G_1} \leq 2$.[29] We name such a mutual embedding between a graph and a linear array a linear-embedding.

Definition 4.1. Let G be an arbitrary graph and L be a linear array. A mutual embedding π between G and L that satisfies $d_{L\to G} \leq 2$ is called a *linear-embedding* of G.

In the following, we show how this embedding facilitates us to connect two arbitrary graphs by a linear array. The main idea follows the proof of Theorem 3.3. First we find a linear embedding σ of G_1 (with L) and then we draw L on G_2 (i.e. find $\tau : L \to G_2$), obtaining a mutual embedding π of G_1 and G_2. Similar to the proof of Theorem 3.3, we find subgraphs $H_i \subset G_2, i = 1, 2, 3$ with a proper boundary diameter and $\Omega(n)$ vertices, such that L enters and leaves one subgraph H_i at least once. W.l.o.g. assume H_1 has a boundary that contains neither $\tau(1)$ nor $\tau(n)$, and suppose p and q are the entry and exit points respectively. To apply the idea in the proof of Theorem 3.3, we need to estimate the distance between $\pi^{-1}(p)$ and $\pi^{-1}(q)$ on G_1, which is determined by the mutual embedding of G_1 and L. Although the distance between $\tau^{-1}(p)$ and $\tau^{-1}(q)$ on L is lower bounded by the number of nodes in H, which is $\Omega(n)$, the distance of $\pi^{-1}(p)$ and $\pi^{-1}(q)$ can be fairly small on G_1. Next, we define the *k-th distortion* of a linear-embedding which will be used to estimate this distance.

Definition 4.2. Let $L = (V_L, E_L)$ be a linear array, $G = (V_G, E_G)$ be an arbitrary graph and ϕ be a linear-embedding of G, where $|V_L| = |V_G| = n$. The k-th distortion, $\lambda_k(\phi)$, of ϕ is defined as $\lambda_k(\phi) = \min_{k \leq d_L(\phi(u),\phi(v)) \leq \frac{n}{3}} d_G(u, v)$.

It is apparent that the $|H|$-th distortion of the linear-embedding of G_1 is a lower bound on the distance of $\pi^{-1}(p)$ and $\pi^{-1}(q)$ on G_1. The following theorem formalizes the above ideas.

Theorem 4.1. Let G_1 and G_2 be two arbitrary graphs with n nodes. If there exist disjoint subgraphs $H_i \subset G_2, i = 1, 2, 3$ such that

(i) $\frac{n}{c} \leq |H_i| \leq \frac{n}{3}$, where c is a fixed constant,
(ii) $D(\partial_{G_2}(H_i)) \leq b$, and,
(iii) there exists a linear-embedding σ of G_1, such that $\lambda_{\frac{n}{c}}(\sigma) \geq \gamma$, then $d_{G_1 \to G_2} d_{G_2 \to G_1} = \Omega(\frac{\gamma}{b})$.

Proof. Let L be a linear array where the nodes are numbered from 1 to n and let σ be a linear-embedding of G_1 (with L). Any mutual embedding $\tau : L \to G_2$ results in a mutual embedding $\pi = \tau \circ \sigma : G_1 \to G_2$. At least one of $H_i \in G_2, i = 1, 2, 3$ contains neither $\tau(1)$ and $\tau(n)$. W.l.o.g. let's assume H_1 is that subgraph and $\tau(p)$ and $\tau(q)$ are the entry and exit points of L in and out of H_1.

First, we get an upper bound on the distance $d_{G_2}(\tau(p), \tau(q))$ and relate it with $d_{G_1 \to G_2}$. Consider the node $(p-1) \in L$, we have $\tau(p-1) \notin H_1$. Moreover,

$$d_{G_1}(\sigma^{-1}(p-1), \sigma^{-1}(p)) \leq 2$$

since σ is a linear-embedding. Therefore,

$$d_{G_1 \to G_2} \geq \frac{d_{G_2}(\tau(p-1), \tau(p))}{d_{G_1}(\sigma^{-1}(p-1), \sigma^{-1}(p))} \geq \frac{1}{2} d_{G_2}(\tau(p), \tau(p-1)).$$

Consider the shortest path from $\tau(p-1)$ to $\tau(p)$ on G_2, it has to pass from the boundary of H_1. So we can find a vertex $r \in \partial_{G_2}(H_1)$ such that

$$d_{G_2}(\tau(p), r) \leq d_{G_2}(\tau(p), \tau(p-1)) \leq 2 d_{G_1 \to G_2}.$$

Reasoning similarly for $\tau(q)$ we get

$$d_{G_2}(\tau(q), s) \leq 2 d_{G_1 \to G_2},$$

where $s \in \partial_{G_2}(H_1)$ and the shortest path from $\tau(q)$ and $\tau(q+1)$ passes from s. Combining the above two inequalities, and adding $d_{G_2}(r, s)$ to both sides, we get

$$\begin{aligned} d_{G_2}(\tau(p), r) + d_{G_2}(\tau(q), s) + d_{G_2}(r, s) &\leq 4 d_{G_1 \to G_2} + d_{G_2}(r, s) \\ d_{G_2}(\tau(p), \tau(q)) &\leq 4 d_{G_1 \to G_2} + b \\ d_{G_2}(\tau(p), \tau(q)) &\leq 4 b d_{G_1 \to G_2}. \end{aligned}$$

The last inequality holds even when $b = 1$ because then $r = s$ and $d_{G_2}(r, s) = 0$.

Second, we obtain a lower bound on the distance of $\sigma^{-1}(p)$ and $\sigma^{-1}(q)$ on G_1. We know that $d_L(p, q) \geq \frac{n}{c}$ since each node of H_1 is mapped to a node of L between p and q. With condition (iii) we have

$$d_{G_1}(\sigma^{-1}(p), \sigma^{-1}(q)) \geq \lambda_{\frac{n}{c}}(\sigma) \geq \gamma.$$

Combining the above two bounds, we get

$$d_{G_2 \to G_1} \geq \frac{d_{G_1}(\sigma^{-1}(p), \sigma^{-1}(q))}{d_{G_2}(\tau(p), \tau(q))} \geq \frac{\gamma}{4 b d_{G_1 \to G_2}}.$$

Hence, $d_{G_1 \to G_2} . d_{G_2 \to G_1} \geq \frac{\gamma}{4b} = \Omega(\frac{\gamma}{b})$. □

Next, we present some examples to illustrate the power of the tri-embedding technique introduced above.

Example 4.1. Mutual embedding between a two-dimensional square mesh and a complete binary tree.

Let G_1 be a mesh and G_2 be a complete binary tree. The linear-embedding of the mesh we adopt here is the one we introduced in Example 3.2, that is we embed the line following from a corner of the mesh and follow a snake pattern. For the complete binary tree we use as H_i the subtrees rooted at the four grandchildren of the root as we did in Example 3.1. Hence for each H_i, we have $|H_i| \geq \frac{n}{4}$ and $D(\partial_{G_2}(H_i)) \leq 1$. We also have $\lambda_{\frac{n}{4}} \geq \frac{\sqrt{n}}{4}$. Applying Theorem 4.1, we have the lower bound $\Omega(\sqrt{n})$.

There is a mutual embedding reaching this lower bound. As we mentioned in Example 3, Heckmann *et al.* showed how to embed complete binary trees into lines and grids (two-dimensional meshes) with optimal dilation.[13] Embedding of the tree into the grid uses the algorithm for embedding into the line as a subroutine. Depending on the height of the tree being even or odd, subtrees of varying sizes are embedded into rows and columns of the mesh. For details of this embedding the interested reader may consult.[13] The dilation of this embedding is $\Theta(\sqrt{n}/\log n)$. Again, the edges of the grid are mapped to the unique paths in the tree, and since the maximum path length is $O(\log n)$, the product of dilations is $\Theta(\sqrt{n})$.

Example 4.2. Mutual embedding of a two-dimensional square mesh and a cycle.

Let $G_1 = C$ denote the cycle and let $G_2 = M$ denote the mesh. We use the following linear-embedding of the cycle C: first embed a vertex in C into the left end of the linear array L, and then embed consecutively all the vertices in C from left to right in L. Clearly, we have $\lambda_{\frac{n}{3}} \geq \frac{n}{3}$ in the cycle. In the mesh M, we find the same subgraphs H_i as we did in Example 3.2. Hence we have $|H_i| \geq \frac{n}{3}$ and $D(\partial_M(H_i)) \leq 2\sqrt{n}$. Applying Theorem 4.1, we have the lower bound $\Omega(\sqrt{n})$. So, in this case the theorem does not give us any new information since $D(M) = 2\sqrt{n}$ and $D(C) = n/2$, and we know that $d_2 \geq \frac{D(C)}{D(M)} = \frac{\sqrt{n}}{4} = \Omega(\sqrt{n})$ from Lemma 1.2.

It is not clear to us whether such a mutual embedding satisfying $d_1 d_2 = O(\sqrt{n})$ exists. We can easily embed the cycle into the mesh with dilation $O(1)$, but in all such examples we observed that $d_2 = \Omega(n)$. Figure 5 shows two example mutual embeddings where $d_1 = O(1)$ and $d_2 = \Omega(n)$. In both mutual embeddings, the cycle is embedded into the mesh using a snake pattern, by tracing cycle edges consecutively. We have $d_1 = 2$ in the first example and $d_1 = 1$ in the second example. However, in both examples there is at least one mesh edge (shown inside an oval) which has to travel roughly half of the cycle edges, resulting in $d_2 = \Omega(n)$.

It is not difficult to find an embedding where $d_2 = O(\sqrt{n})$ though. For example, we can use the mutual embedding presented in Example 3.2 for linear arrays and meshes, with only one change to account for the extra edge in the cycle. The endpoints of the extra cycle edge are mapped to opposite corners of the mesh in this

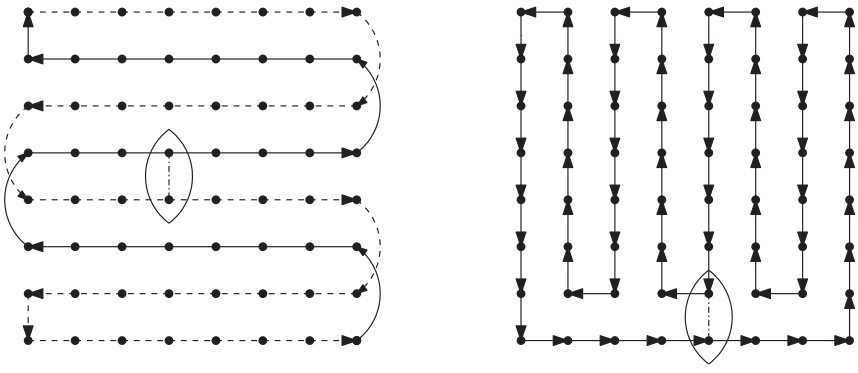

Fig. 5. Two example mutual embeddings between a two-dimensional mesh and a cycle.

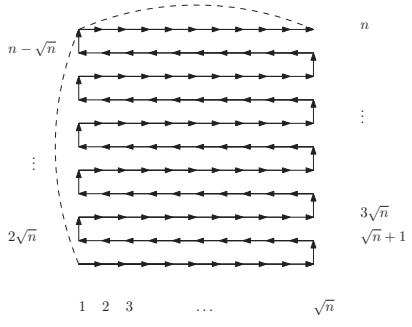

Fig. 6. A mutual embedding between a two-dimensional mesh and a cycle.

case, resulting in $d_1 = 2\sqrt{n}$. As in Example 3.2, we have $d_2 = 2\sqrt{n}$. Figure 6 shows this mutual embedding, more precisely, how the cycle edges are mapped to mesh edges. The mesh edges are mapped to the shortest paths in the cycle between the nodes corresponding to the endpoints of mesh edges. Cycle's node 1 and node n are connected using the edges in the top row and leftmost column (the dashed arcs), causing dilation $2\sqrt{n}$. In Example 3.2, we had $d_1 d_2 = \Omega(\sqrt{n})$ for the linear array and mesh. Yet, in this example too (at least when we use the same mutual embedding), addition of one more edge and completing the cycle leads to an asymptotic difference in the lower bound.

We conjecture that $d_1 d_2 = \Omega(n)$ for the mesh and cycle, similar to the linear array and cycle case. The examples in Figure 5 suggest a way of proving this result, as the dilation of embedding the mesh to the cycle seems to be $\Omega(n)$ whenever the cycle is embedded in the mesh with constant dilation. The investigation of this claim either proves the conjecture or provides a counter-example. The cycle has to be embedded into the mesh with constant dilation to reach the $d_1 d_2 = \Omega(\sqrt{n})$ bound anyway (if our conjecture is false and it is possible to reach this lower bound) since $d_2 = \Omega(\sqrt{n})$ from Lemma 1.2.

Example 4.3. Mutual embedding between a ladder and a cycle.

A ladder is a bi-partite graph defined on $V = (X_L, X_R)$, where $|X_L| = |X_R| = \frac{n}{2}$. All the edges have the form $(x_L, x_R), (x_L, x_{R+1})$ or (x_{L+1}, x_R). The diameter of the ladder is $\frac{n}{2}$, exactly the same with a cycle. So a simple comparison of the diameters gives a trivial lower bound 1.

Let $G_1 = C$ denote the cycle and let $G_2 = LD$ denote the ladder. For the ladder, we define $H_i, i = 1, 2, 3$ as follows. Each H_i contains vertices $\{x_L, x_R : \frac{i-1}{6}n + 1 \leq x_L, x_R \leq \frac{i}{6}n\}$. Therefore, we have $|H_i| \geq \frac{n}{3}$ and $D(\partial(H_i)) \leq 2$. We also have $\lambda_{\frac{n}{3}} \geq \frac{n}{3}$ in the cycle. Applying Theorem 4.1, we have the lower bound $\Omega(n)$.

The $\Omega(n)$ lower bound is tight as it is reached by the following mutual embedding. First we embed 1_L into a vertex in the cycle, and then we embed $2_R, 3_L, 4_R, ...$ into the cycle consecutively. We also embed 1_R into the vertex in the cycle which is the other neighbor of 1_L, and then embed $2_R, 3_L, 4_R, ...$ into the other side of the cycle consecutively. In this way, $d_{C \to LD} = 1$, and $d_{LD \to C} = \frac{n}{2}$ as the edge $(\frac{n}{4}{}_L, \frac{n}{4}{}_R)$ is in the ladder, and the distance of $\frac{n}{4}{}_L$ and $\frac{n}{4}{}_R$ in the cycle is $\frac{n}{2}$.

5. Relationship with Distortion in Metric Space Embeddings

Most of the results we presented so far are about products of dilations in mutual embeddings. On the surface, this quantity looks similar to the notion of distortion in metric space embeddings. There, the aim is to find a low-distortion embedding between two metric spaces (X, D) and (Y, D') where X and Y are sets of points and D and D' are distance functions. The expansion (or stretch) $e(f)$ of an embedding $f : X \to Y$ is defined as

$$e(f) = \max_{x,y \in X, x \neq y} \frac{D'(f(x), f(y))}{D(x, y)}.$$

Similarly, the contraction $c(f)$ of f is defined as

$$c(f) = \max_{x,y \in X, x \neq y} \frac{D(x, y)}{D'(f(x), f(y))}.$$

The distortion of the embedding is defined as the product of expansion and contraction, and it measures how much the distances change in the mapped space. Scaling the distance functions do not change the distortion and the main interest is to find a low-distortion embedding f with the non-contracting constraint, i.e., $D'(f(x), f(y)) \geq D(x, y)\ \forall x, y \in X$.

The product of dilations in mutual embeddings seems similar to the distortion of a metric space embedding, but, there are two key differences:

- We are only interested in the distances of images of adjacent vertices in the graphs, not between the distances between the images of any two vertices of the graph.
- There is no non-contracting constraint.

These differences may result in different asymptotic bounds for the same embedding problem. For example, in Theorem 3.2 and Example 3.1 we showed that $d_1 d_2 =$

$\Omega(n)$ for any mutual embedding between a complete binary tree and a linear array. Following Example 3.1, we presented a mutual embedding reaching this lower bound, using the embedding of complete binary trees into lines given by Heckmann *et al.*[13]

For the same graphs, the distortion lower bound is different in the metric space embedding case. Kumamoto and Miyano showed that the optimal distortion of embedding a complete binary tree into a line is $\Theta(n/\log n)$.[18] Their proof is also based on the proof of Heckmann *et al.* but because of the non-contracting constraint, the vertices of the tree are not simply mapped to numbers 1 to n in the real line, they are stretched further apart to meet the non-contracting constraint. Hence the expansion is not bounded by $\lceil n/\log n \rceil$ as the dilation in the proof of Heckmann *et al.* but bounded by a constant factor times $\lceil n/\log n \rceil$. In contrast, the contraction is bounded by 1 in the metric space embedding whereas the dilation of the mutual embedding in the reverse direction is $\Theta(\log n)$.

6. Mutual Embeddings Between Meshes

In this section we study mutual embeddings between meshes of different dimensions. The problem of finding embeddings between meshes was studied by many researchers.[1,17,28,30] Aleliunas and Rosenberg showed how to embed a rectangular grid into a square grid for various values of expansion and dilation, emphasizing the trade-off between the two.[1] Kosaraju and Atallah showed how to simulate one mesh-connected processor array of arbitrary dimension with another with the help of graph embeddings.[17] The load in their embeddings is not 1, however, they proved an asymptotic lower bound on the dilation of any embedding. Shen and Liang *et al.* studied embeddings between two-dimensional meshes of the same size, and obtained bounds on both congestion and dilation.[30] The embeddings in this section have been studied before by the mentioned authors and others, but we analyze them as mutual embeddings. Even though some of the above mentioned works deal with embeddings between graphs of different number of vertices, they include useful ideas such as folding and the snake pattern.

In Example 3.2 (Section 3), we presented an optimal mutual embedding between a linear array and a two-dimensional square mesh. This construction easily generalizes to mutual embeddings between linear arrays and higher dimensional meshes. As an example, we show how to obtain a mutual embedding between a linear array and a three-dimensional cubic mesh. Note that Rosenberg and Snyder showed that a d-dimensional mesh can be embedded in a linear array with dilation $\Omega(n^{1-1/d})$[27] and Kosaraju and Atallah's lower bound is also a generalization of this specific result.[17]

Theorem 6.1. There is an optimal mutual embedding between a linear array (G_1) and a three-dimensional mesh (G_2) where $c_1 = 1$, $d_1 = 1$, $c_2 = n^{2/3}$, and, $d_2 = \Theta(n^{2/3})$.

Proof. We can embed the linear array into the mesh with dilation and congestion 1 because the linear array is a subgraph of the three-dimensional mesh. The linear

array first snakes through one $n^{1/3} \times n^{1/3}$ level of the three-dimensional mesh, then pops up to the next level, follows another snake pattern on that level and so on. Bounds on c_2 and d_2 follow because the three-dimensional mesh edges on a single level require congestion and dilation $n^{1/3}$, but the $n^{2/3}$ edges between two levels all have the cross the same edge of the linear array. These edges cause congestion $n^{2/3}$ and this is optimal since $\frac{BW(\text{3-d mesh})}{BW(\text{linear array})} = n^{2/3}$. The dilation is also at least $n^{2/3}$ –it is at most $2n^{2/3}$– because after the first node is visited at one level, the node at the same position at the upper level can only be visited after we snake through all the vertices in that level. The value of d_2 is optimal (up to a constant factor), since $\frac{D(\text{linear array})}{D(\text{3-d mesh})} = n^{2/3}$. □

It is easy to generalize Theorem 6.1 to other three-dimensional meshes besides cubes.

Theorem 6.2. Suppose M is a three-dimensional mesh with dimensions $n_1 \leq n_2 \leq n_3$. There is an optimal mutual embedding between a linear array L and M, where $c_{L\to M} = 1$, $d_{L\to M} = 1$, $c_{M\to L} = n_1 n_2$, and, $d_{M\to L} = \Theta(n_1 n_2)$.

Proof. The proof is almost identical to the proof of Theorem 6.1, except that we have to avoid the longest dimension in the snake pattern. The linear array first snakes through one $n_1 \times n_2$ level of the three-dimensional mesh, then pops up to the next level, follows another snake pattern on that level and so on. We have $d_{L\to M} = 1$ and $c_{L\to M} = 1$ since L is a subgraph of M. The $n_1 n_2$ edges between two levels all have the cross the same edge of the linear array. These edges cause congestion $n_1 n_2$ and dilation $\Theta(n_1 n_2)$. □

Finding mutual embeddings between a square mesh and a cubic mesh is more interesting. Let G_1 be a $\sqrt{n} \times \sqrt{n}$ two-dimensional mesh and let G_2 be a $n^{1/3} \times n^{1/3} \times n^{1/3}$ three-dimensional mesh.

Applying Theorem 4.1 we get $d_1 d_2 = \Omega(n^{1/6})$. To see this let H_i be a $n^{1/3} \times n^{1/3} \times \frac{n^{1/3}}{3}$ sub-mesh of G_2 for $i = 1, 2, 3$, s.t. $H_1 \cup H_2 \cup H_3 = G_2$. Then $D(\partial_{G_2}(H_i)) \leq 3n^{1/3}, \forall\ 1 \leq i \leq 3$. For the linear-embedding σ of G_1, let us choose the embedding given in Example 3.2, where the linear array snakes through the square mesh. For this linear-embedding, we have $\lambda_{\frac{n}{3}}(\sigma) \geq 2\sqrt{n}$. Therefore Theorem 4.1 yields $d_1 d_2 = \Omega(\frac{2\sqrt{n}}{3n^{1/3}}) = \Omega(n^{1/6})$. So the theorem does not provide any new information for this mutual embedding; since we have $d_2 \geq \frac{D(\text{2-d mesh})}{D(\text{3-d mesh})} = \frac{\sqrt{n}}{n^{1/3}} = n^{1/6}$ due to Lemma 1.2. For the congestion, we have $c_2 \geq \frac{BW(\text{3-d mesh})}{BW(\text{2-d mesh})} = \frac{n^{2/3}}{\sqrt{n}} = n^{1/6}$.

There are several different embeddings of the three-dimensional mesh into the two-dimensional mesh with congestion $n^{1/6}$ and dilation $\Omega(n^{1/6})$, which is optimal up to a constant factor. For example, suppose that we take each vertical column in the three-dimensional mesh ($n^{1/3}$ nodes), and collapse it down to an $n^{1/6} \times n^{1/6}$ square on the two-dimensional mesh, with the nodes in the column embedded in a

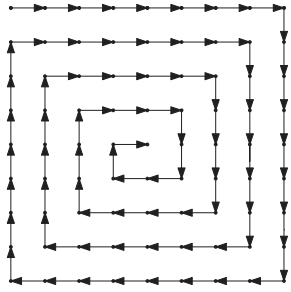

Fig. 7. Embedding of each column of a three-dimensional mesh into a square.

spiral pattern as shown in Figure 7. Each edge of the column is mapped to a single edge inside the corresponding square as shown in the spiral pattern. In fact, this is just another example of embedding a linear array into a mesh.

These squares tile the two-dimensional mesh. Each edge (u, v) between two nodes u and v residing in adjacent columns is mapped to the shortest path between $\pi(u)$ and $\pi(v)$ in the two-dimensional mesh, where $\pi = V_{G_2} \rightarrow V_{G_1}$. Note that $\pi(u)$ and $\pi(v)$ will be in adjacent squares. There are $n^{1/3}$ rows and $n^{1/3}$ columns of squares. A column of the three-dimensional mesh with indices (i, j) will be mapped to the square (i, j) in the two-dimensional mesh. Any two three-dimensional mesh edges are within distance $\Omega(n^{1/6})$ in the two-dimensional mesh: if they lie in the same vertical column of the three-dimensional mesh, then they lie in the same $n^{1/6} \times n^{1/6}$ square on the two-dimensional mesh, and, any two edges between columns of the three-dimensional mesh lie between nodes in adjacent squares of the two-dimensional mesh. The congestion of this embedding is $n^{1/6}$.

It is also not hard to see that the two-dimensional mesh is embedded in the three-dimensional mesh with congestion $n^{1/6}$ and dilation $n^{1/6}$. Any two adjacent two-dimensional mesh nodes are either embedded in the same vertical column of the three-dimensional mesh at distance at most $\Omega(n^{1/6})$ because of the spiral pattern, or embedded in adjacent vertical columns, but at distance at most $n^{1/6}$. Any embedding with dilation $n^{1/6}$ that uses edges in only one dimension has congestion at most $n^{1/6}$. So, this mutual embedding has $c_1 = c_2 = d_1 = d_2 = \Omega(n^{1/6})$. However, it is an open question whether this mutual embedding is optimal.

7. Mutual Embeddings Between Fat-Trees and Meshes

An interconnection network with n nodes and $O(n)$ area is called area-universal if it can simulate any network having $O(n)$ area with $O(\log n)$ slowdown. The proof that certain fat-tree networks[21] are area-universal also provides a mutual embedding between a fat-tree and a square mesh. For example, the area-universal fat-tree described in Ref. 19 has $n/\log^2 n$ leaves, and attached to each leaf is a square mesh with $\log^2 n$ nodes. This fat-tree can be laid out in area $O(n)$. With a little effort, this layout can be converted into a mutual embedding between a $2n$-node fat-tree (where $2n$ includes the leaf nodes of the tree, the internal nodes, and the square

meshes) and a $2n$-node square mesh. The embedding of the fat-tree in the mesh has congestion $O(1)$ and dilation $O(\sqrt{n})$, whereas the embedding of the mesh in the fat-tree has dilation $O(\log n)$ and congestion $O(\log n)$. Note that by Lemma 1.2 there is a lower bound of $O(\sqrt{n}/\log n)$ on the dilation of the embedding of the fat-tree in the mesh.

8. Conclusion and Future Directions

This paper has introduced the notion of mutual embeddings, and provided several examples, along with some lower bounds. Perhaps the most surprising result is that there are classes of graphs, such as linear arrays and cycles, where there are one-way embeddings with constant congestion and dilation, but the product of dilations is $\Omega(n)$ for every mutual embedding. One obvious direction for future work is to discover better mutual embeddings for interesting classes of graphs. Another is to improve our lower bound techniques. We presented techniques to prove bounds on the product of the dilations; investigating techniques that provide lower bounds on congestion, or lower bounds on, e.g., the products of congestion and dilation is another direction for future work.

References

1. R. Aleliunas and A. L. Rosenberg. On embedding rectangular grids in square grids. *IEEE Transactions on Computers*, 31(9):907–913, 1982.
2. Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.
3. Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27(4):930–951, 2006.
4. Marcin Bienkowski, Miroslaw Korzeniowski, and Harald Räcke. A practical algorithm for constructing oblivious routing schemes. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 24–33, 2003.
5. Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 25(3):694–717, March 2003.
6. D. Chen and S. Toledo. Vaidya's preconditioners: Implementation and experimental study. *Electronic Transactions on Numerical Analysis*, 16:30–49, 2003.
7. F.R.K. Chung. Labelings of graphs. In L. Beineke and R. Wilson, editors, *Selected Topics in Graph Theory 3*, pages 151–168. Academic Press, 1988.
8. M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, 34(3):477–495, 1978.
9. F. Gavril. Some NP-complete problems on graphs. In *Proceedings of the 11th Conference on Information Sciences and Systems*, pages 91–95, Baltimore, MD, USA, 1977. Johns Hopkins University.
10. David S. Greenberg, Lenwood S. Heath, and Arnold L. Rosenberg. Optimal embeddings of butterfly-like graphs in the hypercube. *Mathematical systems theory*, 23(1):61–77, 1990.

11. K. Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, Carnegie Mellon University, 1996. CMU-CS-96-123.
12. K. D. Gremban, G. L. Miller, and Marco Zagha. Performance evaluation of a new parallel preconditioner. In *Proceedings of the 9th International Parallel Processing Symposium*, pages 65–69, 1995.
13. Ralf Heckmann, Ralf Klasing, Burkhard Monien, and Walter Unger. Optimal embedding of complete binary trees into lines and grids. *Journal of Parallel and Distributed Computing*, 49(1):40–56, 1998.
14. Juraj Hromkovic, Vladimir Muller, Ondrej Sykora, and Imrich Vrto. On embedding interconnection networks into rings of processors. In Daniel Etiemble and Jean-Claude Syre, editors, *PARLE '92 Parallel Architectures and Languages Europe*, volume 605 of *Lecture Notes in Computer Science*, pages 51–62. Springer Berlin Heidelberg, 1992.
15. Richard R. Koch, F. T. Leighton, Bruce M. Maggs, Satish B. Rao, Arnold L. Rosenberg, and Eric J. Schwabe. Work-preserving emulations of fixed-connection networks. *Journal of ACM*, 44(1):104–147, 1997.
16. Petr Kolman and Christian Scheideler. Improved bounds for the unsplittable flow problem. *Journal of Algorithms*, 61(1):20–44, 2006.
17. S. Rao Kosaraju and Mikhail J. Atallah. Optimal simulations between mesh-connected arrays of processors. *Journal of ACM*, 35(3):635–650, 1988.
18. Masao Kumamoto and Eiji Miyano. Optimal distortion embedding of complete binary trees into lines. *Information Processing Letters*, 112(10):365–370, 2012.
19. F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17(1):157–205, 1994.
20. C. E. Leiserson. Area-efficient graph layouts. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 270–281, Oct 1980.
21. Charles E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, 34(10):892–901, Oct 1985.
22. Bruce M. Maggs, Gary L. Miller, Ojas Parekh, R. Ravi, and Shan Leung Maverick Woo. Finding effective support-tree preconditioners. In *Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 176–185, 2005.
23. Ch. H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Journal of Computing*, 16(3):263–270, 1976.
24. M. S. Paterson, W. L. Ruzzo, and L. Snyder. Bounds on minimax edge length for complete binary trees. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, pages 293–299, New York, NY, USA, 1981. ACM.
25. Richard Peng and Daniel A. Spielman. An efficient parallel solver for SDD linear systems. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 333–342, New York, NY, USA, 2014. ACM.
26. H. Räcke. Minimizing congestion in general networks. In *Proceedings of The 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 43–52, 2002.
27. Arnold L. Rosenberg and Lawrence Snyder. Bounds on the costs of data encodings. *Mathematical Systems Theory*, 12(1):9–39, 1978.
28. M. Rottger and U.-P. Schroeder. Efficient embeddings of grids into grids. *Discrete Applied Mathematics*, 108(1-2):143–173, 2001. Workshop on Graph Theoretic Concepts in Computer Science.

29. Milan Sekanina. On an ordering of the set of vertices of a connected graph. *Publications of the Faculty of Science, University of Brno*, 412:137–142, 1960.
30. Xiaojun Shen, Weifa Liang, and Qing Hu. On embedding between 2D meshes of the same size. *IEEE Transactions on Computers*, 46(8):880–889, 1997.
31. D. Spielman and S. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.