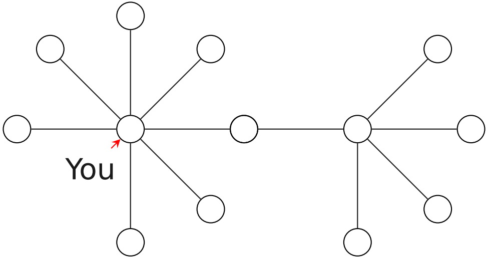
Three-Hop Distance Estimation in Social Graphs

Pascal Welke, Alexander Markowetz, Torsten Suel, Maria Christoforaki

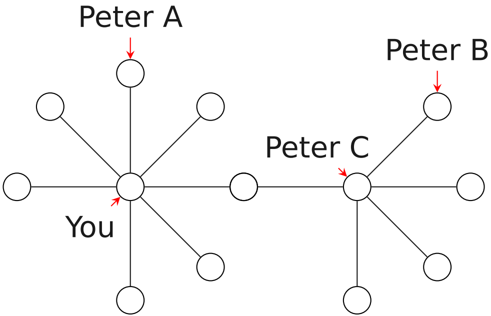
IEEE BigData 2016

Social Search

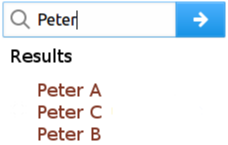
Social Search

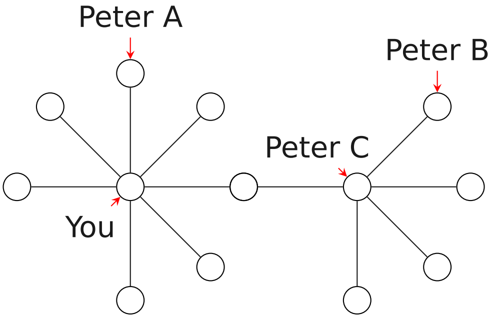


Social Search



Social Search





universitätbonn

Distance Estimation for Social Search

- Social graphs may have millions of vertices and billions of edges
  - Running a shortest path algorithm for each query at runtime is infeasible (runtime constraints)
  - Computing and storing all distances in advance is infeasible (space constraints)

Distance Estimation for Social Search

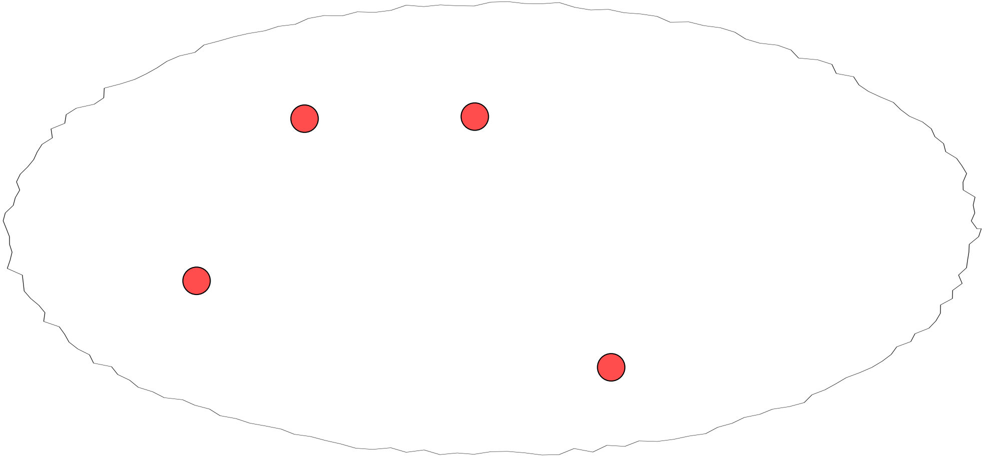
- Social graphs may have millions of vertices and billions of edges
  - Running a shortest path algorithm for each query at runtime is infeasible (runtime constraints)
  - Computing and storing all distances in advance is infeasible (space constraints)
- Distance signals are one factor among many others in social search
  - Exact distances are not always required

Distance Estimation for Social Search

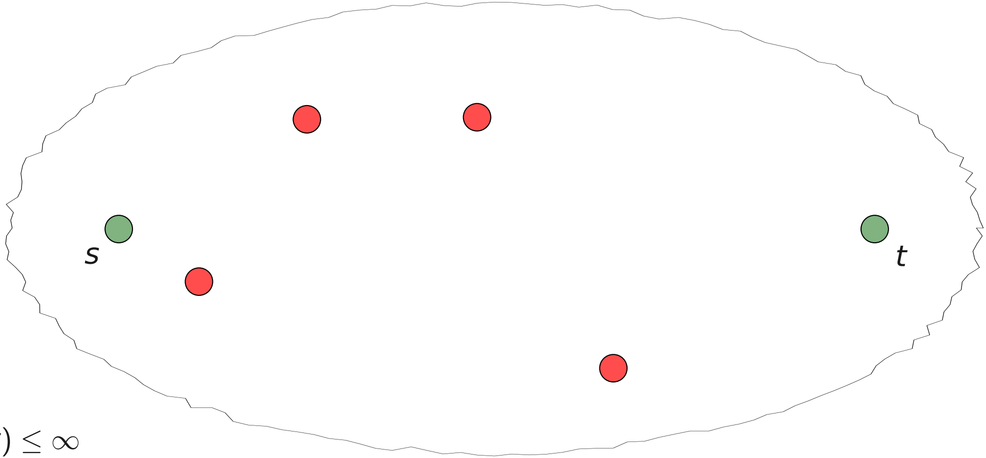
- Social graphs may have millions of vertices and billions of edges
  - Running a shortest path algorithm for each query at runtime is infeasible (runtime constraints)
  - Computing and storing all distances in advance is infeasible (space constraints)
- Distance signals are one factor among many others in social search
  - Exact distances are not always required

Problem: For a graph $G = (V, E)$, compute a data structure of size $O(|V| + |E|)$ that allows fast approximate answers to distance queries for arbitrary pairs of vertices $s, t \in V$.

Classical Approach: Two-Hop Landmarks

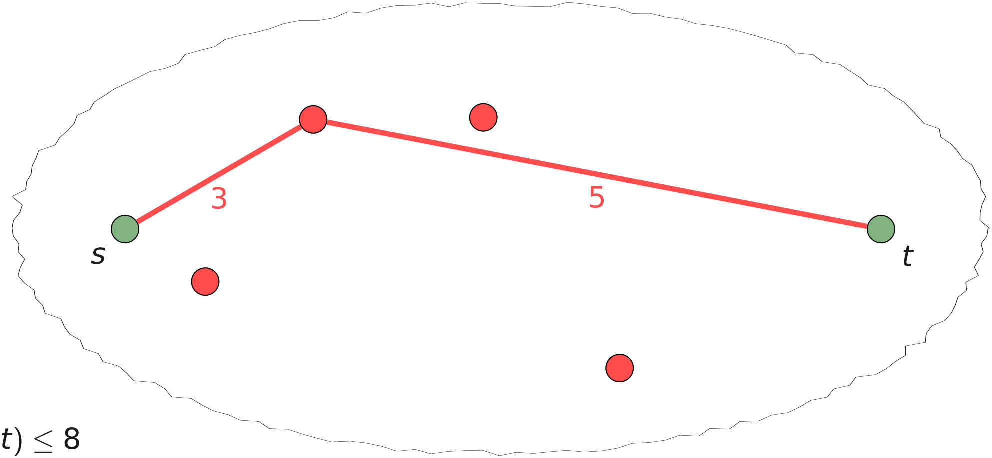


Classical Approach: Two-Hop Landmarks



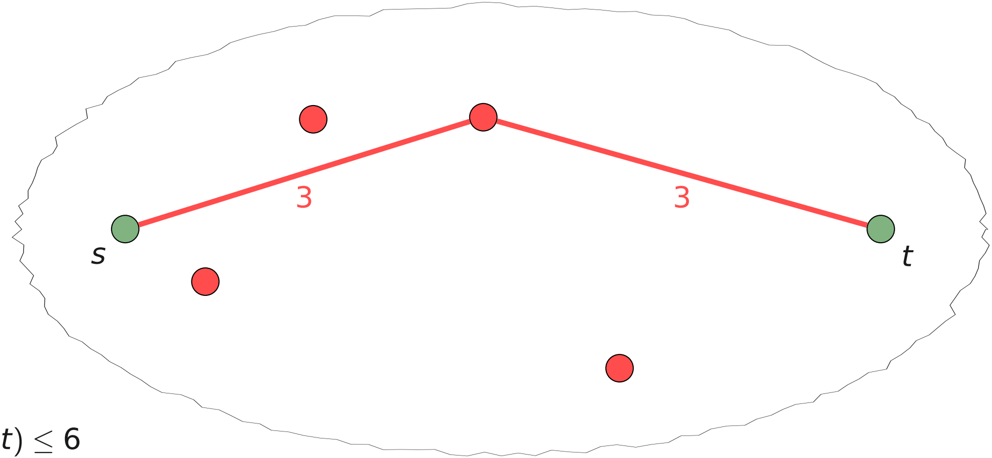
$d(s,t) \leq \infty$

Classical Approach: Two-Hop Landmarks



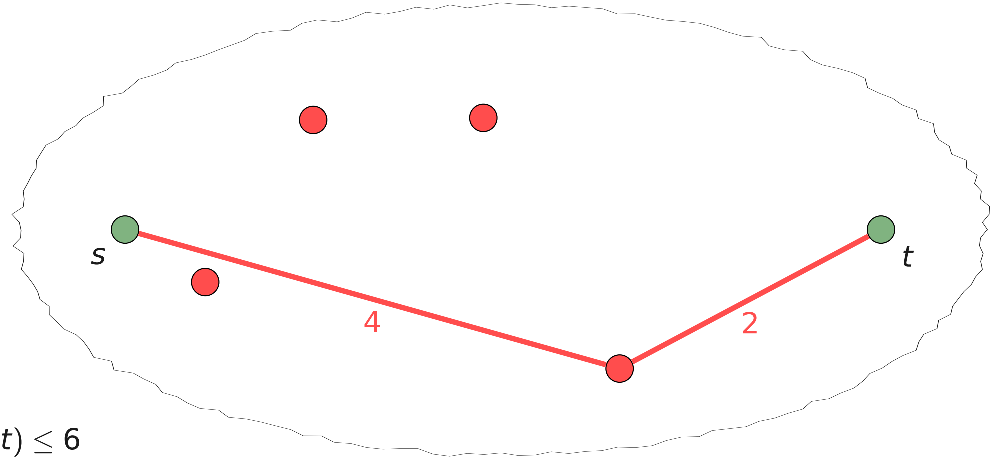
$d(s, t) \leq 8$

Classical Approach: Two-Hop Landmarks



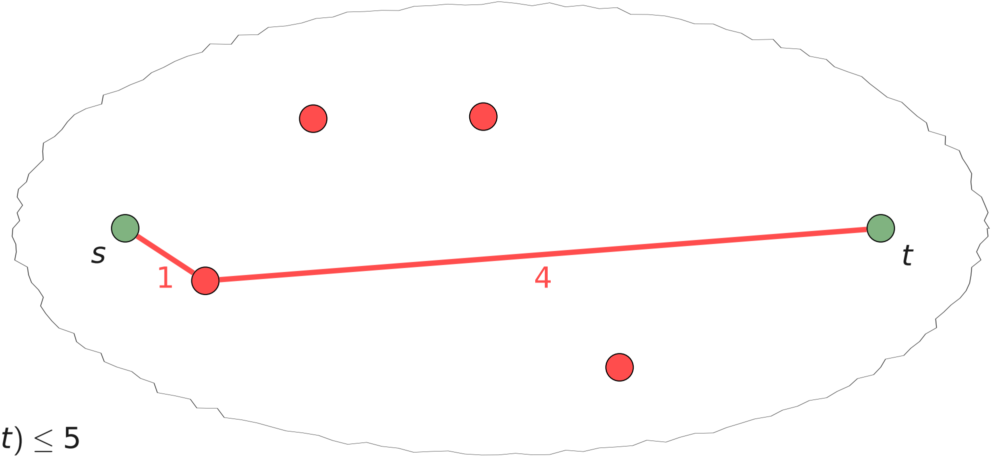
$d(s, t) \leq 6$

Classical Approach: Two-Hop Landmarks



$d(s, t) \leq 6$

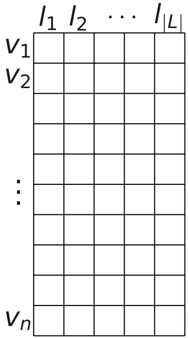
Classical Approach: Two-Hop Landmarks



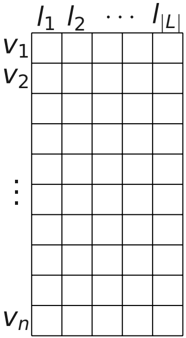
$d(s, t) \leq 5$

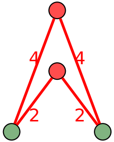
universitätbonn

- We have to store distances from all landmarks to all vertices



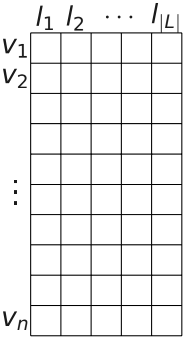
- We have to store distances from all landmarks to all vertices
- We need landmarks close to shortest paths for any given query

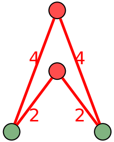




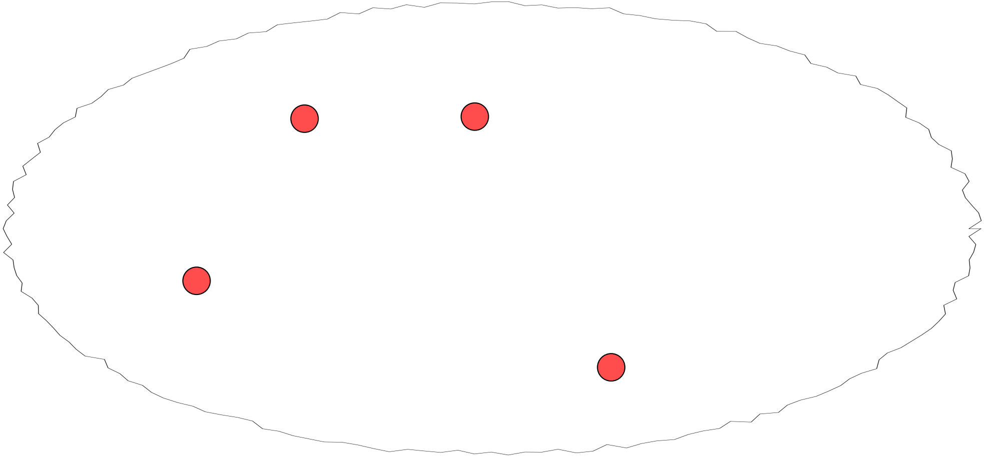
Problems with Two-Hop Landmarks

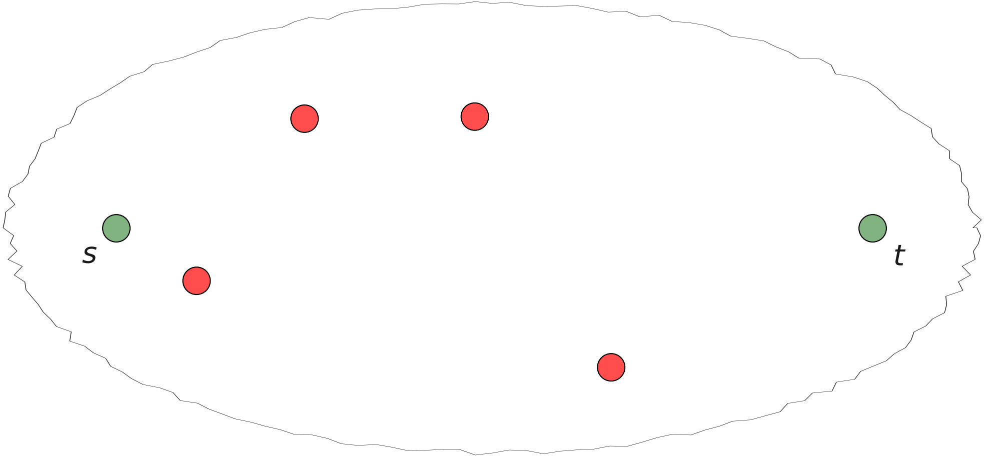
- We have to store distances from all landmarks to all vertices
- We need landmarks close to shortest paths for any given query
- The stored data needs to grow superlinearly for good results



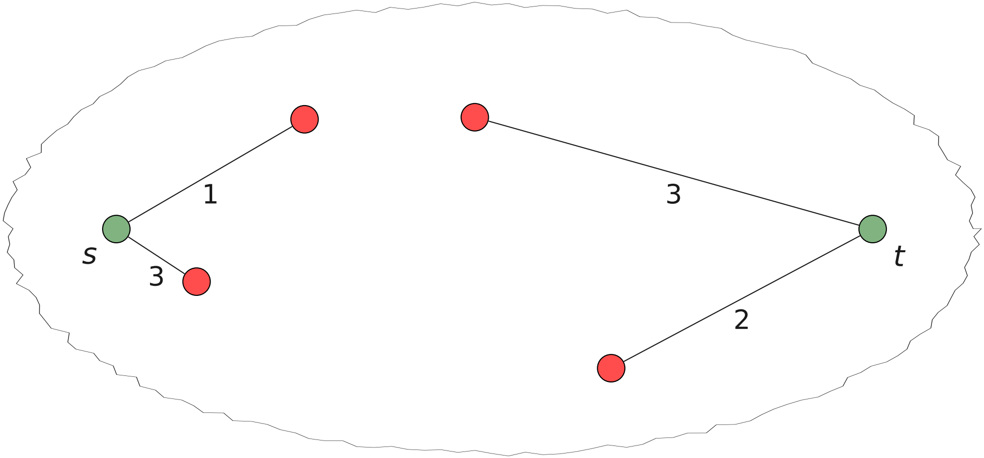


Three-Hop Landmarks



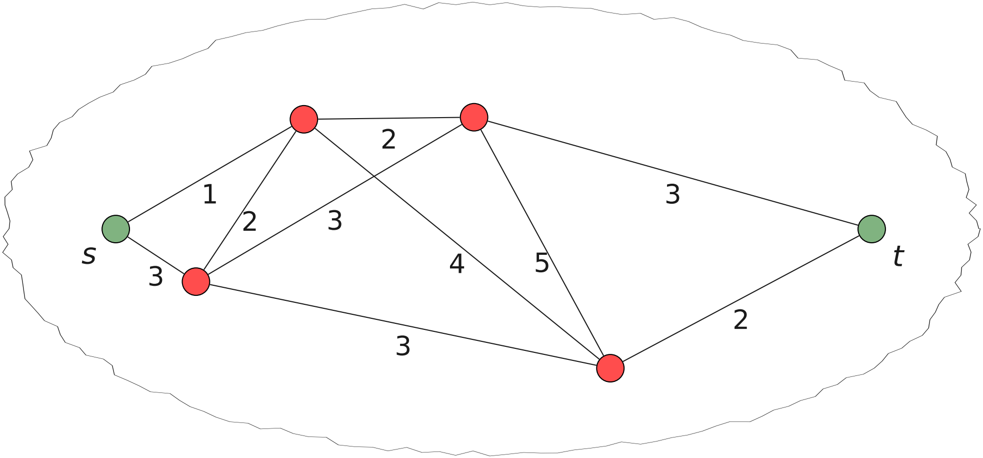

s
t

Three-Hop Landmarks

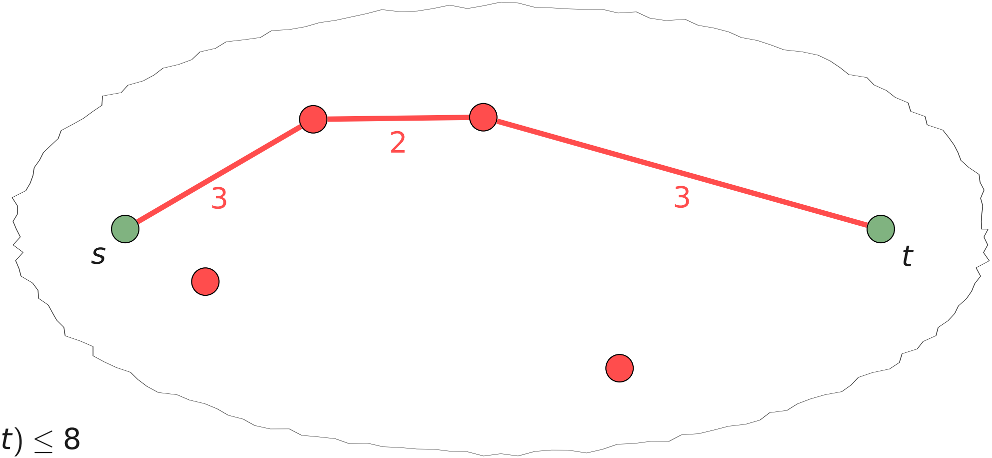


universitätbonn

Three-Hop Landmarks



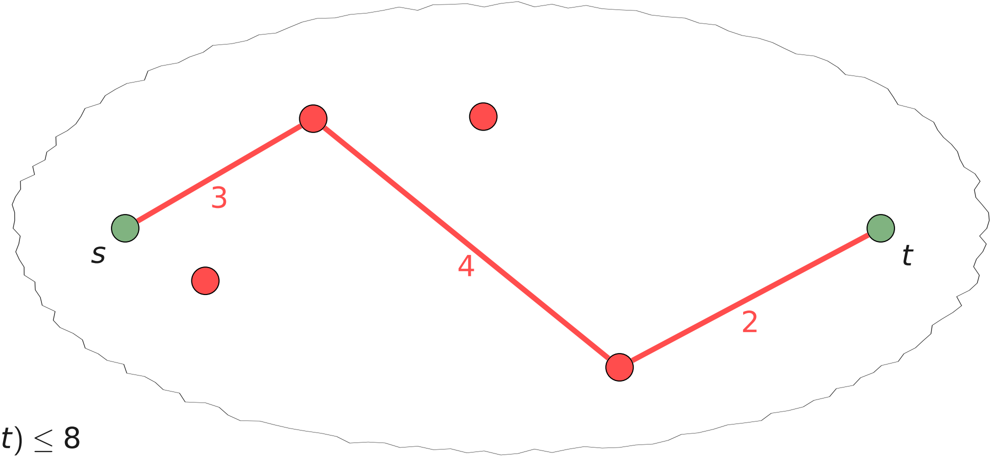
Three-Hop Landmarks



$d(s,t) \leq 8$

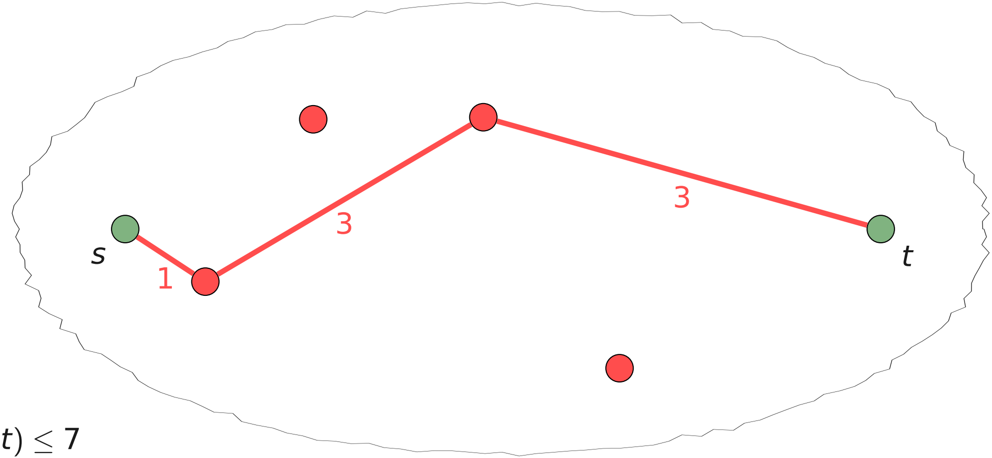
universitätbonn

Three-Hop Landmarks



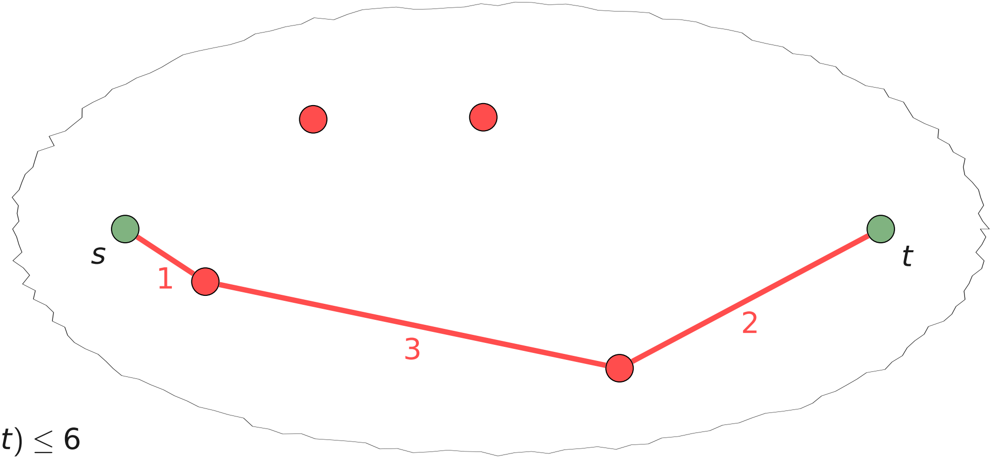
$d(s, t) \leq 8$

Three-Hop Landmarks



$d(s, t) \leq 7$

Three-Hop Landmarks



$d(s,t) \leq 6$

Benefits & Drawbacks of Three-Hop Landmarks

Pros:

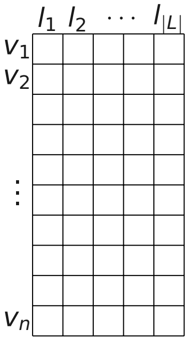
- Close landmarks have a higher likelyhood to be close to shortest paths
- We can have up to $\sqrt{|V|}$ landmarks in a $O(|V|)$ space data structure
- A small number of local landmarks suffices

Cons:

- Going over two landmarks gives less tight bounds
- Algorithms and data structures get more complicated

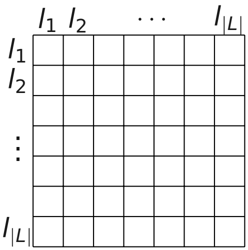
Which Approach is Better?

Two-Hop



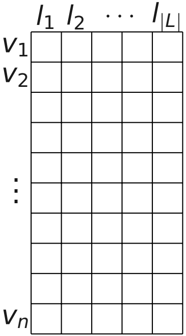
Three-Hop

$$v_1 \rightarrow [(l_{1_1}, 1), \ldots, (l_{1_k}, 2)]$$
$$v_2 \rightarrow [(l_{2_1}, 1), \ldots, (l_{2_k}, 2)]$$
$$\vdots$$
$$v_n \rightarrow [(l_{n_1}, 1), \ldots, (l_{n_k}, 2)]$$



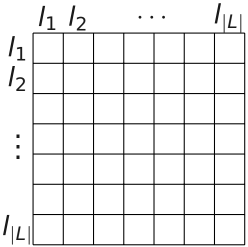
Which Approach is Better?

Two-Hop



$$v_1 \rightarrow [(l_{1_1}, 1), \ldots, (l_{1_k}, 2)]$$
$$v_2 \rightarrow [(l_{2_1}, 1), \ldots, (l_{2_k}, 2)]$$
$$\vdots$$
$$v_n \rightarrow [(l_{n_1}, 1), \ldots, (l_{n_k}, 2)]$$

Three-Hop



Which Approach is Better?

Compressed Size vs. Estimation Error

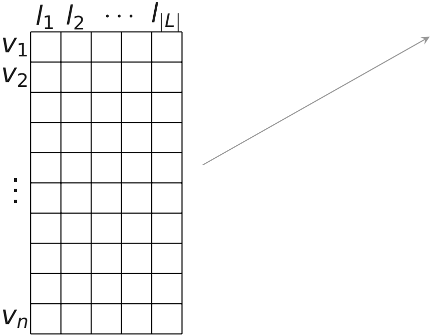
Compressing Two-Hop Landmark Data

	l_1	l_2	$\cdots$		$l_{|L|}$
v_1					
v_2					
$\vdots$					
v_n					

Single Row Compression

	l_1	l_2	$\cdots$		$l_{\lvert L \rvert}$
v_1					
v_2					
$\vdots$					
v_n					

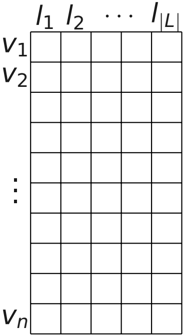
Compressing Two-Hop Landmark Data



Single Row Compression

v: | 3 | 2 | 1 | 4 | 3 |

Compressing Two-Hop Landmark Data



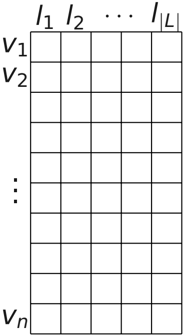
Single Row Compression

v: | 3 | 2 | 1 | 4 | 3 |

– Distances are small

– Use Rice coding

Compressing Two-Hop Landmark Data



Single Row Compression

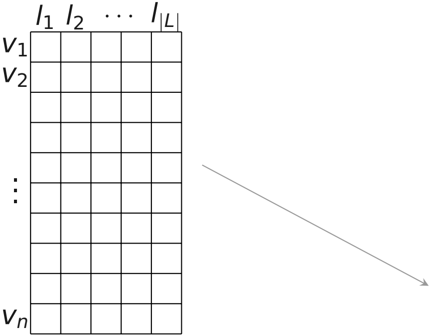
v: | 3 | 2 | 1 | 4 | 3 |

– Distances are small

– Use Rice coding

Neighbor List Compression

Compressing Two-Hop Landmark Data



Single Row Compression

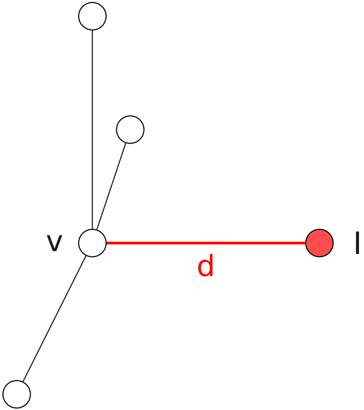
v: | 3 | 2 | 1 | 4 | 3 |

- Distances are small
- Use Rice coding

Neighbor List Compression

v: | 3 | 2 | 1 | 4 | 3 |

Compressing Two-Hop Landmark Data



Single Row Compression

v:	3	2	1	4	3

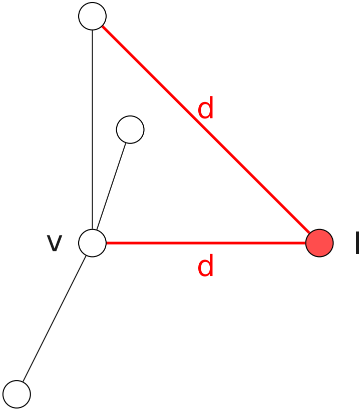
- Distances are small
- Use Rice coding

Neighbor List Compression

- Neighbors are very similar

v:	3	2	1	4	3

Compressing Two-Hop Landmark Data



Single Row Compression

v: | 3 | 2 | 1 | 4 | 3 |

– Distances are small

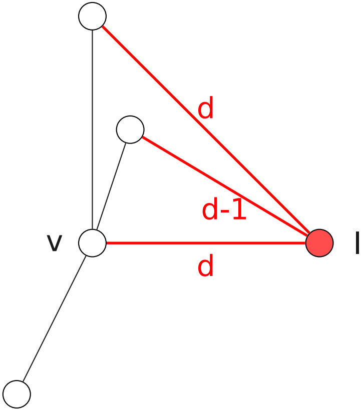
– Use Rice coding

Neighbor List Compression

– Neighbors are very similar

v: | 3 | 2 | 1 | 4 | 3 |

Compressing Two-Hop Landmark Data



Single Row Compression

v: | 3 | 2 | 1 | 4 | 3 |

– Distances are small

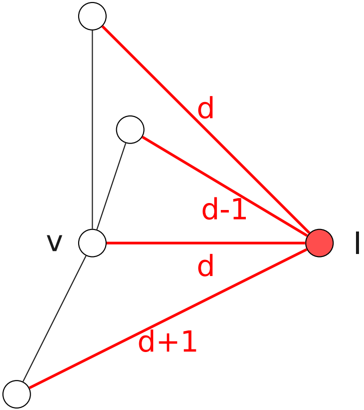
– Use Rice coding

Neighbor List Compression

– Neighbors are very similar

v: | 3 | 2 | 1 | 4 | 3 |

Compressing Two-Hop Landmark Data



Single Row Compression

v:

3	2	1	4	3

– Distances are small

– Use Rice coding

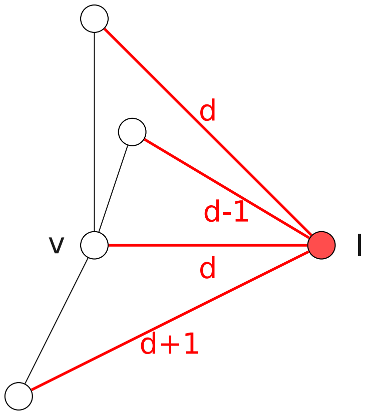
Neighbor List Compression

– Neighbors are very similar

v:

3	2	1	4	3

Compressing Two-Hop Landmark Data



Single Row Compression

v: | 3 | 2 | 1 | 4 | 3 |

- Distances are small
- Use Rice coding

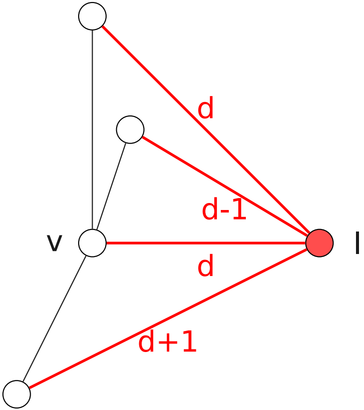
Neighbor List Compression

w: | 3 | 3 | 2 | 3 | 3 |

v: | 3 | 2 | 1 | 4 | 3 |

- Neighbors are very similar

Compressing Two-Hop Landmark Data



Single Row Compression

v: | 3 | 2 | 1 | 4 | 3 |

– Distances are small

– Use Rice coding

Neighbor List Compression

w: | 3 | 3 | 2 | 3 | 3 |

v: | 3 | 2 | 1 | 4 | 3 |

| 0 | -1 | -1 | 1 | 0 |

– Neighbors are very similar

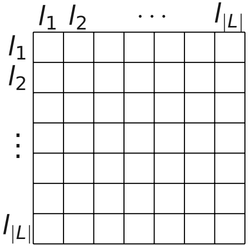
– Use relative encoding

– less than two bits/distance

universitätbonn

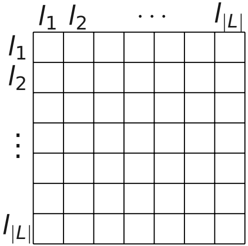
Compressing Three-Hop Landmark Data

	l_1	l_2		$\cdots$			$l_{\vert L \vert}$
l_1							
l_2							
$\vdots$							
$l_{\vert L \vert}$							



Landmark-Landmark Distances

Compressing Three-Hop Landmark Data



Landmark-Landmark Distances

– Random access $\Rightarrow$ fixed lenght encoding

Compressing Three-Hop Landmark Data

$$v_1 \rightarrow [(l_{1_1}, 1), \ldots, (l_{1_k}, 2)]$$
$$v_2 \rightarrow [(l_{2_1}, 1), \ldots, (l_{2_k}, 2)]$$
$$\vdots$$
$$v_n \rightarrow [(l_{n_1}, 1), \ldots, (l_{n_k}, 2)]$$

Landmark-Landmark Distances

– Random access $\Rightarrow$ fixed lenght encoding

Single Row Compression

Compressing Three-Hop Landmark Data

$$v_1 \rightarrow [(l_{1_1}, 1), \ldots, (l_{1_k}, 2)]$$
$$v_2 \rightarrow [(l_{2_1}, 1), \ldots, (l_{2_k}, 2)]$$
$$\vdots$$
$$v_n \rightarrow [(l_{n_1}, 1), \ldots, (l_{n_k}, 2)]$$

Landmark-Landmark Distances

– Random access $\Rightarrow$ fixed lenght encoding

Single Row Compression

$$v_1 \rightarrow [(10, 1), (2, 2), (11, 3)]$$

Compressing Three-Hop Landmark Data

$$v_1 \rightarrow [(l_{1_1}, 1), \ldots, (l_{1_k}, 2)]$$
$$v_2 \rightarrow [(l_{2_1}, 1), \ldots, (l_{2_k}, 2)]$$
$$\vdots$$
$$v_n \rightarrow [(l_{n_1}, 1), \ldots, (l_{n_k}, 2)]$$

Landmark-Landmark Distances

– Random access $\Rightarrow$ fixed lenght encoding

Single Row Compression

$v_1 \rightarrow [(10, 1), (2, 2), (11, 3)]$ – sort by id

$v_1 \rightarrow [(2, 2), (10, 1), (11, 3)]$

Compressing Three-Hop Landmark Data

$$v_1 \rightarrow [(l_{1_1}, 1), \ldots, (l_{1_k}, 2)]$$
$$v_2 \rightarrow [(l_{2_1}, 1), \ldots, (l_{2_k}, 2)]$$
$$\vdots$$
$$v_n \rightarrow [(l_{n_1}, 1), \ldots, (l_{n_k}, 2)]$$

Landmark-Landmark Distances

– Random access $\Rightarrow$ fixed lenght encoding

Single Row Compression

$v_1 \rightarrow [(10, 1), (2, 2), (11, 3)]$ – sort by id

$v_1 \rightarrow [(2, 2), (10, 1), (11, 3)]$ – store gaps

$v_1 \rightarrow [(2, 2), (+8, 1), (+1, 3)]$

Compressing Three-Hop Landmark Data

$$v_1 \rightarrow [(l_{1_1}, 1), \ldots, (l_{1_k}, 2)]$$
$$v_2 \rightarrow [(l_{2_1}, 1), \ldots, (l_{2_k}, 2)]$$
$$\vdots$$
$$v_n \rightarrow [(l_{n_1}, 1), \ldots, (l_{n_k}, 2)]$$

Landmark-Landmark Distances

– Random access $\Rightarrow$ fixed lenght encoding

Single Row Compression

$v_1 \rightarrow [(10, 1), (2, 2), (11, 3)]$ – sort by id

$v_1 \rightarrow [(2, 2), (10, 1), (11, 3)]$ – store gaps

$v_1 \rightarrow [(2, 2), (+8, 1), (+1, 3)]$ – Rice coding

$$v_1 \rightarrow [(l_{1_1}, 1), \ldots, (l_{1_k}, 2)]$$
$$v_2 \rightarrow [(l_{2_1}, 1), \ldots, (l_{2_k}, 2)]$$
$$\vdots$$
$$v_n \rightarrow [(l_{n_1}, 1), \ldots, (l_{n_k}, 2)]$$

Neighbor List Compression

- Similarly, encode vertex information as diff to neighbor
- Take care of changing local landmarks

Experimental Evaluation

- Evaluation on three Social Graphs

Experimental Evaluation

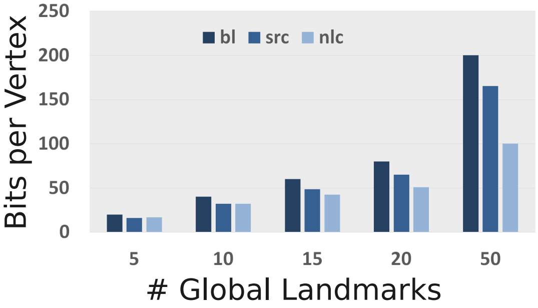
- Evaluation on three Social Graphs
- Here: *loc-gowalla* 197k vertices, 950k edges, diameter 16

Experimental Evaluation

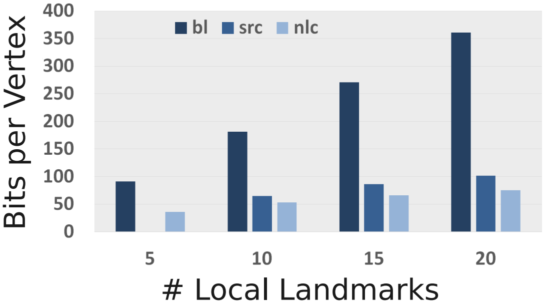
- Evaluation on three Social Graphs
- Here: *loc-gowalla* 197k vertices, 950k edges, diameter 16
- Lots of parameters:
  - How to select landmarks globally and locally?
  - How many local / global landmarks?
  - Which queries are interesting?

Compression Schemes

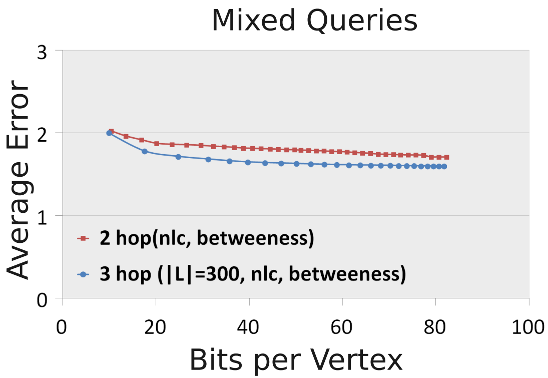
Two-Hop

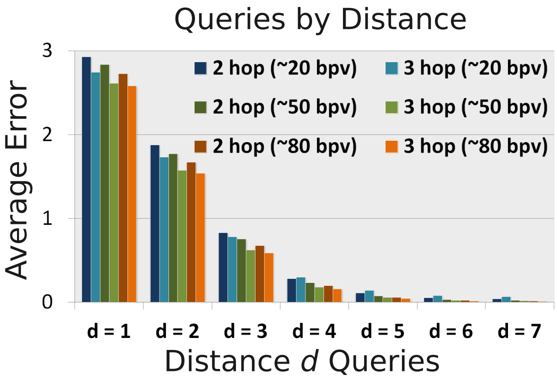


Three-Hop



Space vs. Average Error





Conclusion

- Three-hop landmarks have an asymptotic advantage

Conclusion

- Three-hop landmarks have an asymptotic advantage
- They achieve a modest improvement over two-hop landmarks

Conclusion

- Three-hop landmarks have an asymptotic advantage
- They achieve a modest improvement over two-hop landmarks
- *Sensible Compression makes a huge difference*