Routing in Networks with Low Doubling Dimension

Ittai Abraham[1] Cyril Gavoille[2] Andrew V. Goldberg[3] Dahlia Malkhi[4]

December 2005

Technical Report
MSR-TR-2005-175

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

[1]School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel. ittaia@cs.huji.ac.il

[2]Laboratoire Bordelais de Recherche en Informatique, University of Bordeaux, Bordeaux, France. gavoille@labri.fr

[3]Microsoft Research – Silicon Valley, 1065 La Avenida, Mt. View, CA 94043, USA. goldberg@microsoft.com

[4]School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel, and Microsoft Research – Silicon Valley, 1065 La Avenida, Mt. View, CA 94043, USA. dalia@microsoft.com

Abstract

This paper studies compact routing schemes for networks with low doubling dimension. Two variants are explored, name-independent routing and labelled routing. The key results obtained for this model are the following. First, we provide the first name-independent solution. Specifically, we achieve constant stretch and polylogarithmic storage. Second, we obtain the first truly *scale-free* solutions, namely, the network's aspect ratio is not a factor in the stretch. Scale-free schemes are given for three problem models: name-independent routing on graphs, labelled routing on metric spaces, and labelled routing on graphs. Third, we prove a lower bound requiring linear storage for stretch < 3 schemes. This has the important ramification of separating for the first time the name-independent problem model from the labelled model, since compact stretch-1 $+ \varepsilon$ labelled schemes are known to be possible.

1 Introduction

In this paper we study compact routing schemes for graphs with bounded doubling dimension. In compact routing schemes the goal is to achieve efficient tradeoffs between the *storage*, the maximal number of bits per node, and *stretch*, the maximal ratio between the route and the shortest path over all source destination pairs. Two variants of the problem exists. In the *labelled* model, the designer is allowed to assign nodes with short (typically polylogarithmic) labels that can be used for routing. In the *name-independent* model, the node labels are decided by an adversary. While this model makes the routing task much harder it also enables important network operations like locating nearby copies of replicated objects and tracking of mobile objects [9, 6].

For example, consider the hyper-cube graph $\{0,1\}^{\log n}$. In a labelled scheme, the designer may give each node its natural $\log n$ bit identifier and then stretch 1 routing becomes trivial. The name-independent version is much harder: Suppose an adversary maps nodes to names via a random permutation of $[1...n]$. Intuitively, given a target label, a source node needs to discover the location of the target at a cost that is competitive with the distance between the source and target (which is unknown in advance).

For arbitrary n node graphs, routing schemes with stretch $\approx k$ that require $\approx n^{1/k}$ bits per node are known, for name-independent [9, 10, 1] and labelled [25] models. These results (for both models) are asymptotically tight up to polylogarithmic factors. We refer the reader to Peleg's book [20] and to the surveys of Gavoille and Peleg [13, 14] for more detail and background. Although intuitively the name-independent variant appears harder than the labelled one, until now the same lower bounds hold for both problem models, and moreover the same upper bounds were known up to polylogarithmic factors in storage and constant factors in stretch.

An easier problem than that of routing on a graph is that of routing on a metric space. A metric space is set V and a non-negative, symmetric distance function $d : V \times V \to \mathbb{R}$ that obeys the triangle inequality. In this model the designer needs first to choose for each node a set of outgoing links, this then induces a directed overlay graph on which routing must occur as in the graph model. In addition to minimizing the storage, the goal is also to minimize the number of outgoing links. See [17, 4, 21] for routing schemes on this model.

Another aspect of routing schemes is their dependance on the scale of the network. Let the *aspect ratio* Δ be the ratio between the largest distance and the smallest distance, then many schemes require memory that tends to infinity as Δ increases. One would hope to remove the

dependence on Δ altogether. We will say that a routing scheme is *scale-free* if its memory requirement is independent of the aspect ratio. Obtaining scale-free schemes is a challenging goal: Until now, scale-free compact routing schemes were known only for a fairly restricted class of graphs [5] or with exponential stretch [7, 8].

In recent years, several problems, whose input contains a metric space, were shown to have considerably better solutions when the doubling dimension is low. For example metric space embedding [16, 18], nearest neighbor [19], distance estimation [23, 22], compact routing [21, 11]. The *doubling dimension* of a metric space is the smallest $\alpha > 0$ such that every ball of radius $2r$ can be covered by 2^α balls of radius r. This notion naturally extends to undirected weighted graphs: a graph has doubling dimension α if the metric space induced by the shortest path distances on the graph has doubling dimension α. Yet no name-independent routing scheme on low dimensional spaces was ever given, nor any labelled scale-free scheme provided for low dimensional graphs.

1.1 Our contribution

This paper provides a number of contributions on the topic of compact routing in low-dimensional networks.

- In Section 2 we give the first results on name-independent routing in graphs with low doubling dimension. Our basic scheme achieves *constant* stretch while requiring only $2^{O(\alpha)} \log \Delta \log n$ bits per node.
- We then extend this scheme in Section 3 to provide a constant stretch scale-free version requiring only $2^{O(\alpha)} \log^4 n$ bits. This is the first scale-free compact routing scheme for low-dimensional networks.
- In Section 4, we prove that any name-independent scheme with $o(\alpha n)$ (sub-linear) bits of storage must have a stretch of at least $3-\epsilon$. In particular, this implies that there is no $1+\epsilon$ name-independent scheme with sublinear storage. Thus, we separate for the first time the name-independent and the labelled problem models, since $1+\epsilon$ compact labelled routing is achievable.
- For the labelled case in metric spaces, in Section 5 we provide improvements over the best known stretch $1+\epsilon$ schemes. First, we achieve the optimal $\lceil \log n \rceil$ label size for graphs while requiring lowest known memory. All previous schemes required label sizes which depend on $\alpha, \log \Delta, \log(\frac{1}{\epsilon})$. Second, we give the first truly scale-free scheme on metrics, which also has optimal label size. The best previous scheme still required a $\log \log \Delta$ storage factor and bigger labels.
- In Section 6 we present a stretch $1+\varepsilon$ scale-free labelled scheme for graphs. We are not aware of any previous scale-free schemes for graphs.
- Finally, in Section 7 we provide an additional lower bound on exact (stretch-1) compact routing schemes. It proves that even for the very restricted class of growth bounded graphs, $\Omega(\sqrt{n})$ memory is required at some nodes. This demonstrates the benefits of allowing ϵ slack in the routing stretch.

Table 1 summarizes the best known bounds and this paper's contribution in the different models.

Reference	Model	Stretch	Memory (in bits)	Label/Header (in bits)	Out degree
Awerbuch & Peleg [9]	G, NI	$O(\log^2 n)$	$O(\log \Delta \log^2 n)$	$O(\log n)$	
This paper (Thm. 1)	G, NI	$O(1)$	$2^{O(\alpha)} \log \Delta \log n$	$O(\log n)$	
This paper (Thm. 2)	G, NI	$O(1)$	$2^{O(\alpha)} \log^4 n$	$2^{O(\alpha)} \log^3 n$	
Talwar [23]	G, L	$1+\varepsilon$	$O(\frac{1}{\alpha\epsilon})^{\alpha} \log^{2+\alpha} \Delta$	$O(\alpha \log \Delta)$	
Chan et al. [11]	G, L	$1+\varepsilon$	$(\frac{\alpha}{\varepsilon})^{O(\alpha)} \log \Delta \log n$	$O(\alpha \log \frac{1}{\epsilon} \log \Delta)$	
Slivkins [21] (Thm. 2.1)	G, L	$1+\varepsilon$	$(\frac{1}{\varepsilon})^{O(\alpha)} \log \Delta \log n$	$O(\alpha \log \frac{1}{\epsilon} \log \Delta)$	
Slivkins [21] (Thm. 4.1)	G, L	$1+\varepsilon$	$(\frac{1}{\varepsilon})^{O(\alpha)} \log \Delta \log n \log\log n$	$2^{O(\alpha)} \log n \log(\frac{1}{\epsilon} \log \Delta)$	
This paper (Thm. 4)	G, L	$1+\varepsilon$	$(\frac{1}{\varepsilon})^{O(\alpha)} \log \Delta \log n$	$\lceil \log n \rceil$	
Slivkins [21] (Thm. 4.2)	M, L	$1+\varepsilon$	$(\frac{1}{\varepsilon})^{O(\alpha)} \log n \log\log n \log\log \Delta$	$O(\alpha \log n \log(\frac{1}{\epsilon} \log \Delta))$	$(\frac{1}{\varepsilon})^{O(\alpha)} \log n$
This paper (Thm. 5)	M, L	$1+\varepsilon$	$(\frac{1}{\varepsilon})^{O(\alpha)} \log^2 n$	$\lceil \log n \rceil$	$(\frac{1}{\varepsilon})^{O(\alpha)} \log n$
This paper (Thm. 6)	G, L	$1+\varepsilon$	$(\frac{1}{\varepsilon})^{O(\alpha)} \log^3 n$	$2^{O(\alpha)} \log^3 n$	

Table 1: Our results and comparison with previous results for routing schemes on n nodes of doubling dimension α, aspect ratio Δ and constant $\varepsilon > 0$. Model notation: G for graph, M for metric space, NI for name-independent, L for labelled

1.2 New Techniques

Our solutions employ a variety of novel techniques that are of interest in their own right.

In order to achieve scale-freedom in graph routing (both labelled and name-independent), we use a new sparse-dense decomposition based on [3] that combines scale-based partitions with node-count based partitions. Only doubling the scales may not be scale-free and storing only the $O(\log n)$ scales that have a density change, like the measured decent decomposition [18], leads to $\Omega(\log n)$ stretch. To get constant stretch and polylogarithmic memory, our decomposition at least doubles both the scale *and* the node count at each level. It uses different routing schemes for each level depending on whether it is sparse or dense.

Through all of our constructions, a useful building block we designed for low doubling graphs is a hierarchical network that grows gradually more dense. This is used for spanning low-degree low-diameter trees.

For the labelled case we propose a new labelling scheme based on hierarchical nets and a natural DFS enumeration, and combine this with the ring of neighbor scheme of [21]. This produces optimal $\lceil \log n \rceil$-size labels.

1.3 Model and notation

Given is a weighted graph $G = (V, E, \omega)$ of size $n = |V|$ with a non-negative weight function $\omega : E \to \mathbb{R}^+$. Let the cost of a path be the sum of the weights of its edges. For any $u, v \in V$ let $d(u, v)$ be the cost of a minimum cost path between u and v, observe that (V, d) is a metric space. Let Δ denote the *aspect ratio* (normalized diameter) of G, $\Delta = \max_{u \neq v} d(u, v) / \min_{u \neq v} d(u, v)$. In order to avoid dragging a normalization constant, from here on assume that $\min_{u \neq v} d(u, v) = 1$.

Define the radius r ball around node u, $B(u, r)$, as the set of nodes whose distance is at most r from u, $B(u, r) = \{v \mid d(u, v) \leq r\}$. For any node u, let $T(u)$ denote a minimum cost path spanning tree rooted at u. Given a lexicographic order on the nodes, for any node $u \in V$, set $Z \subseteq V$, and integer $m > 0$ define $N(u, m, Z)$ as the m closest nodes from Z to node u, i.e., as the set $N(u, m, Z) = N$ such that $N \subseteq Z$, $|N| = m$ and for all $x \in N$ and $y \in Z \setminus N$ either

$d(u,x) < d(u,y)$ or $d(u,x) = d(u,y)$ and x is lexicographically smaller than y. Let I denote the set $I = \{0, 1, \ldots, \lceil \log \Delta \rceil\}$. Let K denote the level set $K = \{0, 1, 2, \ldots, \lceil \log n \rceil\}$. Each node has an arbitrary unique network identifier consisting of polylog(n) bits. Using standard hashing techniques it is possible to generalize the model and assume nodes have arbitrarily long unique labels.

Given two sets $U \subseteq V$, we define the following. U in an *r-cover* of V if $\forall v \in V, \exists u \in U$ such that $d(u,v) \leq r$. U is a *minimum r-cover* of V if U is an r-cover of V and for any other r-cover W, $|U| \leq |W|$. U is *r-independent* if $\forall u, v \in U : d(u,v) > r$. U is an *r-net* of V if U is r-independent and is an r-cover of V.

2 A Simple Name-Independent Routing Scheme

We begin by observing that low doubling dimension spaces have efficient sparse covers.

Lemma 1. *For every weighted undirected graph $G = (V, E, \omega)$, $|V| = n$, with doubling dimension α and integer $\rho \geq 1$, there exists a polynomial algorithm that constructs a collection of sets $\mathcal{C}_\rho(G) \subset 2^V$ such that:*

1. *(Cover) For all $v \in V$, there exists $C \in \mathcal{C}_\rho(G)$ such that $B(v, \rho) \subseteq C$.*
2. *(Sparse) For all $v \in V$, $|\{C \in \mathcal{C}_\rho(G) \mid v \in C\}| \leq 4^\alpha$.*
3. *(Small radius) For all $C \in \mathcal{C}_\rho(G)$, $\mathrm{rad}(C) \leq 2\rho$, where $\mathrm{rad}(C) = \min\{r \mid \exists u \in C : C \subset B(u,r)\}$.*

Proof. Let U be a ρ-net of G, choosing U can be done in a simple greedy manner. Define $\mathcal{C}_\rho(G) = \{B(u, 2\rho) \mid u \in U\}$. Property 1 follows since for any v fix $u \in U$ such that $d(u,v) \leq \rho$, so $B(v,\rho) \subseteq B(u, 2\rho)$. Property 3 follows by definition of $\mathcal{C}_\rho(G)$. For Property 2, fix $v \in V$, let W be a minimum $\rho/2$-cover of $B(v, 2\rho)$, then $|W| \leq 4^\alpha$ and $|U \cap B(v, 2\rho)| \leq |W|$ because each ball in W contains at most one point from the ρ-net U. □

Note by comparison that for general spaces, the seminal sparse covers of Awerbuch and Peleg [9] provide $O(n^{1/k})$ sparsity with cover radius $k\rho$, for any chosen integer parameter $k \geq 1$. Thus, the low doubling dimension allows us to substantially improve the partition parameters.

For each $i \in I$ let $\mathcal{C}_{2^i}(G)$ be a set of clusters as in Lemma 1. For each $i \in I$ and $C \in \mathcal{C}_{2^i}(G)$ we apply the following lemma to construct a low degree, low diameter tree $T(C)$ that will be used for routing. Note that $T(C)$ is not necessarily a subgraph of G: it may use edges that are not in G.

Lemma 2. *Given a set $C \subset V$ with $\mathrm{rad}(C) \leq \rho$, there exists a weighted tree $T(C)$ whose nodes are C with the following two properties:*

1. *Each node has out degree at most 4^α.*
2. *Let the weight of edge (u,v) be $d_G(u,v)$. Then the weighted diameter of $T(C)$ is at most 2ρ.*

Proof. Let $U_0 = \{r\}$ be a set containing the center of the cluster C. For $i = 1$ to $\lceil \log \rho \rceil$ let U_i be a $(2^{-i}\rho)$-net of $C \setminus \cup_{0 \leq j < i} U_j$. The tree is formed by connecting each $v \in U_i$ to the closest node in U_{i-1}, denoted $p(v)$. Note that no cycles are created since if $u \in U_i$ it will never be in U_j for $j > i$.

For any $u \in U_{i-1}$ if $p(v) = u$ then $v \in B(u, 2^{-(i-1)}\rho) \cap U_i$. Since U_i is a subset of an $(2^{-i}\rho)$-net then $|B(u, 2^{-(i-1)}\rho) \cap U_i| \leq |W|$ where W is a minimum $(2^{-(i+1)}\rho)$-cover of $B(u, 2^{-(i-1)}\rho)$. Hence $|\{v \mid p(v) = u\}| \leq |W| \leq 4^\alpha$. For the diameter of the tree note that any path from the root is bounded by $\sum_{0 \leq i} 2^{-i}\rho \leq 2\rho$. □

Given such a tree we show an efficient routing scheme.

Lemma 3. *Given a weighted undirected tree $T = (V, E, \omega)$ with root r, $|V| = n$, diameter ρ and maximum out-degree k, there exists a name-independent tree-routing scheme on T with error-reporting that routes on paths of length bounded by 4ρ, each node requires $O(k \log n)$ memory bits and headers are of length $O(\log n)$. Moreover, routing for a non-existent name in T also incurs a (closed) path of length 4ρ until a negative result is reported back to the source.*

Proof. Let $d(v)$ be a depth-first search post-order enumeration of the nodes in T starting from the root. Let $w_1 < w_2 < \cdots < w_n$ be a sorted sequence of the node names (lexicographical order). For any node, let $M(v)$ denote the sequence $\langle d(v), w_{d(v)} \rangle$. Each node v stores $M(v)$ and $M(u)$ for all u that are direct children of v. In addition each node v stores $d(w_{d(v)})$. Observe that this storage requires $O(k \log n)$ bits. Given a name $w = w_i$, it costs at most ρ to get to the root. Then, using $M(\cdot)$ information stored at each node, it is possible to navigate down the tree to the node v such that $d(v) = i$, this costs at most ρ. Specifically, navigation is done by comparing for each child t its stored $w_{d(t)}$ to w; and proceed to the child with largest $w_{d(t)}$ which does not surpass w.

Since v will store $d(w_i)$ it is possible to return to the root (at cost ρ) and route to w using $d(v)$ with an additional cost of ρ. If the name does no exist then it will cost 2ρ to reach the root and back to the source. □

The remaining obstacle is that $T(C)$ is not a subgraph of G. So each node $v \in T(C)$ needs to be able to route to its parent and each of its children in $T(C)$. This is done by storing for each such connection the label of a labelled $(1+\epsilon)$-stretch routing scheme for a fixed $\epsilon < 1$. This requires a node in a tree to store routing information for at most 4^α of its children. For the labelled scheme we can employ the $(1+\epsilon)$-stretch scheme due to Slivkins [21], or the improved labelled scheme we introduce later in this paper in Section 5.

Storage. The storage per node is as follows. For each $i \in I$, a node belongs to at most 4^α clusters. For each cluster a node maintains 4^α connections. Each connection requires labels of size $O(\log n)$ for maintaining $M(\cdot)$. The total is $(\log \Delta)4^{2\alpha}O(\log n)$. In addition each node participates in a labelled routing scheme with $\epsilon = 1/4$ at cost $2^{O(\alpha)} \log \Delta \log n$. The total is $2^{O(\alpha)} \log \Delta \log n$.

Routing. Given a source s, for each $i \in I$, let $C_i \in \mathcal{C}_{2^i}(G)$ be the cluster such that $B(s, 2^i) \subseteq C_i$. Routing for a target t is done iteratively by searching for t on C_i using the scheme of Lemma 3 on the tree of Lemma 2 that spans C_i.

Stretch. For the index i such that $2^{i-1} < d(s,t) \leq 2^i$, s will find t at the i^{th} phase. The cost of a phase j is as follows: It cost at most 2^j to reach the tree root of C_j. The diameter of C_j is bounded by 2^{j+1} and the diameter of the spanning tree of Lemma 2 is bounded by 2^{j+2}. Searching

for t from the cluster root costs stretch $5/4 < 2$ for labelled routing times 2^{j+2} until the label of the target is found, for a total cost of at most 2^{j+3}. Then factor of $5/4 < 2$ for labelled routing and distance at most 2^{j+1} due to cluster diameter for reaching the source or target. The total path is $2^{j+3} + 2^j + 2^{j+2} \leq 2^{j+4}$.

Summing on all levels from 0 to i we get $\sum_{j=0}^{i} 2^{j+4} = 2^{j+5} - 1 < 2^6 d(s,t)$. So the stretch is at most $2^6 = 64$.

Theorem 1. *For each weighted n-node graph with doubling dimension α and aspect ratio Δ, there is a polynomial time constructible name-independent routing scheme with stretch factor 64 that uses $2^{O(\alpha)} \log n \log \Delta$-bit routing tables per node.*

3 A Scale-Free Name-Independent Routing Scheme

We begin with some modifications to Lemma 2 and Lemma 3 in order to make the construction scale-free. For Lemma 2, instead of building the tree by iterating i from 1 to $\lceil \log \rho \rceil$ we iterate only to $\max\{\lceil \rho \rceil, \lceil 3 \log n \rceil\}$. Observe that some nodes may not belong to the tree yet. For each $u \in U_{3 \log n}$ let $C(u)$ be the set of nodes in $B(u, 2^{\rho - 3 \log n})$ whose closest node from $U_{3 \log n}$ is u. Hence $G[C(u)]$ is connected so we build a spanning tree of $C(u)$ rooted at u. Note that now a node can belong to at most one X_i and to at most one $C(u)$, hence a real node simulates two nodes in the tree.

Since the diameter of each $C(u)$ is smaller than $n^{-2}\rho$ we modify Lemma 3 as follows: while searching for the label of the target, if we reach a node in $X_{3 \log n}$ that has some subtree we will simply visit all nodes in that subtree. For this we use the following scheme on the tree.

Lemma 4. **[12, 25]** *For every weighted tree T with n nodes there exists a labelled routing scheme that, given any destination label, routes optimally on T from any source to the destination. The storage per node in T, the label size, and the header size are $O(\log^2 n / \log \log n)$ bits. Given the information of a node and the label of the destination, routing decisions take constant time.*

Given an enumeration of the nodes in $C(u)$ each node stores the label of the next node in the enumeration, hence it is possible to visit all nodes in $C(u)$ at cost of at most ρ, while requiring each node to store $O(\log^2 n)$ bits.

Finally we replace the non scale-free labelled scheme used to route between nodes in the virtual tree with the scale-free version described in Section 6 with $\epsilon = 1/4$.

Summing up the changes, for each tree, a nodes needs to store $2^{O(\alpha)} \log^3 n$ bits per cluster it belongs to.

3.1 Sparse-dense decomposition

We use the decomposition into dense and sparse neighborhoods from [3] with parameter $k = \lceil \log n \rceil$. Let $K = \{0, 1, 2, \ldots, \lceil \log n \rceil\}$. For all $u \in V$ and $i \in K$ define the *range* $a(u,i)$ recursively as follows. Define $a(u,0) = 0$. Then recursively define $a(u, i+1)$ as the smallest positive integer $j > a(u,i)$ such that $|B(u, 2^j)| \geq 2|B(u, 2^{a(u,i)})|$ (or let $a(u, i+1) = \log \Delta$ if there does not exist such an integer). For all $u \in V$ and $i \in K$ denote the *neighborhood* ball $A(u,i)$ as the ball of radius $2^{a(u,i)}$ around u. Formally, $A(u,i) = \{u\}$ for $i = 0$ and $A(u,i) = B(u, 2^{a(u,i)})$ for $i > 0$.

Intuitively, if the gap between $a(u,i)$ and $a(u,i+1)$ is small, then the neighborhood $A(u,i+1)$ is "dense" relative to the neighborhood $A(u,i)$, otherwise $A(u,i+1)$ is "sparse" relative to $A(u,i)$. A central definition capturing this intuitive notion is the following.

Let Φ be a integer parameter. For this section, let $\Phi = 5$.

Definition 1 (Dense level)**.** *For $u \in V$ and $i \in K$, we say that i is a* dense level *for node u if*

$$a(u,i) < a(u,i+1) \leq a(u,i) + \Phi$$

We say that i is a *sparse level* for node u if it is not a dense level. In words, in a dense level, we find at least twice as many nodes as the current level by looking at a ball whose radius is at most 2^{Φ} times the current level.

The high level view of the routing scheme a simple iterative protocol. For phases $i = 1$ to k, search for v as follows: If $A(u,i)$ is sparse, use the sparse neighborhood routing strategy. If $A(u,i)$ is dense, use the dense neighborhood routing strategy.

3.2 Dense levels

For every $u \in V$ define the *range set of node u*, denoted $L(u)$, as $L(u) = \{a(u,i) \mid i \in K\}$ and define the *extended range set* $R(u)$ as,

$$R(u) = \{j \in I \mid \exists a \in L(u) \text{ s.t. } -1 \leq a - j \leq \Phi + 1\} .$$

Define $F(u,i) = B(u, 2^{a(u,i)}/2)$. The main property of dense levels is captured in the following lemma.

Lemma 5 (Dense neighborhoods)**.** *[3] If i is a dense level for u and $v \in F(u,i)$ then $a(u,i) \in R(v)$.*

Proof. Let $v \in F(u,i) = B(u, 2^{a(u,i)-1})$, then $B(v, 2^{a(u,i)-1}) \subseteq A(u,i)$ and hence $|B(v, 2^{a(u,i)-1})| \leq |A(u,i)|$.

Since $a(u,i+1) \leq a(u,i) + \Phi$, then $B(v, 2^{a(u,i)+\Phi+1}) \supseteq A(u,i+1)$, and hence $|B(v, 2^{a(u,i)+\Phi+1})| \geq |A(u,i+1)| \geq 2|A(u,i)|$.

Together, these imply $|B(v, 2^{a(u,i)+\Phi+1})| \geq 2|B(v, 2^{a(u,i)-1})|$. Therefore, there exists some index $a(v,j)$ such that $a(u,i) - 1 \leq a(v,j) \leq a(u,i) + \Phi + 1$. □

For every $i \in I$ define $G_i = (V_i, E_i)$ as the subgraph induced by the nodes $V_i = \{u \mid i \in R(u)\}$. For each G_i we construct the routing scheme for scale 2^i of Section 2. Specifically we build the sparse cover of Lemma 1 of scale 2^i and on each cluster we build the virtual tree and routing scheme as detailed in the beginning of Section 3. Note that G_i may have several connected components, in which case we construct the routing scheme for each such component separately.

Storage. Each node u stores routing scheme for scale 2^i for each $i \in R(u)$. For each such scale, a node stores $2^{O(\alpha)} \log^3 n$ bits.

Routing and Cost. Routing on a dense level i of node u is done by searching for the target on the virtual tree spanning the cluster that contains the ball $B_{G_{a(u,i)}}(u, 2^{a(u,i)})$. The cost of searching on this cluster is bounded by $O(2^{a(u,i)})$.

3.3 Sparse levels

For sparse levels we construct a collection of landmarks $\{C_i \subset V \mid i \in K\}$ due to Slivkins [21]. The following lemma follows from [21](Lemma 3.1). For completeness, we provide a proof of this special case here. For any $v \in V$ and $i \in K$ let $r(u,i)$ be the radius of $N(v, 2^i)$.

Lemma 6. *For every $i \in K$ there exists a set of landmarks $C_i \subseteq V$ such that:*
(1) $d(u, C_i) \leq 5r(u,i)$, for every node u.
(2) If $|B(u,r)| \leq 2^j$ then $|B(u, r/2) \cap C_i| \leq 2^{2\alpha+j-i} + 1$

Proof. Fix some $i \in K$. For each $u \in V$, denote $N_0 = N(u, 2^i)$. From here on, for any ball B, denote by B' the ball obtained by blowing up B's radius twice.

We define a mapping from u to balls $B_u \subseteq B(u, 2r(u,i))$, with the following property: B_u has at least $2^i/2^{2\alpha}$ nodes, and blowing up B_u's radius twice results in a ball that contains at most 2^i nodes. This is possible by the low dimensionality, as follows. There exists a cover of N_0 with balls $N_{0,1}, \ldots, N_{0,2^{2\alpha}}$ whose radius is $r/4$. One ball contains $2^i/2^{2\alpha}$ nodes, denote it N_1, and accordingly N_1' the ball with twice its radius. If N_1' covers less than 2^i nodes, we are done. Otherwise, continue recursively with N_1'. Since we reduce the radius to a half at each step of the recursion, the process must end with a ball N_j with the desired property. Since each ball N_i intersects N_{i-1}', and radii decrease by a factor of two on each element, the final ball satisfies $N_j' \subseteq N_0'$.

We now take a maximal set $\mathcal{F}_i$ of non-intersecting balls B_u, and define C_i to be the set of their center points. First, by counting, a ball $B(s,r)$ with 2^j nodes can trivially contain at most $2^{j-(i-2\alpha)}$ balls from $\mathcal{F}$. Second, since $\mathcal{F}_i$ contains disjoint balls B_u, there can be at most one ball B_u whose center is inside $B(s, r/2)$ and whose radius is $r/2$. Hence, all but one center from C_i inside $B(u, r/2)$ belong to balls that are fully contained in $B(u,r)$. Property (ii) follows.

In order to show (i), fix any node u. If $B_u \in \mathcal{F}_i$, clearly $B(u, 2r(u,i))$ contains the center, and we are done. Otherwise, B_u intersects some other ball $B_v \in \mathcal{F}_i$. Since B_v' contains at most 2^i nodes, it does not strictly contain $B(u, r(u,i)) = N(u, 2^i)$. But since B_v and B_u intersect, and $B_u \subseteq B(u, 2r(u,i))$, we have $\mathrm{rad}(B_v) \leq 3r(u,i)$. Hence, the ball $B(u, 2r(u,i) + \mathrm{rad}(B_v)) \subseteq B(u, 5r(u,i))$ contains the center v, as required. □

For every $u \in V$ and $i \in K$ define the *nearby landmarks* $S(u,i)$ to be the $2^{2\alpha+2}+1$ closest nodes in C_i.

$$S(u,i) = N(u, 2^{2\alpha+2} + 1, C_i)$$

and define $S(u) = \bigcup_{i \in K} S(u,i)$.

Define the *center* $c(u,i)$ as the closest node to u from C_i. and define $C(u) = \bigcup_{i \in K} c(u,i)$.

Let $\Gamma = 4$. Let $E(u,i) = B(u, 2^{a(u,i+1)-\Gamma})$.

The main property of sparse levels is captured in the following Lemma.

Lemma 7 (Sparse neighborhoods)**.** *Let i be a sparse level for u, i.e., $a(u,i+1) > a(u,i) + \Phi$. If $v \in E(u,i)$ then $C(u) \cap S(v) \neq \varnothing$.*

Proof. Denote $a = a(u,i+1)$. Let j be the index such that $2^j \leq |B(u, 2^{a(u,i)}| < 2^{j+1}$ hence $|B(u, 2^{a-\Phi})| \geq 2^j$ so $N(u,2^j) \subseteq B(u, 2^{a-\Phi})$. So $d(u, C_j) \leq 5 \cdot 2^{a-\Phi}$.

We now show that $c(u,j)$ belongs to $S(v,j)$. We do this by constructing a ball B_1 around v that contains $B(u, 5 \cdot 2^{a-\Phi})$ on the one hand, and on the other hand has fewer than $2^{2\alpha+2}$ landmarks from C_j.

For B_1 to contain sufficiently few landmarks, we start with $B_2 = B(v, 2^{a-1} - 2^{a-\Gamma})$, hence $B_2 \subseteq B(u, 2^{a-1})$ so $|B_2| \leq 2^{j+2}$, and by Lemma 6, $B(v, 2^{a-2} - 2^{a-\Gamma-1})$ contains at most $2^{2\alpha+2}+1$ landmarks from C_j. Accordingly, we choose $B_1 \subseteq B(v, 2^{a-2} - 2^{a-\Gamma-1})$.

For $B_1 = B(v,r)$ to contain $B(u, 5 \cdot 2^{a-\Phi})$, it must have radius $r \geq 5 \cdot 2^{a-\Phi} + 2^{a-\Gamma}$. So we will choose $B_1 = B(v, 5 \cdot 2^{a-\Phi} + 2^{a-\Gamma})$.

Hence we need $5 \cdot 2^{a-\Phi} + 2^{a-\Gamma} \leq 2^{a-2} - 2^{a-\Gamma-1}$ which is true when $\Gamma \geq 4$ and $\Phi \geq 5$.

Therefore $c(u,j) \in S(v,j)$ and the lemma follows. □

So if $a(u,i)$ is a sparse level let $J(i)$ be the index such that $2^{J(i)} \leq |B(u, 2^{a(u,i)}| < 2^{J(i)+1}$. The routing scheme uses the following single source scheme.

3.3.1 Single-source routing

The following single-source routing scheme extends Lemma 3 to give constant stretch with poly-logarithmic memory bound. We actually need a stronger result that gives either constant stretch or a predetermined error cost:

Lemma 8. *For any graph $G = (V, E, \omega)$ with doubling dimension α, $|V| = n$, and for any designated root $r \in V$, there exists a name-independent error-reporting single-source routing scheme with the following properties:*

1. *Each node stores $2^{O(\alpha)} \log^3 n$ bits of routing information.*
2. *There exist $O(\log n)$ sub-clusters $X_1 \subset \cdots \subset X_y = V$ such that the following holds for every i: $G[X_i]$ is connected, contains r, $\operatorname{diam}(X_i) \geq 2 \operatorname{diam}(X_{i-1})$, and $|X_i| \geq 2 |X_{i-1}|$.*
3. *For any $j \in \{0, \ldots, y\}$, the root can perform a j-bounded search for destination v. A j-bounded search for v has the following properties: If $v \in X_j$ then it reaches v with stretch $O(1)$; Otherwise it returns a negative response to the root incurring a cost of at most $O(\operatorname{diam}(X_{j-1}))$.*

Proof. Given a cluster X with $|X| = n$ and a root $r \in X$ we build $O(\log n)$ sub-clusters $X_1 \subset \cdots \subset X_y$ in the following manner. Let $X_1 = \{r\}$ and recursively define X_{i+1} as the minimal ball around r such that $|X_{i+1}| \geq 2|X_i|$ and $\operatorname{diam}(X_{i+1}) \geq 2 \operatorname{diam}(X_i)$. If there does not exist such a cluster then set $X_{i+1} = V$ and $y = i+1$. If $|X_{i+1}| = 2|X_i|$ then we say X_{i+1} is *sparse*, otherwise X_{i+1} is *dense*. Note that if X_{i+1} is dense then $\operatorname{diam}(X_{i+1}) = 2 \operatorname{diam}(X_i)$ (unless $y = i+1$, in which case $\operatorname{diam}(X_{i+1}) \leq 2 \operatorname{diam}(X_i)$). We build the scheme of [12, 25] (see Lemma 4) on a shortest path tree rooted at r spanning x, let Λ denote this scheme. For every X_i we define a tree T_i, if X_i is sparse then T_i is the virtual tree on X_{i-1} as defined by Lemma 2 with the modification

that the labels stored by nodes are of the Λ scheme and that that T_i on X_{i-1} stores all the labels of the Λ scheme of nodes in X_i, hence each node must store at most two labels. Otherwise let T_i be the tree of Lemma 2 on X_i, and build the regular scheme from Lemma 3 while storing labels of scheme Λ. Since $y = O(\log n)$, any node can belong to at most $O(\log n)$ trees and each tree requires $O(4^{\alpha} \log^2 n)$ bits per node.

A j-bounded search performs a search from r for t iteratively in $T_1, T_2, \ldots, T_{j-1}$, and if T_j is dense, also in T_j (if $T_{(j-1)}$ is sparse, searching $T_{(j-1)}$ is enough). For any i, if X_i is a dense level then searching as in Lemma 3 costs at most $O(\text{diam}(X_i)) = O(\text{diam}(X_{i-1}))$. If X_i is sparse then searching on X_{i-1} as in Lemma 3 costs at most $O(\text{diam}(X_{i-1}))$. Given a target t, there are two cases. If $t \notin X_j$ then the total cost will be $\sum_{i \leq j} O(\text{diam}(X_{i-1})) = O(\text{diam}(X_{j-1}))$ as required. Otherwise, if $t \in X_\ell$ for $\ell \leq j$ then if X_ℓ is dense then $d(r,t) = \Theta(\text{diam}(X_{\ell-1}))$ and finding t will cost $\sum_{i \leq \ell} O(\text{diam}(X_{i-1})) = O(\text{diam}(X_{\ell-1}))$. Otherwise, if $t \in X_\ell$ and X_ℓ is sparse then $d(r,t) > \text{diam}(X_{\ell-1})$, and finding t will cost $\sum_{i \leq \ell-1} O(\text{diam}(X_{i-1})) + d(r,t)$. □

3.3.2 Sparse level routing

Storage. The storage used per node for sparse level routing is as follows. Each node maintains the routing scheme of Lemma 8 for all the centers v such that $v \in S(u)$.

In addition each node u maintains for each sparse level i the minimal index $bound(i)$ such $A(u,i) \subseteq X_{bound(i)}$, where $X_{bound(i)}$ is the set in Lemma 8 for the closest center of level $J(i)$.

So the amount of memory per sparse level is $2^{2\alpha+2}$ centers times $2^{O(\alpha)} \log^3 n$ bits for Lemma 8, for a total of $2^{O(\alpha)} \log^3 n$ bits

Routing and Cost. On sparse level $a(u,i)$ node u routes to the closest center ℓ from $C_{J(i)}$ and uses the single source routing scheme with bounding parameter $bound(i)$ for Lemma 8 to shortest path tree rooted at ℓ that contains all the nodes v that maintain $\ell \in S(v)$. If the target is not found then the cost is bounded by $O(2^{(a(u,i))})$ this follows from the definition of $bound(i)$ and Lemma 8. If the target is found during the search then by Lemma 8 the stretch is $O(1)$.

3.4 The routing scheme

Theorem 2. *For each weighted n-node graph with doubling dimension α and aspect ratio Δ, there is a polynomial time constructible name-independent routing scheme with stretch factor $O(1)$ that uses $2^{O(\alpha)} \log n \min\{\log^3 n, \log \Delta\}$-bit routing tables per node.*

Proof sketch. For storage, Every node stores $2^{O(\alpha)} \log^3 n$ bits per dense level and $2^{O(\alpha)} \log^3 n$ per sparse level. Since a node maintains at most $\log n$ sparse levels and $|R(u)| = O(\log n)$ dense levels, the total scale-free storage is bounded $2^{O(\alpha)} \log^4 n$ bits per node.

For stretch analysis, let $i \in K$ be the first iteration index in which v is found when search from u. There are two cases to consider.

1. If level $i-1$ is sparse for u, then given that v is not found in iteration $i-1$, by Lemma 7, v is not inside $E(u, i-1)$, and hence, $d(u,v) \geq 2^{a(u,i)-\Gamma}$.

2. Otherwise, level $i-1$ is dense for u, and again, v is not found in iteration $i-1$. In this case, by Lemma 5, v is not inside $F(u,i-1) = B(u, 2^{a(u,i-1)}/2)$. By density, $a(u,i) \leq a(u,i-1)+\Phi$. Putting these two facts together, we have $d(u,v) \geq 2^{a(u,i-1)-1} \geq 2^{a(u,i)-(\Phi+1)}$.

In either case, $d(u,v) \geq C_1 \cdot 2^{a(u,i)}$ for a universal constant C_1. From the analysis of both sparse and dense cases, reaching v on level i will cost at most $C_2 \cdot d(u,v)$ where C_2 is a universal constant.

For the cost of the negative responses, note that the highest level that fails is $i-1$. From the analysis of both sparse and dense cases, the cost of a negative response for level j is bounded by $C_3 \cdot 2^{a(u,j+1)}$ for some universal constant C_3.

Hence, the total cost of negative response is at most $\sum_{j=0}^{i-1} C_3 \cdot 2^{a(u,j+1)} \leq C_3 2^{a(u,i)+1}$. So the stretch is bounded by

$$\frac{C_2 \cdot d(u,v) + C_3 \cdot 2^{a(u,i)+1}}{C_1 \cdot 2^{a(u,i)}} = O(1)$$

□

4 Lower Bounds for Shortest-Path Single-Source Routing Schemes

A (d,m,ϵ)-*star*, for integers $d, m \geq 1$ and $\epsilon \in [0,1)$, is a subdivision G of $K_{1,d}$, each edge of $K_{1,d}$ being subdivided into exactly m nodes. It has a total of $n = md+1$ nodes and the degree-d node of $K_{1,d}$ is called the *center* of G. The edges of G receive weights such that the distances from the center to all the nodes of a same branch range in $[1-\epsilon, 1]$. Finally, the nodes of G have (unique) names taken from $\{1, \ldots, md+1\}$, the center being labelled $md+1$.

We observe that an $(n-1,1,0)$-star is an unweighted n-node planar graph (it is actually a tree isomorphic to $K_{1,n-1}$ with uniform weights $1-\epsilon = 1$ on all its edges), and that a (d,m,ϵ)-star has doubling dimension $\alpha = \log(d+1) \in [1, \log n]$.

Theorem 3. *There is an (d,m,ϵ)-star for which every single-source name-independent routing scheme with stretch factor $< 3(1-\epsilon)$ has memory requirements at least $md \log d - d \log \sqrt{7m}$ bits.*

In particular, for unweighted n-node trees, the stretch factor must be at least 3 if less than $\Omega(n \log n)$ bits are used, and for n-node graphs of doubling dimension α, the stretch must be at least $3-\epsilon$, for any $\epsilon > 0$, if less than $\Omega(\alpha n)$ bits are used.

Proof. Consider any single-source routing scheme R on an (d,m,ϵ)-star G. Assume that the stretch factor of R is $s < 3(1-\epsilon)$ and the source of R is the center of G. The center must route to any destination $y \in \{1, \ldots, md\}$ (whose distance from the center ranges from $[1-\epsilon, 1]$) on the branch containing node y, since otherwise the stretch would be at least $3(1-\epsilon)/1$.

With each possible labeling of the destinations, let us associate the *routing table* of the center, say T. More precisely, T is a table of md entries such that $T[y] \in \{1, \ldots, d\}$ returns the port number of R applied on y. From the above discussion, $T[y]$ must return the branch number to which y belongs.

In the fixed-port model, the number $f(d,m)$ of distinct routing tables T used for all labelings

of G, is:

$$f(d,m) \;=\; \frac{(md)!}{(m!)^d}$$

since there are $(md)!$ labelings for the destinations, and since there are $m!$ possible permutations for each branch (each such permutation given the same table). Clearly, at least one such destination labeling forces the routing table of the center of G to store at least $\log f(d,m)$ bits.

For every $n \geq 1$, we have $n\log(n/e) < \log(n!) < n\log(n/e) + \log\sqrt{7n}$, because, for $n \geq 1$, $(n/e)^n\sqrt{2\pi n} < n! < (n/e)^n\sqrt{2\pi n}e^{1/12-1/360+1/1260} < (n/e)^n\sqrt{7n}$ (cf. formula (9.91) of [15][pp. 481]). Therefore the number of bits required by the center of G is at least:

$$\log f(d,m) \;\geq\; md\log(md/e) - d(m\log(m/e) + \log\sqrt{7m}) \;=\; md\log d - d\log\sqrt{7m}$$

that completes the proof. □

5 Labelled Routing Schemes

In this section we study *labelled* routing schemes for metric spaces. Routing with stretch 1 is trivial using labels with $O(n^2\log n)$ bits that encode the whole graph. Hence an important factor is the label size. Typically this is bounded by a polylogarithmic number of bits.

In all previous labelled stretch $1+\varepsilon$ schemes [23, 11, 21], nodes labels depend on the doubling dimension α and the aspect ratio Δ. Except for [23], they also depend on the stretch ϵ. Here we provide a scheme with optimal labels of only $\lceil \log n \rceil$ bits. Furthermore, the storage is scale-free: The storage in our scheme is $(\frac{1}{\varepsilon})^{O(\alpha)}\log^2 n$, independent of the aspect ratio Δ.

5.1 Optimal label size

In this section, we explain the method for obtaining $\lceil \log \rceil$ bit label sizes. Let $1+\epsilon$ be the desired stretch factor. Assume $0 < \epsilon \leq 2$.

Our scheme builds a hierarchical net in the following intuitive manner. Start with a singleton $2^{\log\Delta}$-net $X_{\log\Delta} = \{r\}$, for an arbitrary choice of $r \in V$. Then recursively construct a 2^i-net X_i out of X_{i+1} by expanding X_{i+1} with nodes (greedily) to obtain a 2^i-net. Thus, $\forall \quad 1 \leq i \leq \log\Delta$: $X_{i-1} \supseteq X_i$.

The hierarchical $\{X_i\}$ net naturally induces a tree T as follows: The root is $r \in X_{\log\Delta}$. For consistency, we count the tree levels from top ($\log\Delta$) to bottom (1). Now iterate on i to connect every node in X_i to the closest node in X_{i+1}. Note that some nodes of V appear multiple times in T. In order to distinguish V members from tree nodes we will call the latter *tnodes*. Also note that, for any node $x \in X_i \cap X_{i+1}$, the closest node to x from X_{i+1} is x itself. Therefore, the tnode in T that corresponds to x at level $i+1$ connects to a tnode that corresponds to x at level i.

The important property maintained by this construction is the follows. Since every tnode s at level j has distance at most 2^j to its parent (at level $j+1$), then a sub-tree rooted at level j has radius 2^{j+1} around its root.

Let $D(v)$ be a DFS enumeration of T that skips duplicate nodes. More precisely, if a node x appears in several levels of T then it is counted the first time it is visited in the DFS order; on

further visits, it is skipped and the DFS enumeration continues with the next tnode in the DFS order.

A *range* of a tnode is the range of the DFS enumeration of its subtree. For a node v, let $L_i(v)$ be the range of the tnode corresponding to v at level i of T ($\perp$ if none), observe that $L_i(v)$ requires $2\lceil \log n \rceil$ bits. The label of a node v is defined simply as $D(v)$ and requires $\lceil \log n \rceil$ bits.

For every $i \in I$ and node x, let $Y_i(x) = B(x, 2^{i+1}) \cap X_i$. Then we have the following.

Claim 9. *For every $i \in I$ and node x either $x \in Y_i(x)$ or there exists $\ell \in Y_i(x)$ such that the range $L_i(\ell)$ contains $D(x)$.*

This follows directly from the structure of DFS enumerations and the fact that if x does not belong to X_i then it has a parent in $\ell \in X_i$ with $d(x, \ell) \leq 2^{i+1}$.

As a side remark, observe that with the above labelling scheme we can easily obtain a $(1+\varepsilon)$-stretch labelled routing scheme with $\lceil \log n \rceil$ label sizes as follows.

A node v maintains $(L_i(x), D(x))$ for every $x \in \left(B(v, 2^{i+2}) \cap X_{i-c}\right)$ and $i \in I$. The parameter $c = \log \frac{1}{\varepsilon} + 3$ guarantees that if $2^i < d(v,u) \leq 2^{i+1}$, then using the labels, v finds a net point $x \in X_{i-c}$ at distance at most $2^{i-c+1} < \frac{\varepsilon}{4} d(v,u)$ from u.

Claim 10. *Stretch is $1+\varepsilon$.*

Proof. Suppose the path of landmarks is $u = x_0, x_1, \ldots, x_t = v$ then each time we route from x_{i-1} towards v through a node x_i we will show $\frac{d(x_{i-1},x_i)}{d(x_{i-1},v)-d(x_i,v)} \leq 1+\epsilon$. Hence $\sum_{i=1}^{t} d(x_{i-1}, x_i) \leq (1+\epsilon)\sum_{i=1}^{t}(d(x_{i-1},v) - d(x_i,v)) \leq (1+\epsilon)d(u,v)$. It remains to observe that $d(x_i, v) \leq \frac{\varepsilon}{4} d(x_{i-1}, v)$ so $\frac{d(x_{i-1},x_i)}{d(x_{i-1},v)-d(x_i,v)} \leq \frac{(1+\frac{\varepsilon}{4})d(x_{i-1},v)}{(1-\frac{\varepsilon}{4})d(x_{i-1},v)} \leq 1+\varepsilon$ using $\epsilon \leq 2$. □

For storage, note that for each i, the number of nodes for which label information is maintained is at most $(2^\alpha)^{c+2} = 2^{O(\alpha \log(1/\varepsilon))}$. The total storage per node is therefore $(\frac{1}{\varepsilon})^{O(\alpha)} \log n \log \Delta$ bits.

Theorem 4. *For any n point undirected weighted graph with doubling dimension α, aspect ratio Δ, and for any $\epsilon \leq 1/2$ there exists a polynomial time constructible labelled routing scheme that assigns $\lceil \log n \rceil$ bit labels, routes on paths of stretch $1+\epsilon$, and requires each node to maintain $(\frac{1}{\varepsilon})^{O(\alpha)} \log n \log \Delta$ bits of routing information.*

5.2 Scale-free labelled routing for metric spaces

Our next goal is to remove the scale factor from the storage. For this we use the concept of the component tree of Thorup [24]. The following construction is applicable in the metric space model.

Let $G(i)$ be the subgraph of G containing all edges e for which $\omega(e) \leq 2^i$. Note that $G(i)$ is not necessarily connected. For a graph G, let $C(G)$ be the number of connected components of G.

We recursively define the *component tree*, a rooted tree $S = (W, p)$. The nodes W of S are connected subgraphs of G. The root node is G and the leafs are all the singleton subgraphs. For a node $H \in W$ let $\mathrm{cal}(H)$, the caliber of H, be the maximum integer i such that $C(H(i)) > 1$. Let $F_1, \ldots, F_{C(H(\mathrm{cal}(H)))}$ be the connected components of $H(\mathrm{cal}(H))$, then the children of H in

the tree S will be the nodes $F_1, \ldots, F_{C(H(\mathrm{cal}(H))}$. This completes the definition of the component tree. The parent function $p : W \to W$ maps each child to its parent in S, such that H is a child of $p(H)$ for any $H \in W$.

Claim 11. *For any subgraph $H \in W$, the length of any simple path in H is $\leq 2^{\mathrm{cal}(H)}(n-1)$*

Let $\overline{W} \subset W$ be the non-leaf nodes of S. We now proceed to assign to each $H \in \overline{W}$ a node in $v \in H$ such that each node in V is assigned at most one node in $\overline{W}$. Initially all nodes are unassigned. We build the assignment from leaves to the root, while maintaining the following invariant for a node $H \in \overline{W}$: If $p(H)$ has no assignment yet, then H contains an unassigned node $v \in H$. Given a node H with children $F_1, \ldots, F_t$, for some $t \geq 2$, each child is either a single node, in which case this node is unassigned or by the induction assumption has an unassigned node. Hence we assign a previously unassigned node to handle H and there remains at least one more unassigned node. We denote by $v(H)$ the node assigned to H. The assignment induces a tree R on a subset of the nodes of V: For a node $v = v(H)$, fix its parent $p(v)$ to be the node assigned to its parent in the component tree S, i.e., $v(p(H))$.

Let $ring(u, i, c) = B(u, 2^{i+2}) \cap X_{i-c}$. Each node u maintains at most two sets of rings. The first set is $ring(u, i, c)$ for all $\mathrm{cal}(u) \leq i \leq \mathrm{cal}(u) + 3\log n + 5 + \log(1/\epsilon)$. If u is assigned to some $H \in \overline{W}$ then it also maintains a second set: $ring(u, i, c)$ for all $\mathrm{cal}(H) \leq i \leq \mathrm{cal}(H) + 3\log n + 5 + \log(1/\epsilon)$. In addition, node u maintains $p(u)$.

Routing: The idea is to climb up the component tree to the root until the target is found. First look for the target in the first ring-set, if does not exist go to the parent. Given a node assigned to a component, use its second set of rings. If the target is not found then move to the parent.

Memory: Each ring requires at most $(\frac{1}{\varepsilon})^{O(\alpha)} \log n$ bits. A node maintains up to $2(3\log n + 5 + \log(1/\epsilon))$ rings, for a total memory of $(\frac{1}{\varepsilon})^{O(\alpha)} \log^2 n$ bits.

Stretch: From a node u, if $d(u,t) \leq n^3 2^{\mathrm{cal}(H)} \frac{2^5}{\varepsilon}$, then some ring in the first set contains a net point that cuts the distance and maintains stretch $1 + \frac{\varepsilon}{2}$ (this is the usual argument as in Claim 10).

Otherwise, each time we go from u to a node $p(u)$ we know that $d(u,t) > n^3 2^{\mathrm{cal}(H)} \frac{2^5}{\varepsilon}$. Since the cost of going from u to $p(u)$ is at most $n2^{\mathrm{cal}(H)}$ then we can repeat this n times and still spend at most $d(s,t)\frac{\varepsilon}{2^5 n}$ until the right ring is found. This process can occur at most n times so the total cost of routing up the component tree to find the appropriate ring is at most $\frac{\varepsilon}{2^5} d(s,t)$.

Theorem 5. *For any n point metric space with doubling dimension α and aspect ratio Δ, and for any $\epsilon \leq 1/2$ there exists a polynomial time constructible labelled routing scheme that assigns $\lceil \log n \rceil$ bit labels, routes on paths of stretch $1+\epsilon$, and requires each node to maintain $(\frac{1}{\varepsilon})^{O(\alpha)} \min\{\log n, \log \Delta\}$ outgoing links and $(\frac{1}{\varepsilon})^{O(\alpha)} \log n \min\{\log n, \log \Delta\}$ bits of routing information.*

6 Scale-Free Labeled Routing Schemes for Graphs

The following scale-free scheme is applicable for the graph model. We use the sparse-dense decomposition of Section 3.1 with parameter $\Phi = \log(\frac{1}{\varepsilon}) + 6$.

We use the same hierarchical net creation $\{X_i\}$ as in Section 5.1 and use rings defined as $ring(u,i,c) = B(u,2^{i+2}) \cap X_{i-c}$ as in Section 5.2. Let $c = \Phi + 6$. Each node maintains $ring(u,i,c)$ for each i such that there exists a level $j \in K$ with $-6 \leq i - a(u,j) \leq \Phi + 3$. For each $\ell \in ring(u,i,c)$ that u maintains, node u stores the a $O(\log n)$ bit range $L_i(\ell)$ and label $d(\ell)$ as described in Section 5.1, and the port of the next hop on a shortest path from u to ℓ.

We build the same centers C_i of Section 3.3. Each node also maintains labelled tree routing scheme of [12, 25] (see Lemma 4) for a shortest path tree routed at each node in $N(u, 2^{2\alpha+3}+1, C_i)$ for all $i \in K$.

The label $\Lambda(u)$ of a node u will consist of two parts: (1) The $O(\log^2 n)$ bit label for the labelled tree routing scheme of Lemma 4 for each tree whose root is in $N(u, 2^{2\alpha+3}+1, C_i)$ for all $i \in K$. (2) The $O(\log n)$ DFS identifier $d(u)$ as defined in Section 5.1. The label's length is dominated by the first part and requires $2^{O(\alpha)} \log^3 n$ bits.

Given a current node u and destination t let i be the maximal index such that $2^{a(u,i)} < d(u,t)$. Denote by $d = d(u,t)$. There are several cases to consider.

1. **Sparse case:** If $2^{a(u,i)}\frac{2^5}{\varepsilon} \leq d < 2^{a(u,i+1)-\Gamma}$ then i must be a sparse level for u.[1] Suppose $2^j \leq |A(u,i)| < 2^{j+1}$ then by Lemma 6 there exists $\ell \in C_j$ such that $d(u,\ell) \leq 5 \cdot 2^{a(u,i)}$. By Lemma 7, for any node $v \in B(u, 2^{a(u,i+1)-\Gamma})$ we have that $\ell \in N(v, 2^{2\alpha+3}+1, C_j)$ hence all the nodes on the path from ℓ to t maintain the tree routing scheme of the tree rooted at ℓ and $\Lambda(t)$ will contain the label of t on ℓ's tree. Hence given t's label $\Lambda(t)$ we can reach the root ℓ and then the target at a cost of at most $(\frac{\varepsilon}{4} + 1 + \frac{\varepsilon}{4})d$.

2. **Otherwise:** If $d < 2^{a(u,i)}\frac{2^5}{\varepsilon}$ or $2^{a(u,i+1)-\Gamma} \leq d < 2^{a(u,i+1)}$ then by the definition of which rings are stored, u maintains $ring(u,j,c)$ where $2^j \leq d < 2^{j+1}$. Hence u can infer from t's label $d(t) \in \Lambda(t)$ and from the labels and ranges of the rings it maintains, a node $\ell \in ring(u,j,c)$ such that $d(\ell,t) \leq \frac{\varepsilon}{8}d$.

   But for a graph this is not enough since we need the nodes on the shortest path from u to ℓ to maintain routing information to reach ℓ. Let P be this path, let $v \in P$ be the first node such that $d/2 \leq d(u,v)$. Actually all we need is that the nodes of P from u up to, and excluding, v will all maintain the required ring and hence maintain ℓ. There are two cases:

   A. Dense case: If indeed each node on the path P from u up to, and excluding, v maintains ℓ in its ring set then once we reach v we continue to route recursively. We prove that $d(u,v)/(d - d(v,t)) \leq 1 + \epsilon$ as in Claim 10. This will guarantee $1+\epsilon$ stretch for the whole path.

   Claim 12. *If $d/2 \leq d(u,v)$ and $d(\ell,t) \leq \frac{\varepsilon}{8}d$ then $d(u,v)/(d - d(v,t)) \leq 1 + \varepsilon$.*

   Proof. Using the triangle inequality on v,t,l and on u,t,l we get $d(v,t) \leq d(1 + 2\frac{\varepsilon}{8}) - d(u,v)$. So $d(u,v)/(d - d(v,t)) \leq d(u,v)/(d(u,v) - 2\frac{\varepsilon}{8}d)$ and using $d/2 \leq d(u,v)$ we get $\leq 1 + 8\frac{\varepsilon}{8}$ when $\varepsilon \leq 1$. □

   B. Back to sparse: Let w be the first node on the path from u to v (excluding v) that does not maintain $ring(w,j,c)$ that contains ℓ. Hence $d(u,w) < d/2$ so $d(w,t) \geq$

[1] Recall that Γ has been defined after Lemma 6.

$d/2 - \varepsilon d \geq d/4$. Therefore, $2^{j-2} \leq d(w,t)$. Let $a(w,i)$ be the maximal level such that $2^{a(w,i)} < d(w,t)$.

Since w does not maintain the ring of distance 2^j then by the definition of what rings are stored due to $a(w,i)$ it must be that $2^{a(w,i)}\frac{2^5}{\varepsilon} \leq d(w,t)$ using $2^{j-2} \leq d(w,t)$. Since the rings due to $a(w,i+1)$ do not store scale 2^j then $d(w,t) < 2^{a(w,i+1)-\Gamma}$. Therefore we are back to case 1, and we can reach t directly using tree routing as detailed above.

Theorem 6. *For any n point undirected weighted graph with doubling dimension α, and for any $\epsilon \leq 1/2$ there exists a polynomial time constructible labelled routing scheme that assigns $2^{O(\alpha)}\log^3 n$ bit labels, routes on paths of stretch $1+\epsilon$, and requires each node to maintain $(\frac{1}{\varepsilon})^{O(\alpha)}\log^4 n$ bits of routing information.*

7 A Lower Bounded for Labeled Shortest Path Routing

A graph (or a metric) has *growth* α (also called *grid dimension*) if $|B(u,2r)|/|B(u,r)| \leq \alpha$ for all node u and radius r. Graphs (or metric) of bounded growth have bounded doubling dimension, but the reverse is false in general. In particular the next lower bound holds for labelled stretch-1 routing on bounded doubling dimension metric. The counter-example is a slight modification on the planar construction presented in [2].

Theorem 7. *There are bounded growth graphs with n nodes and with non-negative integral edge-weights bounded by $O(\sqrt{n})$, for which every labelled stretch-1 routing scheme requires $\Omega(\sqrt{n})$ bits for some local routing tables or some node labels.*

Proof. The counter-example is a slight modification on the graph construction presented in [2]. It depends on an integer $k = k(n) = \Theta(\sqrt{n})$. Let M be a $(k+1)\times(k+1)$ weighted mesh whose each edge of the i-th row has weight $2i$ and each edge of the j-th column has edge $2j$.

Let us check that M has bounded growth. Indeed, let us consider a node $u = (i,j)$ and a radius r. A shortest path from u to any v can be decomposed in a maximal horizontal segment and a maximal vertical segment. Hence, the ball $B(u,r)$ is composed of the points of the mesh located inside the quadriangle defined by the four points around $u = (i,j)$: $(i, j+\delta_j), (i-\delta_i, j), (i, j-\delta_j), (i+\delta_i, j)$, where $\delta_i = \lfloor r/(2j)\rfloor$ and $\delta_j = \lfloor r/(2i)\rfloor$. Summing up the area of the four triangles composing the quadriangle, it follows that $|B(u,r)|$ ranges between $2\delta_i\delta_j$ and $2\delta_i\delta_j + 4(\delta_i+\delta_j)$, where the term $4(\delta_i+\delta_j)$ denotes the perimeter of the quadriangle (some points of the perimeter must possibly be counted twice). Bounding $\delta_i + \delta_j \leq 2\delta_i\delta_j$, we obtain that $|B(u,r)| \in [2\delta_i\delta_j, 10\delta_i\delta_j]$, and thus that $|B(u,r)| = \Theta(r^2/(ij))$. Therefore, $|B(u,2r)|/|B(u,r)| = O(1)$ as claimed.

From M we construct the graph H in which every edge is subdivided in two edges with the following weight updating: around the point (i,j) of the mesh, the weight 1 is assigned to the upward incident edge and to the right incident edge (if any), whereas the downward incident edge and the left incident edge (if any) have weights $2i-1$ and $2j-1$, respectively. Finally, we add a diagonal edge with weight 1 between the upward and right neighbors of (i,j). In H, it follows that the distance between the points $(i-1,j)$ and $(i,j+1)$ (using the diagonal edge) is $2i+2j-1$ whereas the distance between (i,j) and $(i-1,j+1)$ remains $2i+2j$ as in M. More generally, one can show that in H the shortest path between a left-up to a right-down point is unique and

consists to only one vertical down segment, a diagonal edge, followed by a horizontal segment. This corresponds to the distance in M minus 1, whereas the shortest paths between a left-down to a right-up point remains the same than in M. It follows that $B(u,r)$ in H is some neighborhood of $B(u,r)$ in M (i.e., is a quadriangle where some neighbors on the perimeter are added). As H is of bounded degree, H has bounded growth as well. We observe that every graph obtained from H by removing some diagonal edges has bounded growth as well.

Consider a graph A obtained from H by removing independently with probability 1/2 each diagonal edge, and let B be the graph obtained from H by removing the diagonal edges of A. For $i,j \in \{1,\ldots,k\}$, let u_i be the point $(1,i)$ and let $v_j = (1+j, k+1-j)$.

Claim 13. *For all $i+j \leq k$, the distance in A from u_i to v_j (i.e., from a left-up to right-down point) is $2i(j+1)+(2j+1)(k+1-j-1)$ if and only if the diagonal edge of $(i,j+1)$ is missing in A, and otherwise the distance is one less.*

As already said before, A and B have bounded growth. Finally we consider the graph G composed of A and B where the each pair of corresponding points of A and of B are connected by a path of length two (with weight 1 on each edge), so forming a kind of 3D mesh. We check that G has still a bounded growth (the degree is still bounded, and the eccentricity of each node increases by at most two). We note that the graph G has $O(n)$ nodes.

Consider any shortest path routing scheme from some u'_i to some v'_j in G, where u'_i denotes the middle node of the length-2 path connecting the u_i node of A to the u_i node of B (similarly for v'_j). By Claim 13 the shortest path from u'_i to v'_j, with $i+j \leq k$, is unique and goes to u_i in A if and only if the diagonal edge $(i,j+1)$ is missing in A (otherwise the shortest path goes thru B). The graph A contains k^2 random bits (one bit for each diagonal edges), and half of them can be recovered given the routing tables of all the u'_i's and given the label nodes of all the v'_i's. It follows that at least one node has a routing table or a label node of $\Omega(k) = \Omega(\sqrt{n})$ bits. This completes the proof.

Observe that this also proves that any distance labeling for bounded growth graphs requires $\Omega(\sqrt{n})$-bit labels (for polynomial weighted graphs), and $\Omega(n^{1/3})$-bit labels for unweighted bounded growth graphs (by replacing each weight-w edge of G by a length-w unweighted path). □

References

[1] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Routing with improved communication-space trade-off. In 18^{th} *International Symposium on Distributed Computing (DISC)*, volume 3274 of Lecture Notes in Computer Science, pages 305–319. Springer, October 2004.

[2] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Compact routing for graphs excluding a fixed minor. In 19^{th} *International Symposium on Distributed Computing (DISC)*, volume 3724 of Lecture Notes in Computer Science, pages 442–456. Springer, September 2005.

[3] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs for compact routing schemes. Research Report RR-1374-05, LaBRI, University of Bordeaux 1, 351, cours de la Libération, 33405 Talence Cedex, France, November 2005.

[4] Ittai Abraham and Dahlia Malkhi. Compact routing on euclidian metrics. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing (PODC)*, pages 141–149. ACM Press, 2004.

[5] Ittai Abraham and Dahlia Malkhi. Name independent routing for growth bounded networks. In 17^{th} *Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA)*. ACM Press, July 2005. To appear.

[6] Ittai Abraham, Dahlia Malkhi, and Oren Dobzinski. Land: stretch $(1 + \varepsilon)$ locality-aware networks for DHTs. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 550–559. Society for Industrial and Applied Mathematics, 2004.

[7] Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Compact distributed data structures for adaptive routing. In 21^{st} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 479–489. ACM Press, May 1989.

[8] Baruch Awerbuch, Amotz Bar Noy, Nati Linial, and David Peleg. Improved routing strategies with succinct tables. *Journal of Algorithms*, 11(3):307–341, 1990.

[9] Baruch Awerbuch and David Peleg. Sparse partitions. In 31^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 503–513. IEEE Computer Society Press, October 1990.

[10] Baruch Awerbuch and David Peleg. Routing with polynomial communication-space trade-off. *SIAM J. Discret. Math.*, 5(2):151–162, 1992.

[11] Hubert T.-H. Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. In 16^{th} *Symposium on Discrete Algorithms (SODA)*. ACM-SIAM, January 2005.

[12] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of Lecture Notes in Computer Science, pages 757–772. Springer, July 2001.

[13] Cyril Gavoille. Routing in distributed networks: Overview and open problems. *ACM SIGACT News - Distributed Computing Column*, 32(1):36–52, March 2001.

[14] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Journal of Distributed Computing*, 16:111–120, May 2003. PODC 20-Year Issue.

[15] Ronald Lewis Graham, Donald Ervin Knuth, and Oren Patashnik. *Concrete Mathematics, a foundation for computer science (second edition)*. Addison-Wesley, 1994.

[16] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In 44^{th} *Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543. IEEE Computer Society Press, October 2003.

[17] Yehuda Hassin and David Peleg. Sparse communication networks and efficient routing in the plane. *Distributed Computing*, 14(4):205–215, 2001.

[18] R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metrics. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 434–443. IEEE, October 2004.

[19] Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[20] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[21] Aleksandrs Slivkins. Distance estimation and object location via rings of neighbors. In *PODC '05: Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 41–50, New York, NY, USA, 2005. ACM Press.

[22] Aleksandrs Slivkins. Distributed approaches to triangulation and embedding. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages ??–?? Society for Industrial and Applied Mathematics, 2005.

[23] Kunal Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In 36^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 281–290, June 2004.

[24] Mikkel Thorup. Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. *J. Assoc. Comput. Mach.*, 46:362–394, 1999.

[25] Mikkel Thorup and Uri Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. ACM Press, July 2001.