Forbidden-Set Distance Labels for Graphs of Bounded Doubling Dimension

ITTAI ABRAHAM, VMware Research, Palo Alto, CA, USA
SHIRI CHECHIK, Tel-Aviv University, Tel-Aviv, Israel
CYRIL GAVOILLE, Université de Bordeaux, LaBRI, IUF, France
DAVID PELEG, The Weizmann Institute of Science, Rehovot, Israel

This article proposes a forbidden-set labeling scheme for the family of unweighted 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 $O(1+\epsilon^{-1})^{2\alpha}\cdot|F|^2\log n$ time, 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

Additional Key Words and Phrases: Doubling dimension, forbidden sets, fault-tolerance, distance labeling, compact routing

ACM Reference Format:
Ittai Abraham, Shiri Chechik, Cyril Gavoille, and David Peleg. 2016. Forbidden-set distance labels for graphs of bounded doubling dimension. ACM Trans. Algorithms 12, 2, Article 22 (February 2016), 17 pages.
DOI: http://dx.doi.org/10.1145/2818694

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 and efficiently. In many large networks, failures are common, and it is often desirable to quickly compute distances given some set of failures. 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 *distributed* structure for answering distance queries, without having to resort to some central authority. In many large networks, it may be desirable to answer distance queries using a small amount of information. For example, a hand-held device may need to

Supported in part by the Israel Science Foundation (grant 894/09), the United States-Israel Binational Science Foundation (grant 2008348), and the Israel Ministry of Science and Technology (infrastructures grant).
Authors' addresses: I. Abraham, Vmware Research, 3401 Hillview Ave, Palo Alto, CA 94304; email: iabraham@vmware.com; S. Chechik, Tel-Aviv University, Tel-Aviv, Israel; email: shiri.chechik@gmail.com; C. Gavoille, LaBRI - University of Bordeaux, 351, cours de la Liberation, 33405 Talence cedex, France; email: gavoille@labri.fr; D. Peleg, Dept. of Computer Science and Applied Mathematics, The weizmann Institute, Rehovot 7610001, Israel; email: david.peleg@weizmann.ac.il.

DOI: http://dx.doi.org/10.1145/2818694

compute distance queries related to its local region. It would be desirable not to force the device to download a data structure whose size is proportional to the whole graph of the world but only to the relevant region. A similar requirement seems natural for failures. It would be desirable to download only information proportional to the failures relevant to region and query of the hand-held device.

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 this 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 Gavoille and Peleg [2003] 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 toward 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 Dom [2007], and in Krioukov et al. [2004, 2007] it is argued that compact routing schemes are highly applicable to the problem of Internet routing. See also Fraigniaud et al. [2008], Thorup and Zwick [2001], and Chechik [2013].

This article considers a key extension of distance labeling and routing relevant to many network settings. Suppose that, from time to time, the network servers 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 Courcelle and Twigg [2007] and Twigg [2006], which considered this problem for graphs with bounded treewidth or cliquewidth.

In the concrete case of distance labeling, we are interested in 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 $\{L(f) : f \in F\}$, we must 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 [Gavoille et al. 2004], whereas polylogarithmic labels are possible even for arbitrary graphs if a small error is tolerated (see, e.g., Peleg [1999], Talwar [2004], and Thorup [2004]). The situation is similar for compact routing: $\Omega(n^{1/2})$-bit routing tables are required for shortest path routing in bounded growth networks [Abraham et al. 2006] 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 [Abraham et al. 2006; Konjevod et al. 2007, 2008; Slivkins 2005, 2007] or in minor-free graphs [Abraham and Gavoille 2006], 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 article, we extend the forbidden-set distance label paradigm from exact distances in bounded tree-width graph to approximate distances in bounded doubling dimension graphs. 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 α such that, for every $r > 0$, every ball of radius $2r$ can be covered by 2^{α} balls of radius r.

We show the following results.

—Graphs of doubling dimension α have a forbidden-set $(1+\epsilon)$-approximate distance labeling scheme of label length[1] $O(1+\epsilon^{-1})^{2\alpha}\log^2 n$. All the labels can be computed in polynomial time, and each query can be answered in $O(1+\epsilon^{-1})^{2\alpha}\cdot|F|^2\log n$ time.
—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.

From a technical perspective, the main contribution of this article is to distill the core algorithmic ideas of Courcelle and Twigg [2007] and Twigg [2006] that were tailored to exact distances and bounded tree-width graphs and adapt them to approximate distances and bounded doubling dimension graphs. At a high level, our construction has similarities to the approach of Courcelle and Twigg [2007]'s distance query: Take the required labels, algorithmically build a sketch-graph from them, and then run a shortest path algorithm on the sketch graph. On the other hand, the content of our labels and the algorithm that builds our sketch graph is tailored to bounded doubling dimension. These algorithms bear almost no resemblance to the bounded tree-width case. At first sight, our label construction seems to use a standard hierarchical net-point construction (which is quite common in many algorithms for bounded doubling dimension). However, the nontrivial part of our work is in designing the algorithm that constructs the sketch graph from the labels given the faults. To the best of our knowledge, our work is the first result showing that one can concisely capture the shortest path structure after a node failure using a small label. Showing that the required stretch bound of $1+\epsilon$ is nontrivial and requires carefully bounding the stretch loss accumulation over multiple phases. In fact, before our work, it was not clear if the approach of Courcelle and Twigg [2007] could be extended to approximate stretch bounds (the approach of Courcelle and Twigg [2007] is based on monadic second-order logic and does not seem amendable to approximations).

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

[1]Logarithms are in base two.

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 (recovery) of some router v, it adds v to (removes it from) its 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.)

The ability to quickly compute distances from short labels has many practical advantages. The distance labeling approach has recently been implemented to solve real-world large-scale route planning problems [Abraham et al. 2011, 2014]. Due to their data locality, labels are currently the fastest way to compute distances on content-scale road networks. These road networks seem to have low highway dimension [Abraham et al. 2010], which implies low doubling dimension. We believe the ideas in this work can contribute to extend the notion of hub labels to allow dynamic and forbidden-set distance labels. Allowing users to compute distances in road networks given a set of failures (road closures, accidents, etc) could be an important feature of new practical labeling schemes.

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, and 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 Demetrescu and Thorup [2002]. 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 [Bernstein and Karger 2009] and only recently to dual-failure [Duan and Pettie 2009] and to multiple-edge failures [Chechik et al. 2010]. The routing issue has been explored in Khanna and Baswana [2010] for single-vertex failure and in Chechik [2011] for multiple-edge failures.

A scheme for the *forbidden-set distance labeling* problem that allows for the failure (or forbidding) of arbitrary subgraphs is given in Courcelle and Twigg [2007] and Twigg [2006]. The scheme depends on the treewidth or the cliquewidth[2] 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 $k \leqslant n$ for every graph.

Recently, an efficient construction for forbidden-set distance labeling scheme for planar graphs was presented in Abraham et al. [2012]. More specifically, it was shown how to construct, for a given n-vertex planar graph with edge weights in $[1, M]$ and a parameter $\epsilon > 0$, a forbidden-set $(1 + \epsilon)$-approximate distance labeling scheme of label

[2]A graph measure generalizing treewidth.

length $\lambda = O(\epsilon^{-1} \log^2 n \log(nM) \cdot (\epsilon^{-1} + \log n))$. The query time is $O(|F|^2\lambda)$ time, where F is the set of faulty vertices/edges and all the labels can be computed in $O(n\lambda)$ time. Moreover, a general method to transform $(1+\epsilon)$ forbidden-set labeling schemas into a fully dynamic $(1+\epsilon)$ distance oracle was presented in Abraham et al. [2012]. Combining this method with the results of our work presented here, one can construct for a given n-vertex unweighted graph of doubling dimension α a fully dynamic $(1+\epsilon)$ approximate distance oracle of size $\tilde{O}((1+\epsilon^{-1})^{2\alpha}n)$, where each query and update operations takes $\tilde{O}(n^{1/2})$ worst-case time.

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 $O(|F| \log |F|)$ [Courcelle et al. 2008]. A nondistributed solution for arbitrary graphs with a linear size data structure has been given in Pătraşcu and Thorup [2007]. 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 article, 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 $d_{G\setminus F}(u, v) \leqslant \delta(u, v, F) \leqslant s \cdot d_{G\setminus F}(u, v)$.

2. FORBIDDEN-SET LABELING SCHEME

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

2.1. Distance Labeling Scheme

THEOREM 2.1. *Unweighted n-vertex graphs of doubling dimension $\alpha \geqslant 1$ have a forbidden-set $(1+\epsilon)$-approximate distance labeling scheme with $O(1+\epsilon^{-1})^{2\alpha} \log^2 n$-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 2.1. Let G be an unweighted n-vertex graph of doubling dimension $\alpha \geqslant 1$. 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 $B(v, r)$ 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 of G such that $d_G(v, N) \leqslant r$.

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

(1) N_i is a $(2^i - 1)$-dominating set for G for each level $i \geqslant 0$.
(2) $N_i \subseteq N_{i-1}$, for each level $i \geqslant 1$;

Note that $N_0 = V(G)$ is the unique 0-dominating set for G. For every vertex v, let $M_i(v)$ be the net-point in N_i closest to v. Note that $M_0(v) = v$ and $d_G(v, M_i(v)) = d_G(v, N_i) < 2^i$ for each $i \geqslant 0$.

For bounding the label length, we use the following fact while constructing the hierarchy of nets (see Gupta et al. [2003]).

FACT 1. *For a graph G with doubling dimension $\alpha \geqslant 1$, there is an efficiently constructible r-dominating set $W(r)$ such that, for every vertex v of G and radius $R \geqslant r > 0$, the set $B(v, R) \cap W(r)$ has size at most $(4R/r)^\alpha$. If G is unweighted and integral $r \geqslant 1$, $W(r)$ is a $(r-1)$-dominating set.*

Such a set $W(r)$ can be constructed by iteratively selecting for $W(r)$ any not yet covered vertex v and by marking as covered all vertices u such that $d_G(u, v) < r$. At the end of the process, $W(r)$ is an r-dominating set. For unweighted graph and integral $r \geqslant 1$, it is even an $(r-1)$-dominating set. Note that, by this process, vertices in $W(r)$ are pairwise at distance at least r. Let k be the integer such that $r/4 \leqslant R/2^k < r/2$, which is the smallest number of halving R to get a radius $<r/2$. Since G has doubling dimension α, applying recursively k times the definition, we have that $B(v, R)$ can be covered by at most $(2^\alpha)^k$ balls of radius $R/2^k$. Since the radius is $<r/2$ and the vertices of $W(r)$ are pairwise at distance at least r, each $B(v, R)$ contains no more than one vertex in $W(r)$ and thus $|B(v, R) \cap W(r)| \leqslant (2^k)^\alpha$. Now, $r/4 \leqslant R/2^k$ implies that $(2^k)^\alpha \leqslant (4R/r)^\alpha$, completing the proof of Fact 1.

We set $N_i = \bigcup_{j=i}^{\lceil \log n \rceil} W(2^j)$ for each $i \in \{0, \ldots, \lceil \log n \rceil\}$. It is not hard to see that this hierarchy of nets we have constructed indeed satisfies properties (1)&(2). Using Fact 1, we have the following:

LEMMA 2.2. *For a graph G with doubling dimension $\alpha \geqslant 1$, there is an efficiently constructible hierarchy of nets N_i satisfying properties* (1)&(2) *such that, for every vertex v of G, radius $R > 0$, and level $i \geqslant 0$, the set $B(v, R) \cap N_i$ has size at most $2 \cdot (4R/2^i)^\alpha$.*

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 ($F = \varnothing$). We now describe, on a high level, a variant of a distance oracle for this class based on a 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 $I = \{c, \ldots, \lceil \log n \rceil\}$ be the range of levels where $c = c(\epsilon)$ is some constant integer to be fixed later. The label $L(v)$ of each vertex v consists of all the vertices in $B(v, 2^{i+1} - 1) \cap N_{i-c}$ and their distance from v, for each $i \in I$. Note that for the lowest level $i = c$, $L(v)$ contains all the vertices of $B(v, 2^{c+1} - 1)$ since $N_{i-c} = N_0 = V(G)$.

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 $i \geqslant c$ such that

$M_{i-c}(t) \in B(s, 2^{i+1}-1)$ (extracting $M_{i-c}(t)$ from $L(t)$ and performing the check in $L(s)$). We then return the distance estimate $\delta(s,t) = d_G(s, M_{i-c}(t)) + d_G(t, M_{i-c}(t))$.

Clearly, by the triangle inequality, $\delta(s,t) \geqslant d_G(s,t)$. We now claim that setting $c = \max\{0, \lceil \log(2/\epsilon) \rceil\}$ yields the desired stretch of $1+\epsilon$; that is, $\delta(s,t) \leqslant (1+\epsilon) \cdot d_G(s,t)$. To see this, let j be the index such that $2^{j-1} < d \leqslant 2^j$. Note that $d_G(s, M_{j-c}(t)) \leqslant d_G(s,t) + d_G(t, M_{j-c}(t)) \leqslant 2^j + 2^{j-c} - 1 \leqslant 2^{j+1} - 1$. We get that, for this index j, $M_{j-c}(t) \in B(s, 2^{j+1}-1)$. In particular, the index i returned by the oracle satisfies $i \leqslant j$. By property (2), $N_j \subseteq N_i$, and thus $d_G(s, M_{i-c}(t)) \leqslant d_G(s, M_{j-c}(t))$ and $d_G(t, M_{i-c}(t)) \leqslant d_G(t, M_{j-c}(t))$. Therefore, the distance estimate $\delta(s,t)$ satisfies:

$$\begin{aligned}
d_G(s,t) \leqslant \delta(s,t) &= d_G(s, M_{i-c}(t)) + d_G(t, M_{i-c}(t)) \\
&\leqslant d_G(s, M_{j-c}(t)) + d_G(t, M_{j-c}(t)) \\
&\leqslant [d_G(s,t) + d_G(t, M_{j-c}(t))] + d_G(t, M_{j-c}(t)) \\
&< d_G(s,t) + 2 \cdot 2^{j-c} \leqslant d_G(s,t) + 2^{j+1-\log(2/\epsilon)} \\
&\leqslant d_G(s,t) + (\epsilon/2) \cdot 2^{j+1} = d_G(s,t) + \epsilon \cdot 2^j \\
&\leqslant (1+\epsilon) \cdot d_G(s,t).
\end{aligned}$$

Finally, let us bound the label length. Consider the label $L(v)$ of some vertex v. Using Lemma 2.2, for each index $i \in I$, the number of net-points added to $L(v)$ is at most $2 \cdot (4R/2^{i-c})^\alpha$ where $R < 2^{i+1}$. This is less than $2 \cdot (2^{3+c})^\alpha = 2^{3\alpha+1} \cdot 2^{\alpha c} = 2^{3\alpha+1} \cdot 2^{\alpha \max\{0, \lceil \log(2/\epsilon) \rceil\}} \leqslant 2^{3\alpha+1} \cdot \max\{2^0, 2^{\log(2/\epsilon)+1}\}^\alpha \leqslant 2^{3\alpha+1} \cdot (1 + (2/\epsilon)^\alpha) = 2^{O(\alpha)} \cdot (1 + 1/\epsilon)^\alpha$. So the total label length is at most $2^{O(\alpha)} \cdot (1+1/\epsilon)^\alpha \cdot \log^2 n = (O(1+\epsilon^{-1}))^\alpha \cdot \log^2 n$ bits, where one $\log n$ factor is due to the number $|I|$ of levels and one is due to the fact that it takes $O(\log n)$ bits to store each net-point and distance.

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 we wish 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 $L(v)$ also contains edges that do not touch v but instead 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 contains faulty vertices, we instead 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 region 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, whereas, 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 $I = \{c+1, \ldots, \lceil \log n \rceil\}$ be the range of levels where $c = c(\epsilon)$ is some constant integer to be fixed later.

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 some parameter radius r_i (to be fixed later) 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. Denote the ball of radius r_i around v by $B_i(v) = B(v, r_i)$. λ_i is a parameter (to be fixed later on) for 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 $N_{i-c-1} \cap B_i(v)$ 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:

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

—Edge weights:

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

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 $PB_i(v) = B(v, \lambda_i)$. This definition is used

in the query phase, where on level i we take into H only edges e in the labels of every $v \in \{s,t\} \cup F$ such that e is not inside any protected ball $PB_i(f)$ 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 are sufficient for checking if a vertex $x \in N_{i-c-1}$ is in the protected ball $PB_i(v)$. The label $L_i(v)$ contains all edges (v,y) such that $y \in N_{i-c-1}$ and $d_G(v,y) \leqslant \lambda_i$. Therefore, if the label $L_i(v)$ does not contain the edge (v,x), then x is not on the protected ball $PB_i(v)$.

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:

$$\begin{aligned} E(H) \;=\; & \{e = (x,y) \mid \exists i \in I \text{ such that:} \\ & \quad e \in H_i(v) \text{ for some } v \in \bar{F}, \text{ and} \\ & \quad \{x,y\} \not\subseteq PB_i(f) \text{ for every } f \in F\} \\ \cup\; & \{e = (x,y) \mid e \in H_{c+1}(v) \text{ for some} \\ & \quad v \in \bar{F}, \omega(x,y) = 1 \text{ and } x,y \notin F\}. \end{aligned}$$

—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 in 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 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 (to be fixed later). 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 because 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 earlier, 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. Toward proving the existence of such a path P', we prove Claim 1. As discussed earlier, 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 $2 \leqslant c < \lceil \log n \rceil$.

CLAIM 1. *(a)* $\lambda_i \geqslant \rho_i + \rho_{i+1} + 2^i$, *and (b)* $N_{\lceil \log n \rceil - c - 1} \subseteq B_{\lceil \log n \rceil}(v)$ *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 2.3. *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 $d_G(x, y) \leqslant \lambda_i$, and, for every $f \in F$, either x or y is not in the level-i protected ball $PB_i(f)$. Consider a vertex $f \in F$. Without loss of generality, assume that $x \notin PB_i(f)$ (i.e., $d_G(x, f) > \lambda_i$). Assume, toward 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 $G \setminus F$. □

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., $B(v, \mu_i) \cap F = \varnothing$). 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 just described is a forbidden-set distance scheme.

LEMMA 2.4. *Given* $\epsilon > 0$, *fix* $c = \max\{\lceil \log(6/\epsilon) \rceil, 2\}$. *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 $Q = (s = v_1, \ldots, v_d = t)$ be some shortest-path from s to t in $G \setminus F$. 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 $\hat{M}_{j+x(j)}$ for sufficiently large $x(j)$, where we denote $\hat{M}(v_j)$ by $\hat{M}_j$ for every $1 \leqslant j \leqslant d$. We show that the distance from $\hat{M}_j$ to $\hat{M}_{j+x(j)}$ is "close" to the distance from v_j to $v_{j+x(j)}$. We then show that by concatenating all these subpaths we get a path from s to t of length "close" to their actual distance in G. We also show that all these subpaths also exist in H. Therefore, we have a path from s to t in H of length "close" to their actual distance in G. Figure 1 illustrates this path. □

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

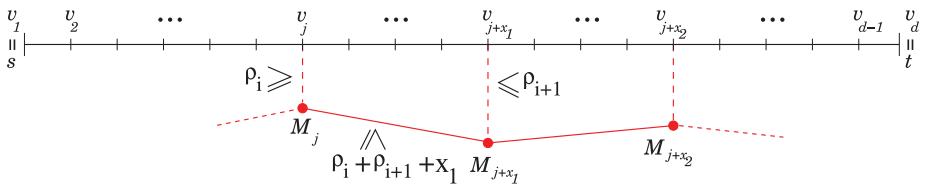

Fig. 1. Illustration of the path from s to t in H whose existence is asserted in the proof of Lemma 2.4. Here $x_1 = x(j)$ and $x_2 = x(j + x_1)$.

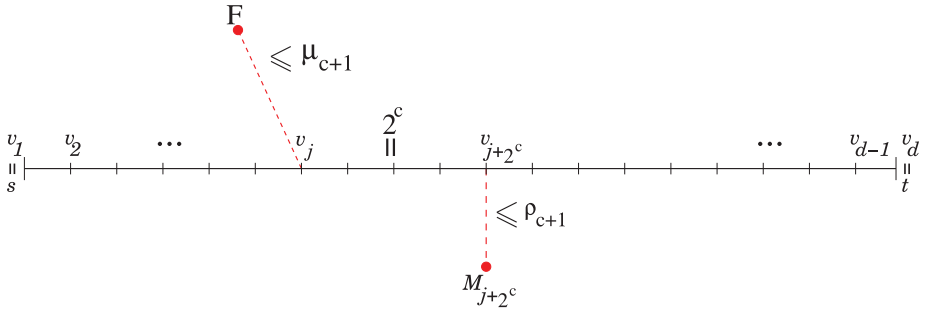

Fig. 2. The case $\ell = c$ and $\ell' = c + 1$.

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 $i(v_j)$, $\mu_\ell < \tau \leqslant \mu_{\ell+1}$. As $d_G(v_j, v_{j+2^\ell}) = 2^\ell$, we get that $\tau - 2^\ell \leqslant \tau' \leqslant \tau + 2^\ell$. This implies that $\ell - 1 \leqslant \ell' \leqslant \ell + 1$, as $\mu_{\ell-1} = \rho_{\ell-1} + \lambda_{\ell-1} = 2^{\ell-c-1} + 2^\ell < \mu_\ell - 2^\ell < \tau' \leqslant \mu_{\ell+1} + 2^\ell \leqslant \rho_{\ell+1} + \lambda_{\ell+1} + 2^\ell < \mu_{\ell+2}$.

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

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 in this case $\hat{M}_j = v_j$. To see this, recall that for nodes v that satisfy $i(v) = c$, $\hat{M}(v)$ is v itself. Since $\ell = i(v_j) = c$, we get that $\hat{M}_j = \hat{M}(v_j) = v_j$.

Moreover, note that all edges in the path $(v_j = \hat{M}_j, \ldots, v_{j+2^c})$ are added to H. To see this, observe that for all $j \leqslant r < j + 2^c$, both v_r and v_{r+1} are in $B(f', r_{c+1})$ as $d_G(v_r, f') \leqslant \mu_{c+1} + 2^c < r_c < r_{c+1}$ and also $d_G(v_{r+1}, f') \leqslant \mu_{c+1} + 2^c < r_c < r_{c+1}$, and in addition none of them belongs 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 $(v_j, \ldots, v_{j+2^c})$ 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}. Because $d_G(v_{j+2^c}, \hat{M}_{j+2^c})$ 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 $B(f', r_{c+1})$ because the distance from every vertex on that path to f' is at most $\mu_{c+1} + 2^c + \rho_{c+1} = r_c$ (see Figure 2). Moreover, as already shown, all edges in the path $(v_j = \hat{M}_j, \ldots, v_{j+2^c})$ 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 $c \geqslant \log(4/\epsilon)$, we get that $d(\hat{M}_j, \hat{M}_{j+2^c}) \leqslant (1 + \epsilon/2) d(v_j, v_{j+2^c}) = (1 + \epsilon/2) \cdot 2^\ell$.

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

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

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

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 $PB_\ell(f)$. 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 $PB_\ell(f)$ for all $f \in F$. To show 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 $\ell = \lceil \log n \rceil$. In that case, both $\hat{M}_j$ and $\hat{M}_{j+2^\ell}$ belong to $B_{\lceil \log n \rceil}(s)$. To see this, note that $\ell = \lceil \log n \rceil$ and $\ell' \geqslant \lceil \log n \rceil - 1$, so both $\hat{M}_j$ and $\hat{M}_{j+2^\ell}$ belong to $N_{\lceil \log n \rceil - c - 1} \subseteq B_{\lceil \log n \rceil}(s)$, by Claim 1(b).

The second case is when $c < \ell < \lceil \log n \rceil$. Note that by the maximality of $\ell = i(v_j)$, it must be that there exists a vertex $f \in F$ such that $d_G(v_j, f) \leqslant \mu_{\ell+1}$. Because $r_\ell = \mu_{\ell+1} + 2^\ell + \rho_{\ell+1}$, we get that both $\hat{M}_j$ and $\hat{M}_{j+2^\ell}$ 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, because $\ell' \geqslant \ell - 1$, then $\hat{M}_{j+2^\ell}$ 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 $c \geqslant \log(6/\epsilon)$, we get $d_H(\hat{M}_j, \hat{M}_{j+2^\ell}) \leqslant 2^\ell + \rho_\ell + \rho_{\ell+1} = 2^\ell + 2^{\ell-c} + 2^{\ell-c+1} = 2^\ell + 2^\ell \cdot 3 \cdot 2^{-c} = (1 + \epsilon/2) \cdot 2^\ell$. □

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 a distance query in the failure-free setting. More specifically, assume without loss of generality that $d < 2^\ell$ and that $\ell \leqslant \ell'$, and let i be such that $2^{i-1} \leqslant d \leqslant 2^i$. Note that $i \leqslant \ell$. Let $z \in N_{i-c}$ be the closest net point to t in N_{i-c}. Because $r_i = \mu_{i+1} + 2^i + \rho_{i+1}$, we get that z is in the ball $B_i(s)$. In addition, $d_G(s, F) > \mu_\ell \geqslant \mu_i = \rho_i + \lambda_i$. We get that the vertex s and the edge between s and z are not in the protective ball $PB_i(f)$ for all $f \in F$. Moreover, $d_G(s, z) \leqslant \rho_i + d \leqslant \lambda_i$. 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 $c \geqslant \log(6/\epsilon)$ 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 $\hat{M}_{1+2^\ell}$ of length at most $\rho_\ell + 2^\ell$. Note that by taking again $c \geqslant \log(6/\epsilon)$, we get $d_H(s, \hat{M}_{1+2^\ell}) \leqslant (1 + \epsilon/2) \cdot 2^\ell$. By Claim 2, H contains a path from $\hat{M}_{1+2^\ell}$ 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 $d_H(s, \hat{M}_j) \leqslant (1 + \epsilon/2)d'$.

Note that, using similar arguments as before, we get that H contains an edge between $\hat{M}_j$ to t of weight $d_H(\hat{M}_j, t) \leqslant d'' + \rho_{\ell'+1}$ (where $\ell' - 1 \leqslant i(v_j) \leqslant \ell' + 1$ 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 $d_H(s,t) \leqslant d_H(s, \hat{M}_j) + d_H(\hat{M}_j, t) \leqslant (1+\epsilon/2)d' + d'' + \rho_{\ell'+1} \leqslant \rho_{\ell'+1} + (1+\epsilon/2)d \leqslant (1+\epsilon)d$.

We finally bound the label length.

LEMMA 2.5. *The label length is* $O(1+\epsilon^{-1})^{2\alpha} \cdot \log^2 n$.

PROOF. By Lemma 2.2, for each index $c \leqslant i \leqslant \log n$, 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 $r_i = \mu_{i+1} + 2^i + \rho_{i+1} = 2\rho_{i+1} + \lambda_{i+1} + 2^i = 2^{i-c+2} + 2^{i+2} + 2^i < 2^{i+3}$ as $c \geqslant 2$. Therefore, for a specific index i, the total length added to $L(v)$ is at most $\max\{512^{2\alpha}, (1536/\epsilon)^{2\alpha}\}$, because for each pair of vertices we might store an edge. Hence, the label length is at most $\log^2 n \cdot \max\{O(1)^{2\alpha}, (O(1)/\epsilon)^{2\alpha}\} = O(1+\epsilon^{-1})^{2\alpha} \cdot \log^2 n$. □

We now bound the query time.

LEMMA 2.6. *The query time is* $O(1+\epsilon^{-1})^{2\alpha} \cdot |F|^2 \log n$.

PROOF. By the analysis of Lemma 2.5, the number of edges stored in every label is $O(1+\epsilon^{-1})^{2\alpha} \cdot \log n$. Thus, the number of edges in all labels of $\bar{F}$ is $O(1+\epsilon^{-1})^{2\alpha} \cdot |F| \log n$. For each such edge e, the algorithm checks if it is safe by checking if the edge e is not inside any protected ball $PB_i(f)$ for any $f \in F$. For every $f \in F$, this can be done in $O(1)$ time (by using perfect hashing). Hence, checking if an edge is safe takes $O(|F|)$ time, and constructing the graph H takes $O(1+\epsilon^{-1})^{2\alpha} \cdot |F|^2 \log n$ time. The shortest path distance from s to t in H can be computed in time $O(1+\epsilon^{-1})^{2\alpha} \cdot |F| \log n$. Hence, the total query time is $O(1+\epsilon^{-1})^{2\alpha} \cdot |F|^2 \log n$. □

Edge Faults. To simplify notations, the construction we describe handles 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, 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. The headers have length at most $O(|V(H)|)$ times the maximum length of a vertex name, which is usually $O(|V(H)| \log n)$ (the standard assumption is that the vertex names are of size $O(\log n)$). (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 shown:

THEOREM 2.7. *Unweighted n-vertex graphs of doubling dimension $\alpha \geqslant 1$ 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 $c \leqslant n$ is the number of connected components of the graph.

THEOREM 3.1. *Every forbidden-set connectivity labeling scheme for unweighted n-vertex graphs of doubling dimension $\alpha \geqslant 1$ 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, $\mathcal{O}_G(i, j, F) = \textit{true}$ 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 integers. We assume that d is even and $d, p \geqslant 2$. We consider the graphs $G_{p,d}$ and $H_{p,d}$, two variants of the d-dimensional grid of $p \times \cdots \times p$ vertices. The vertices of $G_{p,d}$ and $H_{p,d}$ are sequences $(x_1, \ldots, x_d)$ where $x_i \in \{0, \ldots, p-1\}$. Two vertices $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ 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 $\sum_i |x_i - y_i| \leqslant d/2$. 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 $|E(H_{p,d})| \leqslant \frac{1}{2} m_{p,d}$.

The doubling dimension of $G_{p,d}$ is $\leqslant d$ because any ball of radius $2r$, for any $r > 0$, centered at $(x_1, \ldots, x_d)$, can be covered by no more than 2^d balls of radius r centered

at the vertices $(|x_1 + c_1|, \ldots, |x_d + c_d|)$, 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 $\mathcal{F}_{n,\alpha}$.

Consider a ball B of radius $2r$ in G centered at vertex v. Let B' be the ball of radius $2r$ in $G_{p,d}$ centered at vertex v. Note that B is contained in B'. 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 $r/2$ of $G_{p,d}$. Since G contains $H_{p,d}$, G is a s-spanner of $G_{p,d}$ for some $s \leqslant 2$. In particular, any ball of radius $r/2$ in $G_{p,d}$ is contained in a ball of radius $rs \leqslant r$ of G. It follows that B (and even the ball B') is covered by at most $(2^d) \cdot (2^d) = 2^{2d} = 2^\alpha$ balls of radius r in G. Thus G, and all the graphs of $\mathcal{F}_{n,\alpha}$, have n vertices and doubling dimension $\leqslant \alpha$. The number of graphs in the family $\mathcal{F}_{n,\alpha}$ is

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

By the preceding 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 connectivity 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 $n \geqslant 4$. 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 that 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 $L(x) = L(y)$: a contradiction. Hence, there are at least $n-2$ labels.

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

ACKNOWLEDGMENTS

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

REFERENCES

Ittai Abraham, Shiri Chechik, and Cyril Gavoille. 2012. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*. 1199–1218.

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. 2011. A Hub-based labeling algorithm for shortest paths in road networks. In *10th International Conference on Experimental Algorithms (SEA)*. 230–241.

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, Ruslan Savchenko, and Renato F. Werneck. 2014. Hub labels: Theory and practice. In *Proceedings of the 13rd International Conference on Experimental Algorithms (SEA)*. 259–270.

Ittai Abraham, Amos Fiat, Andrew V. Goldberg, Ruslan Savchenko, and Renato F Werneck. 2010. Highway dimension, shortest paths, and provably efficient algorithms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 782–793.

Ittai Abraham and Cyril Gavoille. 2006. Object location using path separators. In *Proceedings of the 25th ACM Symposium on Principles of Distributed Computing (PODC)*. 188–197.

Ittai Abraham, Cyril Gavoille, Andrew V. Goldberg, and Dahlia Malkhi. 2006. Routing in networks with low doubling dimension. In *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS)*.

Aaron Bernstein and David Karger. 2009. A nearly optimal oracle for avoiding failed vertices and edges. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC)*. 101–110.

Bruno Courcelle, Cyril Gavoille, Mamadou Mustapha Kanté, and David Andrew Twigg. 2008. Connectivity check in 3-connected planar graphs with obstacles. In *Proceedings of the International Conference on Topological & Geometric Graph Theory*, vol. 31. 151–155.

Shiri Chechik. 2011. Fault-tolerant compact routing schemes for general graphs. In *Proceedings of the 38th International Coll. on Automata, Languages and Programming (ICALP)*. 101–112.

Shiri Chechik. 2013. Compact routing schemes with improved stretch. In *Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC)*. 33–41.

Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. 2010. f-sensitivity distance oracles and routing schemes. In *Proceedings of the 18th European Symposium on Algorithms (ESA)*. 84–96.

Bruno Courcelle and David Andrew Twigg. 2007. Compact forbidden-set routing. In *Proceedings of the 24th Symposium on Theoretical Aspects of Computer Science (STACS)*, LNCS 4393. 37–48.

Michael Dom. 2007. Compact routing. In *Algorithms for Sensor and Ad Hoc Networks*, LNCS 4621. 187–202.

Ran Duan and Seth Pettie. 2009. Dual-failure distance and connectivity oracles. In *Proceedings of the 20th Symposium on Discrete Algorithms (SODA)*. 506–515.

Camil Demetrescu and Mikkel Thorup. 2002. Oracles for distances avoiding a link-failure. In *Proceedings of the 13th Symposium on Discrete Algorithms (SODA)*. 838–843.

Pierre Fraigniaud, Emmanuelle Lebhar, and Laurent Viennot. 2008. The inframetric model for the internet. In *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM)*. 1085–1093.

Anupam Gupta, Robert Krauthgamer, and James R. Lee. 2003. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, 534–543, IEEE.

Cyril Gavoille and David Peleg. 2003. Compact and localized distributed data structures. *Distributed Computing* 16, 2–3, 111–120.

Cyril Gavoille, David Peleg, Stéphane Pérennès, and Ran Raz. 2004. Distance labeling in graphs. *Journal of Algorithms* 53, 1, 85–112.

Neelesh Khanna and Surender Baswana. 2010. Approximate shortest path oracle under vertex failure. In *Proceedings of the 26th Symposium on Theoretical Aspects of Computer Science (STACS)*.

Dmitri Krioukov, K. C. Claffy, Kevin Fall, and Arthur Brady. 2007. On compact routing for the internet. *ACM SIGCOMM Computer Communication Review* 37, 3, 43–52.

Dmitri Krioukov, Kevin Fall, and Xiaowei Yang. 2004. Compact routing on internet-like graphs. In *Proceedings of the 23rd Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*.

Goran Konjevod, Andréa Werneck Richa, and Donglin Xia. 2007. Optimal scale-free compact routing schemes in doubling networks. In *Proceedings of the 18th Symposium on Discrete Algorithms (SODA)*. 939–948.

Goran Konjevod, Andréa Werneck Richa, and Donglin Xia. 2008. Dynamic routing and location services in metrics of low doubling dimension. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC)*, LNCS 5218. 379–393.

David Peleg. 1999. Proximity-preserving labeling schemes and their applications. In *Proceedings of the 25th Workshop on Graph- Theoretic Concepts in Computer Science*. 30–41.

C. Greg Plaxton, Rajmohan Rajaraman, and Andréa Werneck Richa. 1997. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 311–320.

Mihai Pătraşcu and Mikkel Thorup. 2007. Planning for fast connectivity updates. In *Proceedings of the 48th IEEE Symposium on Foundations of Computer Science (FOCS)*. 263–271.

Aleksandrs Slivkins. 2005. Distance estimation and object location via rings of neighbors. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC)*. 41–50.

Aleksandrs Slivkins. 2007. Distance estimation and object location via rings of neighbors. *Distributed Computing* 19, 4, 313–333.

Kunal Talwar. 2004. Bypassing the embedding: Algorithms for low dimensional metrics. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*. 281–290.

Mikkel Thorup. 2004. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM* 51, 6, 993–1024.

Mikkel Thorup and Uri Zwick. 2001. Compact routing schemes. In *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 1–10.

David Andrew Twigg. 2006. *Forbidden-set Routing*. PhD thesis, University of Cambridge (King's College).

Received June 2014; revised June 2015; accepted June 2015