On Space-Stretch Trade-Offs: Upper Bounds

Ittai Abraham
School of Computer Science and Engineering
Hebrew University of Jerusalem
Jerusalem, Israel
ittaia@cs.huji.ac.il

Cyril Gavoille*
Laboratoire Bordelais de Recherche en Informatique
University of Bordeaux
Bordeaux, France
gavoille@labri.fr

Dahlia Malkhi
School of Computer Science and Engineering
Hebrew University of Jerusalem
and Microsoft Research, Silicon Valley Center
dalia@microsoft.com

ABSTRACT

One of the fundamental trade-offs in compact routing schemes is between the *space* used to store the routing table on each node and the *stretch* factor of the routing scheme – the maximum ratio over all pairs between the cost of the route induced by the scheme and the cost of a minimum cost path between the same pair. All previous routing schemes required storage that is dependent on the diameter of the network. We present a new *scale-free* routing scheme, whose storage and header sizes are independent of the aspect ratio of the network. Our scheme is based on a decomposition into sparse and dense neighborhoods. Given an undirected network with arbitrary weights and n arbitrary node names, for any integer $k \geq 1$ we present the first scale-free routing scheme with asymptotically optimal space-stretch trade-off that does *not* require edge weights to be polynomially bounded. The scheme uses $\widetilde{O}(n^{1/k})$ space routing table at each node, and routes along paths of asymptotically optimal linear stretch $O(k)$.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Networks]: Network Architecture and Design – *Distributed networks*; G.2.2 [Discrete Mathematics]: Graph Theory – *Network problems, Graph labeling.*

General Terms: Algorithms, Theory.

Keywords: Compact Routing.

1. INTRODUCTION

One of the most basic functionalities of any distributed network is the ability to route messages between pairs of nodes. Given that each node has an arbitrary network identifier, a routing scheme allows any source node to route messages to any destination node, given the destination's network identifier. It is natural to consider a weighted network in which the cost of routing a message is proportional to the cost of the path taken from source to destination. In such a model it is desirable to minimize routing costs by routing on short paths. In this sense the efficiency of a routing scheme is measured by its *stretch factor*, the maximum ratio over all source destination pairs, between the cost of routing from the source to the destination and the cost of a minimum cost path. The trivial solution to routing on shortest paths with stretch factor 1 is for each node to store a routing table with $(n-1)$ entries that contains the next hop of an all pairs shortest path algorithm. This solution is very expensive as it requires each node to store $\Omega(n \log n)$ bits. Thus, network designers are faced with two conflicting goals: reduce both the stretch factor and the size of the routing tables.

For a weak variant of this problem, called *labeled routing*, both lower bounds and asymptotically optimal upper bounds are known (see [29]). In this version of the problem, the designer of a solution may pick node names that contain (bounded size) information about their location in the network. This variant is useful in many aspects of network theory, but less so in practice: Knowledge of the labels needs to be disseminated to all potential senders, as these labels are not the addresses by which nodes of an *existing* network, e.g., an IP network, are known. Furthermore, if the network may admit new joining nodes, all the labels may need to be re-computed and distributed to any potential sender. Finally, various recent applications pose constraints on nodes addresses that cannot be satisfied by existing labeled-routing schemes. E.g., Distributed Hash Tables (DHTs) require nodes names in the range $[1..n]$, or ones that form a binary prefix.

In this paper we assume a network with arbitrary node names and arbitrary edge weights. This model is called the *name-independent* model because the designer of the routing scheme has no control over node names. This routing problem may appear daunting: In order to route to a node, we must first somehow gain knowledge about its location in the network, but we must do so without exceeding the distance to the target.

A fundamental difficulty in all previous schemes is their heavy dependence on the scale of the network. Let the *aspect ratio* $\Delta = \max d(u,v)/\min_{u\neq v} d(u,v)$ be the ratio between the largest distance and the smallest distance, then many schemes require memory that tends to infinity as Δ increases. This suggests that there might exist a lower bound associated with the aspect ratio. However, the best known lower bound for name-independent routing [29] does not con-

*Supported by the projects "PairAPair" and "GeoComp" of the ACI Masses de Données.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SPAA'06, July 30–August 2, 2006, Cambridge, Massachusetts, USA.
Copyright 2006 ACM 1-59593-452-9/06/0007 ...$5.00.

tain such dependence. Hence 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, the only scale-free schemes for general graphs have exponential stretch [7, 8, 6]. Specifically, when each node stores $\widetilde{O}(n^{1/k})$ bit routing information the best stretch bound achieved is $O(2^k)$. Obtaining such scale free solutions was raised as an open question in [6, 22]. In this paper we fully answer this problem and provide an exponential improvement from $O(2^k)$ to asymptotically optimal $O(k)$ stretch.

1.1 Our contribution

We construct for any $k \geq 1$ a routing scheme with linear stretch factor of $O(k)$ and with $\widetilde{O}(n^{1/k})$-bit routing tables per node and $\widetilde{O}(1)$-bit headers[1].

Theorem 1. *For each weighted n-node graph, and integer $k \geq 1$, there is a polynomial time constructible name-independent routing scheme with stretch factor $O(k)$ that uses $O(k^2 n^{1/k} \log^3 n)$-bit routing tables per node.*

1.2 Techniques

Broadly speaking, there are two main techniques used to construct routing schemes. The first technique is *random sampling*, in which landmark nodes are selected randomly. This technique has been successful in labeled routing, providing asymptotically optimal space-stretch tradeoffs [29]. For name-independent schemes, random sampling based schemes were used for optimal trade-offs for stretch 3 schemes with $\widetilde{O}(\sqrt{n})$ space [5]. However, all general schemes with $\widetilde{O}(n^{1/k})$ space, based on random sampling, obtained exponential stretch $O(2^k)$ (see [7, 8, 6]).

The second technique is *sparse covers*, in which the graph is covered by clusters of bounded diameter such that each node belongs to a small number of clusters. This technique was used for several name-independent schemes [25, 9, 10, 6, 3]. In all the schemes above that use sparse covers, a cover with clusters of diameter $< 2^i$ is constructed for each $i \in \{1, \ldots, \lceil \log \Delta \rceil\}$, hence these schemes are inherently *not* scale-free.

Our scheme is based on a new decomposition into dense and sparse neighborhoods. It uses a subtle combination of random sampling and sparse cover routing techniques depending on the density or sparsity of each neighborhood. This decomposition allows us to remove any dependence on the aspect ratio. Informally, a sparse neighborhood is one where the number of nodes does not increase to much if the radius is increased by a constant factor. For sparse neighborhoods, we use random sampling techniques that turns out to be efficient in this case. Specifically, nodes maintain a tree-routing scheme for all the nearby landmarks. For dense neighborhoods, we use sparse cover based routing techniques. Since dense neighborhoods imply that the number of nodes is multiplied when the diameter is increased by a constant, the number of dense neighborhood scales a node belongs to is $O(\log n)$. This fact allows to use sparse cover techniques in a scale-free manner. While our decomposition was developed independently of [20], it bears some similarities to the "measured decent" approach of Krauthgamer et al. [20]. However, using the "measured decent" approach for routing fails since it chooses scales for each density change. Hence searching on a ball with $\Omega(\log n)$ density changes may incur $O(\log n) \gg k$ stretch which is unacceptable. Our decomposition circumvents this by decomposing both by density change and by diameter change.

Our sparse/dense decomposition technique has interest in its own, as a general approach to remove the aspect ratio parameter in many other constructions. For example for labeled and name-independent routing schemes for networks with low doubling dimension [2].

1.3 Related work

The space-stretch trade-off has been extensively studied under various models and extensions. We refer the reader to Peleg's book [24] and to the surveys of Gavoille and Peleg [17, 18] for comprehensive background.

Peleg and Upfal [25] were the first to study this trade-off in a parameterized manner. For unit cost networks they achieve $O(k)$ stretch with a total of $O(k^3 n^{1+1/k} \log n)$ bits for all routing tables. For weighted networks, Awerbuch et al. [7, 8] present a scale-free routing scheme with exponential stretch of $O(k^2 9^k)$ that requires $\widetilde{O}(n^{1/k})$ bits per node. Arias et al. [6] improve the stretch to $O(k^2 2^k)$ with the same memory bound.

More recently, constant stretch routing schemes have been designed for networks whose induced metric space has a low doubling dimension [2, 19], and for unweighted graphs excluding a fixed minor [4] (including trees and planar graphs). These schemes require a polylogarithmic space, hidding a multiplicative constant depending on the doubling dimension, or on the minor excluded.

Awerbuch and Peleg use *sparse covers* [9] in order to build a hierarchal routing scheme [10]. Their scheme is based on tree covers with geometrically increasing radii. Therefore there is an inherent geometric factor in their memory requirement. They achieve stretch $O(k^2)$ with $\widetilde{O}(n^{1/k} \log \Delta)$-bit routing tables, and $O(\log \Delta)$ headers, where Δ is the maximum weighted distance between any two nodes divided by the minimum weighted distance between any two unique nodes. Abraham, Gavoille, and Malkhi [3] improve the stretch factor to $O(k)$ with the same memory requirement. This solution is adequate if the network weights are polynomial in the number of nodes. However for arbitrary networks the diameter may be arbitrarily large, for instance $\Delta = \Omega(2^n)$, and solutions based on the aspect ratio of the network may become unusable.

A weaker variant of compact routing is based on the *labeled routing* model. Instead of assuming nodes have arbitrary names, in this model, the network designer is allowed to name the nodes in a topology dependent manner. This paradigm does not provide for a realistic network design, however, the tools devised for its solution have proven useful as building blocks of full routing schemes.

Eilam et al. [14] present a stretch 5 labeled scheme with $\widetilde{O}(n^{1/2})$ memory, whereas Cowen [13] presents a stretch 3 labeled scheme with $\widetilde{O}(n^{2/3})$ memory. Later, Thorup and Zwick [29] improve to stretch 3 using $\widetilde{O}(n^{1/2})$ bits. These three schemes uses $O(\log n)$-bit node names. Thorup and Zwick also give in [29] a generalization of their scheme and using techniques from their distance oracles [30], achieve labeled schemes with stretch $4k-5$ (and even $2k-1$ with hand-

[1]The notation $\widetilde{O}(\cdot)$ indicates complexity similar to $O(\cdot)$ up to poly-logarithmic factors.

shaking) using $\widetilde{O}(n^{1/k})$-bit routing tables and $o(k\log^2 n)$-bit node names. Labeled routing on a trees is explored in [15, 29], achieving stretch 1 with $O(\log^2 n/\log\log n)$ bits for local tables and for headers, and this is tight [16].

Thorup [28] showed that planar graphs support stretch $1+\varepsilon$ labeled routing schemes with polylogarithmic space. This has been generalized by Abraham and Gavoille [1] to graphs excluding a fixed minor with same stretch and space bounds. For low doubling dimension networks, strech $1+\varepsilon$ labeled schemes exist [27, 12, 26], but all of them have a dependency in the aspec ratio Δ in the memory bounds.

A variation of our sparse-dense decomposition was recently used to provide some scale-free labeled and name-independent routing schemes for networks with low doubling dimension [2], this solved a question raised by Slivkins [26]. While the decomposition in that paper is superficially similar, the techniques used in this paper are significantly different. Specifically, the sparse level case uses a landmark property of Lemma 3 together with a new error-reporting tree routing scheme of Lemma 4. The dense level case is based on applying spares covers of [9] with the routing extensions of [3].

2. SPARSE AND DENSE NEIGHBORHOOD DECOMPOSITION

2.1 Preliminaries

Given is a weighted graph $G=(V,E,\omega)$ of size $n=|V|$ with a non-negative weight function $\omega: E\rightarrow \mathbb{R}^+$. Let the cost of a path be the sum of the weights of its edges. For any two nodes $u,v\in V$ let $d(u,v)$ denote the cost of a minimum cost path between u and v. 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, $B(u,r)=\{v \mid d(u,v)\le r\}$. 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 $I=\{0,1,\ldots,\lceil\log\Delta\rceil\}$. Given a parameter k, let K denote the level set $K=\{0,1,2,\ldots,k\}$. 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.

Our solution is based on using a new decomposition into a series of balls around each node that have a combined combinatorial and geometric restriction. Each ball has at least $n^{1/k}$ more nodes than the previous *and* its radius is at least twice the radius of the previous.

Definition 1. *For all $u\in V$ and $i\in K$ define the* range *$a(u,i)$ as follows. Let $a(u,0)=0$. Then recursively let $a(u,i+1)$ be the smallest positive integer $j>0$ such that*

$$|B(u,2^j)|\ge n^{1/k}|B(u,2^{a(u,i)})|$$

(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 who's radius is $2^{a(u,i)}$ 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.

Definition 2 (Dense level). *For $u\in V$ and $i\in K$, define that i is a* dense level *for node u if*

$$a(u,i) \;<\; a(u,i+1) \;\le\; a(u,i)+3$$

Define 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 $n^{1/k}$ times as many nodes as the current level by looking at a ball whose radius is at most 2^3 times the current level.

2.2 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,

$$R(u)=\{i\in I\mid \exists a\in L(u), -1\le a-i\le 4\}\ .$$

Define $F(u,i)=B(u,2^{a(u,i)-1})$. The main property of dense levels is captured in the following lemma.

Lemma 2 (Dense neighborhoods). *If i is a dense level for u and $v\in F(u,i)$, then $a(u,i)\in R(v)$.*

PROOF. Recall that $F(u,i)=B(u,2^{a(u,i)-1})$. Let $v\in F(u,i)$, then $B(v,2^{a(u,i)-1})\subseteq A(u,i)$ (see Figure 1), and hence

$$|B(v,2^{a(u,i)-1})| \;\le\; |A(u,i)|\ .$$

Since $a(u,i+1)\le a(u,i)+3$, then $B(v,2^{a(u,i)+4})\supseteq A(u,i+1)$, and hence

$$|B(v,2^{a(u,i)+4})| \;\ge\; |A(u,i+1)| \;\ge\; |A(u,i)|\cdot n^{1/k}\ .$$

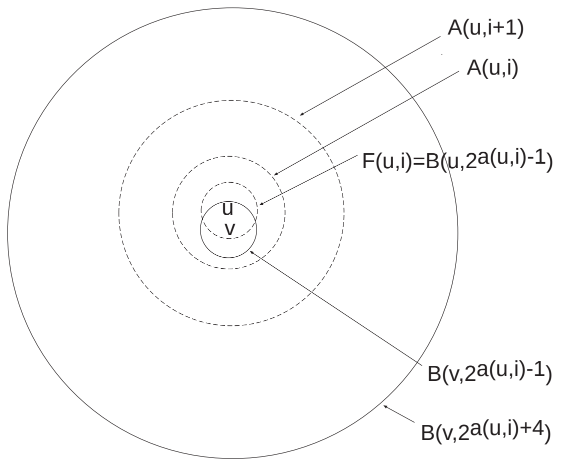

Figure 1: Example of a level i dense neighborhood for node u and a node $v\in F(u,i)$.

Together, these imply $|B(v, 2^{a(u,i)+4})| \geq n^{1/k} \cdot |B(v, 2^{a(u,i)-1})|$. Therefore, there exists some index $a(v, j)$ such that $a(u, i) - 1 \leq a(v, j) \leq a(u, i) + 4$. $\square$

2.3 Sparse Levels

We use a low discrepancy cover of 'landmark' nodes. We use $k + 1$ sets $V = C_0 \supseteq C_1 \supseteq \cdots \supseteq C_k = \varnothing$ of landmarks defined as follows. Let $C_0 = V$. For $i = 1$ to $k - 1$ iteratively set C_i to contain each element of C_{i-1} independently, with probability $(n/\ln n)^{-1/k}$. The randomized procedure can be de-randomized using the method of conditional probabilities and pessimistic estimators. Let $\mathcal{B} = \left\{B(u, 2^i) \mid u \in V, i \in I\right\}$, note that $|\mathcal{B}| \leq |V|^2$. We will use two simple properties.

Claim 1. *With high probability, for any $B \in \mathcal{B}$, if $4(\ln n)^{(k-j)/k} n^{j/k} \leq |B|$ for $j \in K$ then $B \cap C_j \neq \varnothing$.*

Proof. Union bound and $\Pr[B \cap C_j = \varnothing] \leq (1 - (n/\ln n)^{-j/k})^{4(\ln n)^{(k-j)/k} n^{j/k}} \leq e^{4 \ln n}$. $\square$

Claim 2. *With high probability, for any $B \in \mathcal{B}$, if $|B| < 4(\ln n)^{(k-(j+1)/k)} n^{(j+2)/k}$ for $j \in K$ then $|B \cap C_j| \leq 16n^{2/k} \ln n$.*

Proof. Union bound, Chernoff bound and $E[|B \cap C_j|] \leq 4(\ln n)^{(k-(j+1)/k)} n^{(j+2)/k} (n/\ln n)^{-j/k} \leq 4n^{2/k} (\ln n)^{(k-1)/k}$. $\square$

If $x \in C_j$, and $x \notin C_{j+1}$, define that node x has *rank* j. For every $u \in V$ and $i \in K$ define the *nearby landmarks* $S(u, i)$ to be the $n^{2/k} \log n$ closest nodes in C_i.

$$S(u, i) = N(u, 16n^{2/k} \log n, C_i)$$

and define $S(u) = \bigcup_{i \in K} S(u, i)$. Define $m(u, i)$ as the highest rank of any node in $A(u, i)$. Formally,

$$m(u, i) = \max \left\{\ell \in K \mid A(u, i) \cap C_\ell \neq \varnothing\right\} .$$

Define the *center* $c(u, i)$ as the closest node to u from $C_{m(u,i)}$. Let $E(u, i) = B(u, 2^{a(u,i+1)}/6)$. The main property of sparse levels is captured in the following lemma.

Lemma 3 (Sparse neighborhoods). *Let i be a sparse level for u, i.e., $a(u, i+1) > a(u, i) + 3$. If $v \in E(u, i)$, then $c(u, i) \in S(v)$.*

Proof. Recall that $E(u, i) = B(u, 2^{a(u,i+1)}/6)$ and $m(u, i)$ is the highest rank of any node in $A(u, i)$. Formally,

$$m(u, i) = \max \left\{\ell \in K \mid A(u, i) \cap C_\ell \neq \varnothing\right\} .$$

Let $j \in K$ be the index such that $4(\ln n)^{(k-j/k)} n^{j/k} \leq |A(u, i)| < 4(\ln n)^{(k-(j+1)/k)} n^{(j+1)/k}$ then from Claim 1 it follows that $m(u, i) \geq j$. For any $v \in E(u, i)$, we have (see Figure 2)

$$c(u, i) \in A(u, i) \subseteq E(u, i) \subseteq B(v, 2^{a(u,i+1)}/3) \subseteq B(u, 2^{a(u,i+1)}/2) .$$

Since level i is sparse for u, by definition there are strictly fewer than $n^{1/k}|A(u, i)|$ nodes in $B(u, 2^{a(u,i+1)}/2)$. Therefore

$$|B(v, 2^{a(u,i+1)}/3)| < n^{1/k}|A(u, i)| \leq 4(\ln n)^{(k-(j+1)/k)} n^{(j+2)/k} .$$

From Claim 2 and since $m(u, i) \geq j$ it follows that there are less than $16n^{2/k} \log n$ nodes of rank $m(u, i)$ in $B(v, 2^{a(v,i+1)}/3)$. Since $c(u, i) \in B(v, 2^{a(v,i+1)}/3)$ then $c(u, i) \in S(v)$ as required. $\square$

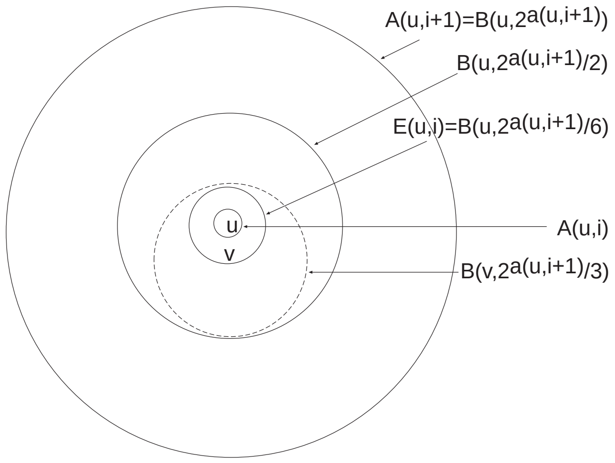

Figure 2: Example of a level i sparse neighborhood for node u and a node $v \in E(u, i)$.

3. A SCALE-FREE ROUTING SCHEME

The goal of this section is to present the construction for the following upper bound:

Theorem 1. *For each weighted n-node graph, and integer $k \geq 1$, there is a polynomial time constructible name-independent routing scheme with stretch factor $O(k)$ that uses $O(k^2 n^{1/k} \log^3 n)$-bit routing tables per node.*

The high level view of the routing scheme is 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. We begin by describing the two routing strategies.

3.1 Sparse neighborhood routing strategy

For every center $c(u, i)$ define $T(c(u, i))$ as a minimum cost path tree rooted at $c(u, i)$ that spans all nodes v such that $c(u, i) \in S(v)$. Routing on these trees is done using the following name-independent tree-routing scheme, which is an enhancement of Laing's algorithm [21].

Lemma 4. *For any $k \geq 1$, and for any weighted tree $T = (V, E, \omega)$ and for any designated root $r \in V$, there exists a name-independent error-reporting tree-routing scheme with the following properties:*

1. *Each node stores $O(kn^{1/k} \log^2 n)$ bits of routing information.*

2. *For any $j \in K$, the root can perform a j-*bounded search *for destination v.*

   A j-bounded search for v has the following properties:

   (a) *If $v \in N(r, n^{j/k}, V)$ then it reaches v with stretch $2j - 1$;*

   (b) *Otherwise it returns a negative response to the root incurring a cost of at most $(2j - 2) \max \left\{d(r, v) \mid v \in N(r, n^{(j-1)/k}, V)\right\}$.*

PROOF. Given a tree $T = (V, E)$ with $m \leq n$ nodes and a root $t \in T$. We give each node $v \in V$ three names. Let $a_0, \ldots, a_{m-1}$ denote the nodes of T sorted by increasing distance from the root. Formally, for any two indexes, if $i < j$ then $d_T(r, a_i) < d_T(r, a_j)$ or $d_T(r, a_i) = d_T(r, a_j)$ and a_i is lexicographically smaller than a_j. The first name we give nodes, called their primary name, makes use of words of the alphabet $\Sigma = \left\{0, 1, 2, \ldots n^{1/k} - 1\right\}$. Specifically, name $a_0 = r$ as the empty word (ε). Then name $a_1, \ldots, a_{n^{1/k}}$ respectively with one digit in Σ in increasing order $(0), (1), \ldots, (n^{1/k} - 1)$ respectively. Then name each of the next $n^{2/k}$ nodes $a_{1+n^{1/k}}, \ldots, a_{1+n^{1/k}+n^{2/k}}$ by a name in Σ^2 in increasing lexicographic order, $(0, 0), (0, 1), \ldots, (0, n^{1/k} - 1), (1, 0), (1, 1), \ldots, (1, n^{1/k}-1), \ldots, (n^{1/k}-1, 0), \ldots, (n^{1/k}-1, n^{1/k} - 1)$ respectively. Let $j_i = \sum_{j=0}^{i} n^{j/k}$. Continue this naming process, naming nodes $a_{j_{i-1}+1}, \ldots, a_{j_i}$ respectively by a name in Σ^i until all nodes in T are exhausted, up to at most a k-digit node name in Σ^k. For $0 \leq j \leq k$ let V_j denote the set of nodes whose name contains at most j digits. Next, we give v its name based on the labeled tree routing of Thorup and Zwick [29] and Fraigniaud and Gavoille [15]:

Lemma 5. **[15, 29]** *For every integer $k > 1$ and every weighted tree T with m nodes there exists a labeled routing scheme that, given any destination label, routes optimally on T from any source to the destination. The storage per node is $O(m^{1/k} \log m)$ bits, the label size, and the header size are $O(k \log m)$ bits.*

For a tree T containing a node v, let $\mu(T, v)$ denote the routing information of node v and $\lambda(T, v)$ denote the destination label of v in T as required from Lemma 5. We require from each node v to store $\mu(T, v)$. The second name we assign v is $\lambda(T, v)$. The third node-name makes use of a hash function $h : T \to \Sigma^k$. We require that, for all $0 \leq j \leq k$, $\max_{u \in \Sigma^{j-1}} |\{v \in V_j \mid u \text{ is a prefix of } h(v)\}| \leq |\Sigma| \log n = n^{1/k} \log n$. This requirement can be fulfilled with high probability using a $\Theta(\log n)$-wise independent hash function that requires $\Theta(\log^2 n)$ bits of storage [11, 23]. A node $u \in V$ with name $(x_1, \ldots, x_j)$ stores:

1. Information for labeled tree routing $\mu(T, u)$. This requires $O(n^{1/k} \log n)$ bits.
2. The labels $\lambda(T, v)$ of all the nodes v whose name is $(x_1, \ldots, x_j, y)$ for all $y \in \sigma$. This requires $O(kn^{1/k} \log n)$ bits.
3. The map $v \to \lambda(T, v)$ of the $n^{1/k} \log n$ closest nodes v (from the root) whose first j indexes of their hash $h(v)$ equal $x_1, \ldots, x_j$. Formally, define $Z = \{z \mid h(z)[1 \ldots j] = x_1, \ldots, x_j\}$ and store $v \to \lambda(T, v)$ for all $v \in N(r, n^{1/k} \log n, Z)$. This requires $O(kn^{1/k} \log^2 n)$ bits.

To search from the root r for a node t whose hash is $y_1, \ldots, y_k$ on a j-bounded search, do the following:

1. *current* $:= r$; *round* $:= 1$;
2. if *round* $= j$ and *current* does not know of t then return to root r with negative response.
3. Otherwise if *current* knows of t's label then route to t.
4. Otherwise route to the node whose name is $(y_1, \ldots, y_{round})$, set *round* $:=$ *round* $+ 1$, and update *current* to be the current node.
5. goto 2;

Suppose the destination is a node t whose hash is $y_1, \ldots, y_k$ and whose name has length i ($t \in V_i$ and $t \notin V_{i+1}$). The destination will be found after at most i iterations of the algorithm. This is true since if $i = 1$, then the root knows about the destination. Otherwise, at the $i - 1$th iteration we reach the node x whose name is $(y_1, \ldots, y_{i-1})$ and due to the properties of the hash function we know that $|\{v \in V_i \mid (y_1, \ldots, y_{i-1}) \text{ is a prefix of } h(v)\}| \leq |\Sigma| \log n$ and hence x stores the label of t.

Since all the nodes that are visited have smaller names than i, it is easy to see that the stretch is bounded by $2i-1 \leq 2k - 1$.

If a j-bounded search is performed and $j < i$ then t may not be found. At the $(j-1)$th iteration, a node whose name is $(y_1, \ldots, y_{j-1})$ will report the error to the root. Since all nodes visited have names with at most $j - 1$ digits then the total cost is bounded by $(2j - 2) \max_{v \in V_{j-1}} \{d(r, v)\}$.

The storage per node v is $O(\log^2 n) + O(|\Sigma| \log n) + O(k|\Sigma| \log n) + O(k|\Sigma| \log^2 n) = O(kn^{1/k} \log^2 n)$, for the hash function h, routing information $\mu(T, v)$, the primary name entries, and the hash name entries respectively. $\square$

For all $u \in V$ and $i \in K$ recall that $E(u, i) = B(u, 2^{a(u,i+1)}/6)$. Given a sparse level i the algorithm routes to the root $c(u, i)$. We prove that $E(u, i) \subseteq T(c(u, i))$ and hence searching from $c(u, i)$ for $v \in E(u, i)$ on the tree $T(c(u, i))$ will succeed with stretch of at most $2k-1$. In order to bound the cost incurred if $v \notin E(u, i)$ every node $u \in V$ stores for every $i \in K$ the index $b(u, i)$. Where $b(u, i)$ is the minimal integer j such that a j-bounded search on $T(c(u, i))$ of any node in $E(u, i)$ succeeds. Since $b(u, i)$ is defined as the minimal index to find in $T(c(u, i))$ all nodes of $E(u, i)$ then we prove that a negative result from this search has cost proportional at most to the diameter of $E(u, i)$. More precisely, we now define the information stored by every node, and the routing algorithm for sparse neighborhoods.

3.2 Storage for sparse neighborhood strategy

For any tree T and node $u \in T$, let $\tau(T, u)$ denote the information stored on u that is induced by the name-independent tree-routing scheme of Lemma 4. Every node $u \in V$ stores $\tau(T(v), u)$ for all $v \in S(u)$. In addition, for every $i \in K$, u records $c(u, i)$ and $b(u, i)$.

3.3 Routing algorithm for sparse neighborhood strategy

1. route to the root $c(u, i)$.
2. Perform a $b(u, i)$-bounded search on $T(c(u, i))$ for destination v.
3. If destination is not found, continue to iteration $i + 1$.

3.4 Dense neighborhood routing strategy

If i is a dense level, the routing strategy uses tree covers that have a bounded radius. A source of difficulty in arbitrarily weighted graphs is that the logarithm of the diameter may be arbitrarily large ($\gg n$, for example $\Delta = 2^n$).

Hence, tree covers of geometrically increasing radii, e.g., as used in [9, 3], require each node to participate in too many partition levels any may require $\Omega(n)$ bits per node. Our solution for general graphs is to let each node maintain routing information only for a limited number of radii that surround its range set. However, special care must be taken to make sure that the destination and all the nodes along the way store information for the same radius that the source is using. Recall that the range set is defined as $L(u) = \{a(u,i) \mid i \in K\}$, and the *extended range set* $R(u)$ is defined as, $R(u) = \{i \in I \mid \exists a \in L(u), -1 \le a - i \le 4\}$. For every $i \in I$ define $G_i = (V_i, E_i)$ as the subgraph induced by the nodes $V_i = \{u \mid i \in R(u)\}$. We prove that in a dense level i for u, if v is in $B(u, 2^{a(u,i)-1})$ then $i \in R(v)$ and moreover both source and destination are connected in G_i. Hence a tree cover in G_i may be used for routing. The tree cover is built using the construction of [9] with the improvements of [3].

Lemma 6. **[9, 3]** *For every weighted graph $G = (V, E, \omega)$, $|V| = n$ and integers $k, \rho \ge 1$, there exists a polynomial algorithm that constructs a collection of rooted trees $\mathcal{TC}_{k,\rho}(G)$ such that:*

1. *(Cover) For all $v \in V$, there exists $T \in \mathcal{TC}_{k,\rho}(G)$ such that $B(v, \rho) \subseteq T$.*
2. *(Sparse) For all $v \in V$, $|\{T \in \mathcal{TC}_{k,\rho}(G) \mid v \in T\}| \le 2kn^{1/k}$.*
3. *(Small radius) For all $T \in \mathcal{TC}_{k,\rho}(G)$, $\mathrm{rad}(T) \le (2k-1)\rho$, where $\mathrm{rad}(T) = \max_u \{d_T(r, u)\}$.*
4. *(Small edges) For all $T \in \mathcal{TC}_{k,\rho}(G)$, $\mathrm{maxE}(T) \le 2\rho$, where $\mathrm{maxE}(T) = \max_{e \in E(T)} \{\omega(e)\}$.*

For every $i \in I$ we build a tree cover $\mathcal{TC}_{k,2^i}(G_i)$ only on the graph G_i (note that G_i may have several connected components, so a tree cover is built for each connected component separately). Since $|R(u)| = O(k)$ then every node u participates only in $O(k)$ such tree covers. For all $u \in V$ and $i \in K$, denoting $j = a(u,i)$, define $W(u,i) \in \mathcal{TC}_{k,2^j}(G_j)$ to be the tree such that $B(u, 2^j) \subseteq W(u,i)$. On a dense level i, routing towards a destination begins by routing to the root of the tree $W(u,i)$ and then using a name-independent tree-routing scheme. For all $i \in I$ and $T \in \mathcal{TC}_{k,\rho}(G_i)$ we use the name-independent error-reporting tree-routing scheme of [3] with an improved analysis.

Lemma 7. *For every tree $T = (U, E, \omega)$, $|U| = m$, $U \subset V$, $|V| = n$, and integer k there exists a name-independent tree-routing scheme on T with error-reporting that routes on paths of length bounded by $4\mathrm{rad}(T) + 2k\mathrm{maxE}(T)$, each node requires $O(kn^{1/k}\log n)$ memory bits, and headers are of length $O(\log^2 n)$. Moreover, routing for a non-existent name in T also incurs a (closed) path of length $4\mathrm{rad}(T) + 2k\mathrm{maxE}(T)$ until a negative result is reported back to the source.*

Proof sketch. We use the same construction of [3], the only change is to use Lemma 5 that requires $O(k \log n)$ bit labels instead of $O(\log^2 n / \log\log n)$ bit labels used in [3]. Since each node stores only one such label, the space requirement is only $O(kn^{1/k}\log n)$ bits (all other labels used require only $O(\log n)$ bits). □

We prove that if i is a dense level for u then $\forall v \in F(u,i)$ we have $i \in R(v)$ (Lemma 2) and hence the tree routing scheme on $W(u,i)$ will reach any node in $F(u,i)$ or report a negative response at a cost proportional to k times the radius of $F(u,i)$ (recall $F(u,i) = B(u, 2^{a(u,i)-1})$). We now define the information stored by every node, and the routing algorithm for dense neighborhoods.

3.5 Storage for dense neighborhood strategy

For any tree T and node $u \in T$, let $\phi(T,u)$ denote the information stored on u that is induced by the name-independent error-reporting tree-routing scheme of Lemma 7. For every $u \in V$ and $i \in R(u)$ node u stores $\phi(T,u)$ for all $T \in \mathcal{TC}_{k,2^i}(G_i)$ such that $u \in T$. In addition, every node $u \in V$ records $w(u,i)$ the root of the tree $W(u,i)$.

3.6 Routing algorithm for dense neighborhood strategy

1. Route to the root $w(u,i)$.
2. Route on the tree $W(u,i)$ to either find destination v or get a negative response.
3. If destination is not found, continue with iteration $i+1$.

3.7 Analysis

Throughout the description below, the source node is denoted u and the destination node is denoted v. Routing from u to v is done by iteratively expanding the search through the neighborhoods of u, $A(u,1), A(u,2), \ldots, A(u,k)$ until the destination is found. The routing strategy in each neighborhood $A(u,i)$ depends on whether level i is sparse or dense for node u. If $A(u,i)$ is sparse, we use the sparse neighborhood strategy. If $A(u,i)$ is dense we use the dense neighborhood routing strategy.

The following technical lemmata are used to prove Theorem 1.

Lemma 8. *Let i be a dense level for u, and let $v \in F(u,i)$. Then a dense neighborhood routing strategy from u will reach v.*

Proof. Recall that $R(u) = \{i \mid \exists \ell \in L(u), -1 \le \ell - i \le 4\}$. Denote $a(u,i) = j$, by the properties of dense neighborhood stated in Lemma 2, for every $x \in F(u,i)$ node x has $a(u,i) \in R(x)$ and hence $x \in G_j$. Recall that the tree $W(u,i) \in \mathcal{TC}_{k,2^j}(G_j)$ is defined to be the tree such that $B_{G_j}(v, 2^j) \subseteq W(u,i)$. Therefore, $F(u,a) \subseteq W(u,i)$ and it is possible to reach v by routing on $W(u,i)$. □

Lemma 9. *Let $i \in K$ be a dense level for $u \in V$. Then the dense neighborhood routing strategy requires $O(k^3 n^{2/k}\log n)$ memory bits and incurs a cost of $O(k \cdot 2^{a(u,i)})$ either to reach v if $v \in B(u, 2^{a(u,i)-1})$ or otherwise to return a negative response to the source u.*

Proof. *Storage*: For the dense level routing strategy every node maintains $O(k)$ ranges. For each range, every node participates in $O(kn^{1/k})$ trees, each tree requires $O(kn^{1/k}\log n)$ bits. For a total of $O(k^3 n^{2/k}\log n)$ bits.

Routing cost: By combining Lemma 6 and Lemma 7 the cost of searching for v on $G_{a(u,i)}$ is at most $O(k \cdot 2^{a(u,i)})$. From Lemma 8, the destination will actually be found on $G_{a(u,i)}$ if $v \in F(u,i) = B(u, 2^{a(u,i)-1})$. □

Lemma 10. *Let i be a sparse level for u, and let $v \in E(u,i)$. Then the sparse neighborhood routing strategy from u will reach v.*

Proof. Denote $c = c(u,i)$, and recall that c is the closest landmark of highest rank in $A(u,i)$. By the properties of sparse neighborhoods stated in Lemma 3, every node $v \in E(u,i)$ has $c \in S(v)$. Hence, all nodes in $E(u,i)$ store $\tau(T(c))$, including all nodes on the shortest path of $T(c)$ from u to c. Since $b(u,i)$ is defined so that a $b(u,i)$-bounded search on $T(c)$ will find all nodes in $E(u,i)$ then $v \in E(u,i)$ will be found. $\square$

Lemma 11. *Let $i \in K$ be a sparse level for $u \in V$. Then the sparse neighborhood routing strategy requires $O(k^2 n^{3/k} \log^3 n)$ memory bits and incurs a cost of $O(k \cdot (d(u,v) + 2^{a(u,i)}))$ to reach v if $v \in B(u, 2^{a(u,i+1)}/6)$ or otherwise returns a negative response to the source at a cost $O(k \cdot 2^{a(u,i+1)})$.*

Proof. *Storage*: For the sparse level routing strategy every node maintains k closest landmark sets $S(u,1),\dots,S(u,k)$. For each set, a node participates in $16n^{2/k}\log n$ trees, each tree requires $O(kn^{1/k}\log^2 n)$ bits. For a total of $O(k^2 n^{3/k}\log^3 n)$ bits.

Routing cost: If $v \in E(u,i) = B(u, 2^{a(u,i+1)}/6)$, then by Lemma 10 the routing strategy will route from u to $c(u,i)$ and then to v. By definition $c(u,i) \in A(u,i)$ so the path to $c(u,i)$ incurs a cost of at most $d(u,c(u,i)) \le 2^{a(u,i)}$. Then by *Lemma 4* the $b(u,i)$-bounded search will find v at cost at most $(2k-1)d(c(u,i),v) \le (2k-1)(d(u,v)+d(u,c(u,i))) \le (2k-1)(d(u,v)+2^{a(u,i)})$. So the total cost is bounded by $2^{a(u,i)} + (2k-1)(d(u,v)+2^{a(u,i)}) = O(k(d(u,v)+a(u,i)))$ as required. Otherwise if $v \notin E(u,i) = B(u,2^{a(u,i+1)}/6)$ then the round-trip cost of reaching $c(u,i)$ and returning is at most $2^{a(u,i)+1}$. Then by Lemma 4 the cost of a $b(u,i)$-bounded search is bounded by $(2k-2)2^{a(u,i+1)}/6$. This is true since $b(u,i)$ is defined so that all the $n^{b(u,i)/k}$ closest nodes in $T(c(u,i))$ all belong to $E(u,i)$. Hence the total cost for a negative answer is $2^{a(u,i)+1} + (2k-2)2^{a(u,i+1)} = O(k \cdot 2^{a(u,i+1)})$. $\square$

We now prove the main theorem.

Proof of Theorem 1. For storage, by combining Lemma 11 and Lemma 9 it follows that every node stores $O(k^2 n^{3/k}\log^3 n)$-bit routing tables. For stretch analysis, let $i \in K$ be the first iteration index in which v is found. 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 11, v is not inside $E(u,i-1)$, and hence, $d(u,v) \ge 2^{a(u,i)}/6$.

2. Otherwise, level $i-1$ is dense for u, and again, v is not found in iteration $i-1$. In this case, by Lemma 9, v is not inside $F(u,i-1) = B(u,2^{a(u,i-1)}/2)$. By density, $a(u,i) \le a(u,i-1)+3$. Putting these two facts together, we have $d(u,v) \ge 2^{a(u,i-1)}/2 \ge 2^{a(u,i)-3}/2$.

In either case, $2^{a(u,i)} = O(d(u,v))$. From Lemma 11 and Lemma 9, reaching v on level i will cost $O(k \cdot 2^{a(u,i)})$ if i is dense or $O(k(d(u,v)+2^{a(u,i)}))$ if i is sparse. Hence in both cases, level i searches are bounded by $O(k \cdot d(u,v))$. For the cost of the negative responses, note that the highest level that fails is $i-1$. From Lemma 11 and Lemma 9 the cost of a negative response for level j is either $O(k \cdot 2^{a(u,j)})$ for a dense level, or $O(k \cdot 2^{a(u,j+1)})$ for a sparse level. Hence, the total cost of negative response is at most $\sum_{j=0}^{i-1} O(k \cdot 2^{a(u,j+1)}) = O(k \cdot 2^{a(u,i)}) = O(k \cdot d(u,v))$. $\square$

4. CONCLUSION

Our routing scheme can be adopted to work on strongly connected directed graphs, this extension will appear in the full paper. Our upper bounds have asymptotically optimal stretch with poly-logarithmic storage overhead. There are two natural open questions. First, what is the exact lowest stretch obtainable with a $\widetilde{O}(n^{1/k})$ memory? Even for the labeled case, the bound is not known to be tight. Second, what is memory requirement for schemes with $\Theta(\log n)$ stretch? Our upper bounds requires $O(\log^5 n)$ bits in such cases. We believe at least one logarithm can be removed by improving the hash function used by Lemma 5.

5. REFERENCES

[1] Ittai Abraham and Cyril Gavoille. Object location using path separators. In 25rd *Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, July 2006.

[2] Ittai Abraham, Cyril Gavoille, A.V. Goldberg, and Dahlia Malkhi. Routing in networks with low doubling dimension. Technical Report MSR-TR-2005-175, Miscrosoft Research, December 2005. To appear in The 26th International Conference on Distributed Computing Systems (ICDCS 06).

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

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

[5] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In 16th *Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA)*, pages 20–24. ACM Press, July 2004.

[6] Marta Arias, Lenore J. Cowen, Kofi Ambrose Laing, Rajmohan Rajaraman, and Orjeta Taka. Compact routing with name independence. In 15th *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 184–192. ACM Press, June 2003.

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

[8] Baruch Awerbuch, Amotz Bar-Noy, Nathan 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 31th *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 Journal on Discrete Mathematics*, 5(2):151–162, May 1992.

[11] J. Lawrence Carter and Mark N. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[12] T.-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. In 16th *Symposium on Discrete Algorithms (SODA)*, pages 762–771. ACM-SIAM, January 2005.

[13] Lenore J. Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, 38:170–183, 2001.

[14] Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. *Journal of Algorithms*, 46:97–114, 2003.

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

[16] Pierre Fraigniaud and Cyril Gavoille. A space lower bound for routing in trees. In 19th *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2285 of Lecture Notes in Computer Science, pages 65–75. Springer, March 2002.

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

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

[19] Goran Konjevod, Andréa Werneck Richa, and Donglin Xia. On optimal stretch name-independent compact routing in doubling metrics. In 25rd *Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, July 2006.

[20] Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. In 45th *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 434–443. IEEE Computer Society Press, October 2004.

[21] Kofi Ambrose Laing. Brief announcement: name-independent compact routing in trees. In 23rd *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 382–382. ACM Press, July 2004.

[22] Kofi Ambrose Laing and Rajmohan Rajaraman. Brief announcement: A space lower bound for name-independent compact routing in trees. In 17th *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, page 216. ACM Press, July 2005.

[23] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

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

[25] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, July 1989.

[26] Aleksandrs Slivkins. Distance estimation and object location via rings of neighbors. In 24th *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 41–50. ACM Press, July 2005.

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

[28] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, November 2004.

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

[30] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, January 2005.