On Approximate Distance Labels and Routing Schemes with Affine Stretch

Conference Paper · September 2011

DOI: 10.1007/978-3-642-24100-0_39 · Source: DBLP

2 authors:

Ittai Abraham
Microsoft
118 PUBLICATIONS 3,447 CITATIONS
SEE PROFILE

Cyril Gavoille
University of Bordeaux
231 PUBLICATIONS 3,839 CITATIONS
SEE PROFILE

On Approximate Distance Labels and Routing Schemes with Affine Stretch

Ittai Abraham[1] and Cyril Gavoille[2,⋆]

[1] Microsoft Research, Silicon Valley Center, USA
ittaia@microsoft.com
[2] Université de Bordeaux, LaBRI, France
gavoille@labri.fr

Abstract. For every integral parameter $k > 1$, given an unweighted graph G, we construct in polynomial time, for each vertex u, a distance label $L(u)$ of size $\tilde{O}(n^{2/(2k-1)})$. For any $u, v \in G$, given $L(u), L(v)$ we can return in time $O(k)$ an *affine* approximation $\hat{d}(u, v)$ on the distance $d(u, v)$ between u and v in G such that $d(u, v) \leqslant \hat{d}(u, v) \leqslant (2k-2)d(u, v) + 1$. Hence we say that our distance label scheme has affine stretch of $(2k-2)d + 1$. For $k = 2$ our construction is comparable to the $O(n^{5/3})$ size, $2d + 1$ affine stretch of the distance oracle of Pǎtraşcu and Roditty (FOCS '10), it incurs a $o(\log n)$ storage overhead while providing the benefits of a distance label. For any $k > 1$, given a restriction of $o(n^{1+1/(k-1)})$ on the total size of the data structure, our construction provides distance labels with affine stretch of $(2k-2)d+1$ which is better than the stretch $(2k-1)d$ scheme of Thorup and Zwick (J. ACM '05). Our second contribution is a compact routing scheme with poly-logarithmic addresses that provides affine stretch guarantees. With $\tilde{O}(n^{3/(3k-2)})$-bit routing tables we obtain affine stretch of $(4k-6)d+1$, for any $k > 1$. Given a restriction of $o(n^{1/(k-1)})$ on the table size, our routing scheme provides affine stretch which is better than the stretch $(4k-5)d$ routing scheme of Thorup and Zwick (SPAA '01).

1 Introduction

A *distance label* scheme is a pair of protocols. A *pre-processing* protocol takes a graph as input and maps each vertex u to a label $L(u)$ and a *query* protocol that takes two labels $L(u), L(v)$ as input and outputs an estimate $\hat{d}(u, v)$. Typically, distance label schemes are measured by (1) the time complexity of the prepossessing protocol, (2) the size of the labels, (3) the time complexity of the query algorithm, and (4) the quality of the distance estimation. Thorup and Zwick [17] prove that for any integer $k > 1$ it is possible to preprocess a graph in polynomial time to obtain labels of size at most[1] $\tilde{O}(kn^{1/k})$. Given any $L(u), L(v)$,

⋆ Also member of "Institut Universitaire de France". Supported by the ANR-project "ALADDIN", and équipe-projet commune LaBRI-INRIA Bordeaux Sud-Ouest "CÉPAGE". This work has been partially done during a visit at Microsoft in April 2011.
[1] Tilde-O notation are similar to O notation up to factors poly-logarithmic in n.

in time $O(k)$, a distance estimation $d(u,v) \leqslant \hat{d}(u,v) \leqslant (2k-1)d(u,v)$ is provided. Distance labels are a special type of a *Distance Oracle*, a global data structure that returns estimations on the all-pairs distance matrix. Distance labels have the benefit of allowing to easily partition the distance oracle into sub-regions. For example, suppose we have a distance label scheme for the whole world road network, it is easy to distribute to each mobile phone in a succinct manner the part of the map in the world that is relevant to that particular phone.

In this paper we focus on unweighted graphs and consider distance estimations $\hat{d}$ that have both a linear multiplicative term and an additive term. We say that a distance estimation $\hat{d}$ has *affine stretch* of $\alpha d+\beta$ if $d(u,v) \leqslant \hat{d}(u,v) \leqslant \alpha d(u,v)+\beta$ for all u,v. It is known that any distance oracle of $o(n^2)$ size must have affine stretch $\alpha d + \beta$ such that $\alpha + \beta \geqslant 3$, and this is true even if we restrict our attention to unweighted graphs (see [17]).

Pătraşcu and Roditty [14] prove that unweighted graphs have a distance oracle of $O(n^{5/3})$ expected size that provides affine stretch of $2d+1$. Given a restriction of $o(n^2)$ on the size of the distance oracle, the $2d+1$ affine stretch of [14] is better than the stretch $3d$ distance oracle of [17]. (The [17] scheme uses only $O(n^{3/2})$ memory). However, the scheme of [14] seems to require a global data structure. We could not see an obvious way to distribute this information into balanced labels of size $\tilde{O}(n^{2/3})$.

Our first contribution is a distance label scheme that for any integer $k > 1$, can be pre-processed in polynomial time and produce labels of size $\tilde{O}(n^{2/(2k-1)})$ that provide distance estimation in $O(k)$ time with affine stretch of $(2k-2)d+1$. Relative to the $2d+1$ result of [14], our $k=2$ scheme requires $O(\log^{2/3} n)$ more total memory but provides the benefits of a distance label. Note that for any $k > 1$, our $(2k-2)d+1$ scheme requires more memory than the [17] $(2k-1)d$ scheme (but has better stretch) and less memory than the [17] $(2k-3)d$ scheme (but has worse stretch). To highlight our contribution relative to the best previous results, consider a problem where there is some external restriction of $o(n^{1+1/(k-1)})$ on the total size of the distance oracle, and ask for the best affine stretch under such restrictions. Our construction provides distance labels with affine stretch of $(2k-2)d+1$ which is better than the best previous solution which obtains stretch $(2k-1)d$. Note that according to an Erdös-Simonovits conjecture about density of large girth graphs [8], any distance oracle on unweighted graphs with $o(n^{1+1/(k-1)})$ memory must have an affine stretch $\alpha d+\beta$ with $\alpha+\beta \geqslant 2k-1$.

Technically, we make two contributions. First we observe that the scheme of [14] naturally[2] generalizes to a distance oracle with $O(n^{1+2/(2k-1)})$ memory and affine stretch of $(2k-2)d+1$ using the hierarchical sampling scheme of [17]. The main technical contribution of [14] was a new ball growing protocol. Our second contribution it to show that for distance labels one can instead use the sampling technique of [16] to obtain similar bounds.

For stretch $2d+1$ with $\tilde{O}(n^{2/3})$ labels the high level idea is to build clusters $B(u)$ around each node u such that $|B(u)| = \tilde{O}(n^{1/3})$ and for every $w \in B(u)$,

[2] We have learnt that M. Patrascu, L. Roditty, M. Thorup have independently made a similar observation.

$|\{v : w \in B(v)\}| = \tilde{O}(n^{1/3})$. These clusters are built using the sampling technique of [16]. Given two nodes s and t the proof then proceeds much like [1,14], by separating into two cases. If $B(s) \cap B(t) \neq \varnothing$ then we can get the exact distance. Otherwise if $B(s)$ and $B(t)$ are disjoint, then we use the observation of [14] that the diameter of the smallest of the two balls is at most $(d(s,t)+1)/2$.

In the second part we study compact routing schemes with affine stretch. A routing scheme maps each vertex to some routing information (its routing table) and a poly-log size label (its address). Given a target label, the source needs to decide with its routing table how to forward a message using a poly-log sized header. The affine stretch of a routing scheme is a bound on the worst case route length taken by the routing scheme relative to the shortest path. Routing schemes are more challenging than distance labels since the source has access only to a poly-log size label of the target (while in the distance label model we had symmetric information about the source and target). Intuitively, this asymmetry generates difficulties in maintaining similar space-stretch trade-off in comparison with distance labeling or distance oracles. On the other hand, when progressing to the target, a message can profit from vertices it traverses. Nevertheless, for small distances in a sparse graph this advantage seems negligible. Given space bound of $\tilde{O}(n^{1/k})$ and $k > 2$, there are gaps between the best known distance labeling and the best known compact routing. Distance labeling achieve stretch $2k-1$ [17], whereas the best known routing schemes only achieves stretch $4k-5$ [16].

Our second contribution is a compact routing scheme that with $\tilde{O}(n^{3/(3k-2)})$-bit routing tables obtains affine stretch of $(4k-6)d+1$, for any $k > 1$. Our $(4k-6)d+1$ scheme requires more memory than the [16] $(4k-5)d$ scheme (but has better stretch) and less memory than the [16] $(4k-9)d$ scheme (but uses less memory). To highlight our contribution relative to the best previous results, consider a problem where there is some external restriction of $o(n^{1/(k-1)})$ on the size of the routing tables, and ask for the best affine stretch under such restrictions. Our construction provides a compact routing scheme with affine stretch of $(4k-6)d+1$ which is better than the best previous solution which obtains stretch $(4k-5)d$ (see [16]).

Our stretch $2d+1$ routing scheme is based on our approach for $2d+1$ labeling, but requires some additional ideas. The main new difficulty is when the smaller ball is around the source. This requires to redefine the ball around the source, spread routing information among vertices, and using hashing to find the relevant information. Such ideas were previously used (to the best of our knowledge) only for name-independent routing (for example, see [1]). For the general case we need an additional step. When the smaller ball is around the source we first try routing near the ball to gather information but if this fails we go back to the source and then proceed as in [16].

Related Work. One idea to reduce the size of the representation of distance information is to use sparse graph spanners, that is a spanning subgraph of the original graph using possibly less edges while preserving distances between vertices up to some estimation. A simple extension of the Kruskhal's greedy algorithm [3] shows that, for every $k \geqslant 1$, every weighted graph with n vertices has a spanner of size $O(n^{1+1/k})$ and stretch $(2k-1)d$. As quoted in the introduction, according to some girth conjecture, there are unweighted graphs on which every stretch $\alpha d + \beta$ spanner has size $\Omega(n^{1+1/k})$ if $\alpha + \beta < 2k+1$ (this is proved for $k = 1, 2, 3, 5$, and for all k if $\alpha = 1$ [19]). In the last decade many constructions have been obtained for different trade-offs between α and β, see for instance [5,6,7,13,18]. It seems that the picture of possible trade-offs is far from complete.

Graph spanners give a trivial compact data structure to represent approximate distances, however they do not give a fast way to extract approximate distances or to extract the best path in the spanner. Typically, sparse graphs (say with $o(n^{1+1/k})$ edges) have trivial spanners (the graph itself), but extracting short paths in the spanner is as hard as extracting shortest paths in the original graph. So, other more "structured" constructions of spanners have been given, based on tree-covers or partitions [4,12,17], that support fast approximate distance queries, and so lead to compact and fast distance oracles. They achieve space $\tilde{O}(n^{1+1/k})$, affine stretch $O(kd)$, and query time $O(k)$ (and even time $O(1)$ for [12]). Interestingly, [15] have showed that every distance oracle on sparse graphs that supports time t stretch αd distance queries must have space $n^{1+\Omega(1/(\alpha t))}$. So, the space bound of a distance oracle is constrained not only by the representation of approximate distances (or spanner representation) but also by the query time: fast distance oracles imply large space data structures, independent of the density of the input graph. In this context [14] have showed that, under a conjecture about set intersecting data structures, any distance oracle on sparse unweighted graphs of $\tilde{O}(n)$ edges with affine stretch $< 2d+1$ requires space $\Omega(n^{1.5})$.

Compact Routing has been already investigated in the late '70s for the very first interconnected computer networks [11]. Trade-offs between the size of the routing tables and the stretch on the route length are similar to the one achieved by spanners and distance oracles. Many results have been publish in this fields in the last decade, in particular for routing in specific graph classes (low dimension networks, scale free networks, planar networks, road networks, and so on), capturing the topology of real networks. For general graphs, the state-of-the-art is the Thorup and Zwick routing scheme [16] that, with $\tilde{O}(n^{1/k})$ routing tables and polylog addresses, routes along paths with affine stretch $(4k-5)d$. Like distance oracles, under a girth conjecture, the lower bound on the stretch is $\alpha + \beta \geqslant 2k-1$ for any stretch $\alpha d + \beta$ routing scheme of space $o(n^{1/(k-1)})$. So, optimal routing schemes is known only for $k = 1$ and 2. If we consider routing schemes with affine stretch $d + \beta$, then a lower bound of $\Omega(n/\beta^2)$ on the space exists [10]. This latter lower bound indicates, even if girth and set intersecting conjecture do not hold, that trade-offs in compact routing are different from those for spanners. While graph spanners of $o(n)$ edge density exist for affine stretch $d+2$ ([2]) and $d+6$ ([5]), affine stretch $d+\beta$, for *any* constant β, cannot be guaranteed by a $o(n)$ memory routing scheme even with arbitrarily large routing decision time.

2 Fast Approximate Distance Labeling

Theorem 1. *Let $k \geqslant 2$ be an integer. Every unweighted n-vertex graph enjoys a distance labeling with affine stretch $s(d) = (2k-3)d + 2\lceil d/2 \rceil \leqslant (2k-2)d + 1$, labels of length $\tilde{O}(n^{2/(2k-1)})$, and query time $O(k)$. Construction of the labels takes polynomial time.*

Hereafter, our constructions are described as randomized. Most of randomness comes from the construction of some small hitting sets, or vertex coloring, for which deterministic versions of polynomial time complexity exist.

For the sake of the presentation, we will sketch the first case of our distance labeling, whenever $k = 2$. It achieves stretch $d + 2\lceil d/2 \rceil \leqslant 2d + 1$ and labels of length $\tilde{O}(n^{2/3})$. Most of the ideas of this base case will be reused for the general construction, and also for routing.

Let $G = (V, E)$ be any unweighted graph with n vertices. The construction has some similarities with the technique used in the Thorup-Zwick compact routing scheme [16]. It is based on the selection of some subset of vertices $L \subseteq V$ named hereafter *landmarks*.

Given a set of landmarks L and a base set $W \subseteq V$, we define, for every vertex $u \in V$, its *ball* with respect to W and L as: $B_{W,L}(u) = \{v \in W : d(u,v) < d(u,L)\}$ where $d(u,L) = \min\{d(u,\ell) : \ell \in L\}$. In other words, $B_{W,L}(u)$ consists of all the closest vertices of W around u up to the closest landmark. For every $v \in W$, we define $C_{W,L}(v) = \{u \in V : v \in B_{W,L}(u)\}$.

We slightly generalize Theorem 3.1 of Thorup-Zwick [16] (proof in full version):

Lemma 1. *Given a graph $G = (V, E)$, size parameter s and base set $W \subseteq V$, one can construct in polynomial time a landmark set L such that for every vertex u of G, $|B_{W,L}(u)| \leqslant 4|W|/s$, $|C_{W,L}(u)| \leqslant 4|V|/s$, and, in expectation, $|L| \leqslant 4s \log |V|$.*

The key property of Lemma 1 is that *all* $C_{W,L}$'s balls are bounded by $O(|V|/s)$, and not only in expectation. In the remaining of this section, we will choose $W = V$, and for convenience, we will drop subscript V from B's and C's balls. So, $B_L(u) = B_{V,L}(u)$, and $C_L(v) = C_{V,L}(u)$. An important observation is that $u \in C_L(v)$ if and only if $v \in B_L(u)$.

We apply Lemma 1, so with $W = V$, and with $s = (n^2/\log n)^{1/3}$, so that $|L| = O((n \log n)^{2/3})$ and $|B_L(u)|, |C_L(u)| = O((n \log n)^{1/3})$.

Storage for Vertex u. It stores in its labels the set of vertices:

$$I(u) \ = \ L \cup B_L(u) \cup \left(\bigcup_{v \in B_L(u)} C_L(v) \right) .$$

It also stores the distances from u to each vertex $v \in I(u)$, and its closest landmark, say $\ell(u)$ (breaking ties arbitrary). We easily check that $|I(u)| = O((n \log n)^{2/3})$. Thus the label length is $O(n^{2/3} \log^{5/3} n)$ bits. The labeling scheme is clearly polynomially constructible.

Querying between s and t.

If $t \in I(s)$, then returns $d(s,t)$, else
returns $\min\{d(s,\ell(s)) + d(\ell(s),t), d(t,\ell(t)) + d(\ell(t),s)\}$.

The query can be solved using the labels of s and t only. It takes constant time to determine if $t \in I(s)$ using linear size and worst-case constant time static membership data-structures. Then, a linear size hashing tables can extract from $I(s)$ in constant time $d(s,w)$ for any vertex $w \in I(s)$ (see [17] for further details). So answering query (s,t) is done in constant time.

Stretch Analysis. If $t \in I(s)$, then the returned value is indeed the distance $d(s,t)$. Assume $t \notin I(s)$. We observe that $B_L(s)$ and $B_L(t)$ must be disjoint in that case. If not, say $v \in B_L(s) \cap B_L(t)$, then $v \in B_L(t)$ implies $t \in C_L(v)$. Since $v \in B_L(s)$ too, then $C_L(v) \subset I(s)$. Therefore, $t \in I(s)$: a contradiction.

Let $d = d(s,t)$, and let $\hat{d} = \min\{d(s,\ell(s)) + d(\ell(s),t), d(t,\ell(t)) + d(\ell(t),s)\}$ be the value returned by the oracle. Without loss of generality, assume that $d(s,\ell(s)) \leqslant d(t,\ell(t))$. Using the triangle inequality between $\ell(s)$ and t, we have:

$$\hat{d} \leqslant d(s,\ell(s)) + d(\ell(s),t) \ \leqslant\ d(s,\ell(s)) + (d(\ell(s),s) + d(s,t)) = 2d(s,\ell(s)) + d\ .$$

Consider a shortest path P from s to t. Let $x \in P \cap B_L(s)$ be the farthest from s, and, similarly, let $y \in P \cap B_L(t)$ be the farthest from t. Note that, since $B_L(s)$ and $B_L(t)$ are disjoint, $d(x,y) \geqslant 1$. Because x,y are on P, we have:

$$d \ = \ d(s,x) + d(x,y) + d(y,t) \ \geqslant\ d(s,x) + d(y,t) + 1\ . \qquad (1)$$

By definition of x, the neighbor of x on P at distance $d(s,x)+1$ from s is not in $B_L(s)$. So its distance to s is $d(s,x)+1 \geqslant d(s,\ell(s))$. The same argument applies to y and t, so that $d(t,y)+1 \geqslant d(t,\ell(t))$.

Plugging in Eq. (1), and combining with the assumption $d(s,\ell(s)) \leqslant d(t,\ell(t))$, we obtain:

$$d \ \geqslant\ d(s,\ell(s)) + d(t,\ell(t)) - 1 \ \geqslant\ 2d(s,\ell(s)) - 1$$

In other words, $d(s,\ell(s)) \leqslant \lfloor (d+1)/2 \rfloor = \lceil d/2 \rceil$. Since, $\hat{d} \leqslant 2d(s,\ell(s)) + d$, we have proved that $\hat{d} \leqslant d + 2\lceil d/2 \rceil$.

General Construction: $k \geqslant 2$. For the general case, we will make the use of *k levels* of landmark sets: $L_0, L_1, \ldots, L_{k-1}$ where $L_0 = V$ for convenience. We denote by $\ell_i(u)$ the closest landmark from u in L_i (breaking ties arbitrary). Note that $\ell_0(u) = u$. The notations $\ell(u)$ and L for case $k=2$ simply denotes here $\ell_1(u)$ and L_1 respectively.

The collection of landmark sets is constructed by applying $k-1$ times Lemma 1. The first time, we apply it with base set $W_1 = L_0 = V$ and size parameter $s_1 = n^{1-1/(2k-1)}$ so $|B_{L_1}(u)|, |C_{L_1}(u)| = O(n/s_1) = \tilde{O}(n^{1/(2k-1)})$. Then, for each $i \in [2, k-1]$, we fix the base set $W_i = L_{i-1}$ and $s_i = n^{1-(2i-1)/(2k-1)}$. Note that $s_{k-1} = n^{2/(2k-1)}$, and that $s_{i-1}/s_i = n^{2/(2k-1)}$ for all $i \geqslant 1$.

Storage for Vertex u. It stores in its labels the set of vertices:

$$I(u) \;=\; L_{k-1} \cup \left(\bigcup_{i=1}^{k-1} B_{L_{i-1},L_i}(u) \right) \cup \left(\bigcup_{v \in B_{L_1}(u)} C_{L_1}(v) \right) .$$

It also stores in its label the distances from u to each vertex $v \in I(u)$, and its sequence of closest landmarks $\ell_0(u), \ldots, \ell_{k-1}(u)$. Note that the distance to each landmark $\ell_i(u)$ is already in the label of u.

Applying Lemma 1, the size of $I(u)$ is (details in the full version): $|I(u)| = \tilde{O}(s_{k-1}) + \tilde{O}\left(\sum_{i=2}^{k-1}(s_{i-1}/s_i)\right) + O(n/s_1)^2 = \tilde{O}(n^{2/(2k-1)})$.

Querying between s and t.

1. If $d(s,\ell_1(s)) > d(t,\ell_1(t))$, exchange the role of s and t in the next two steps.
2. Compute the smallest index i_0 such that $\ell_{2i_0}(t) \in I(s)$ or $\ell_{2i_0+1}(s) \in I(t)$.
3. If $\ell_{2i_0}(t) \in I(s)$ returns $d(s,\ell_{2i_0}(t)) + d(\ell_{2i_0}(t),t)$, else returns $d(s,\ell_{2i_0+1}(s)) + d(\ell_{2i_0+1}(s),t)$.

Intuitively, the answering algorithm tries to approximate the distance successively thanks to the sequence of landmarks $\ell_0(t), \ell_1(s), \ell_2(t), \ell_3(s), \ldots$, respectively with the paths $s \to \ell_i(t) \to t$ for even i, and $s \to \ell_i(s) \to t$ for odd i. The query can be solved using the labels of s and t only. It takes $O(k)$ to determine i_0 using static dictionary. Then, it takes constant time to return the distance using hash table.

Stretch Analysis for $k \geqslant 2$. Appears in the full version.

3 Compact Routing

Theorem 2. *Given an unweighted connected graph with n vertices and an integral parameter $k \geqslant 2$, there exists a polynomial algorithm that produces a labeled routing scheme which requires $\tilde{O}(n^{3/(3k-2)})$-bit routing tables per vertex, $o(k \log^2 n)$ labels, and $o(\log^2 n)$ headers, that performs routing decisions in $O(k)$ time and routes along paths of affine stretch at most $(4k-7)d + 2\lceil d/2 \rceil \leqslant (4k-6)d + 1$.*

To prove Theorem 2, we need to introduce a third kind of ball, defined by volume. For a vertex u, set L, and parameter t, let $E_L(u,t)$ be the subset of L that consists of the t closest vertices to u (breaking ties with any uniform policy). We can check that B, C, E balls share the following monotonicity property which is important for routing:

Property 1. Let $X(\cdot)$ denote one type of ball among $B_L(\cdot)$, $C_L(\cdot)$, and $E_L(\cdot,t)$, for a given set $L \subseteq V$ and parameter t. Then, if $t \in X(s)$ and u is on a shortest path from s to t, then $t \in X(u)$.

A key technique in the proof of Theorem 2 is to spread routing information using a commonly known hash function. This idea is standard in *Name-Independent* routing (see for example [1]). To the best of our knowledge this is the first use of such technique for *Labeled* routing (where vertices are given poly-log size labels).

We also make use of the following labeled routing scheme for trees:

Lemma 2. [9,16] *Every spanning tree T of a graph with n vertices has a labeled routing scheme that, given any destination label, routes optimally on T from any source to the destination. The storage per vertex, the label size, and the header size are $O(\log^2 n/\log\log n)$ bits. Given the stored information of a vertex and the label of the destination, routing decisions take constant time.*

For a tree T containing a vertex v, let $\mu(T,v)$ denote the routing information stored at vertex v and $\lambda(T,v)$ denote the destination label of v in T as defined by the labeled routing scheme of Lemma 2.

For the sake of the presentation, we present first the basic construction for $k=2$. Then, we present the construction for $k=3$ which introduces new tools needed for $k>2$.

Proof of Theorem 2 for $k=2$. The routing tables in this case have size $\tilde{O}(n^{3/4})$, and the routing scheme has affine stretch $d+2\lceil d/2\rceil \leqslant 2d+1$.

Let L be a set of landmarks such that for all $u \in V$, $|B_L(u)|, |C_L(u)| = \tilde{O}(n^{1/4})$. Such a set can be obtained with $|L| = \tilde{O}(n^{3/4})$, by choosing in Lemma 1 $s=(n\log n)^{3/4}$ and $W=V$.

Let $E(u)=E_V(u, 2n^{2/4}\log n)$ and let $e(u)=\max\{d(u,v) : v\in E(u)\}$ be the radius of the ball $E(u)$. Let $\ell(u)$ be the closest vertex in L to u (this vertex defines the radius of $B_L(u)$). Let $c: V \to [1, n^{1/4}]$ be a hash function that maps vertices into $n^{1/4}$ colors, with the following two properties:

(1) $\forall u\in V$ and $j\in[1,n^{1/4}]$, $E(u)$ contains a vertex $\ell\in L$ such that $c(\ell)=j$;
(2) $\forall j\in[1,n^{1/4}]$, the number of vertices with $c(u)=j$ is at most $2n^{3/4}$.

Clearly a $O(\log n)$-wise independent hash function has this property but known constructions of such functions require non-constant time (poly-log) to evaluate $c(u)$. To get constant time, we use the construction of [1] to obtain a hash function that can be represented using $O(n^{3/4})$ bits and allows computing $c(u)$ is constant time.

For every $x\in V$, let $T(x)$ be a spanning shortest path tree rooted at x.

High Level Idea. Given a source u and target v there will be three cases. First, if $E(u)\cap B(v)\neq\varnothing$ then we will route with affine stretch d (shortest path). Otherwise the balls $E(u)$ and $B(v)$ are disjoint. If the radius of $B(v)$ is smaller than the one of $E(u)$ then we route on shortest path to the closest landmark to v and then on a shortest path to v, so the affine stretch is $2d+1$. Otherwise, to get affine stretch $2d+1$ we need to route on shortest path to some landmark $w\in E(u)$ and then on shortest path from w to v. The problem is that w needs to know the label of v on $T(w)$. The label of v cannot store this information because there are too many landmarks. So this information must be stored in w. So the landmarks in $E(u)$ need to collectively store labels of all n destinations. By using a random hash function c we speared this $\tilde{O}(n)$ bits of information over the $O(n^{1/4})$ landmarks in $E(u)$ so that each landmark in $E(u)$ stores only $\tilde{O}(n^{3/4})$ bits of information.

Label for a vertex v is $\langle v, \ell(v), d(v, \ell(v)), \lambda(T(\ell(v)), v)\rangle$ of $O(\log^2 n/\log\log n)$ bits from Lemma 2.

Storage for a Vertex u:

(1) For every $x \in E(u)$, store routing information $\mu(T(x), u)$ of the tree $T(x)$. Also store $e(u)$.
(2) For every $v \in C_L(w)$ such that $w \in E(u)$ and w is on the shortest path from u to v, store $\langle v, w, \lambda(T(w), v)\rangle$.
(3) For every vertex $\ell \in L$, store routing information $\mu(T(\ell), u)$ of the tree $T(\ell)$.
(4) If $u \in L$, then for every vertex v such that $c(v) = c(u)$, store $\lambda(T(u), v)$.

Storing (1) requires $\tilde{O}(n^{2/4})$ space since $|E(u)| = \tilde{O}(n^{2/4})$. (2) requires $\tilde{O}(n^{3/4})$ space since $|C_L(w)| = \tilde{O}(n^{1/4})$ for each $w \in E(u)$. (3) requires $\tilde{O}(n^{3/4})$ space since $|L| = \tilde{O}(n^{3/4})$. (4) requires $\tilde{O}(n^{3/4})$ space from the second property of c. Overall, the storage for u is $\tilde{O}(n^{3/4})$.

Routing from u to v. Given the label of v, routing from u to v at distance d is done in the following manner:

1. If exists $w \in E(u)$ such that $w \in B_L(v)$ (this can be checked using (2) and in constant time using a static dictionary), then route to w using (1) and from w to v using (2). The affine stretch is d. Otherwise, it must be that $E(u)$ and $B_L(v)$ are disjoint (here we use $x \in B_L(v)$ implies $y \in B_L(v)$ for all $d(v, y) \leqslant d(v, x)$ - note that this is not necessary true for $E(u)$). Since the graph is unweighted then $\min\{e(u), d(v, \ell(v))\} \leqslant \lceil d/2 \rceil$.
2. If $e(u) > d(v, \ell(v))$ (checked using (1) and v's label) then route to $\ell(v)$ using (3). Then route from $\ell(v)$ to v using (3) and the label of v (that contains $\lambda(T(\ell(v)), v)$). From the triangle inequality, the affine stretch is $d + 2\lceil d/2 \rceil$.
3. Otherwise $e(s) \leqslant d(v, \ell(v))$ then route to some $\ell \in L \cap E(u)$ such that $c(\ell) = c(v)$ (using (3)) then using (4) on ℓ route to v. From the triangle inequality, the affine stretch is $d + 2\lceil d/2 \rceil$.
   Note that such an ℓ exists from the first property of c.

In all the cases, the affine stretch is $d + 2\lceil d/2 \rceil \leqslant 2d + 1$ as required. Using standard dictionary and hashing techniques, each of the routing decisions above can be done in constant time.

Proof of Theorem 2 for $k = 3$. Let L_1 be a set of landmarks such that for all $u \in V$, $|B_{L_1}(u)|, |C_{L_1}(u)| = \tilde{O}(n^{1/7})$. Such a set can be obtained with $|L_1| = \tilde{O}(n^{6/7})$, by choosing in Lemma 1 $s = (n \log n)^{6/7}$ and $W = V$.

Using Lemma 1 with $W = L_1$ and $s = (n \log n)^{3/7}$ let $L_2 \subset L_1$ be a set of *super landmarks* such that: (1) $|L_2| = \tilde{O}(n^{3/7})$; (2) for all $u \in V$, $|B_{L_1,L_2}(u)| = \tilde{O}(n^{3/7})$; (3) for all $\ell \in L_1$, $|C_{L_1,L_2}(\ell)| = O(n^{4/7})$.

Let $\ell_1(u)$ be the closest node in L_1 to u. Let $\ell_2(u)$ be the closest node in L_2 to u. Let $E_1(u) = E_V(u, 2n^{2/7} \log n)$ and $e_1(u) = \max\{d(u,v) : v \in E_1(u)\}$. Let $E_2(u) = E_{L_1}(u, 2n^{3/7} \log n)$ and $e_2(u) = \max\{d(u,v) : v \in E_2(u)\}$. Let $c : V \to [1, n^{1/7}]$ be a random coloring that maps vertex into $n^{1/7}$ colors. We need the following properties:

(1) $\forall u \in V$ and $j \in [1, n^{1/7}]$, $E_1(u)$ contains a vertex $\ell \in L_1$ such that $c(\ell) = j$;
(2) $\forall u \in L_1$ and $j \in [1, n^{1/7}]$, $|\{w \in V \cap C_{L_1,L_2}(u) : c(w) = j\}| \leqslant 2n^{3/7} \log n$.

Again we use a similar construction to that of [1] to obtain c with the above properties that requires $\tilde{O}(n^{3/7})$ bits and can be evaluated in constant time.

High Level Idea. Given a source u and target v there will again be three main cases. First, if $E_1(u) \cap B_{L_1}(v) \neq \varnothing$ then we will route with affine stretch d (shortest path). Otherwise the balls $E_1(u)$ and $B_{L_1}(v)$ are disjoint. If $B_{L_1}(v)$ is smaller than $E_1(u)$ then we try to route using $\ell_1(v)$ and get affine stretch of $2d+1$, but if u does not know about $\ell_1(v)$ then we use $\ell_2(v)$. As in [16] each time we move from $\ell_i(v)$ to $\ell_{i+1}(v)$ we apply the triangle inequality an loose $4d$ each step. For $k = 3$ we do this once to obtain affine stretch of $6d+1$.

Otherwise, (as in the $k = 2$ case) we route to the $\ell \in L_1 \cap E_1(u)$ such that $c(\ell) = c(v)$. If ℓ knows about v then get affine stretch of $2d+1$. Otherwise we route back to u (and pay $2d+1$). We show that $d(v, \ell_2(v)) \leqslant d + (d+1)/2$. So we try to route using $\ell_2(v)$ to get affine stretch of $(2d+1)+(4d+1) \leqslant 6d+1$. In the general case, if u does not know about $\ell_i(v)$, then we test if u knows about $\ell_{i+1}(v)$. As in [16] each time we move from $\ell_i(v)$ to $\ell_{i+1}(v)$ we apply the triangle inequality and loose $4d$ in each such iteration.

Label for a vertex v is $\langle v, \ell_1(v), d(v, \ell_1(v)), \lambda(T(\ell_1(v)), v), \ell_2(v), \lambda(T(\ell_2(v)), v)\rangle$, which is $O(\log^2 n / \log\log n)$ bits from Lemma 2.

Storage for a Vertex u:

(1) For every vertex $x \in E_1(u)$, store routing information $\mu(T(x), u)$ of the tree $T(x)$. Also store $e_1(u)$.
(2) For every $v \in C_{L_1}(w)$ such that $w \in E_1(u)$ and w is on the shortest path from u to v, store $\langle v, w, \lambda(T(w), v)\rangle$. This is $\tilde{O}(n^{3/7})$ storage since $|E_1(u)| \cdot \max|C_{L_1}(w)| = \tilde{O}(n^{3/7})$.
(3) For every vertex $\ell \in E_2(u)$, store routing information $\mu(T(\ell), u)$ of the tree $T(\ell)$. This is $\tilde{O}(n^{3/7})$ storage since $|E_2(u)| = \tilde{O}(n^{3/7})$.
(4) If $u \in L_1$ then for every vertex $v \in C_{L_1,L_2}(u)$ such that $c(v) = c(u)$ store $\lambda(T(u), v)$. This is $\tilde{O}(n^{3/7})$ storage by the property of c.
(5) For every vertex $\ell \in L_2$, store routing information $\mu(T(\ell), u)$ of the tree $T(\ell)$. This is $\tilde{O}(n^{3/7})$ storage since $|L_2| = \tilde{O}(n^{3/7})$.

Routing from u to v. Given the label of v, routing from u to v at distance d is done in the following manner:

1. If exists $w \in E_1(u)$ such that $w \in B_{L_1}(v)$ then route to w using (1) and from w to v using (2). The routing has affine stretch of d (shortest path). Otherwise it must be that $E(u)$ and $B_L(v)$ are disjoint. Since the graph is unweighted then $\min\{e(u), d(v, \ell(v))\} \leqslant \lceil d/2 \rceil$.
2. If $e_1(u) > d(v, \ell_1(v))$ (this can be checked using (1) and v's label) then:
   (a) If $\ell_1(v) \in E_2(u)$ then route to $\ell_1(v)$ using (3) and from $\ell_1(v)$ to v using (3) and the label of v. From the triangle inequality, the affine stretch is $d + 2\lceil d/2 \rceil$.
   (b) Otherwise from the triangle inequity it must be that $e_2(u) \leqslant d + \lceil d/2 \rceil$ and since $|E_2(u)| = \Theta(n^{3/7})$ it follows that $d(u, L_2) \leqslant e_2(u)$. So from the triangle inequality $d(v, L_2) \leqslant 2d + \lceil d/2 \rceil$. So using (5) route from u to $\ell_2(v)$ and from $\ell_2(v)$ to v using (5) and using v's label that contains $\lambda(T(\ell_2(v)), v))$. From the triangle inequality, the affine stretch is at most $(3d + \lceil d/2 \rceil) + (2d + \lceil d/2 \rceil) \leqslant 5d + 2\lceil d/2 \rceil \leqslant 6d + 1$ as required.
3. Otherwise $e_1(u) \leqslant d(v, \ell_1(v))$ then:
   (a) Route to some $\ell \in L_1 \cap E_1(u)$ such that $c(\ell) = c(v)$ (using (3)). Note that such a ℓ must exist from the properties of c. If $v \in C_{L_1,L_2}(\ell)$ then using (4), route on $T(\ell)$ to v. From the triangle inequality, the affine stretch is $d + 2\lceil d/2 \rceil$.
   (b) Otherwise, $\ell \notin B_{L_1,L_2}(v)$ hence from the triangle inequality, $d(v, \ell_2(v)) \leqslant 2d + \lceil d/2 \rceil$. So route from ℓ back to u on $T(\ell)$ using (3) and u's label. From u route to $\ell_2(v)$ and from $\ell_2(v)$ to v using (5) and since v's label contains $\lambda(T(\ell_2(v)), v))$. The affine stretch of going to ℓ and back to u is at most $d + \lceil d/2 \rceil$. The affine stretch of going from u to $\ell_2(v)$ and then to v is at most $(2d + \lceil d/2 \rceil) + (d + \lceil d/2 \rceil) \leqslant 3d + 2\lceil d/2 \rceil$. So the total affine stretch is $4d + 3\lceil d/2 \rceil \leqslant 5d + 1 + \lceil d/2 \rceil \leqslant 5d + 2\lceil d/2 \rceil \leqslant 6d + 1$.

In all the cases, the affine stretch is $5d+2\lceil d/2 \rceil \leqslant 6d+1$ as required. Again using standard dictionary and hashing techniques all routing decision can be made in constant time.

4 Conclusion

We have provided new distance label and compact routing schemes that provide affine stretch guarantees for unweighted graphs. Our results obtain new space-stretch trade-offs that neither dominate or are dominated by the best known previous constructions. So there exist restrictions on either the space or the stretch, for which our results provide improved bounds.

There are several questions that remain open. We believe the most intriguing problem we leave open is:

> Design a constant time (or poly-log time) approximate distance oracle for unweighted graphs with affine stretch $3d + 2$ and space $o(n^{3/2})$, or show that such construction is not possible.

Note that it is easy to construct a spanner with $O(n^{4/3})$ edges and affine stretch $3d+2$, for instance using the construction of [5] for $k=3$ with size $O(n^{1+1/k})$ and stretch $kd+k-1$.

When focusing just on affine stretch of $\alpha d+1$, the natural question is whether the memory requirements of our schemes can be improved. Specifically the current state of upper bounds indicates four different bounds: for spanners, distance oracles, fast query labels, and routing schemes.

For this abstract we did not optimize the time construction and poly-log factors in the label or routing table size. We also plan to extend our result to weighted sparse graphs using the same approach as [14].

References

1. Abraham, I., Gavoille, C., Malkhi, D., Nisan, N., Thorup, M.: Compact name-independent routing with minimum stretch. ACM Trans. on Algo. 3 (2008)
2. Aingworth, D., Chekuri, C., Indyk, P., Motwani, R.: Fast estimation of diameter and shortest paths (without matrix multiplication). SIAM J. on Comp. 28 (1999)
3. Althöfer, I., Das, G., Dobkin, D.P., Joseph, D.A., Soares, J.: On sparse spanners of weighted graphs. Discr. & Comp. Geom. 9, 81 (1993)
4. Awerbuch, B., Peleg, D.: Sparse partitions. In: FOCS, p. 503. IEEE Press, Los Alamitos (1990)
5. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: Additive spanners and (α,β)-spanners. ACM Trans. on Algo. 7, A.5 (2010)
6. Elkin, M.: Computing almost shortest paths. ACM Trans. on Algo. 1, 323 (2005)
7. Elkin, M., Peleg, D.: $(1+\epsilon,\beta)$-spanner constructions for general graphs. SIAM J. on Comp. 33, 608 (2004)
8. Erdös, P.: Extremal problems in graph theory, p. 29. Publ. House Cszechoslovak Acad. Sci., Prague (1964)
9. Fraigniaud, P., Gavoille, C.: Routing in trees. In: Yu, Y., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 757–772. Springer, Heidelberg (2001)
10. Gavoille, C., Sommer, C.: Sparse spanners vs. compact routing. In: SPAA, p. 225. ACM Press, New York (2011)
11. Kleinrock, L., Kamoun, F.: Hierarchical routing for large networks; performance evaluation and optimization. Computer Networks 1, 155 (1977)
12. Mendel, M., Naor, A.: Ramsey partitions and proximity data structures. In: FOCS, p. 109. IEEE Comp. Soc. Press, Los Alamitos (2006)
13. Pettie, S.: Low distortion spanners. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 78–89. Springer, Heidelberg (2007)
14. Pătraşcu, M., Roditty, L.: Distance oracles beyond the Thorup-Zwick bound. In: FOCS, p. 815. IEEE Comp. Soc. Press, Los Alamitos (2010)
15. Sommer, C., Verbin, E., Yu, W.: Distance oracles for sparse graphs. In: FOCS, p. 703. IEEE Comp. Soc. Press, Los Alamitos (2009)
16. Thorup, M., Zwick, U.: Compact routing schemes. In: SPAA, p. 1. ACM Press, New York (2001)
17. Thorup, M., Zwick, U.: Approximate distance oracles. J. ACM 52, 1 (2005)
18. Thorup, M., Zwick, U.: Spanners and emulators with sublinear distance errors. In: SODA, p.802. ACM-SIAM (2006)
19. Woodruff, D.P.: Lower bounds for additive spanners, emulators, and more. In: FOCS, p. 389. IEEE Comp. Soc. Press, Los Alamitos (2006)