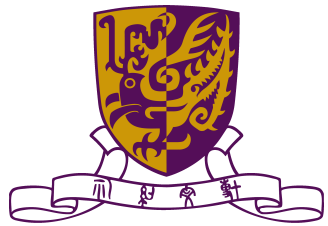
Approximate Shortest Distance Computing: A Query-Dependent Local Landmark Scheme

Miao Qiao, Hong Cheng, Lijun Chang and Jeffrey Xu Yu



The Chinese University of Hong Kong
{mqiao, hcheng, ljchang, yu}@se.cuhk.edu.hk

March 30, 2012

Landmark Embedding Scheme for Shortest Distance Query

Landmark Embedding

- *Graph*: $G(V, E, W)$.
- *Landmarks*: $S = \{l_1, \ldots, l_k\} \subseteq V$.
- *Embedding*: $\forall l \in S, v \in V$, compute $\delta(l, v)$.
- *Querying*: $q = (a, b)$, estimate

$$\widetilde{\delta}(a, b) = \min_{l_i \in S} \{\delta(l_i, a) + \delta(l_i, b)\}$$

Widely used in

- Road networks [11, 4];
- Social networks and web graphs [8, 7, 10, 2];
- Internet graphs[1, 5];

Landmark Embedding Scheme for Shortest Distance Query

Landmark Embedding

- *Graph*: $G(V, E, W)$.
- *Landmarks*: $S = \{l_1, \ldots, l_k\} \subseteq V$.
- *Embedding*: $\forall l \in S, v \in V$, compute $\delta(l, v)$.
- *Querying*: $q = (a, b)$, estimate

$$\widetilde{\delta}(a, b) = \min_{l_i \in S} \{\delta(l_i, a) + \delta(l_i, b)\}$$

Widely used in

- Road networks [11, 4];
- Social networks and web graphs [8, 7, 10, 2];
- Internet graphs[1, 5];

Major Differences among Existing Landmark Embedding Approaches

Landmark selection strategy

- Optimal landmark set selection: NP-Hard!
  - Betweenness centrality based criterion(Potamias et al. [7]).
  - Minimum K-center formulation(Francis et al. [1]).
- Random selection([3, 12, 8, 10, 2]).
- Graph measure based heuristics.
  - Approximate centrality based selection (Potamias et al. [7]).
- Drawback: the performance largely depends on graph properties.

Landmark organization

- Hierarchical(Kriegel et al. [4]).
- Sketch-based Landmark(Sarma et al. [10], Gubichev et al. [2]).

Error bound or not

- $(2k-1)$-approximation with $O(kn^{1+1/k})$ memory(Thorup and Zwick [12]).

Major Differences among Existing Landmark Embedding Approaches

Landmark selection strategy

- Optimal landmark set selection: NP-Hard!
  - Betweenness centrality based criterion(Potamias et al. [7]).
  - Minimum K-center formulation(Francis et al. [1]).
- Random selection([3, 12, 8, 10, 2]).
- Graph measure based heuristics.
  - Approximate centrality based selection (Potamias et al. [7]).
- Drawback: the performance largely depends on graph properties.

Landmark organization

- Hierarchical(Kriegel et al. [4]).
- Sketch-based Landmark(Sarma et al. [10], Gubichev et al. [2]).

Error bound or not

- $(2k-1)$-approximation with $O(kn^{1+1/k})$ memory(Thorup and Zwick [12]).

Major Differences among Existing Landmark Embedding Approaches

Landmark selection strategy

- Optimal landmark set selection: NP-Hard!
  - Betweenness centrality based criterion(Potamias et al. [7]).
  - Minimum K-center formulation(Francis et al. [1]).
- Random selection([3, 12, 8, 10, 2]).
- Graph measure based heuristics.
  - Approximate centrality based selection (Potamias et al. [7]).
- Drawback: the performance largely depends on graph properties.

Landmark organization

- Hierarchical(Kriegel et al. [4]).
- Sketch-based Landmark(Sarma et al. [10], Gubichev et al. [2]).

Error bound or not

- $(2k-1)$-approximation with $O(kn^{1+1/k})$ memory(Thorup and Zwick [12]).

Major Differences among Existing Landmark Embedding Approaches

Landmark selection strategy

- Optimal landmark set selection: NP-Hard!
  - Betweenness centrality based criterion(Potamias et al. [7]).
  - Minimum K-center formulation(Francis et al. [1]).
- Random selection([3, 12, 8, 10, 2]).
- Graph measure based heuristics.
  - Approximate centrality based selection (Potamias et al. [7]).
- Drawback: the performance largely depends on graph properties.

Landmark organization

- Hierarchical(Kriegel et al. [4]).
- Sketch-based Landmark(Sarma et al. [10], Gubichev et al. [2]).

Error bound or not

- $(2k-1)$-approximation with $O(kn^{1+1/k})$ memory(Thorup and Zwick [12]).

Other Shortest Distance Computing Approaches

Spatial Networks

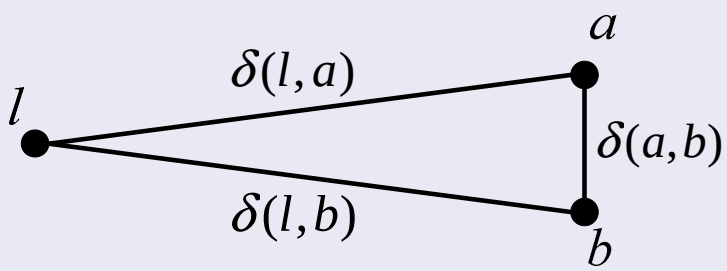
- Euclidean distance as a lower bound to prune the search space (Papadias et al. [6]).
- Path-distance oracle of size $O(n \cdot \max(s^d, \frac{1}{\epsilon}^d))$, and answer a query with ϵ-approximation in $O(\log n)$ time (Sankaranarayanan et al. [9]).
- Drawback: coordinate dependent.

Exact distance indexing on non-spatial Networks

- Tree-width decomposition based [13].
- Drawback: not scalable.

Global Landmark Embedding Drawbacks

Large error for nearby query nodes



$$\tilde{\delta}(a,b) = \delta(l,a) + \delta(l,b) >> \delta(a,b)$$

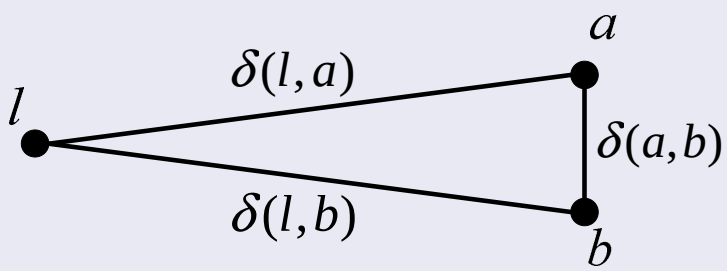
Improving accuracy by increasing $|S|$ draws large overhead

Global Landmark Embedding Complexity:

- Query time: $O(|S|)$
- Index time: $O(|S||V|\log(|V|))$
- Index size: $O(|S||V|)$

Global Landmark Embedding Drawbacks

Large error for nearby query nodes



$$\tilde{\delta}(a,b) = \delta(l,a) + \delta(l,b) >> \delta(a,b)$$

Improving accuracy by increasing $|S|$ draws large overhead

Global Landmark Embedding Complexity:

- Query time: $O(|S|)$
- Index time: $O(|S||V|\log(|V|))$
- Index size: $O(|S||V|)$

Roadmap

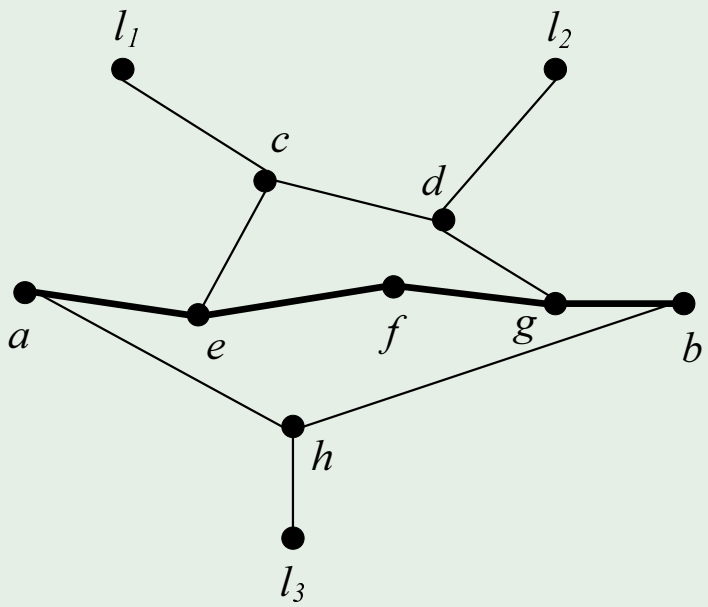
- Introduction
- Query-Dependent Local Landmark Scheme
- Optimization Techniques
- Experiments
- Conclusion

Query-Dependent Local Landmarks

Example

Landmark set $S = \{l_1, l_2, l_3\}$.

For a query (a, b),
$P(a, b) = (a, e, f, g, b)$.



$P(l_1, a) = (l_1, c, e, a)$,
$P(l_1, b) = (l_1, c, d, g, b)$
Based on landmark l_1,

$$\widetilde{\delta}(a, b) = \delta(l_1, a) + \delta(l_1, b).$$

But based on node c,

$$\widetilde{\delta}^L(a, b) = \delta(c, a) + \delta(c, b)$$

is more accurate because

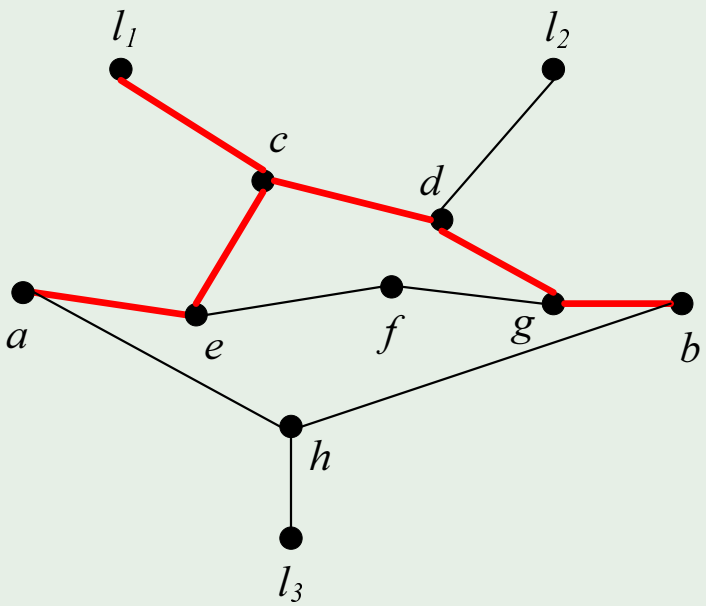
$$\widetilde{\delta}(a, b) = \widetilde{\delta}^L(a, b) + 2\delta(l_1, c).$$

Query-Dependent Local Landmarks

Example

Landmark set $S = \{l_1, l_2, l_3\}$.

For a query (a, b),
$P(a, b) = (a, e, f, g, b)$.



$P(l_1, a) = (l_1, c, e, a)$,
$P(l_1, b) = (l_1, c, d, g, b)$
Based on landmark l_1,

$$\widetilde{\delta}(a, b) = \delta(l_1, a) + \delta(l_1, b).$$

But based on node c,

$$\widetilde{\delta}^L(a, b) = \delta(c, a) + \delta(c, b)$$

is more accurate because

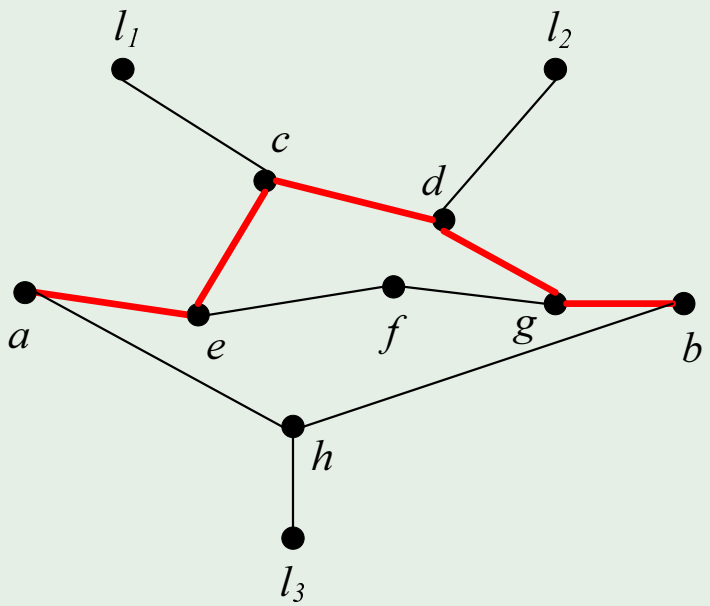
$$\widetilde{\delta}(a, b) = \widetilde{\delta}^L(a, b) + 2\delta(l_1, c).$$

Query-Dependent Local Landmarks

Example

Landmark set $S = \{l_1, l_2, l_3\}$.

For a query (a, b),
$P(a, b) = (a, e, f, g, b)$.



$P(l_1, a) = (l_1, c, e, a)$,
$P(l_1, b) = (l_1, c, d, g, b)$
Based on landmark l_1,

$$\widetilde{\delta}(a, b) = \delta(l_1, a) + \delta(l_1, b).$$

But based on node c,

$$\widetilde{\delta}^L(a, b) = \delta(c, a) + \delta(c, b)$$

is more accurate because

$$\widetilde{\delta}(a, b) = \widetilde{\delta}^L(a, b) + 2\delta(l_1, c).$$

Query-Dependent Local Landmarks

Definition (Query-Dependent Local Landmark Function)

Given a global landmark set S a query (a, b),

$$L_{ab}(S) : V^k \mapsto V.$$

A more accurate distance estimation, $r_{a,b} \leftarrow L_{ab}(S)$

$$\widetilde{\delta}^L(a, b) = \delta(r_{a,b}, a) + \delta(r_{a,b}, b).$$

Requirements for a local landmark function:

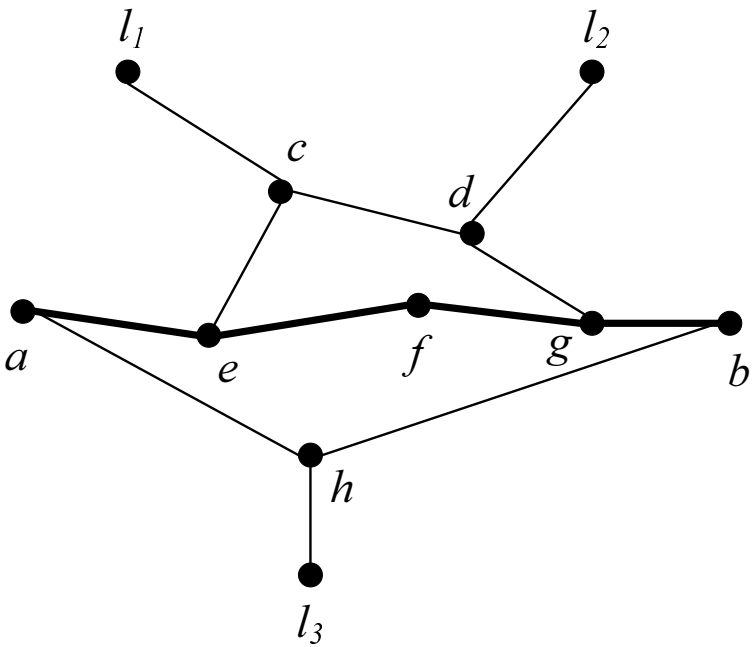
- Efficiently retrieve $L_{ab}(S)$;
- Efficiently compute $\delta(L_{ab}(S), a)$, $\delta(L_{ab}(S), b)$;
- Compact index.

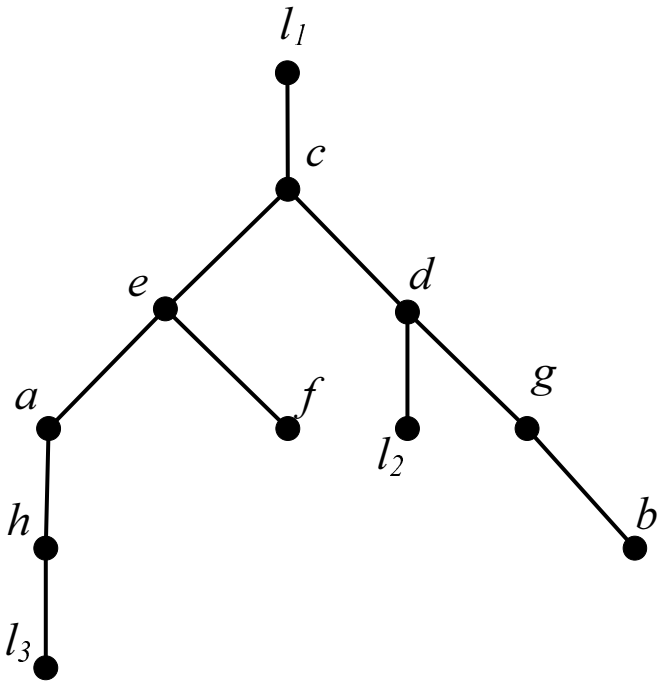
SPT Based Local Landmark Function

Definition (SPT Based Local Landmark Function)

$$L_{ab}(S) = \arg \min_{r \in \{LCA_{T_l}(a,b) | l \in S\}} \{\delta(r,a) + \delta(r,b)\}$$

T_l: The shortest path tree rooted at $l \in S$.
$LCA_{T_l}(a,b)$: The Least common ancestor of a and b in T_l.



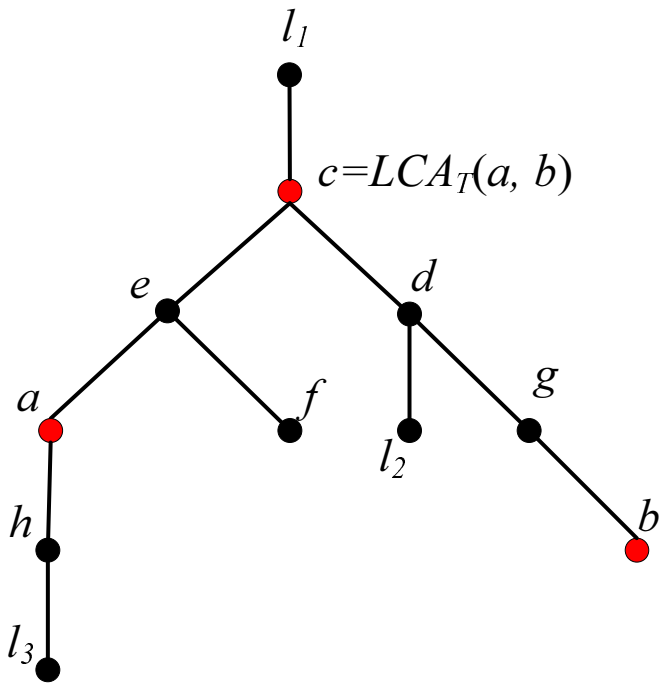


SPT Based Local Landmark Function

Definition (SPT Based Local Landmark Function)

$$L_{ab}(S) = \arg \min_{r \in \{LCA_{T_l}(a,b) | l \in S\}} \{\delta(r,a) + \delta(r,b)\}$$

T_l: The shortest path tree rooted at $l \in S$.
$LCA_{T_l}(a,b)$: The Least common ancestor of a and b in T_l.



$$\begin{aligned}\widetilde{\delta}^L(a,b) &= \delta(c,a) + \delta(c,b) \\ &= \delta(l_1,a) + \delta(l_1,b) - 2\delta(l_1,c)\end{aligned}$$

SPT Based Local Landmark Function

Accuracy

$\forall a, b \in V$,

$$\delta(a, b) \leq \widetilde{\delta}^{L}(a, b) \leq \widetilde{\delta}(a, b)$$

Complexity

- LCA Query Time: $O(1)$.
- Online Query Time: $O(|S|)$.
- Offline Embedding Space: $O(|S||V|)$.
- Offline Embedding Time$O(|S||V| \log(|V|))$.
- SAME complexities as the global landmark embedding.

SPT Based Local Landmark Function

Accuracy

$\forall a, b \in V$,

$$\delta(a, b) \leq \widetilde{\delta}^{L}(a, b) \leq \widetilde{\delta}(a, b)$$

Complexity

- LCA Query Time: $O(1)$.
- Online Query Time: $O(|S|)$.
- Offline Embedding Space: $O(|S||V|)$.
- Offline Embedding Time$O(|S||V| \log(|V|))$.
- SAME complexities as the global landmark embedding.

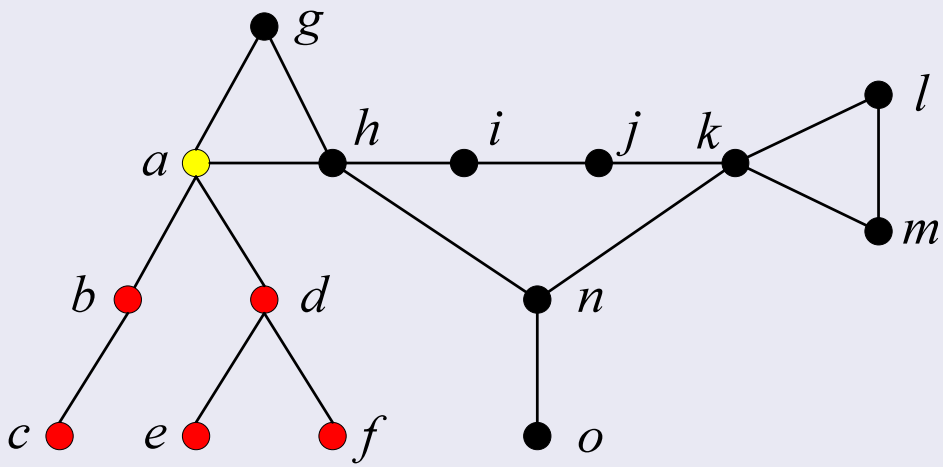
Roadmap

- Introduction
- Query-Dependent Local Landmark Scheme
- Optimization Techniques
  - Index Reduction with Graph Compression
  - Improving the Accuracy by Local Search
- Experiments
- Conclusion

Index Reduction with Graph Compression

Lossless graph compression by removing two special types of nodes:

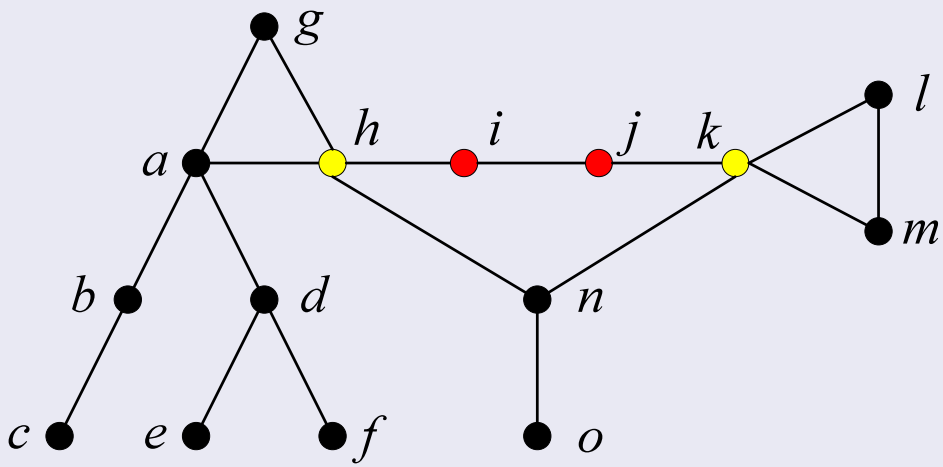
- Tree node
  - Map to the entry node.
  - Remove the tree node.
- Chain node
  - Map to two end nodes.
  - Remove the chain node.



Index Reduction with Graph Compression

Lossless graph compression by removing two special types of nodes:

- Tree node
  - Map to the entry node.
  - Remove the tree node.
- Chain node
  - Map to two end nodes.
  - Remove the chain node.



Index Reduction with Graph Compression

For $\forall a \in V$, case a:

- *Tree node*: $map(a)$ contains a *entry node*.
- *Chain node*: $map(a)$ contains two *end nodes*.
- Otherwise: $map(a)$ contains only itself.

Query time: $O(|S|)$

$$\widetilde{\delta}^L(a,b) = \min_{r_a \in map(a), r_b \in map(b)} \{\delta(a,r_a) + \widetilde{\delta}^L(r_a,r_b) + \delta(b,r_b)\}.$$

Index size

Reduce size from $O(|S||V|)$ to $O(|V| + (|S|-1)|V'|)$.
$|V'|$: the number of nodes in the compressed graph.

Index Reduction with Graph Compression

For $\forall a \in V$, case a:

- *Tree node*: $map(a)$ contains a *entry node*.
- *Chain node*: $map(a)$ contains two *end nodes*.
- Otherwise: $map(a)$ contains only itself.

Query time: $O(|S|)$

$$\widetilde{\delta}^L(a,b) = \min_{r_a \in map(a), r_b \in map(b)} \{\delta(a,r_a) + \widetilde{\delta}^L(r_a,r_b) + \delta(b,r_b)\}.$$

Index size

Reduce size from $O(|S||V|)$ to $O(|V| + (|S|-1)|V'|)$.
$|V'|$: the number of nodes in the compressed graph.

Index Reduction with Graph Compression

For $\forall a \in V$, case a:

- *Tree node*: $map(a)$ contains a *entry node*.
- *Chain node*: $map(a)$ contains two *end nodes*.
- Otherwise: $map(a)$ contains only itself.

Query time: $O(|S|)$

$$\widetilde{\delta}^L(a,b) = \min_{r_a \in map(a), r_b \in map(b)} \{\delta(a,r_a) + \widetilde{\delta}^L(r_a,r_b) + \delta(b,r_b)\}.$$

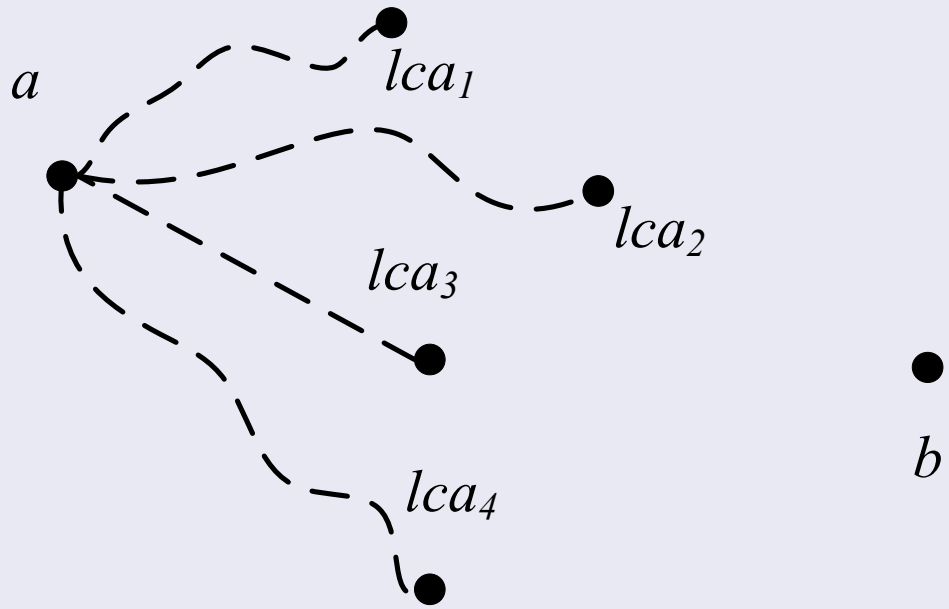
Index size

Reduce size from $O(|S||V|)$ to $O(|V| + (|S|-1)|V'|)$.
$|V'|$: the number of nodes in the compressed graph.

Improving the Accuracy by Local Search

Cost-effective Local Search Procedure

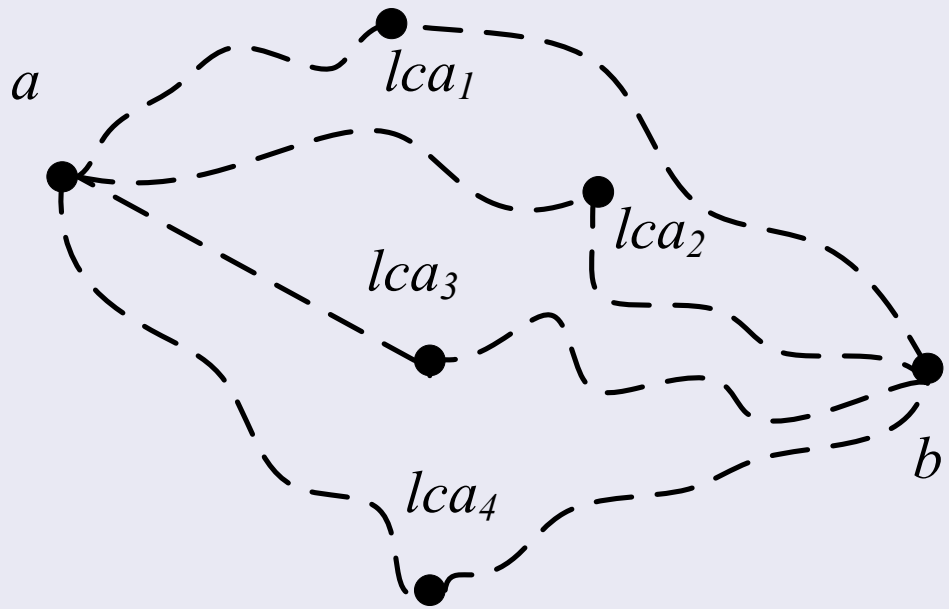
- Connect two query nodes to local landmarks through the shortest paths.
- Expand each node to include its c-hop neighbors.
- The expanded nodes may form shortcuts which provide tighter distance estimation.



Improving the Accuracy by Local Search

Cost-effective Local Search Procedure

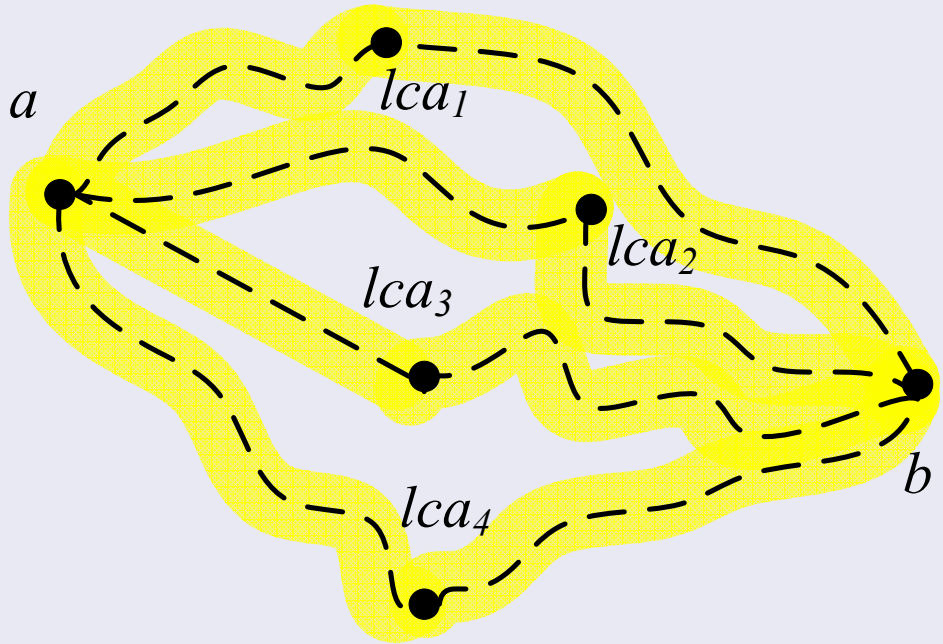
- Connect two query nodes to local landmarks through the shortest paths.
- Expand each node to include its c-hop neighbors.
- The expanded nodes may form shortcuts which provide tighter distance estimation.



Improving the Accuracy by Local Search

Cost-effective Local Search Procedure

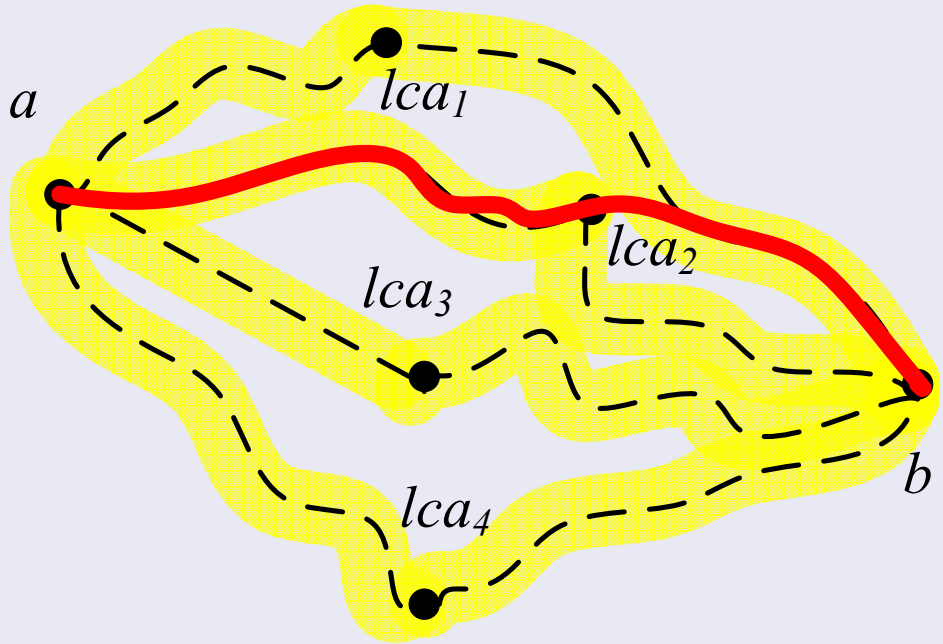
- Connect two query nodes to local landmarks through the shortest paths.
- Expand each node to include its c-hop neighbors.
- The expanded nodes may form shortcuts which provide tighter distance estimation.



Improving the Accuracy by Local Search

Cost-effective Local Search Procedure

- Connect two query nodes to local landmarks through the shortest paths.
- Expand each node to include its c-hop neighbors.
- The expanded nodes may form shortcuts which provide tighter distance estimation.



Roadmap

- Introduction
- Query-Dependent Local Landmark Scheme
- Optimization Techniques
- Experiments
- Conclusion

Comparison Methods and Metrics

The embedding methods for comparison:

- Global Landmark Scheme (GLS)
- Local Landmark Scheme (LLS)
- Local Search (LS)
- 2RNE (Kriegel et al. [4])
- TreeSketch (Gubichev et al. [2])

Evaluation Metrics: relative error

$$err = \frac{|\widetilde{\delta}(s,t) - \delta(s,t)|}{\delta(s,t)}$$

Comparison Methods and Metrics

The embedding methods for comparison:

- Global Landmark Scheme (GLS)
- Local Landmark Scheme (LLS)
- Local Search (LS)
- 2RNE (Kriegel et al. [4])
- TreeSketch (Gubichev et al. [2])

Evaluation Metrics: relative error

$$err = \frac{|\widetilde{\delta}(s,t) - \delta(s,t)|}{\delta(s,t)}$$

.

Dataset Description

Table: Network Statistics

Dataset	$\|V\|$	$\|E\|$	$\|V'\|$	$\|E'\|$
Slashdot	77,360	905,468	36,012	752,478
Google	875,713	5,105,039	449,341	4,621,002
Youtube	1,157,827	4,945,382	313,479	4,082,046
Flickr	1,846,198	22,613,981	493,525	18,470,294
NYRN	264,346	733,846	164,843	532,264
USARN	23,947,347	58,333,344	7,911,536	24,882,476

$|V'|$ and $|E'|$ denote the number of nodes and edges in the compressed graph.

Dataset Description

Table: Network Statistics

Dataset	$\|V\|$	$\|E\|$	$\|V'\|$	$\|E'\|$
Slashdot	77,360	905,468	36,012	752,478
Google	875,713	5,105,039	449,341	4,621,002
Youtube	1,157,827	4,945,382	313,479	4,082,046
Flickr	1,846,198	22,613,981	493,525	18,470,294
NYRN	264,346	733,846	164,843	532,264
USARN	23,947,347	58,333,344	7,911,536	24,882,476

$|V'|$ and $|E'|$ denote the number of nodes and edges in the compressed graph.

Average Relative Error

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.6309	0.5072	0.6346	0.5131	0.1825	0.1121
LLS	0.1423	0.0321	0.0637	0.0814	0.0246	0.0786
LS	0.0000	0.0046	0.0009	0.0001	0.0071	0.0090

Average Relative Error

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.6309	0.5072	0.6346	0.5131	0.1825	0.1121
LLS	0.1423	0.0321	0.0637	0.0814	0.0246	0.0786
LS	0.0000	0.0046	0.0009	0.0001	0.0071	0.0090

Average Relative Error

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.6309	0.5072	0.6346	0.5131	0.1825	0.1121
LLS	0.1423	0.0321	0.0637	0.0814	0.0246	0.0786
LS	0.0000	0.0046	0.0009	0.0001	0.0071	0.0090

Query Time and Index Size

Online Query Time in Milliseconds

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.002	0.005	0.008	0.009	0.006	0.020
LLS	0.006	0.021	0.015	0.014	0.036	0.067
LS	0.158	2.729	2.818	4.735	0.681	58.289

Index Size in MB

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	6.2	57.9	90.7	124.7	21.2	1915.8
LLS	10.4	122.7	103.2	156.1	85.3	4424.6
LS	16.4	159.7	135.9	303.9	89.6	4623.6

Query Time and Index Size

Online Query Time in Milliseconds

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.002	0.005	0.008	0.009	0.006	0.020
LLS	0.006	0.021	0.015	0.014	0.036	0.067
LS	0.158	2.729	2.818	4.735	0.681	58.289

Index Size in MB

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	6.2	57.9	90.7	124.7	21.2	1915.8
LLS	10.4	122.7	103.2	156.1	85.3	4424.6
LS	16.4	159.7	135.9	303.9	89.6	4623.6

Query Time and Index Size

Online Query Time in Milliseconds

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.002	0.005	0.008	0.009	0.006	0.020
LLS	0.006	0.021	0.015	0.014	0.036	0.067
LS	0.158	2.729	2.818	4.735	0.681	58.289

Index Size in MB

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	6.2	57.9	90.7	124.7	21.2	1915.8
LLS	10.4	122.7	103.2	156.1	85.3	4424.6
LS	16.4	159.7	135.9	303.9	89.6	4623.6

Query Time and Index Size

Online Query Time in Milliseconds

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.002	0.005	0.008	0.009	0.006	0.020
LLS	0.006	0.021	0.015	0.014	0.036	0.067
LS	0.158	2.729	2.818	4.735	0.681	58.289

Index Size in MB

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	6.2	57.9	90.7	124.7	21.2	1915.8
LLS	10.4	122.7	103.2	156.1	85.3	4424.6
LS	16.4	159.7	135.9	303.9	89.6	4623.6

Query Time and Index Size

Online Query Time in Milliseconds

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.002	0.005	0.008	0.009	0.006	0.020
LLS	0.006	0.021	0.015	0.014	0.036	0.067
LS	0.158	2.729	2.818	4.735	0.681	58.289

Index Size in MB

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	6.2	57.9	90.7	124.7	21.2	1915.8
LLS	10.4	122.7	103.2	156.1	85.3	4424.6
LS	16.4	159.7	135.9	303.9	89.6	4623.6

Query Time and Index Size

Online Query Time in Milliseconds

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	0.002	0.005	0.008	0.009	0.006	0.020
LLS	0.006	0.021	0.015	0.014	0.036	0.067
LS	0.158	2.729	2.818	4.735	0.681	58.289

Index Size in MB

	SlashD	Google	Youtube	Flickr	NYRN	USARN
	$k = 20$					
GLS	6.2	57.9	90.7	124.7	21.2	1915.8
LLS	10.4	122.7	103.2	156.1	85.3	4424.6
LS	16.4	159.7	135.9	303.9	89.6	4623.6

Comparison with Other Methods

Dataset	Algorithm	AvgErr	Query Time(ms)	Index Size(MB)
SlashD	2RNE	0.8345	0.001	6.2
	TreeSketch	0.0011	0.176	37.4
	LS	0.0000	0.158	16.4
Google	2RNE	0.5750	0.001	57.9
	TreeSketch	0.0048	3.549	383.7
	LS	0.0046	2.729	159.7
Youtube	2RNE	0.7138	0.001	90.7
	TreeSketch	0.0005	5.317	587.7
	LS	0.0009	2.818	135.9
Flickr	2RNE	0.6233	0.001	124.7
	TreeSketch	0.0001	7.333	959.6
	LS	0.0001	4.735	303.9
NYRN	2RNE	0.4748	0.001	21.2
	TreeSketch	0.0156	1.074	120.5
	LS	0.0071	0.681	89.6
USARN	2RNE	0.4240	0.002	1915.8
	TreeSketch	0.0379	104.769	14555.5
	LS	0.0090	58.289	4623.6

Comparison with Other Methods

Dataset	Algorithm	AvgErr	Query Time(ms)	Index Size(MB)
SlashD	2RNE	0.8345	0.001	6.2
	TreeSketch	0.0011	0.176	37.4
	LS	0.0000	0.158	16.4
Google	2RNE	0.5750	0.001	57.9
	TreeSketch	0.0048	3.549	383.7
	LS	0.0046	2.729	159.7
Youtube	2RNE	0.7138	0.001	90.7
	TreeSketch	0.0005	5.317	587.7
	LS	0.0009	2.818	135.9
Flickr	2RNE	0.6233	0.001	124.7
	TreeSketch	0.0001	7.333	959.6
	LS	0.0001	4.735	303.9
NYRN	2RNE	0.4748	0.001	21.2
	TreeSketch	0.0156	1.074	120.5
	LS	0.0071	0.681	89.6
USARN	2RNE	0.4240	0.002	1915.8
	TreeSketch	0.0379	104.769	14555.5
	LS	0.0090	58.289	4623.6

Comparison with Other Methods

Dataset	Algorithm	AvgErr	Query Time(ms)	Index Size(MB)
SlashD	2RNE	0.8345	0.001	6.2
	TreeSketch	0.0011	0.176	37.4
	LS	0.0000	0.158	16.4
Google	2RNE	0.5750	0.001	57.9
	TreeSketch	0.0048	3.549	383.7
	LS	0.0046	2.729	159.7
Youtube	2RNE	0.7138	0.001	90.7
	TreeSketch	0.0005	5.317	587.7
	LS	0.0009	2.818	135.9
Flickr	2RNE	0.6233	0.001	124.7
	TreeSketch	0.0001	7.333	959.6
	LS	0.0001	4.735	303.9
NYRN	2RNE	0.4748	0.001	21.2
	TreeSketch	0.0156	1.074	120.5
	LS	0.0071	0.681	89.6
USARN	2RNE	0.4240	0.002	1915.8
	TreeSketch	0.0379	104.769	14555.5
	LS	0.0090	58.289	4623.6

Comparison with Other Methods

Dataset	Algorithm	AvgErr	Query Time(ms)	Index Size(MB)
SlashD	2RNE	0.8345	0.001	6.2
	TreeSketch	0.0011	0.176	37.4
	LS	0.0000	0.158	16.4
Google	2RNE	0.5750	0.001	57.9
	TreeSketch	0.0048	3.549	383.7
	LS	0.0046	2.729	159.7
Youtube	2RNE	0.7138	0.001	90.7
	TreeSketch	0.0005	5.317	587.7
	LS	0.0009	2.818	135.9
Flickr	2RNE	0.6233	0.001	124.7
	TreeSketch	0.0001	7.333	959.6
	LS	0.0001	4.735	303.9
NYRN	2RNE	0.4748	0.001	21.2
	TreeSketch	0.0156	1.074	120.5
	LS	0.0071	0.681	89.6
USARN	2RNE	0.4240	0.002	1915.8
	TreeSketch	0.0379	104.769	14555.5
	LS	0.0090	58.289	4623.6

Comparison with Other Methods

Dataset	Algorithm	AvgErr	Query Time(ms)	Index Size(MB)
SlashD	2RNE	0.8345	0.001	6.2
	TreeSketch	0.0011	0.176	37.4
	LS	0.0000	0.158	16.4
Google	2RNE	0.5750	0.001	57.9
	TreeSketch	0.0048	3.549	383.7
	LS	0.0046	2.729	159.7
Youtube	2RNE	0.7138	0.001	90.7
	TreeSketch	0.0005	5.317	587.7
	LS	0.0009	2.818	135.9
Flickr	2RNE	0.6233	0.001	124.7
	TreeSketch	0.0001	7.333	959.6
	LS	0.0001	4.735	303.9
NYRN	2RNE	0.4748	0.001	21.2
	TreeSketch	0.0156	1.074	120.5
	LS	0.0071	0.681	89.6
USARN	2RNE	0.4240	0.002	1915.8
	TreeSketch	0.0379	104.769	14555.5
	LS	0.0090	58.289	4623.6

Roadmap

- Introduction
- Problem Statement
- Query-Dependent Local Landmark Scheme
- Optimization Techniques
- Experiments
- Conclusion

Conclusion

Query-Dependent Local Landmark Scheme

- Cost-effective in improving accuracy.
- Linear index size, and thus scalable.
- Allows optimization techniques for further reducing index size and relative error.
- Outperforms state-of-the-art landmark embedding approaches in experiments.

Q&A

Thanks!


P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang.
IDMaps: A global internet host distance estimation service.
IEEE/ACM Trans. Networking, 9(5):525–540, 2001.


A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum.
Fast and accurate estimation of shortest paths in large graphs.
In *Proc. Int. Conf. Information and Knowledge Management (CIKM'10)*, 2010.


J. Kleinberg, A. Slivkins, and T. Wexler.
Triangulation and embedding using small sets of beacons.
In *Proc. Symp. Foundations of Computer Science (FOCS'04)*, pages 444–453, 2004.


H.-P. Kriegel, P. Kröger, M. Renz, and T. Schmidt.
Hierarchical graph embedding for efficient query processing in very large traffic networks.
In *Proc. Int. Conf. Scientific and Statistical Database Management (SSDBM'08)*, pages 150–167, 2008.


T. S. E. Ng and H. Zhang.

Predicting internet network distance with coordinates-based approaches.
In *Int. Conf. on Computer Communications (INFOCOM'01)*, pages 170–179, 2001.


D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao.
Query processing in spatial network database.
In *Proc. Int. Conf. Very Large Data Bases (VLDB'03)*, pages 802–813, 2003.


M. Potamias, F. Bonchi, C. Castillo, and A. Gionis.
Fast shortest path distance estimation in large networks.
In *Proc. Int. Conf. Information and Knowledge Management (CIKM'09)*, pages 867–876, 2009.


M. J. Rattigan, M. Maier, and D. Jensen.
Using structure indices for efficient approximation of network properties.
In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'06)*, pages 357–366, 2006.


J. Sankaranarayanan, H. Samet, and H. Alborzi.
Path oracles for spatial networks.
PVLDB, pages 1210–1221, 2009.


A. D. Sarma, S. Gollapudi, M. Najork, and R. Panigrahy.
A sketch-based distance oracle for web-scale graphs.
In *Proc. Int. Conf. Web Search and Data Mining (WSDM'10)*, pages 401–410, 2010.


C. Shahabi, M. Kolahdouzan, and M. Sharifzadeh.
A road network embedding technique for k-nearest neighbor search in moving object databases.
In *Proc. 10th ACM Int. Symp. Advances in Geographic Information Systems (GIS'02)*, pages 94–100, 2002.


M. Thorup and U. Zwick.
Approximate distance oracles.
Journal of the ACM, 52(1):1–24, 2005.


F. Wei.
TEDI: Efficient shortest path query answering on graphs.

In *Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'10)*, pages 99–110, 2010.