

Forbidden-Set Distance Labels for Graphs of Bounded Doubling Dimension

Ittai Abraham

Microsoft Research
Silicon Valley Center, USA
ittai@microsoft.com

Cyril Gavoille[†]

Université de Bordeaux
LaBRI, IUF, France
gavoille@labri.fr

Shiri Chechik^{*}

Dept. of Computer Science
Weizmann Institute, Israel
shiri.chechik@weizmann.ac.il

David Peleg^{*}

Dept. of Computer Science
Weizmann Institute, Israel
david.peleg@weizmann.ac.il

ABSTRACT

The paper proposes a forbidden-set labeling scheme for the family of graphs with doubling dimension bounded by α . For an n -vertex graph G in this family, and for any desired precision parameter $\epsilon > 0$, the labeling scheme stores an $O(1+\epsilon^{-1})^{2\alpha} \log^2 n$ -bit label at each vertex. Given the labels of two end-vertices s and t , and the labels of a set F of “forbidden” vertices and/or edges, our scheme can compute, in time polynomial in the length of the labels, a $1+\epsilon$ stretch approximation for the distance between s and t in the graph $G \setminus F$. The labeling scheme can be extended into a forbidden-set labeled routing scheme with stretch $1+\epsilon$ for graphs of bounded doubling dimension.

Categories and Subject Descriptors: F.2.2 Analysis of Algorithms and Problem Complexity: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Theory

Keywords: doubling dimension, forbidden sets, fault-tolerance, distance labeling, compact routing

1. INTRODUCTION

A fundamental problem in networks concerns efficiently representing the network so that queries of interest, notably the distance between vertex pairs, can be answered quickly. Beyond estimating distances, one may wish to actually route on shortest paths in the network. Since the network is often managed in a distributed manner, it is desirable to have a

^{*}Supported by the French-Israeli “Multi-Computing” project.

[†]Supported by the ANR-projects “ALADDIN” and “GRAAL” and the équipe-projet INRIA “CÉPAGE”.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PODC’10, July 25–28, 2010, Zurich, Switzerland.

Copyright 2010 ACM 978-1-60558-888-9/10/07 ...\$10.00.

distributed structure for answering distance queries, without having to resort to some central authority.

This motivation is captured by the *distance labeling* problem: given an n -vertex graph G , preprocess G so as to produce a collection of *distance labels* to be assigned to the vertices. Then, a distance query involving two vertices u, v should be answered using only the labels assigned to u and v , using some decoder function. We are interested in bounding the *label length* (i.e., the maximum length of a label) of the scheme. One motivation for that is that if these distance labels are sufficiently small, say of length polylogarithmic in n , then they may be used as the “addresses” for vertices in the network (see [12] for an overview). To perform routing, one can extend the idea so that the labels for u and v convey enough information so that u can determine its next hop neighbour on a short path towards v . Again, it is hoped that the labels are small; in general, if the label length is sublinear in n , then the resulting scheme is said to be a *compact routing scheme*. A recent survey on compact routing can be found in [7], and in [15, 16] it is argued that compact routing schemes are highly applicable to the problem of Internet routing. See also [10].

This paper considers a key extension of distance labeling and routing, relevant to many network settings. Suppose that from time to time, the network servers may learn that some subset F of nodes or links has failed, or is simply “forbidden” (inaccessible). It is then required to answer queries on the “surviving” graph $G \setminus F$, without having to recompute the labels. This type of extension was suggested in [6, 26], which considered this problem for graphs with bounded treewidth or cliquewidth.

In the concrete case of distance labeling, we are interested in a label-based distributed data structures. First, preprocess G and assign to each vertex v of G a label $L(v)$. Now, given the set of labels $\{L(f) : f \in F\}$, it is required to answer queries involving distances on $G \setminus F$. Specifically, given additionally two labels $L(u)$ and $L(v)$, it should be possible to answer a query concerning the distance between u and v in $G \setminus F$.

Computing the exact distance (or shortest route) between pairs of vertices may be costly in term of memory requirements. Typically, polynomial space is needed for exact distances in sparse graphs like planar graphs [13], whereas polylogarithmic labels are possible even for arbitrary graphs if a

small error is tolerated (see, e.g., [19, 24, 25]). The situation is similar for compact routing: $\Omega(n^{1/2})$ -bit routing tables are required for shortest path routing in bounded growth networks [2] or planar graphs, whereas routing tables of size $(\epsilon^{-1} \log n)^{O(1)}$ suffice for $(1 + \epsilon)$ -stretch routing scheme in bounded doubling dimension graphs [2, 17, 18, 22, 23] or in minor-free graphs [1], some much wider classes of networks.

The schemes presented herein demonstrate, somewhat surprisingly, that similar bounds can be achieved even in the (considerably harder) “forbidden-set” scenario.

Our contributions.

In this paper we consider undirected unweighted n -vertex graphs. Our scheme applies to any unweighted graph G , however, its efficiency depends exponentially on the *doubling dimension* of G , defined as the smallest $\alpha > 0$ such that, for every $r > 0$, every ball of radius $2r$ can be covered by 2^α balls of radius r . A related notion widely used for the design of Distributed Hash Tables [20] is the *grid dimension*¹. However, the family of bounded grid dimension graphs is strictly included in that of bounded doubling dimension. Concerning the issue of routing in real networks, the reader is referred to [10] for further discussions about the impact of the doubling dimension of the underlying graph.

We show the following results.

- Graphs of doubling dimension α have a forbidden-set $(1 + \epsilon)$ -approximate distance labeling scheme of label length² $O(1 + \epsilon^{-1})^{2\alpha} \log^2 n$. All the labels can be computed in polynomial time, and each query can be answered in time polynomial in the length of the labels occurring in the query.
- The scheme extends to a forbidden-set routing labeling scheme with stretch $1 + \epsilon$ and $O(1 + \epsilon^{-1})^{2\alpha} \log^2 n$ -bit routing tables.
- The exponential term in α appearing in the label length bound in our schemes is in fact necessary, even for a *connectivity* labeling scheme (equivalent to a $(1 + \epsilon)$ -approximate distance scheme with very large ϵ). More precisely, we show that any forbidden-set connectivity labeling scheme on the family of unweighted graphs of doubling dimension α requires labels of length $\Omega(2^{\alpha/2} + \log n)$. This should be contrasted with the failure-free scenario, where $\lceil \log n \rceil$ -bit labels suffice to solve connectivity in arbitrary graphs. Hence, the label length of our scheme is tight, up to a polylogarithmic factor.

An attractive feature of our solution is that the label and routing tables are not affected by the size of forbidden sets. Since our scheme is based on labels of polylogarithmic length associated with each vertex of the input graph, as a byproduct we obtain for bounded doubling dimension graphs an $(1 + \epsilon)$ -approximate distance oracle of size $O(\epsilon^{-1} n \log n)$, which is independent of the number of faults it is required to tolerate.

¹The maximum ratio of the volumes of radius- $2r$ balls over radius- r balls centered at the same vertex.

²Logarithms are in base two.

Applications.

Our motivating scenario is a network where, whenever a failure has occurred, we wish to recover without delay. After a failure of some collection of routers (vertices) or links (edges), network traffic must be quickly rerouted without loss, and without having to wait for the recomputation of the routing tables. Such a recomputation of the routes may be performed in the background, but during this time we still wish to ensure good performance, which means being able to route along short paths in $G \setminus F$, rather than just any paths that avoid the failed set F .

One way in which such a forbidden-set routing scheme may be used for fast recovery from failures is as follows: each router keeps track of a set F of “failed” routers, and it makes distance queries with respect to the surviving graph $G \setminus F$. Routers are routinely updated about the operational status (failures and recoveries) of other routers, either directly (by probing the neighbouring routers), or through other routers. As soon as a router u discovers a failure (respectively, recovery) of some router v , it adds v to (resp., removes it from) its set own set F_u and propagates this information to other routers. The propagation may be done using some flooding mechanism, or piggybacked onto data sent on some routes (and thus all routers on this path will learn about the failure of v). Clearly, it is possible for a router to begin routing on a path that is going to be cut by a failed set, but as soon as the packet reaches a router that is aware of the failure, it can make a new query and the packet can be rerouted back again on a new shortest path, without waiting for the update time incurred by a route maintenance algorithm.

Another important scenario is when a router decides to change its own routing policy. For example, for economic or security reasons, a part of the network may become forbidden. The local forbidden-set of the router can be accordingly modified, and it can update its route immediately without having to invoke a global route maintenance mechanism. (This application may require including information on the policy in the message header.)

Related work.

There is a vast literature on labeling schemes and information oracles in the classical (failure-free) setting, and it covers many aspects: distance, routing, exact queries, approximation, global or localized data-structures.

Data-structures supporting limited failures exist. A scheme for answering distance queries when a single edge may fail, a.k.a. the *distance sensitivity* problem, is presented in [9]. In this model, the failed edge e is specified at query time, along with two vertices u, v , and it is desired to compute the distance between u, v in $G \setminus \{e\}$. They showed how to construct an oracle for such queries of size $O(n^2 \log n)$ and query time $O(\log n)$, for general directed graphs. This has been extended to a single edge and/or vertex failure [3], and only recently to dual-failure [8] and to multiple edge failures [5]. The routing issue has been explored in [14] for single vertex failure and in [5] for two edge failures. Extending these solutions to three or more failures, particularly in a distributed representation, is clearly a challenging problem.

A scheme for the *forbidden-set distance labeling* problem that allows for the failure (or forbidding) of arbitrary subgraphs is given in [6, 26]. The scheme depends on the

treewidth or the cliquewidth³ of the graph, and can be distributed in the form of labels assigned to vertices. For graphs of treewidth or cliquewidth k , the label length is $O(k^2 \log^2 n)$ bits, and $k \leq n$ for every graph.

On a related note, the problem of answering forbidden-set connectivity queries (answering whether the vertices u and v are connected in $G \setminus F$) has also been considered. For the case of planar graphs, it has recently been shown how to construct $\Theta(\log n)$ bit labels so that queries about connectivity of the graph $G \setminus F$ can be answered in time $O(|F| \log |F|)$ [4]. A non-distributed solution for arbitrary graphs with a linear size data-structure has been given in [21]. The distribution of the representation is an open problem.

Preliminaries.

Denote the distance between u and v in the graph G by $d_G(u, v)$. A *forbidden-set distance oracle* is a data-structure designed for a graph G that supports distance queries in the graph $G \setminus F$ for every subset $F \subset V(G) \cup E(G)$. In other words, such an oracle $\mathcal{O}_G(u, v, F)$ must return $d_{G \setminus F}(u, v)$ for all u, v, F .

In this paper we are interested in oracles that can be *distributed* over the vertices of the graph itself, so that a query involving, say, vertices x, y and z , can be answered using the parts of the oracle stored at x, y and z , without accessing any other source of information.

More formally, a *forbidden-set distance labeling scheme* for a graph family \mathcal{F} is a pair (L, f) , where L is called the *labeling* or *marker* and f the *decoder*, such that for every graph G of the family \mathcal{F} ,

- the marker associates with each vertex u of G a binary label $L(u, G)$ (we simply write $L(u)$ when G is clear from the context); and
- for any $u, v \in V(G)$ and any subset $F \subset V(G) \cup E(G)$, the decoder returns $d_{G \setminus F}(u, v)$ given the labels of u and v and the labels of all vertices and edges of F . (The label of an edge (a, b) of F is specified by the pair $(L(a), L(b))$.)

The *label length* of a labeling scheme is the length (in bits) of the largest label it assigns to a vertex of a graph of \mathcal{F} . Observe that one can construct an oracle \mathcal{O}_G for G from the labeling scheme by storing in some table T the label of each vertex u , i.e., $T[u] = L(u, G)$. Hence the size of the oracle is at most n times the label length of the labeling scheme. A forbidden-set distance query (u, v, F) is answered by loading from T all the required labels and running the decoder f on those labels. The required labels are $T[u], T[v], T[x]$ for all vertices x of F , and $(T[a], T[b])$ for all edges (a, b) of F .

A forbidden-set distance oracle (or labeling scheme) is *s-approximate* if the value $\delta(u, v, F)$ it returns for a query (u, v, F) satisfies $d_{G \setminus F}(u, v) \leq \delta(u, v, F) \leq s \cdot d_{G \setminus F}(u, v)$.

2. FORBIDDEN-SET LABELING SCHEME

In this section we describe our forbidden-set distance and routing labeling schemes.

2.1 Distance labeling scheme

THEOREM 1. *Unweighted n -vertex graphs of doubling dimension α have a forbidden-set $(1 + \epsilon)$ -approximate distance*

³A graph measure generalizing treewidth.

labeling scheme $O(1 + \epsilon^{-1})^{2\alpha} \log^2 n$ -bit labels. All the labels can be computed in polynomial time, and each query can be answered in time polynomial in the label length of the query.

Section 2.1 is devoted to the proof of Theorem 1. Let G be an unweighted n -vertex graph of doubling dimension α . For a vertex x and a set $W \subseteq V(G)$, denote by $d_G(x, W)$ the distance between x and the closest $w \in W$. For every vertex v and radius r , let $B(v, r, G)$ denote the ball of radius r in G centered at v . We say that a subset $N \subseteq V(G)$ is an r -dominating set of G if for every vertex $v \in V(G)$ there exists a vertex $v' \in N$ such that $d_G(v, v') \leq r$.

Define a hierarchy of *nets*, namely, vertex sets in G , denoted by N_i , one for each level $i \in \{0, \dots, \lceil \log n \rceil\}$, with the following properties:

- (1) $N_0 = V(G)$;
- (2) $N_i \subset N_{i-1}$, for all $i > 0$;
- (3) N_i is a 2^i -dominating set for G .

For every vertex v , let $M_i(v)$ be the net-point in N_i closest to v . Note that $d_G(v, M_i(v)) \leq 2^i$.

For bounding the label length, we use the following fact while constructing the hierarchy of nets (see [11]).

FACT 1. *For a graph with doubling dimension α , there is an efficiently constructible r -dominating set $W(r)$, such that for every vertex $v \in V(G)$ and radius $R \geq r$, the set $B(v, R, G) \cap W(r)$ has size at most $(4R/r)^\alpha$.*

We set $N_{\lceil \log n \rceil} := W(2^{\lceil \log n \rceil})$ and $N_i = W(2^i) \cup N_{i+1}$ for $0 \leq i \leq \lceil \log n \rceil - 1$. It's not hard to see that this hierarchy of nets indeed satisfy properties (1) – (3). Using Fact 1 we have the following:

LEMMA 1. *For a graph with doubling dimension $\alpha > 1$, there is an efficiently constructible hierarchy of nets N_i , one for each level $i \in \{0, \dots, \lceil \log n \rceil\}$, satisfying properties (1) – (3) such that for every vertex $v \in V(G)$ and radius $R \geq r$, the set $B(v, R, G) \cap N_i$ has size at most $(8R/2^i)^\alpha$.*

Overview of the failure-free case.

Consider first the problem of constructing a distance oracle for graphs of bounded doubling dimension in the failures free setting ($F = \emptyset$). We now describe, on a high level, a variant of a distance oracle for this class based on labeling scheme, that is close to our scheme in the fault-tolerant setting. The idea is to give for each vertex v a short label $L(v)$, such that given the labels $L(s)$ and $L(t)$ of two vertices s and t , it is possible to approximate the distance between them with a stretch of at most $1 + \epsilon$. Let $I = \{c, \dots, \beta\}$ be the range of levels where $c = c(\epsilon)$ is some constant integer to be fixed later on and $\beta = \lceil \log n \rceil$. The label $L(v)$ of each vertex v consists of the distances from v to all vertices in $B(v, 2^{i+2}, G) \cap N_{i-c}$ for every $i \in I$. Note that for the lowest level, $L(v)$ contains all the vertices of the ball $B(v, 2^{c+2}, G)$ since N_0 contains all vertices. Now, when getting the labels $L(s)$ and $L(t)$ of two vertices s and t , whose distance needs to be estimated, do the following. Find the smallest index i such that $M_{i-c}(t)$ is in $B(s, 2^{i+2}, G)$ (extracting $M_{i-c}(t)$ from $L(t)$ and performing the check in $L(s)$). We then return $d' = d_G(s, M_{i-c}(t)) + d_G(t, M_{i-c}(t))$.

We now claim that setting $c = \max\{\lceil \log(2/\epsilon) \rceil, 0\}$ yields the desired stretch of $1 + \epsilon$. To see this, let d' be the distance estimate returned by the oracle and let $d = d_G(s, t)$. Let j be the index such that $2^j \leq d \leq 2^{j+1}$. Note that $d_G(s, M_{j-c}(t)) \leq d_G(s, t) + d_G(t, M_{j-c}(t)) \leq 2^{j+1} + 2^{j-c} < 2^{j+2}$. We get that for this index j , $M_{j-c}(t) \in B(s, 2^{j+2}, G)$. Hence the index i returned by the oracle satisfies $i \leq j$, so the distance estimate d' satisfies

$$\begin{aligned} d' &\leq d_G(s, M_{j-c}(t)) + d_G(M_{j-c}(t), t) \\ &\leq d_G(s, t) + d_G(t, M_{j-c}(t)) + d_G(t, M_{j-c}(t)) \\ &= d + 2 \cdot 2^{j-c} \leq d + 2^{j+1-\log(2/\epsilon)} \\ &= d + \epsilon/2 \cdot 2^{j+1} = d + \epsilon \cdot 2^j \\ &\leq (1 + \epsilon)d \end{aligned}$$

Finally, let us bound the label length. Consider the label $L(v)$ of some vertex v . Using Lemma 1, for each index $c \leq i \leq \log n$, the number of net-points added to $L(v)$ is at most $(8 \cdot 2^{i+2}/2^{i-c})^\alpha = (2^{5+\max\{\lceil \log(2/\epsilon) \rceil, 0\}})^\alpha = \max\{2^{5\alpha}, 2^{(6+\log(2/\epsilon))\alpha}\} = \max\{2^{5\alpha}, (128/\epsilon)^\alpha\}$, so the total label length is at most $\log^2 n \cdot \max\{2^{5\alpha}, (128/\epsilon)^\alpha\} = O(1 + \epsilon^{-1})^{2\alpha} \cdot \log^2 n$ bits (where one $\log n$ factor is due to the number of levels and one due to the fact that it takes $\log n$ bits to store each vertex).

The case of non-empty forbidden-set.

We now turn to the general case, where F is non-empty. In this setting F fails and it is desired to approximate the distance between s and t in the surviving graph $G \setminus F$, given the labels of the vertices s and t as well as the labels of all faulty vertices in the set F (for the sake of simplicity, we first consider forbidden vertex-set only).

The label $L(v)$ represents a sparse subgraph with virtual edges whose lengths are part of the label. In the failure free setting, $L(v)$ contains only edges that touch v , i.e., edges from v to some net-points in the ball around v . In contrast, in the fault-tolerant setting the label $L(v)$ contains also edges that do not touch v , but rather connect pairs of net-points in the ball around v .

In the failure-free setting we look for a net-point M relatively close to t (relative in perspective to the distance between s and t), such that the virtual edges (s, M) and (M, t) are contained in the label $L(s)$ and $L(t)$ respectively. Each of these virtual edges represents a shortest path in the original graph G . In the fault-tolerant setting, complications may arise when these paths contain some faulty vertices. To overcome this problem, instead of looking for one long edge (long in the sense that the path it represents is long), where this edge may represent a path that contain faulty vertices, we rather construct a virtual graph H that contains a small number of edges, where the idea is that in regions where there are no faulty vertices nearby we add a few long edges and in regions where there are faulty vertices nearby we add many shorter edges. Eventually, this virtual graph will contain a virtual path from s to t that represents a path in $G \setminus F$. Only “safe” edges will be added to H , safe in the sense that the path they represent is guaranteed not to contain a faulty vertex. After building this graph, we invoke a shortest path algorithm from s to t in order to estimate the distance $d_{G \setminus F}(s, t)$.

As in the failure free setting, the label $L(v)$ of each vertex v contains distances between low-level net-points in the re-

gion close to v , and distances between high level net-points in more remote regions.

Using the labels $L(s)$ and $L(t)$, we construct the graph H , by adding long edges as long as these edges are far away from every faulty vertex. It could be that some of the edges we wanted to add to the graph H are too close to some faulty vertices. In that case we can use the labels of the faulty vertices to replace these long edges with shorter ones, where in the regions that are very close to faulty vertices we add many very short edges.

We now dive into the details of the algorithm, explaining the label structure of each vertex v and showing how to construct the graph H given the labels of the vertices of $\{s, t\} \cup F$.

Let $I = \{c + 1, \dots, \beta\}$ be the range of levels where $c = c(\epsilon)$ is some constant integer to be fixed later on and $\beta = \lceil \log n \rceil$.

Labels.

The label $L(v)$ of each vertex v consists of a list of its level- i labels $L_i(v)$, where each $L_i(v)$ encodes an edge-weighted graph $H_i(v)$, for every $i \in I$.

For each level $i \in I$, we store some net-points (and some edges between them) in the label $L(v)$ of each vertex v . For each vertex v and for each level i , the label $L(v)$ stores all net-points of N_{i-c-1} that are inside a ball of radius r_i around v . Denote the *domination radius* of the net-points N_{i-c} by $\rho_i = 2^{i-c}$. The net-points stored on level i are taken from a ρ_{i-1} -dominating set for G , for reasons that will become clearer later on. Denote the ball of radius r_i around v by $B_i(v) = B(v, r_i, G)$. Denote by λ_i the maximum length of the edges added to the labels on level i . In other words, for every vertex v and level i , the label $L(v)$ stores all net-points N_{i-c-1} and all *short* edges between every pair of the net-points and also between v and the net-points, where an edge is considered to be short if its length is at most λ_i . On the lowest level $c + 1$, $L(v)$ stores all edges in the original graph G that are in $B_{c+1}(v)$.

Formally, the graph $H_i(v)$ is defined as follows:

- Vertex-set:

$$V(H_i(v)) = N_{i-c-1} \cap B_i(v) \quad \forall i \in I.$$

- Edge-set:

$$\begin{aligned} E(H_i(v)) = \\ \{(x, y) : d_G(x, y) \leq \lambda_i \text{ and } x, y \in V(H_i(v))\} \\ \forall i \in I. \end{aligned}$$

- Edge weights:

$$w(x, y) = d_G(x, y) \text{ for every } (x, y) \in E(H_i(v)).$$

Distance queries.

A *distance query* (s, t, F) involves a source s , a target t and a subset F of *faulty* (or *forbidden*) vertices. Its input consists of the labels $\{L(s), L(t)\} \cup \{L(v) : v \in F\}$. Answering a query (s, t, F) based on the given labels, namely, finding an approximate distance from s to t in the graph $G \setminus F$, is schematically done as follows:

1. Compute a weighted graph $H = H(s, t, F)$.
2. Compute a shortest path distance from s to t in H and return it.

More specifically, let us consider a query (s, t, F) , and let $\bar{F} = F \cup \{s, t\}$. Define the *level- i protected ball* of radius λ_i around v as $P_i(v) = B(v, \lambda_i, G)$. This definition is used in the query phase, where on level i we take only edges e in the labels of every $v \in \{s, t\} \cup F$ such that e is not inside any protected ball $P_i(f)$ for any $f \in F$. (We say that an edge is not inside a ball if at least one of its endpoints is outside the ball.) Notice that by taking to H , on level i , only edges of length at most λ_i that are not inside some protected ball, we get that these edges are far enough from every faulty vertex $f \in F$. Thus all edges added to H are safe in the sense that they represent paths that do not contain a faulty vertex.

Note that the data stored at the label of a vertex v is sufficient for checking if a vertex $x \in N_{i-c-1}$ is in the protected ball $P_i(v)$. The label $L_i(v)$ contains all edges (v, y) such that $y \in N_{i-c-1}$ and $d_G(v, y) \leq \lambda_i$. Therefore, if the label $L_i(v)$ does not contain the edge (v, x) , then x is not on the protected ball $P_i(v)$.

Formally, define the graph H as follows:

- Vertex-set: $V(H) = \bigcup_{v \in \bar{F}} \bigcup_{i \in I} V(H_i(v))$.
- Edge-set:

$$E(H) = \{e = (x, y) \mid \exists i \in I \text{ such that:}$$

$$e \in H_i(v) \text{ for some } v \in \bar{F}, \text{ and}$$

$$\{x, y\} \not\subseteq P_i(f) \text{ for every } f \in F\}$$

$$\cup \{e = (x, y) \mid e \in H_{c+1}(v) \text{ for some}$$

$$v \in \bar{F}, \omega(x, y) = 1 \text{ and } x, y \notin F\}.$$
- Edge weights - unchanged: $\omega(x, y) = d_G(x, y)$.

The intuition behind our parameter setting is as follows. We later look at a path P from s to t in $G \setminus F$ and claim that the computed graph H contains a path “close” to P . We construct a path P' corresponding to P by choosing some nodes in P and determining, for each chosen node v , some net-point “relatively” close to it. We later show that this chosen net-point appears either in the label of s or t or one of the failed vertices. Both the level of the net-point and the next chosen node v' on the path P are determined by the distance from v to F . The idea is that the net-points of v and v' must satisfy two requirements. First, they need to be far enough from any failed node, in order to allow us to claim that the edge connecting v and v' is “safe”, in the sense that the path it represents is in $G \setminus F$. Second, the net-points need to be on a sufficiently high level, so the node $f \in F$ closest to v will contain these net-points and the edge connecting them in its label. Another constraint is that the distance from v to its net-point needs to be small compared to the distance from v to the next chosen node v' in P , in order to ensure the desired stretch. The distance from a node v on the path P to F is captured by the parameter μ_i . More specifically, let i be the maximum index such that the ball of radius μ_i around v is free of faults. The next node in the path P is chosen at distance 2^i from v . Therefore the parameter ρ_i is fixed to be significantly smaller than 2^i . We now consider the net-point of v' (the next chosen node on the path P). We later show that as the distance from v to v' is 2^i and the distance from v to F is at most μ_{i+1} , the distance from v' to F is less than μ_{i+2} , which implies that the net-point of v' is on level at most $i+1$. Additional necessary constraints are that μ_i be sufficiently large so that the edge between the net-points of v and v' is “safe”, i.e., far

enough from every node in F , and thus represents a path in $G \setminus F$, and that r_i be large enough, so that the node of F closest to v contains in its label both the net-points of v and v' and the edge connecting them.

We thus set the parameters as follows: $\rho_i = 2^{i-c}$, $\lambda_i = 2^{i+1}$, $\mu_i = \rho_i + \lambda_i$, $r_i = \mu_{i+1} + 2^i + \rho_{i+1}$.

Analysis.

As explained above, we later look at a path P from s to t in $G \setminus F$ and construct a corresponding path P' to P in H . Towards proving the existence of such a path P' , we prove Claim 1. As discussed above, on each level i we add to H edges of length at most λ_i . The first part of the claim is essential for showing that the length of the edge between two consecutive net-points on the constructed path P' is at most λ_i . The purpose of the second part of the claim is to handle the case where there are some nodes on the path P that are very far from F . In this case we show that the net-points corresponding to these nodes are contained in the labels of s or t . It is not hard to verify that the chosen parameters satisfy the following claim for every $2 \leq c < \beta$.

CLAIM 1. (a) $\lambda_i \geq \rho_i + \rho_{i+1} + 2^i$, and (b) $N_{\beta-c-1} \subseteq B_{\beta}(v)$ for all $v \in V(G)$.

We now prove the correctness of this query algorithm. The following lemma proves that all edges added to H are “safe”, in other words, for every edge $e = (u, v)$ in H there is a corresponding path from u to v in $G \setminus F$ of length $\omega(e)$.

LEMMA 2. If (x, y) is an edge of H , then x and y are connected in $G \setminus F$ and their distance in $G \setminus F$ is $d_{G \setminus F}(x, y) = \omega(x, y)$.

PROOF. Consider an edge $(x, y) \in E(H)$. Note that $(x, y) \in H_i(v)$ for some i and v . If $\omega(x, y) = 1$, then x and y are neighbors in G and both x and y are not in F , therefore there is a path of length 1 from x to y in $G \setminus F$. Now suppose $\omega(x, y) > 1$. Then the graph G contains a shortest path $P(x, y)$ from x to y of length $d_G(x, y) \leq \lambda_i$, and for every $f \in F$, either x or y are not in the level- i protected ball $P_i(f)$. Consider a vertex $f \in F$. Without loss of generality, assume that $x \notin P_i(f)$, i.e., $d_G(x, f) > \lambda_i$. Assume, towards contradiction, that $f \in P(x, y)$. Therefore, $d_G(x, y) > d_G(x, f) > \lambda_i$, a contradiction. We get that $f \notin P(x, y)$ for every $f \in F$, and therefore $P(x, y)$ exists in $G \setminus F$. \square

For every vertex v , let $i(v)$ be the largest index $i \in I$ such that there is no $f \in F$ at distance μ_i from v in G , i.e., $B(v, \mu_i, G) \cap F = \emptyset$. If no such index exists, then set $i(v) = c$. Denote the nearest net point to v on that level by $\hat{M}(v) = M_{i(v)-c}(v)$. Note that for $i(v) = c$, $\hat{M}(v)$ is v itself. The next lemma establishes that the label scheme described above is a forbidden-set distance scheme.

LEMMA 3. Given $\epsilon > 0$, fix $c = \lceil \log(6/\epsilon) \rceil$. Consider two vertices $s, t \in V(G)$. If s and t are connected in $G \setminus F$ by a path of length d then they are connected by a path of length $(1 + \epsilon)d$ in H .

PROOF. Consider $s, t \in V(G)$ and let $d = d_{G \setminus F}(s, t)$. Let $Q = (s = v_1, \dots, v_d = t)$ be some shortest-path from s to t in $G \setminus F$. In order to show that there exists a path from s to t in H of length close to the length of Q , we show the

existence of an edge between \hat{M}_j and \hat{M}_{j+x} for sufficiently large x , where we denote $\hat{M}(v_j)$ by \hat{M}_j for every $1 \leq j \leq d$. We show that the distance from \hat{M}_j to \hat{M}_{j+x} is “close” to the distance from v_j to v_{j+x} . Loosely speaking the analysis is as follows. We show that the distance from s to \hat{M}_{z_1} and from \hat{M}_{z_1} to \hat{M}_{z_2} and so on are close to their corresponding distances in the path Q (i.e., to the paths: v_1 to v_{z_1} , v_{z_1} to v_{z_2} and so on). For the last z_k , we show that the distance from \hat{M}_{z_k} to t is relatively very short. Therefore we get a path from s to t in H of length “close” to their actual distance in G . Fig. 1 illustrates this path.

CLAIM 2. *Given $\epsilon > 0$, fix $c = \lceil \log(6/\epsilon) \rceil$. Consider a vertex v_j on the path Q and let $i(v_j) = \ell$. If $1 \leq j \leq d - 2^\ell$ then H contains an edge between \hat{M}_j and \hat{M}_{j+2^ℓ} and its weight is at most $(1 + \epsilon/2) \cdot 2^\ell$.*

PROOF. Let τ be the distance from v_j to F , i.e., $\tau = d_G(v_j, F)$ and let $\tau' = d_G(v_{j+2^\ell}, F)$. Let $\ell' = i(v_{j+2^\ell})$. By definition of $i(v_j)$, $\mu_\ell < \tau \leq \mu_{\ell+1}$. As $d_G(v_j, v_{j+2^\ell}) = 2^\ell$, we get that $\tau - 2^\ell \leq \tau' \leq \tau + 2^\ell$. This implies that $\ell - 1 \leq \ell' \leq \ell + 1$, as $\mu_{\ell-1} = \rho_{\ell-1} + \lambda_{\ell-1} = 2^{\ell-c-1} + 2^\ell < \mu_\ell - 2^\ell < \tau' \leq \mu_{\ell+1} + 2^\ell \leq \rho_{\ell+1} + \lambda_{\ell+1} + 2^\ell < \mu_{\ell+2}$.

We now prove the claim by case analysis. First assume that $\ell = c$, i.e., there exists a vertex $f' \in F$ such that $d_G(v_j, f') \leq \mu_{c+1}$.

The label $L(f)$ of every vertex $f \in F$ stores all edges (x, y) such that x and y are at distance at most r_{c+1} from f . Of those edges, we add to H all those whose endpoints are not in F .

First note that all edges in the path $(v_j = \hat{M}_j, \dots, v_{j+2^c})$ are added to H . To see this, observe that for all $j \leq r < j + 2^c$, both v_r and v_{r+1} are in $B(f', r_{c+1})$ as $d_G(v_r, f') \leq \mu_{c+1} + 2^c < r_c < r_{c+1}$ and also $d_G(v_{r+1}, f') \leq \mu_{c+1} + 2^c < r_c < r_{c+1}$, and in addition none of them belong to F .

We now consider two subcases. The first is when $\ell' = c$. In this subcase $\hat{M}_{j+2^c} = v_{j+2^c}$, hence H contains a path from \hat{M}_j to \hat{M}_{j+2^c} of length 2^c , namely, the path (v_j, \dots, v_{j+2^c}) itself.

The second subcase is when $\ell' = c + 1$. In this subcase, there is no vertex $f \in F$ at distance μ_{c+1} from v_{j+2^c} . As $d_G(v_{j+2^c}, \hat{M}_{j+2^c})$ is at most $\rho_{c+1} < \mu_{c+1}$, all edges in the path connecting v_{j+2^c} to \hat{M}_{j+2^c} are nonfaulty. In addition, this path belongs to $B(f', r_{c+1})$, as the distance from every vertex on that path to f' is at most $\mu_{c+1} + 2^c + \rho_{c+1} = r_c$ (see Fig. 2). Moreover, as already shown all edges in the path $(v_j = \hat{M}_j, \dots, v_{j+2^c})$ are added to H . We get a path from \hat{M}_j to \hat{M}_{j+2^c} of length at most $2^c + \rho_{c+1}$. Note that, by setting c to be any integer such that $c \geq \log(4/\epsilon)$, we get that $d(\hat{M}_j, \hat{M}_{j+2^c}) \leq (1 + \epsilon/2)d(v_j, v_{j+2^c}) = (1 + \epsilon/2) \cdot 2^\ell$.

Now assume that $\ell > c$ and that $j + 2^\ell \leq d$. Note that

$$\begin{aligned} d_G(\hat{M}_j, \hat{M}_{j+2^\ell}) &\leq d_G(\hat{M}_j, v_j) + d_G(v_j, v_{j+2^\ell}) + \\ &\quad d_G(v_{j+2^\ell}, \hat{M}_{j+2^\ell}) \\ &\leq \rho_\ell + 2^\ell + \rho_{\ell'} \leq \rho_\ell + \rho_{\ell+1} + 2^\ell \\ &\leq \lambda_\ell, \end{aligned}$$

where the last inequality follows from Claim 1(a).

In order to show that the edge $(\hat{M}_j, \hat{M}_{j+2^\ell})$ is in H , it suffices to show that both of these points belong to $B_\ell(v)$ for some $v \in \bar{F}$ and that for every $f \in F$, one of these points

does not belong to the protective ball $P_\ell(f)$. Note that $d_G(v_j, F) > \mu_\ell$, and therefore $d_G(\hat{M}_j, F) > \mu_\ell - \rho_\ell = \lambda_\ell$. It follows that \hat{M}_j and the edge $(\hat{M}_j, \hat{M}_{j+2^\ell})$ are not in the protective ball $P_\ell(f)$ for all $f \in F$. For showing that \hat{M}_j and \hat{M}_{j+1} belong to $B_\ell(v)$ for some $v \in \bar{F}$, we need to consider two cases. The first case is when $\ell = \beta$. In that case, both \hat{M}_j and \hat{M}_{j+2^ℓ} belong to $B_\beta(s)$. To see this, note that $\ell = \beta$ and $\ell' \geq \beta - 1$, so both \hat{M}_j and \hat{M}_{j+2^ℓ} belong to $N_{\beta-c-1} \subseteq B_\beta(s)$, by Claim 1(b).

The second case is when $c < \ell < \beta$. Note that by the maximality of $\ell = i(v_j)$, it must be that there exists a vertex $f \in F$ such that $d_G(v_j, f) \leq \mu_{\ell+1}$. As $r_\ell = \mu_{\ell+1} + 2^\ell + \rho_{\ell+1}$, we get that both \hat{M}_j and \hat{M}_{j+2^ℓ} belong to $B_\ell(f)$. In addition, both these points belong to $N_{\ell-c-1}$. To see this, recall that \hat{M}_j is a net-point of $N_{\ell-c}$ and as $\ell' \geq \ell - 1$, then \hat{M}_{j+2^ℓ} is a net-point of $N_{\ell-c-1}$, in addition $N_{\ell-c} \subseteq N_{\ell-c-1}$. Therefore the edge $(\hat{M}_j, \hat{M}_{j+2^\ell})$ is in H . By setting $c \geq \log(6/\epsilon)$, we get $d_H(\hat{M}_j, \hat{M}_{j+2^\ell}) \leq 2^\ell + \rho_\ell + \rho_{\ell+1} = 2^\ell + 2^{\ell-c} + 2^{\ell-c+1} = 2^\ell + 2^\ell \cdot 3 \cdot 2^{-c} = (1 + \epsilon/2)2^\ell$. \square

To ensure the existence of a path from s to t in H , we also need to show that there is a path from s to some \hat{M}_{z_1} and from some \hat{M}_{z_2} to t , where we explain later how to choose z_1 and z_2 . We then bound the length of this path using a careful analysis.

Let $\ell = i(v_1)$ and let $\ell' = i(v_d)$. We consider several cases.

The first case is when $d < 2^\ell$ or when $d < 2^{\ell'}$. Handling this case is similar to the failure-free setting. In this case the region around the path connecting s and t in the graph $G \setminus F$ is free from faults, therefore we can handle it as handling a distance query in the failure-free setting. More specifically, assume w.l.o.g that $d < 2^\ell$ and that $\ell \leq \ell'$, and let i be such that $2^{i-1} \leq d \leq 2^i$. Note that $i \leq \ell$. Let $z \in N_{i-c}$ be the closest net point to t in N_{i-c} . As $r_i = \mu_{i+1} + 2^i + \rho_{i+1}$, we get that z is in the ball $B_i(s)$. In addition, $d_G(s, F) > \mu_\ell \geq \mu_i = \rho_i + \lambda_i$. We get that the vertex s and the edge between s and z are not in the protective ball $P_i(f)$ for all $f \in F$. Moreover $d_G(s, z) \leq \rho_i + d \leq \lambda_i$. Therefore the edge between s and z is added to H . By a similar reasoning, the edge from t to z is also added to H . We get a path from s to t of length $2\rho_i + d$. As in the failure-free setting, taking any $c \geq \log(2/\epsilon)$ yields the desired $1 + \epsilon$ stretch.

The next case to consider is when $d > 2^\ell$ and $d > 2^{\ell'}$. In this case, using the same arguments as before we get a path from $s = v_1$ to \hat{M}_{1+2^ℓ} of length at most $\rho_\ell + 2^\ell$. Note that by taking again $c \geq \log(6/\epsilon)$, we get $d_H(s, \hat{M}_{1+2^\ell}) \leq (1 + \epsilon/2)2^\ell$. By Claim 2, H contains a path from \hat{M}_{1+2^ℓ} to $\hat{M}_{1+2^\ell+2^{z_1}}$ and from that vertex to $\hat{M}_{1+2^\ell+2^{z_1}+2^{z_2}}$ and so on, where $z_1 = i(v_{1+2^\ell})$ and $z_2 = i(v_{1+2^\ell+2^{z_1}})$. This continues until the route reaches a vertex \hat{M}_j corresponding to a vertex v_j such that $d_G(v_j, t) < 2^{i(v_j)}$.

Let $d_G(s, v_j) = d'$ and $d_G(v_j, t) = d''$. We get that $d_H(s, \hat{M}_j) \leq (1 + \epsilon/2)d'$.

Note that using similar arguments as before, we get that H contains an edge between \hat{M}_j to t of weight $d_H(\hat{M}_j, t) \leq d'' + \rho_{\ell'+1}$ (where $\ell' - 1 \leq i(v_j) \leq \ell' + 1$ again by similar arguments as before).

By setting $c = \max\{\lceil \log(6/\epsilon) \rceil, 2\}$ and concatenating all these paths together, we get that H contains a path from s to t of length $d_H(s, t) \leq d_H(s, \hat{M}_j) + d_H(\hat{M}_j, t) \leq (1 + \epsilon/2)d' + d'' + \rho_{\ell'+1} \leq \rho_{\ell'+1} + (1 + \epsilon/2)d \leq (1 + \epsilon)d$.

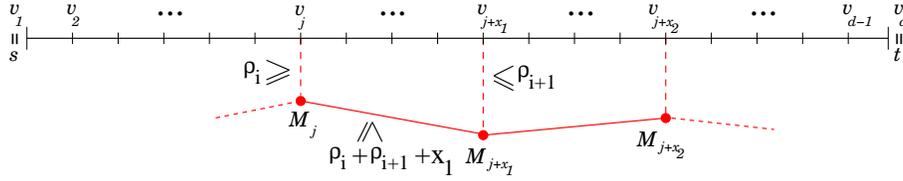


Figure 1: Illustration of the path from s to t in H .

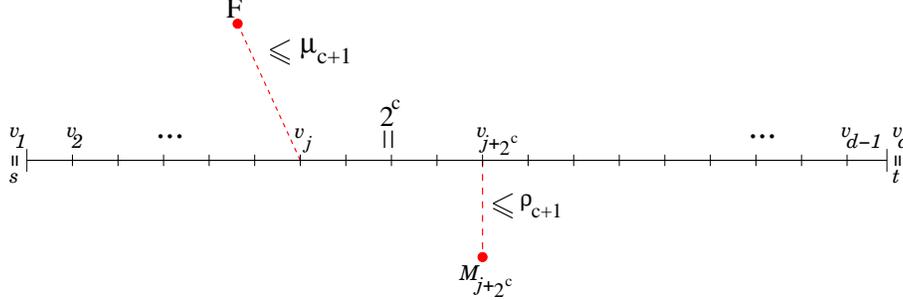


Figure 2: The case $\ell = c$ and $\ell' = c + 1$.

We finally bound the label length.

LEMMA 4. *The label length is $O(1 + \epsilon^{-1})^{2\alpha} \cdot \log^2 n$.*

PROOF. By Lemma 1, for each index $c \leq i \leq \log n$, the number of net-points added to $L(v)$ is at most $(8r_i/2^{i-c-1})^\alpha < (8 \cdot 2^{i+3}/2^{i-c-1})^\alpha = (8 \cdot 2^{4+c})^\alpha = \max\{512^\alpha, (1536/\epsilon)^\alpha\}$, where the first inequality is due to the fact that $r_i = \mu_{i+1} + 2^i + \rho_{i+1} = 2\rho_{i+1} + \lambda_{i+1} + 2^i = 2^{i-c+2} + 2^{i+2} + 2^i < 2^{i+3}$. Therefore, for a specific index i , the total length added to $L(v)$ is at most $\max\{512^{2\alpha}, (1536/\epsilon)^{2\alpha}\}$, as for each pair of vertices we might store an edge. Hence the label length is at most $\log^2 n \cdot \max\{O(1)^{2\alpha}, (O(1)/\epsilon)^{2\alpha}\} = O(1 + \epsilon^{-1})^{2\alpha} \cdot \log^2 n$. \square

Edge Faults.

To simplify notations, the construction we describe handle only vertex faults. However, it's not hard to see that only small modifications are needed for this construction to handle both vertex and edge failures together. The only modification needed is in the query phase, in the low level instead of adding all edges of G that are in $H_{c+1}(v)$ whose both endpoints are not faulty, rather just add all edges of G where both endpoints are not faulty and also the edge itself is not faulty (where we assume that for a faulty edge (x, y) we get the labels $(L(x), L(y))$).

2.2 Routing scheme

We can easily transform our forbidden-set labeling scheme to a forbidden-set compact routing scheme. Each vertex u stores its label $L(u)$, and for each vertex x of G contained in $L(u)$, vertex u stores the port of the out-going edge on a shortest path that leads to x from u .

The total storage is $O(|V(H)| \log n)$, where $|V(H)|$ is the number of vertices of H in $L(u)$'s label. From the previous section, $|V(H)| = O(1 + \epsilon^{-1})^{2\alpha} \log n$, and so the label length complexity is not affected by the routing extension. Note that the vertex names are preserved, so on $\lceil \log n \rceil$ bits, and headers have length at most $O(|V(H)| \log n)$. (If the method

is used by a router v for implementing a private routing policy, by forbidding the use of certain set F_v , then the header size will have to include a description of the policy, increasing it accordingly.)

It is easy to see that given this routing information, and given the labels of s, t, F , then for any edge (x, y) in H , from any vertex along the shortest path between x and y one can route to x or to y with stretch 1. This is due to the fact that every vertex z on a shortest path from x to y in G , assuming that (x, y) is an edge of H , contains x and y in its label. This implies that the stretch of the forbidden-set compact routing scheme is exactly the stretch of the forbidden-set label scheme. Therefore, we have showed:

THEOREM 2. *Unweighted n -vertex graphs of doubling dimension α have a forbidden-set routing labeling scheme with stretch $1 + \epsilon$ and $O(1 + \epsilon^{-1})^{2\alpha} \log^2 n$ -bit routing tables. All the labels and routing tables can be computed in polynomial time, and each query can be answered in time polynomial in the label length of the query.*

3. LOWER BOUND

In this section we provide a lower bound on the size (namely, total number of bits) of a (distance or connectivity) oracle. The bound tells us that the exponential term in α appearing in the label length bound in our scheme is in fact necessary, even for connectivity oracles, for the class of graphs of doubling dimension α . Note that a lower bound on the size of a connectivity oracle implies the same lower bound on the size of any approximate distance oracle, and thus also on its label length. Observe also that in the failure-free case, connectivity queries can be supported with $\lceil \log c \rceil$ -bit labels, where $c \leq n$ is the number of connected components of the graph.

THEOREM 3. *Every forbidden-set connectivity labeling scheme for unweighted n -vertex graphs of doubling dimension α requires labels of length $\Omega(2^{\alpha/2} + \log n)$.*

PROOF. Consider a graph family \mathcal{F} . For each graph G of \mathcal{F} , we consider any forbidden-set connectivity oracle \mathcal{O}_G for

G . Formally, $\mathcal{O}_G(i, j, F) = \text{true}$ if i and j belong to the same connected component of $G \setminus F$, and *false* otherwise, where i, j are vertices of G and $F \subset V(G) \cup E(G)$.

We claim that there is a graph G_0 such that its connectivity oracle \mathcal{O}_{G_0} has size $\log |\mathcal{F}|$. Consider a graph $G \in \mathcal{F}$. For given endpoints i and j , denote the “everywhere failure” set of G outside i and j by $F(i, j) = V(G) \setminus \{i, j\}$. For every two vertices i, j , $\mathcal{O}_G(i, j, F(i, j))$ is *true* if and only if i is adjacent to j in G . Indeed, the graph $G \setminus F(i, j)$ consists of either the edge (i, j) (in case i and j are neighbors in G) or the isolated vertices i and j (in case i and j are not adjacent). It follows that invoking the connectivity oracle and testing connectivity $\mathcal{O}_G(i, j, F(i, j))$ for all pairs of vertices determines the structure of the graph G . Consequently, the number of distinct connectivity oracles for \mathcal{F} , i.e., the cardinality of $\{\mathcal{O}_G : G \in \mathcal{F}\}$, is at least the number of elements of $|\mathcal{F}|$. Therefore, for at least one graph $G_0 \in \mathcal{F}$, the size of the oracle \mathcal{O}_{G_0} is at least $\log |\mathcal{F}|$.

We have seen that every labeling scheme using k -bit labels has a corresponding oracle of length nk . Thus, there must exist a label of length at least $\frac{1}{n} \log |\mathcal{F}|$ in every forbidden-set connectivity labeling scheme for the family \mathcal{F} . We shall consider now a specific family \mathcal{F} .

Let d, p be two positive integers. For simplicity, we assume that d is even and $p \geq 2$. We consider the graphs $G_{p,d}$ and $H_{p,d}$, two variants of the d -dimensional grid of $p \times \dots \times p$ vertices. The vertices of $G_{p,d}$ and $H_{p,d}$ are sequence (x_1, \dots, x_d) where $x_i \in \{0, \dots, p-1\}$. Two vertices $x = (x_1, \dots, x_d)$ and $y = (y_1, \dots, y_d)$ are adjacent in $G_{p,d}$ if and only if $\max_i |x_i - y_i| = 1$. They are adjacent in $H_{p,d}$ if and only if $\max_i |x_i - y_i| = 1$ and $\sum_i |x_i - y_i| \leq d/2$. The number of edges of $G_{p,d}$ is $m_{p,d} = \Omega(2^d p^d)$, its minimum degree being $2^d - 1$. The number of edges of $H_{p,d}$ is $|E(H_{p,d})| = \frac{1}{2} m_{p,d}$.

The doubling dimension of $G_{p,d}$ is $\leq 2^d$, because any ball of radius $2r$, for any $r > 0$, centered at (x_1, \dots, x_d) , can be covered by no more than 2^d balls of radius r centered at the vertices $(|x_1 + c_1|, \dots, |x_d + c_d|)$, where $c_i \in \{-r, +r\}$. Note also that $H_{p,d}$ is a 2-spanner of $G_{p,d}$, i.e., a spanning subgraph in which any pair of neighbor vertices in $G_{p,d}$ are at distance at most two in $H_{p,d}$.

We consider the family $\mathcal{F}_{n,\alpha}$ of n -vertex graphs composed of all the subgraphs of $G_{p,d}$ containing $H_{p,d}$, where $n = p^\alpha$ and $\alpha = 2d$ (with d an even integer). Let G be a graph of $\mathcal{F}_{n,\alpha}$. Since G contains $H_{p,d}$, G is a s -spanner of $G_{p,d}$ for some $s \leq 2$. In particular, any ball B of radius $2r$ in G is contained in a ball B' of radius $2rs \leq 4r$ of $G_{p,d}$. Using the doubling dimension of $G_{p,d}$, ball B' (and so ball B) can be covered by at most $(2^d) \cdot (2^d)$ balls of radius r of $G_{p,d}$. Each ball of radius r in $G_{p,d}$ is included in some ball of radius r in G , G is a subgraph of $G_{p,d}$. It follows that ball B can be covered by no more than $2^{2d} = 2^\alpha$ balls of radius r of G . Thus G , and all the graphs of $\mathcal{F}_{n,\alpha}$, have n vertices and doubling dimension $\leq \alpha$. The number of graphs in the family $\mathcal{F}_{n,\alpha}$ is

$$\begin{aligned} |\mathcal{F}_{n,\alpha}| &= 2^{|E(G_{p,d})| - |E(H_{p,d})|} \\ &= 2^{\frac{1}{2} m_{p,d}} \geq 2^{\Omega(2^d p^d)} = 2^{\Omega(2^{\alpha/2} n)}. \end{aligned}$$

By the above discussion, every forbidden-set connectivity labeling scheme for $\mathcal{F}_{n,\alpha}$ requires labels of length $\frac{1}{n} \log |\mathcal{F}_{n,\alpha}| = \Omega(2^{\alpha/2})$.

To conclude, let us show that any forbidden-set connectiv-

ity labeling scheme on $\mathcal{F}_{n,\alpha}$ requires at least $n - 2$ different labels. Assume the scheme assigns at most $n - 3$ labels to the vertices of all graphs of $\mathcal{F}_{n,\alpha}$. Consider the n -vertex path P_n where $n \geq 4$. Observe that $P_n \in \mathcal{F}_{n,\alpha}$ since $P_n = G_{n,1}$. Among the vertices of P_n receiving the same label (there are at least three such vertices), we select two non-neighboring vertices x, y , one of which is not an end-vertex of P_n . Let $P_n(x, y)$ be the subpath of P_n going from x to y , excluding x and y . Without loss of generality assume y is not an endpoint of P_n , and let z be the neighbor of y that is not in $P_n(x, y)$. Note that $P_n(x, y)$ contains at least one vertex, say w , and $w \notin \{x, y, z\}$. If w is faulty, then z and x are not in the same component, whereas z and y are. This implies that testing connectivity queries $(z, x, \{w\})$ and $(z, y, \{w\})$ should lead to different results. However, the input labels given to the decoder are the same since $L(x) = L(y)$; contradiction. Hence there are at least $n - 2$ labels.

Therefore, every forbidden-set connectivity labeling scheme requires labels of length at least $\max\{\Omega(2^{\alpha/2}), \log(n - 2)\} = \Omega(2^{\alpha/2} + \log n)$. \square

Acknowledgement.

We thank A. Twigg and B. Courcelle for helpful discussions about labeling schemes in the forbidden-set setting.

4. REFERENCES

- [1] Ittai Abraham and Cyril Gavoille. Object location using path separators. In *25th ACM Symp. on Principles of Distributed Computing (PODC)*, 188–197, 2006.
- [2] Ittai Abraham, Cyril Gavoille, Andrew V. Goldberg, and Dahlia Malkhi. Routing in networks with low doubling dimension. In *26th Int. Conf. on Distributed Computing Systems (ICDCS)*, IEEE, 2006.
- [3] Aaron Bernstein and David Karger. A nearly optimal oracle for avoiding failed vertices and edges. In *41st ACM Symp. on Theory of Computing (STOC)*, 101–110, 2009.
- [4] Bruno Courcelle, Cyril Gavoille, Mamadou Mustapha Kanté, and David Andrew Twigg. Connectivity check in 3-connected planar graphs with obstacles. In *Int. Conf. on Topological & Geometric Graph Theory*, volume 31, 151–155. Electronic Notes in Discrete Mathematics, 2008.
- [5] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f -sensitivity distance oracles and routing schemes. Unpublished manuscript, 2009.
- [6] Bruno Courcelle and David Andrew Twigg. Compact forbidden-set routing. In *24th Symp. on Theoretical Aspects of Computer Science (STACS)*, LNCS 4393, 37–48. SV, 2007.
- [7] Michael Dom. Compact routing. In *Algorithms for Sensor and Ad Hoc Networks*, LNCS 4621, 187–202. SV, 2007.
- [8] Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *20nd Symp. on Discrete Algorithms (SODA)*, 506–515. ACM-SIAM, 2009.
- [9] Camil Demetrescu and Mikkel Thorup. Oracles for distances avoiding a link-failure. In *13th Symp. on Discrete Algorithms (SODA)*, 838–843. ACM-SIAM, 2002.

- [10] Pierre Fraigniaud, Emmanuelle Lebhar, and Laurent Viennot. The inframetric model for the internet. In *27th IEEE Conf. on Computer Communications (INFOCOM)*, 1085–1093, 2008.
- [11] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th IEEE Symp. on Foundations of Computer Science (FOCS)*, 534–543, IEEE, 2003.
- [12] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Distributed Computing*, 16(2-3):111–120, 2003.
- [13] Cyril Gavoille, David Peleg, Stéphane Pérennès, and Ran Raz. Distance labeling in graphs. *J. Algorithms*, 53(1):85–112, 2004.
- [14] Neelesh Khanna and Surender Baswana. Approximate shortest path oracle under vertex failure. In *26th Symp. on Theoretical Aspects of Computer Science (STACS)*, 2010.
- [15] Dmitri Krioukov, KC Claffy, Kevin Fall, and Arthur Brady. On compact routing for the internet. *ACM SIGCOMM Computer Communication Review*, 37(3):43–52, 2007.
- [16] Dmitri Krioukov, Kevin Fall, and Xiaowei Yang. Compact routing on internet-like graphs. In *23rd Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, volume 1, –219, 2004.
- [17] Goran Konjevod, Andréa Werneck Richa, and Donglin Xia. Optimal scale-free compact routing schemes in doubling networks. In *18th Symp. on Discrete Algorithms (SODA)*, 939–948. ACM-SIAM, 2007.
- [18] Goran Konjevod, Andréa Werneck Richa, and Donglin Xia. Dynamic routing and location services in metrics of low doubling dimension. In *22nd Int. Symp. on Distributed Computing (DISC)*, LNCS 5218, 379–393. SV, 2008.
- [19] David Peleg. Proximity-preserving labeling schemes and their applications. In *Proc. 25th Workshop on Graph- Theoretic Concepts in Computer Science*, 30–41, 1999.
- [20] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa Werneck Richa. Accessing nearby copies of replicated objects in a distributed environment. In *9th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, 311–320, 1997.
- [21] Mihai Pătraşcu and Mikkel Thorup. Planning for fast connectivity updates. In *48th IEEE Symp. on Foundations of Computer Science (FOCS)*, 263–271, IEEE, 2007.
- [22] Aleksandrs Slivkins. Distance estimation and object location via rings of neighbors. In *24th ACM Symp. on Principles of Distributed Computing (PODC)*, 41–50, 2005.
- [23] Aleksandrs Slivkins. Distance estimation and object location via rings of neighbors. *Distributed Computing*, 19(4):313–333, 2007.
- [24] Kunal Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In *36th ACM Symp. on Theory of Computing (STOC)*, 281–290, 2004.
- [25] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.
- [26] David Andrew Twigg. *Forbidden-set Routing*. PhD thesis, University of Cambridge (King’s College), 2006.