2013/09/28 @ NII Shonan Meeting

Shortest-Path Distance Queries on Large Networks by *Pruned Landmark Labeling*

(SIGMOD'13 Research Track Full Paper)

Takuya Akiba *(U Tokyo)*

Yoichi Iwata *(U Tokyo)*

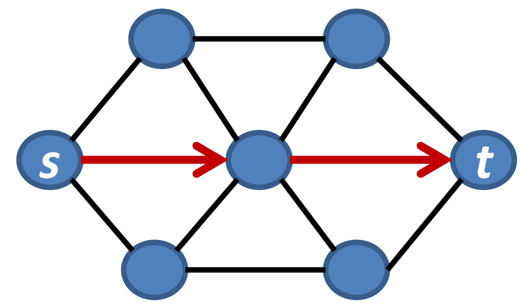
Yuichi Yoshida *(NII & PFI)*

(Implementation available: git.io/pll)

Problem Definition

Given a graph $G = (V, E)$

1. construct an index to
2. answer distance $d_G(s, t)$



Goal: Good trade-off

Scalability
Indexing time
Index size

Query Performance
Query time
Precision

First Notice

- Empirical method for real-world networks
  - No non-trivial theoretical bounds, but works well

Summary of Results:

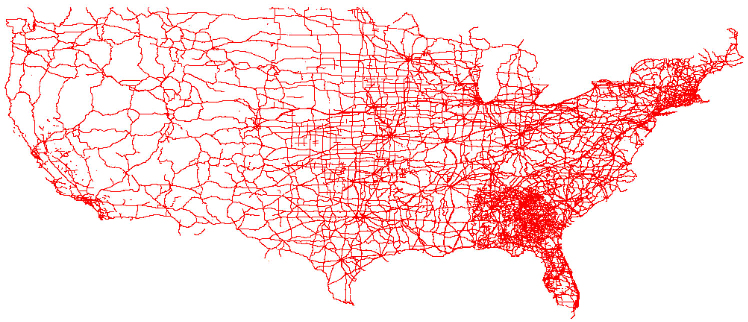
Network with 100M edges → 10GB index, 10 μs query

- Compact data structure?
  - → Much more compact than *distance matrices*
- Data structure?
  - ☹ index = just $2n$ arrays *(room for improvement?)*

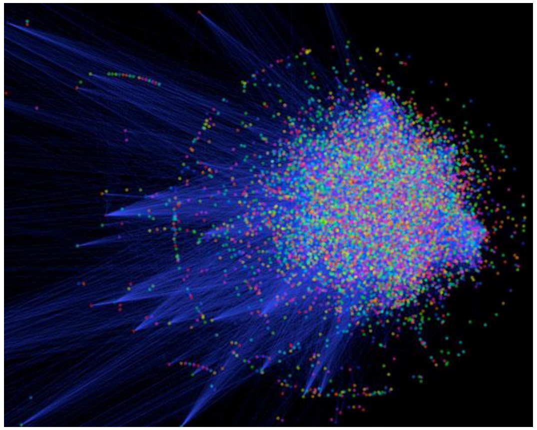
Number of vertices

Real-world Networks

Road Networks



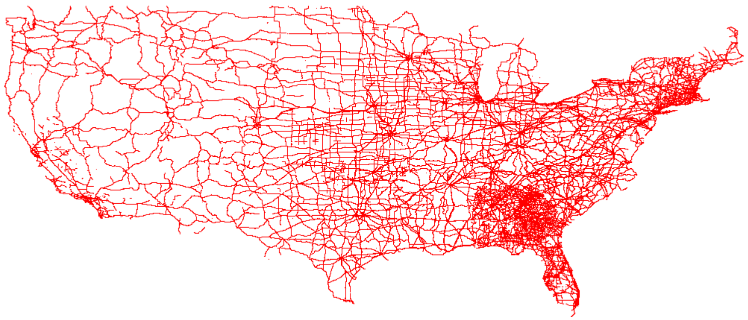
Complex Networks



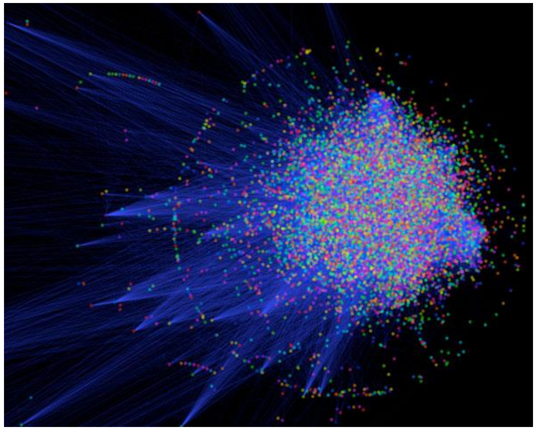
- Social Networks
- Web Graphs
- Computer Networks
- Biological Networks

Real-world Networks

Road Networks



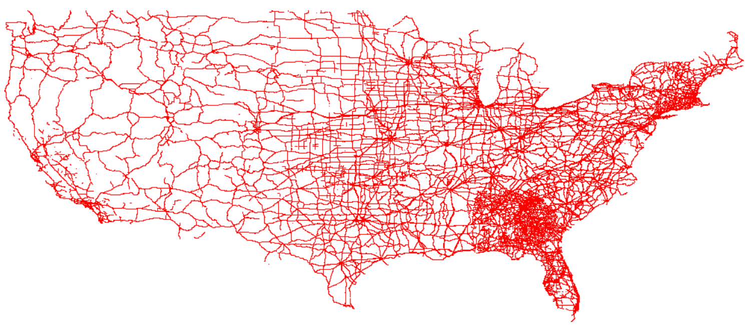
Complex Networks



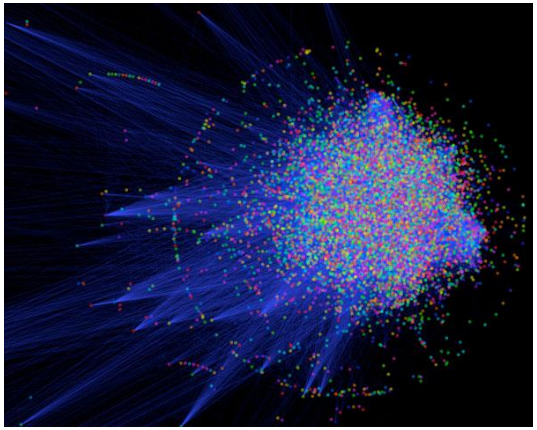
Structural differences
→ Different methods

Real-world Networks

Road Networks



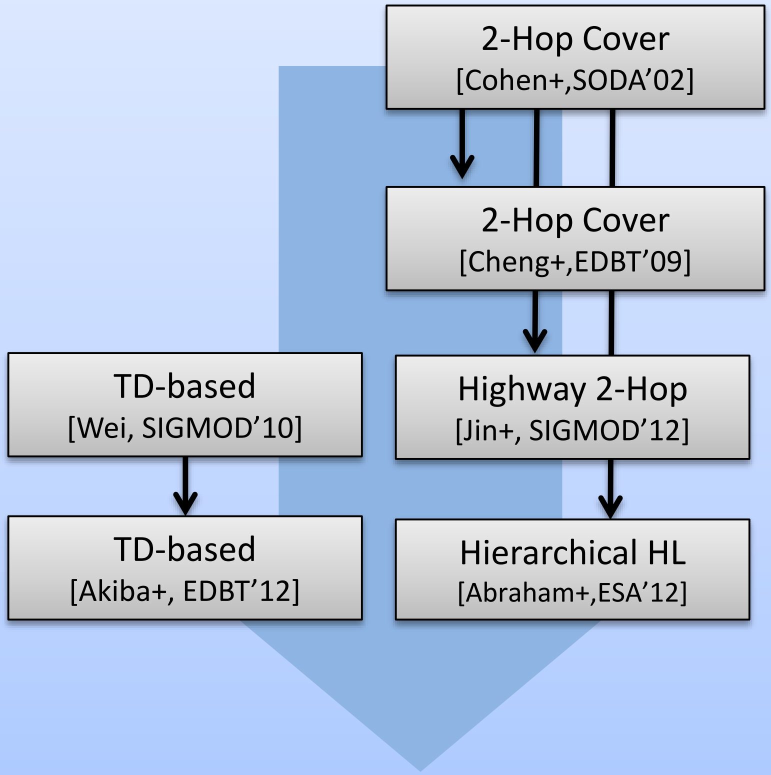
Complex Networks



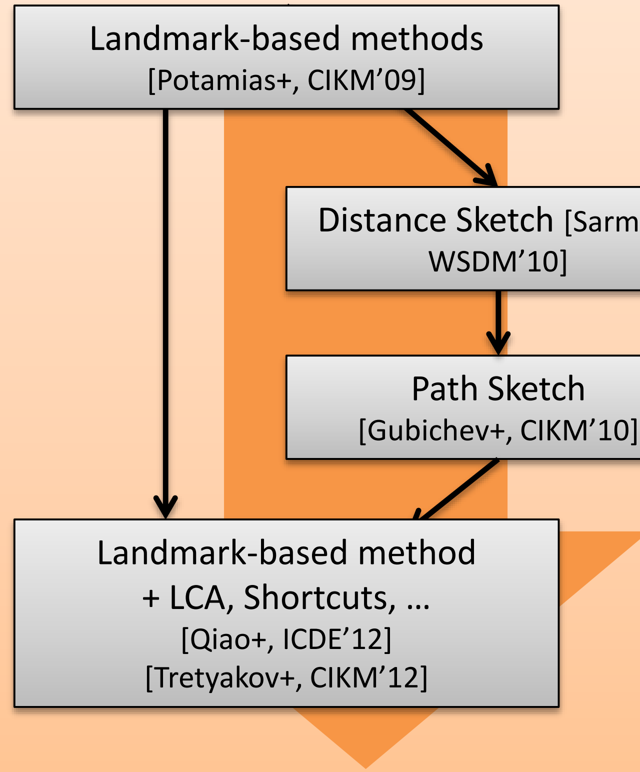
Today's topic:
methods for complex networks

Previous Methods

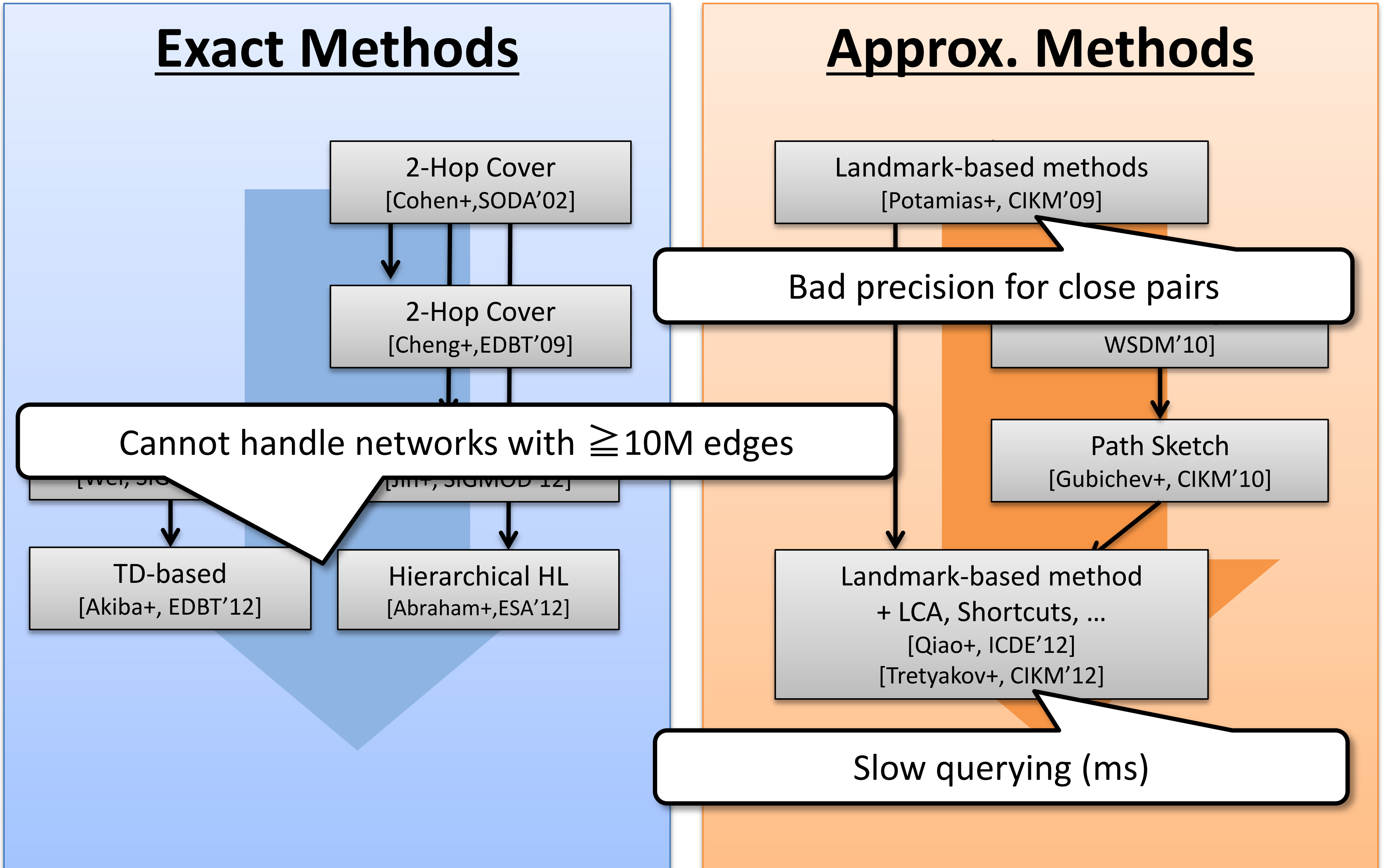
Exact Methods



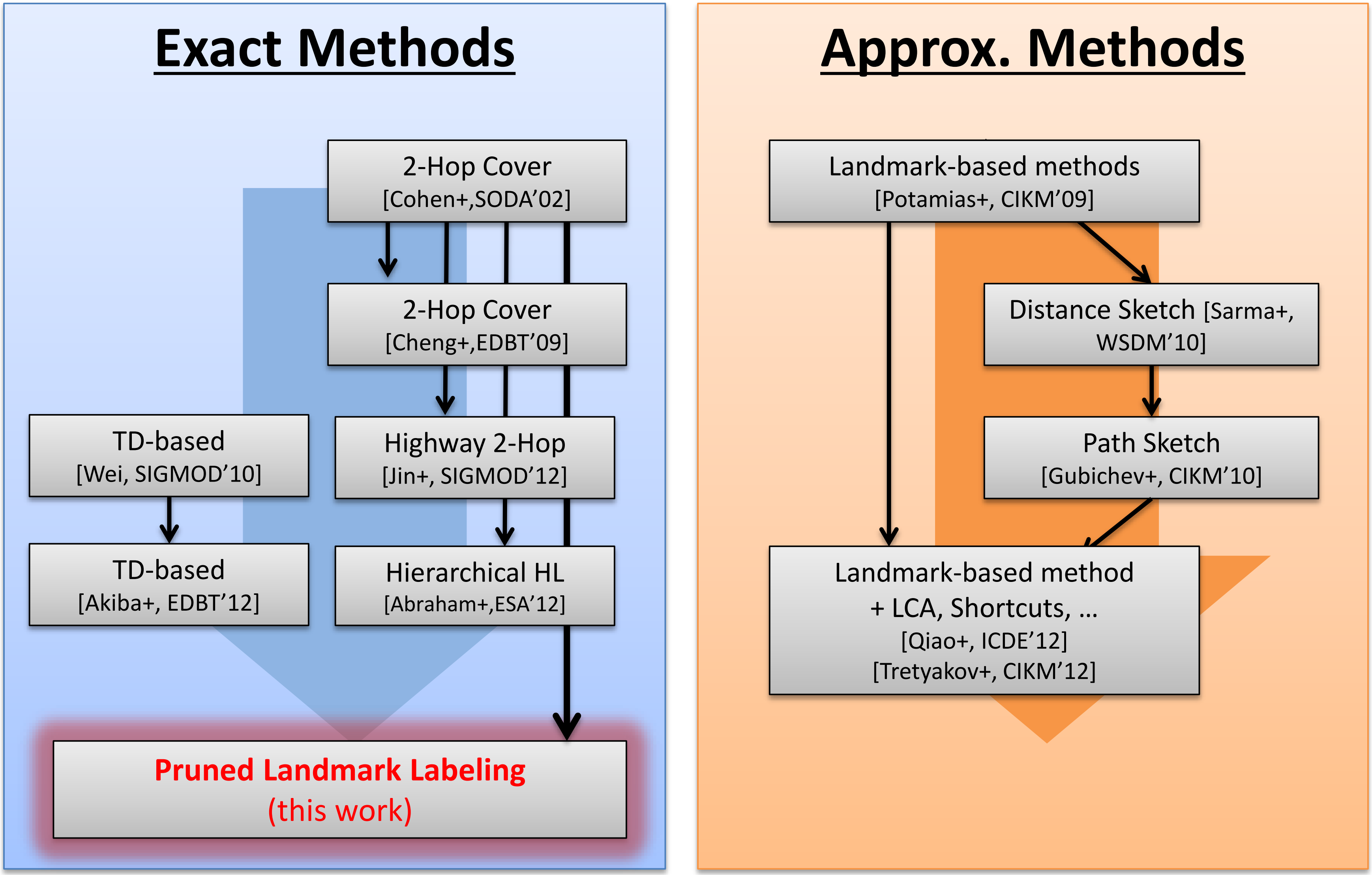
Approx. Methods



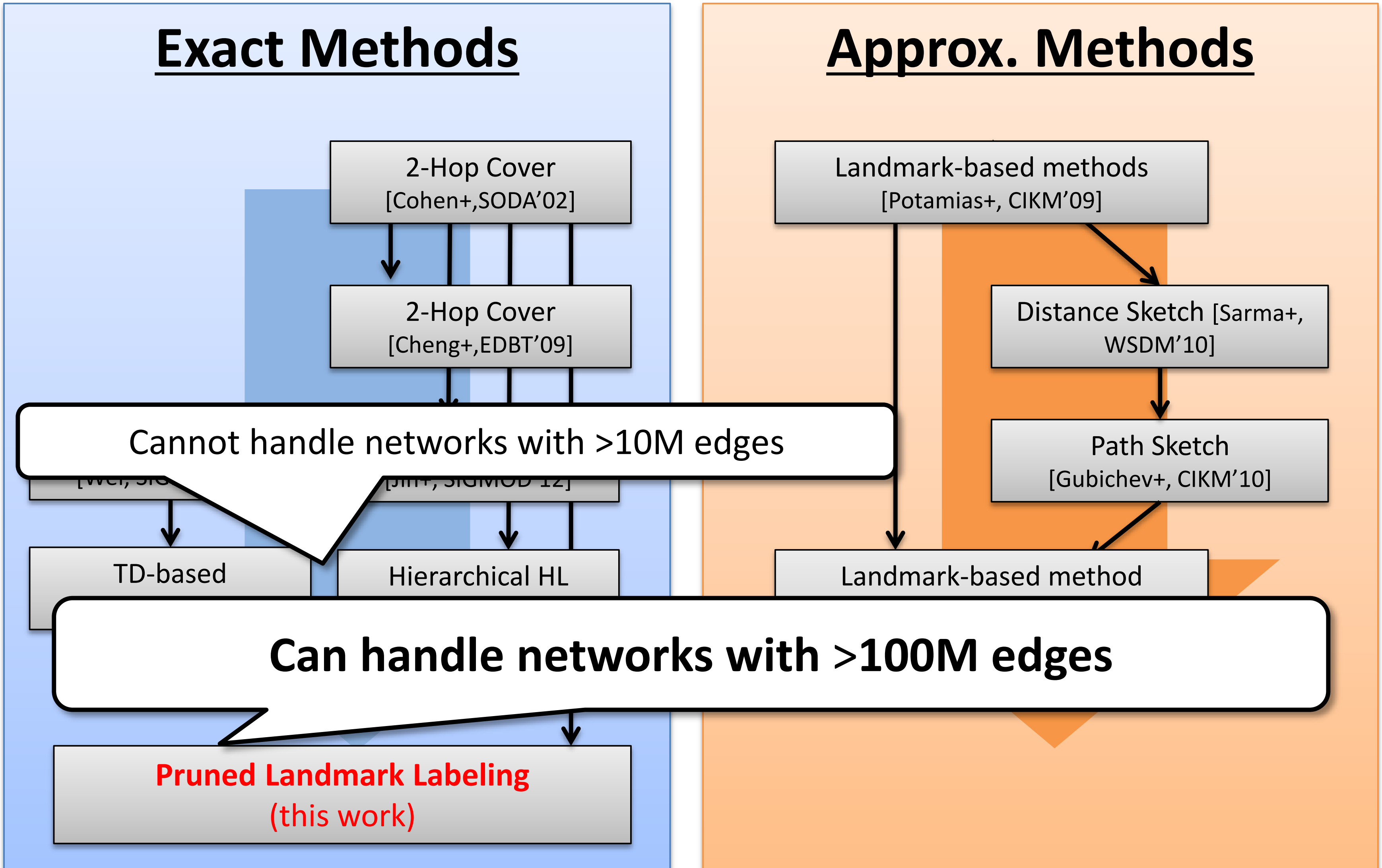
Previous Methods: Problems



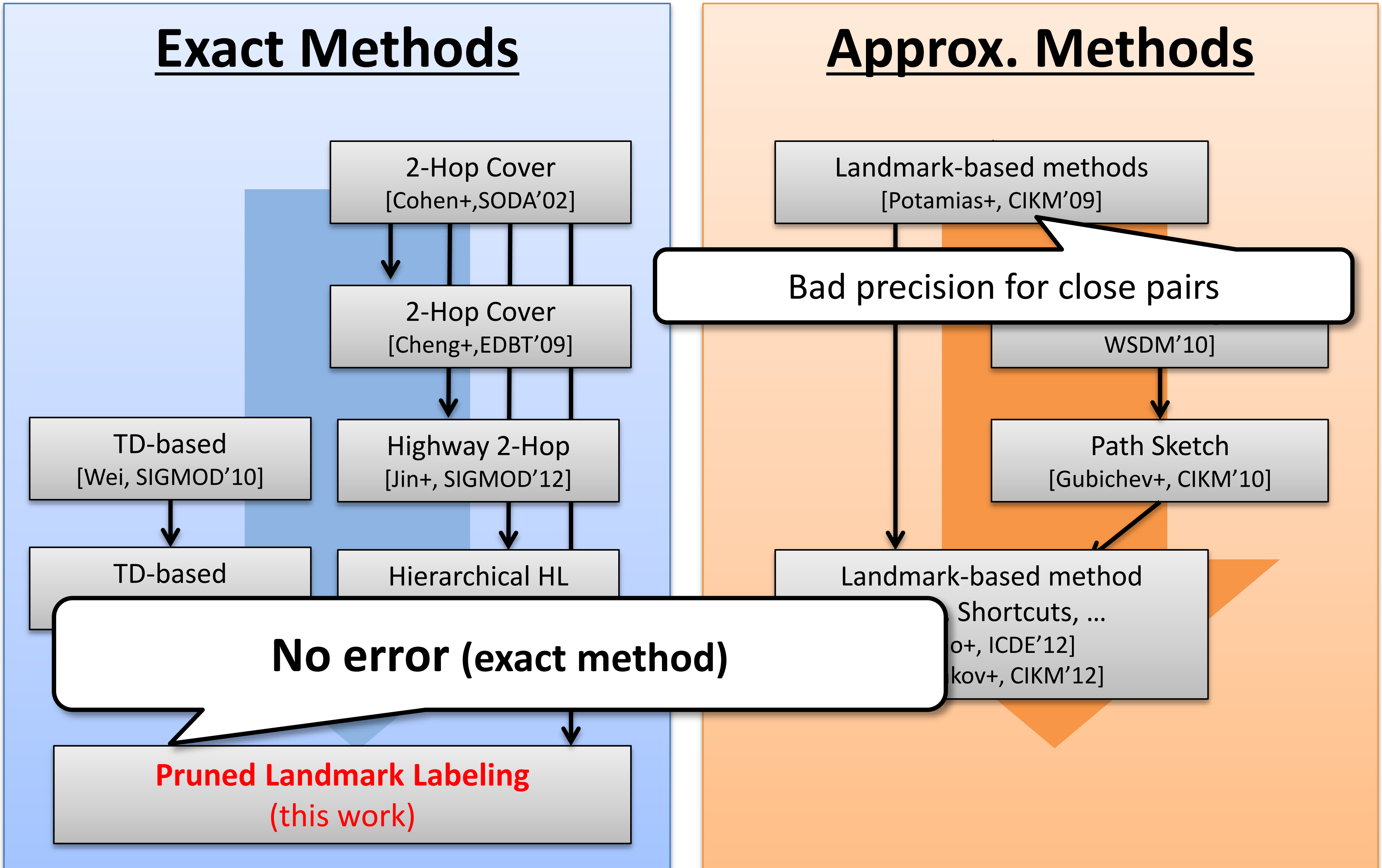
Proposed Method: *Pruned Landmark Labeling*



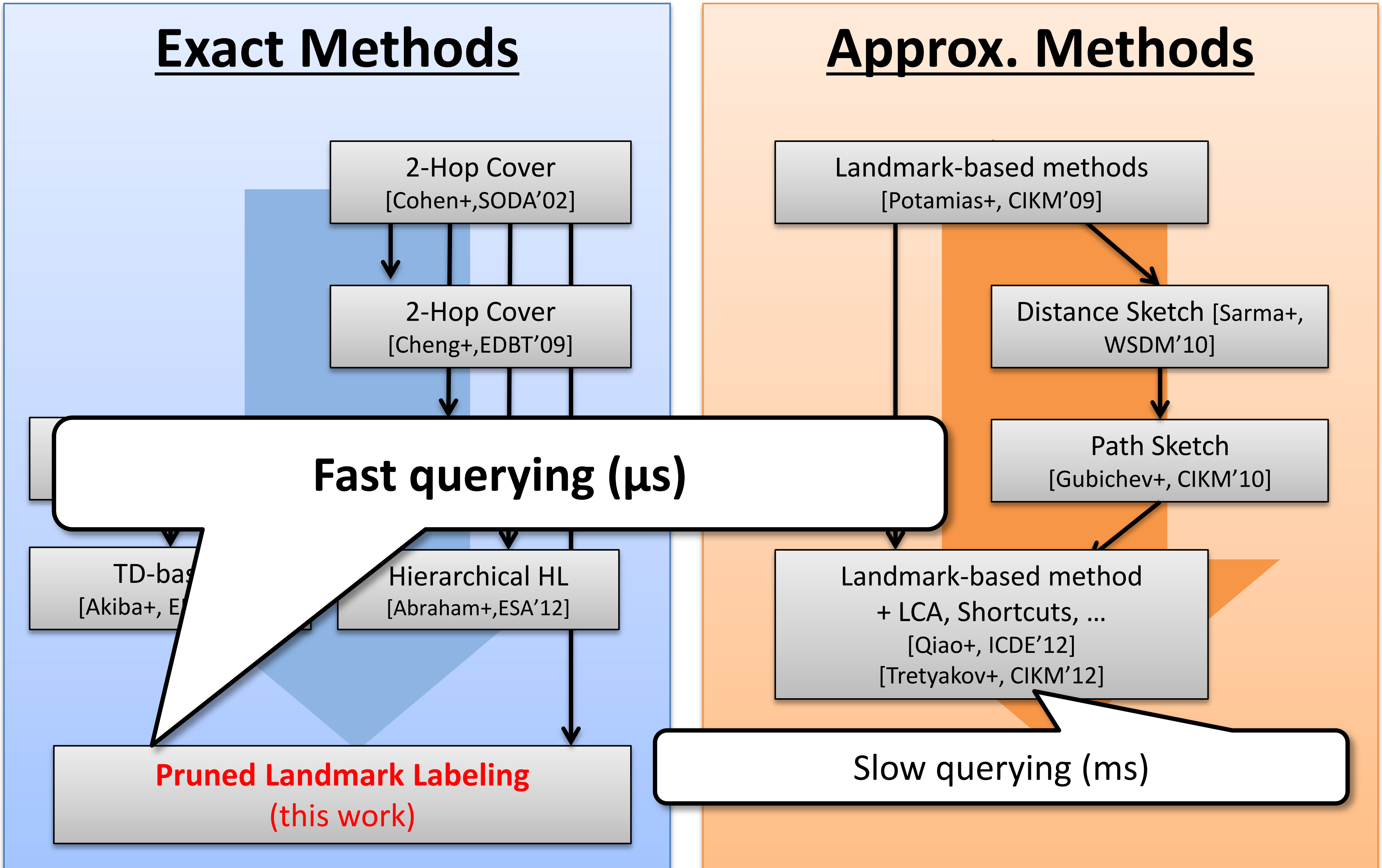
Proposed Method: Advantages



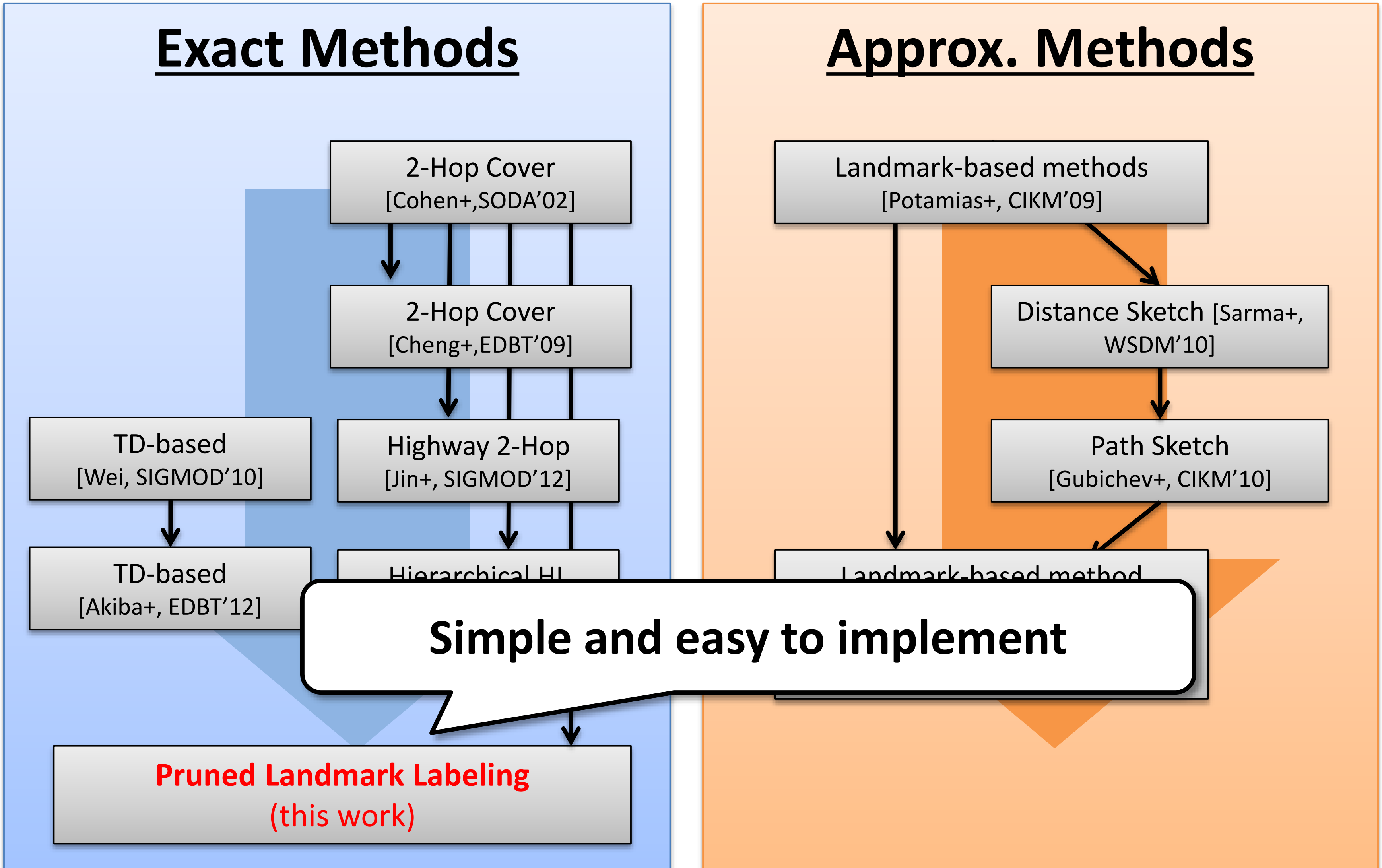
Proposed Method: Advantages



Proposed Method: Advantages



Proposed Method: Advantages



Preliminaries

Assumption

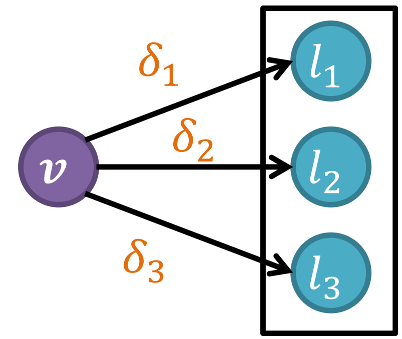
- **Undirected**
- **Unweighted**

(we can easily obtain **directed and/or weighted** version)

2-Hop Labeling: Index Data Structure

- Commonly used framework *(= Data Structure + Query Algo.)*
  - [Cohen+'02], [Cheng+'09], [Jin+'12], [Abraham+'12] and **ours**

- Index: label $L(v) = \{(l_1, \delta_1),\ (l_2, \delta_2),\ \ldots\}$
  - $l_i \in V,\ \ \delta_i = d_G(v, l_i)$



Example

$L(1)$:

Vertex	1	4	5	7	10
Distance	0	3	2	4	5

$d_G(1{,}10) = 5$

$L(2)$:

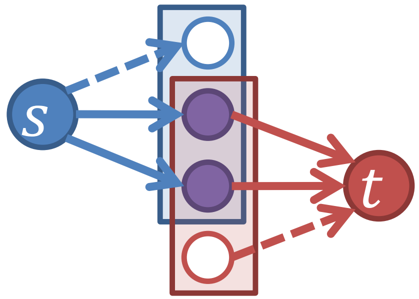
Vertex	2	4	6	12
Distance	0	1	5	3

$L(3)$:

Vertex	2	3	4	6	7
Distance	5	0	4	7	2

2-Hop Labeling: Query Algorithm

- Query: $d_G(s,t) = \min_{l \in L(s) \cap L(t)} d_G(s,l) + d_G(l,t)$
  - Paths through common vertices



Example

$L(1)$:

Vertex	1	4	5	7	10
Distance	0	3	2	4	5

$L(3)$:

Vertex	2	3	4	6	7
Distance	5	0	4	7	2

Distance between vertex **1** and **3**：

- **1 –…– 4 –…– 3** ： 3 + 4 = 7
- **1 –…– 7 –…– 3** ： 4 + 2 = 6

Answer min{6, 7} = **6**

2-Hop Labeling: Challenge

Challenge: computing labels

- Correctness *(Exactness)*
- Sizes of labels *(Index Size & Query Time)*
- Efficiency *(Scalability)*

Previous approach [Cohen+'02], [Cheng+'09], [Jin+'12], [Abraham+'12]

- Reduce to optimization problems

Our approach

- Directly assign label entries by graph searches

Pruned Landmark Labeling

Our Approach

1. Naïve Landmark Labeling

Conduct a BFS from every vertex

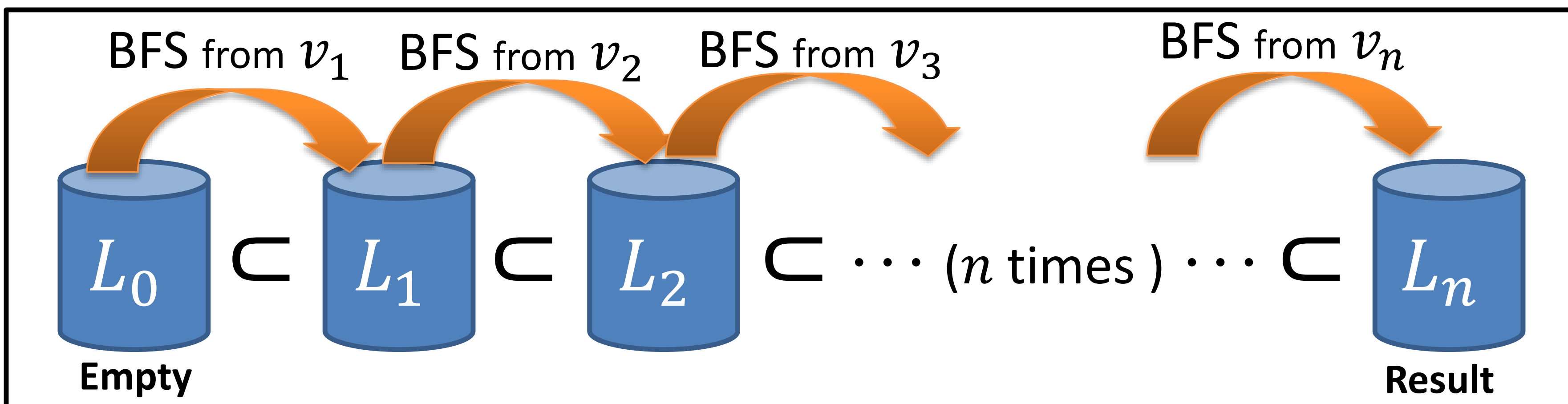
2. Pruned Landmark Labeling

(proposed method)

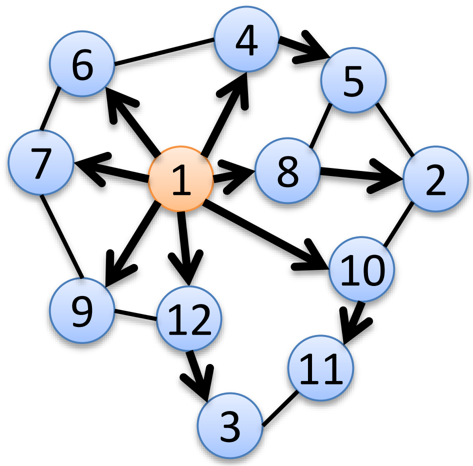
Pruning during BFSs

Naïve Landmark Labeling (w/o pruning)

1. $L_0 \leftarrow$ an empty index
2. For each vertex $v_1, v_2, \dots, v_n$
   - Conduct a BFS from v_i
   - Label all the visited vertices
     - $L_i(u) = L_{i-1}(u) \cup \left(v_i, d_G(u, v_i)\right)$



Naïve Landmark Labeling (w/o pruning)



After a BFS from 1

$L_1(1)$:

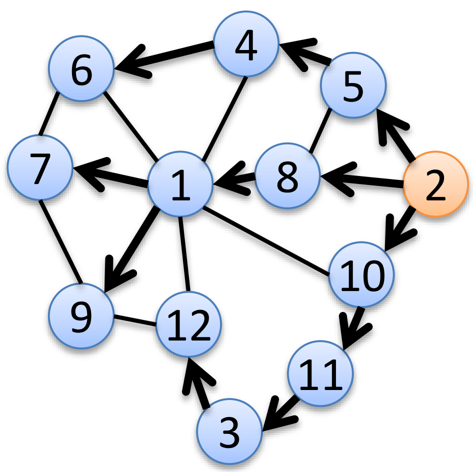
Vertex	1
Distance	0

$L_1(2)$:

Vertex	1
Distance	2

$L_1(3)$:

Vertex	1
Distance	2



After a BFS from 2

$L_2(1)$:

Vertex	1	2
Distance	0	2

$L_2(2)$:

Vertex	1	2
Distance	2	0

$L_2(3)$:

Vertex	1	2
Distance	2	3

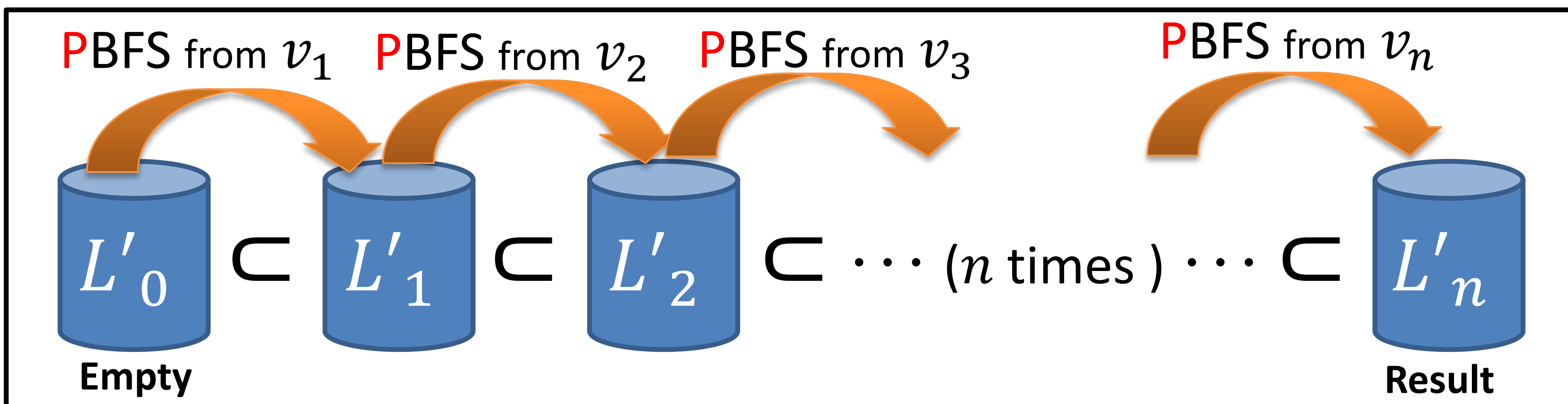
Naïve Landmark Labeling (w/o pruning)

1. $L_0 \leftarrow$ an empty index
2. For each vertex $v_1, v_2, \dots, v_n$
   - Conduct a BFS from v_i
   - Label all the visited vertices
     - $L_i(u) = L_{i-1}(u) \cup \left(v_i, d_G(u, v_i)\right)$

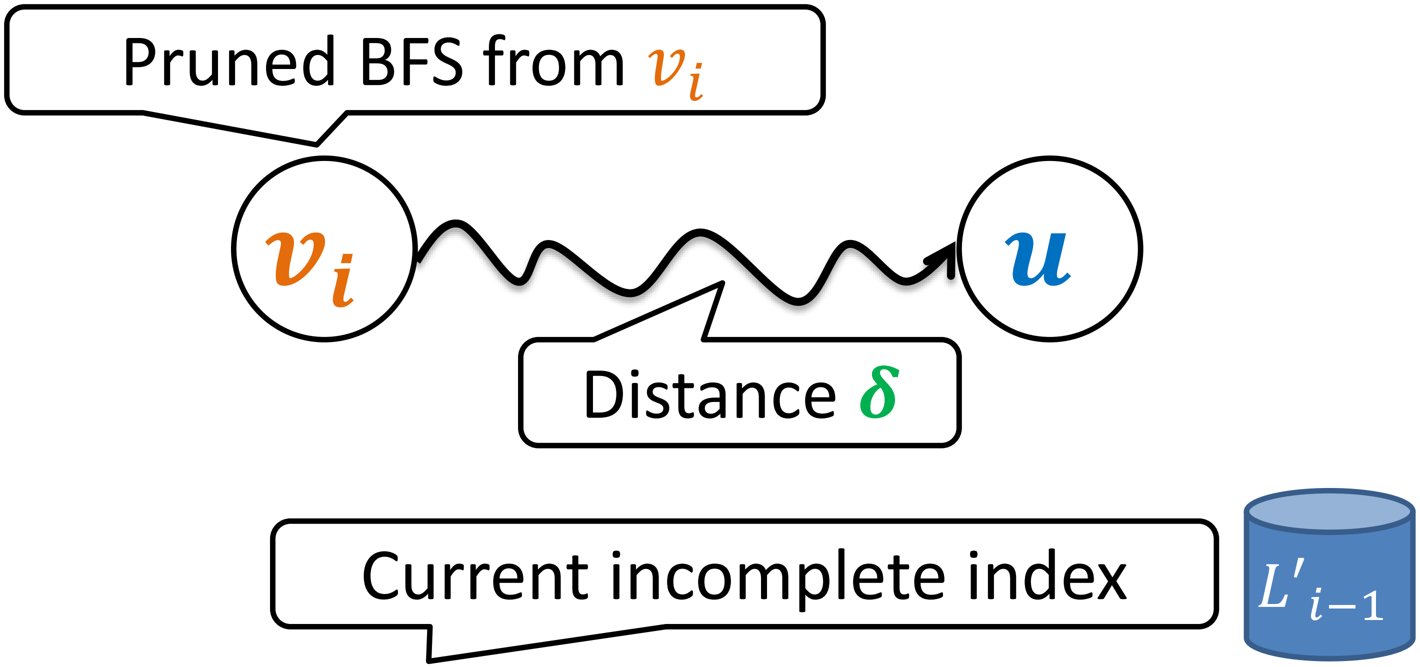
- $\Theta(nm)$ preprocessing time
- $\Theta(n^2)$ space
- Inpractical!

Pruned Landmark Labeling

1. $L'_0 \leftarrow$ an empty index
2. For each vertex $v_1, v_2, \ldots, v_n$
   - Conduct a pruned BFS from v_i
   - Label all the visited vertices
     - $L'_i(u) = L'_{i-1}(u) \cup \left(v_i, d_G(u, v_i)\right)$



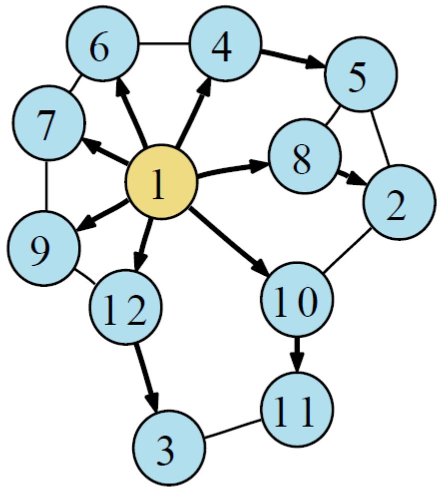
Pruned BFS



If $\text{Query}(v_i, u, L'_{i-1}) \leq \delta \rightarrow$ **Prune** u

- We do not label u this time
- We do not traverse edges from u

Example



First BFS from vertex 1

$L'_1(1)$:

Vertex	1
Distance	0

$L'_1(2)$:

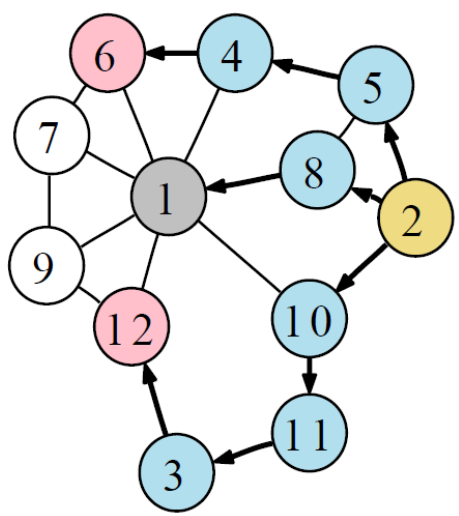
Vertex	1
Distance	2

⋮

$L'_1(6)$:

Vertex	1
Distance	1

⋮

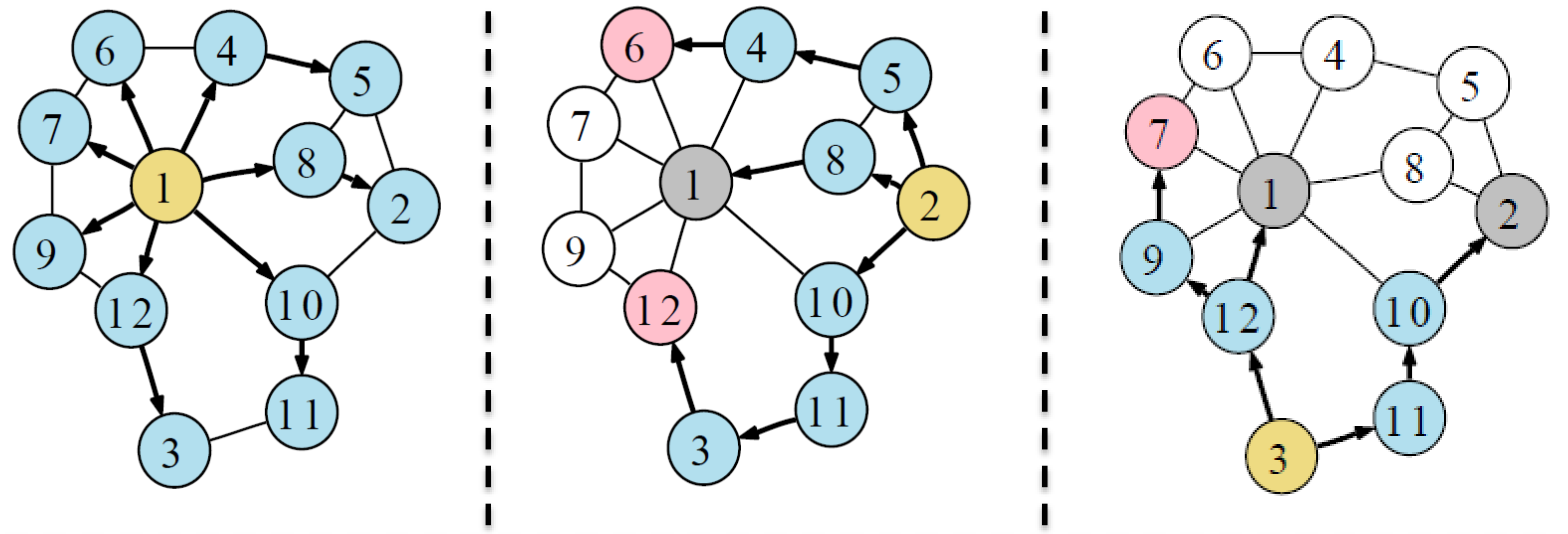


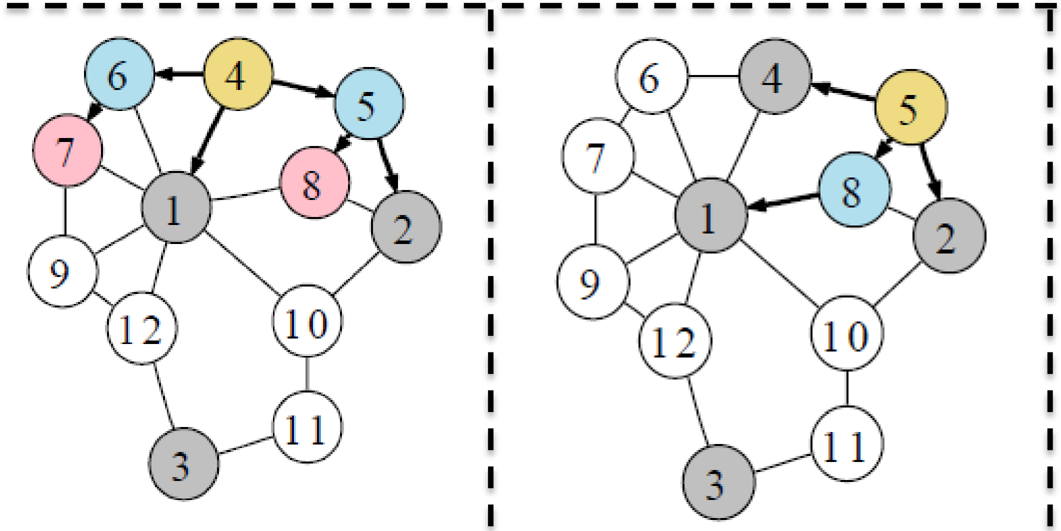
Second BFS from vertex 2

$\textsc{Query}(2, 6, L'_1) = 2 + 1 = 3 = d(2,6)$

→ Vertex 6 is pruned.

Example





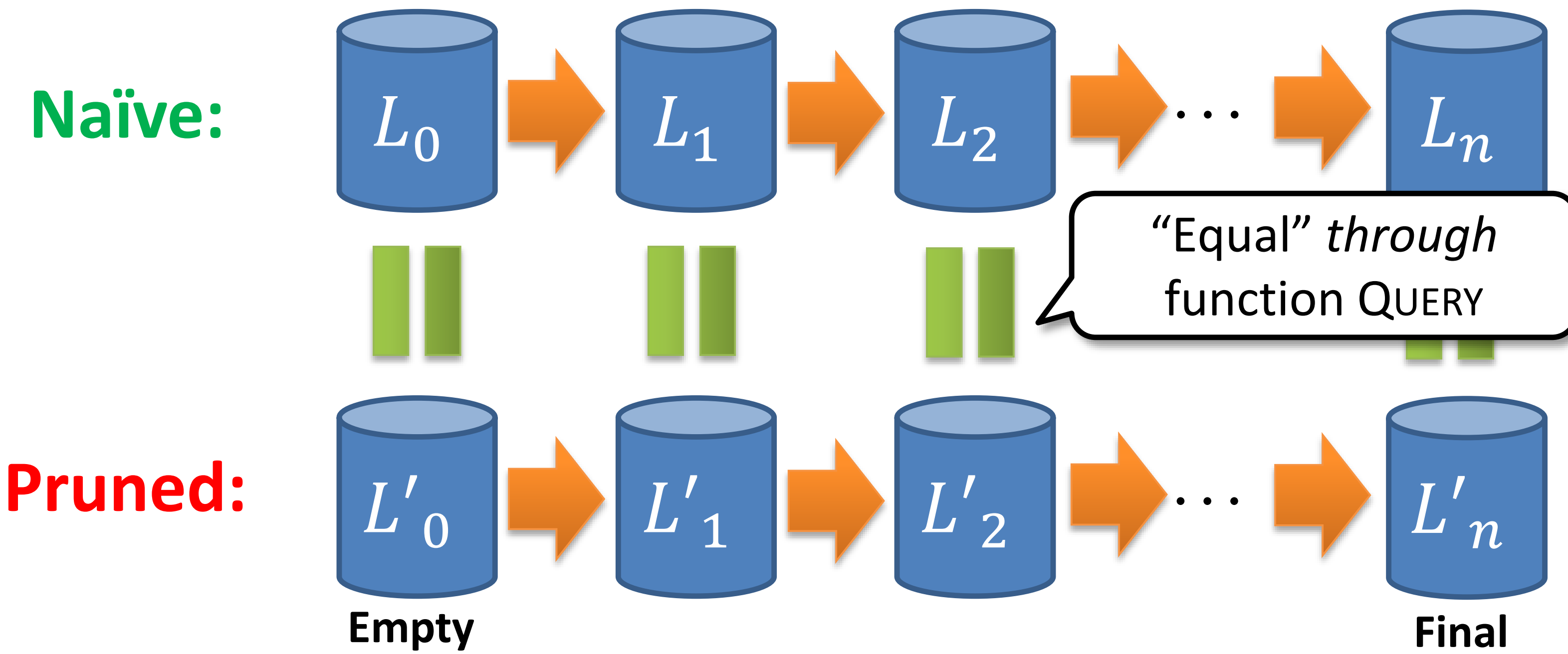
The search space gets smaller and smaller

Theorems: Correctness

Theorem 4.1

$$\textsc{Query}(s, t, L'_i) = \textsc{Query}(s, t, L_i)$$

for **any** s, t and i.



Theorems: Correctness

Theorem 4.1

$$\textsc{Query}(s, t, L'_i) = \textsc{Query}(s, t, L_i)$$

for **any** s, t and i.

Corollary 4.1 (Correctness)

$$\textsc{Query}(s, t, L'_n) = d_G(s, t)$$

for any s, t

i.e., our method is exact.

Theorems: Minimality

Theorem 4.2 (Minimality)

L'_n (the constructed index) is minimal.

i.e., we cannot remove any label entry from the index.

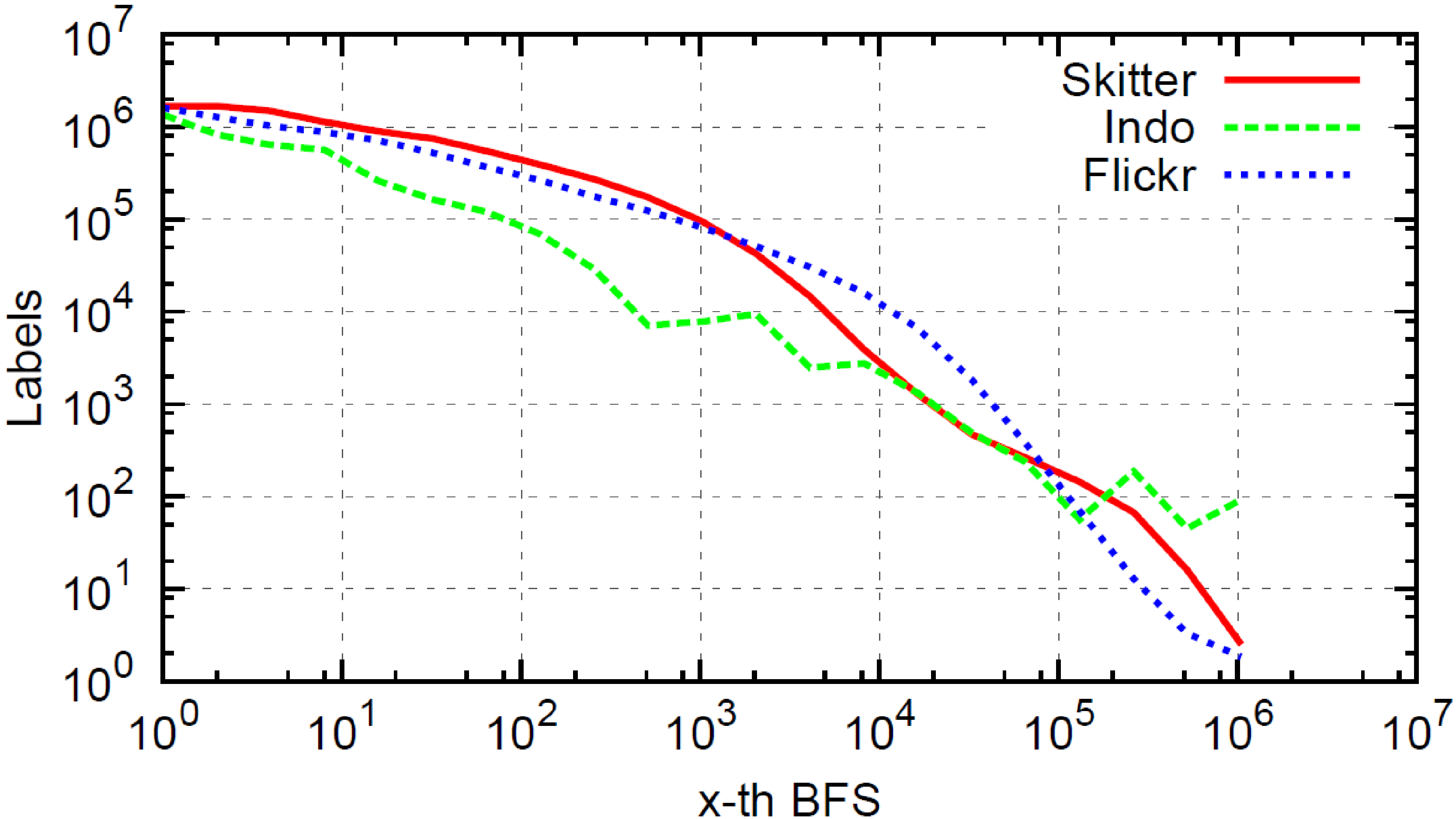
Vertex Ordering Strategies

Choosing the order of vertices to conduct BFSs from.

- To prune later BFSs as much as possible,
- *Central* vertices should come first

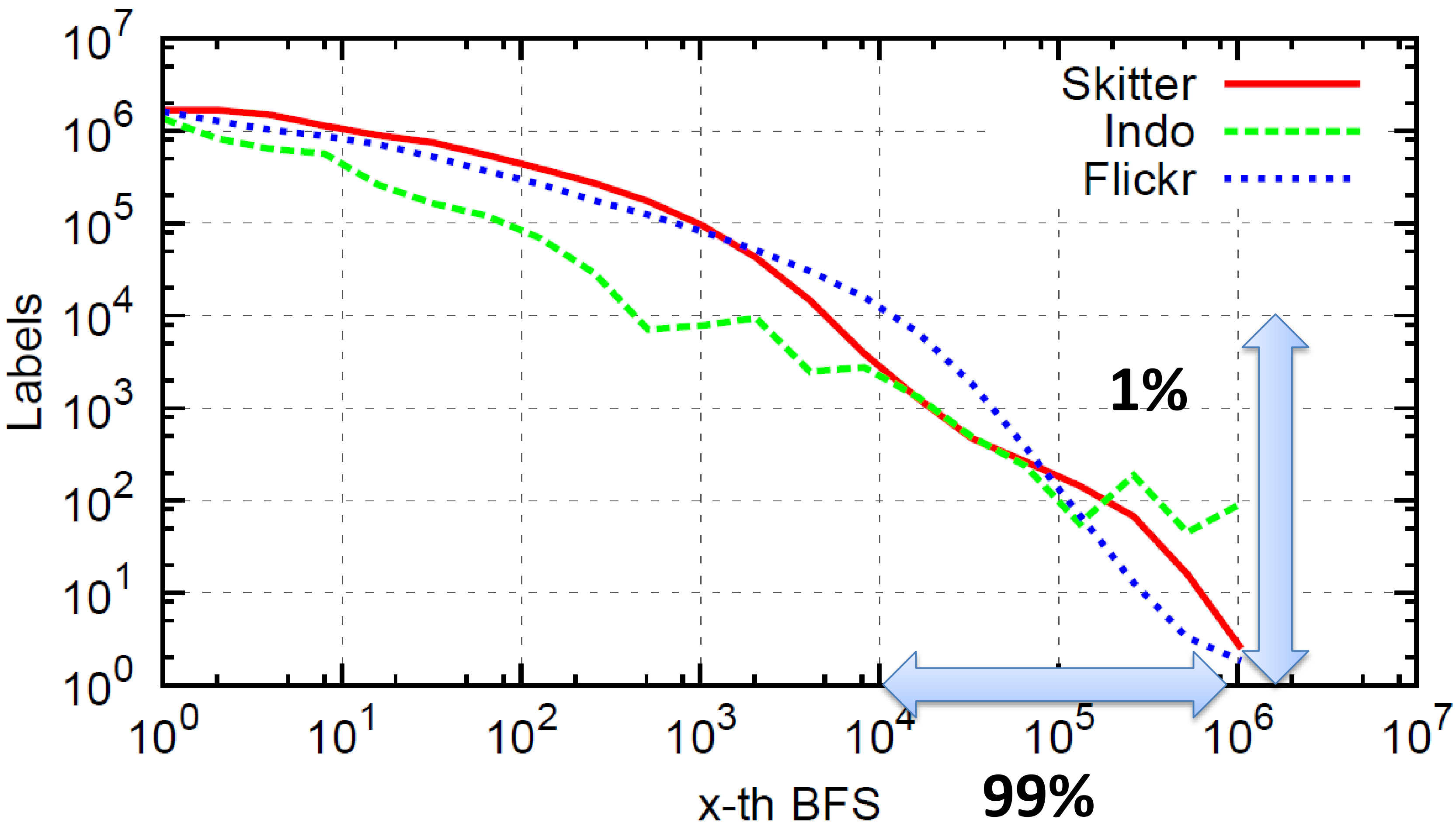
We conduct BFSs from vertices with **higher degree**

Pruning on Real-world Networks



Number of vertices labeled in each pruned BFS.

Pruning on Real-world Networks



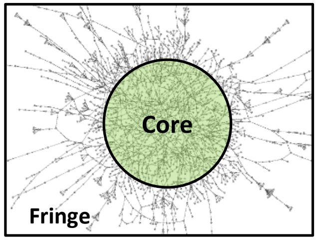
Number of vertices labeled in each pruned BFS.

Theorems: Relations between other methods

Landmark-based approx. methods
[Potamias+'09][Gubichev+'10][Sarma+'10][Qiao+'12][Tretyakov+'12]
Attain high average precision *by exploiting hubs*

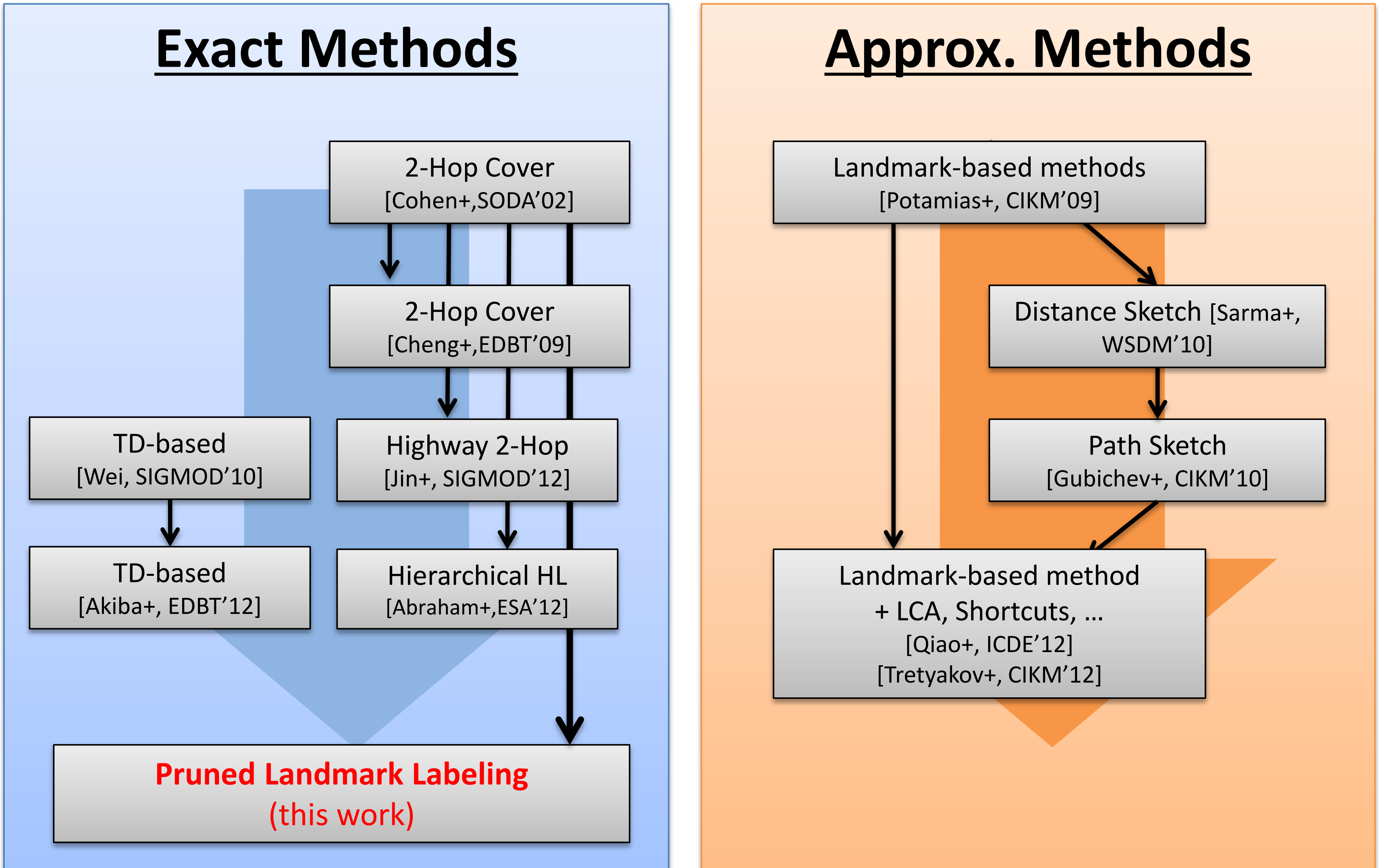
Tree-decomposition-based methods
[Wei'10], [Akiba+'12]
Attain good performance *by exploiting tree-like fringes*

Theorem 4.3
If landmark-based methods attain good precision on a graph
→ then PLL will have small label sizes
PLL can also exploit hubs (highly central vertices)

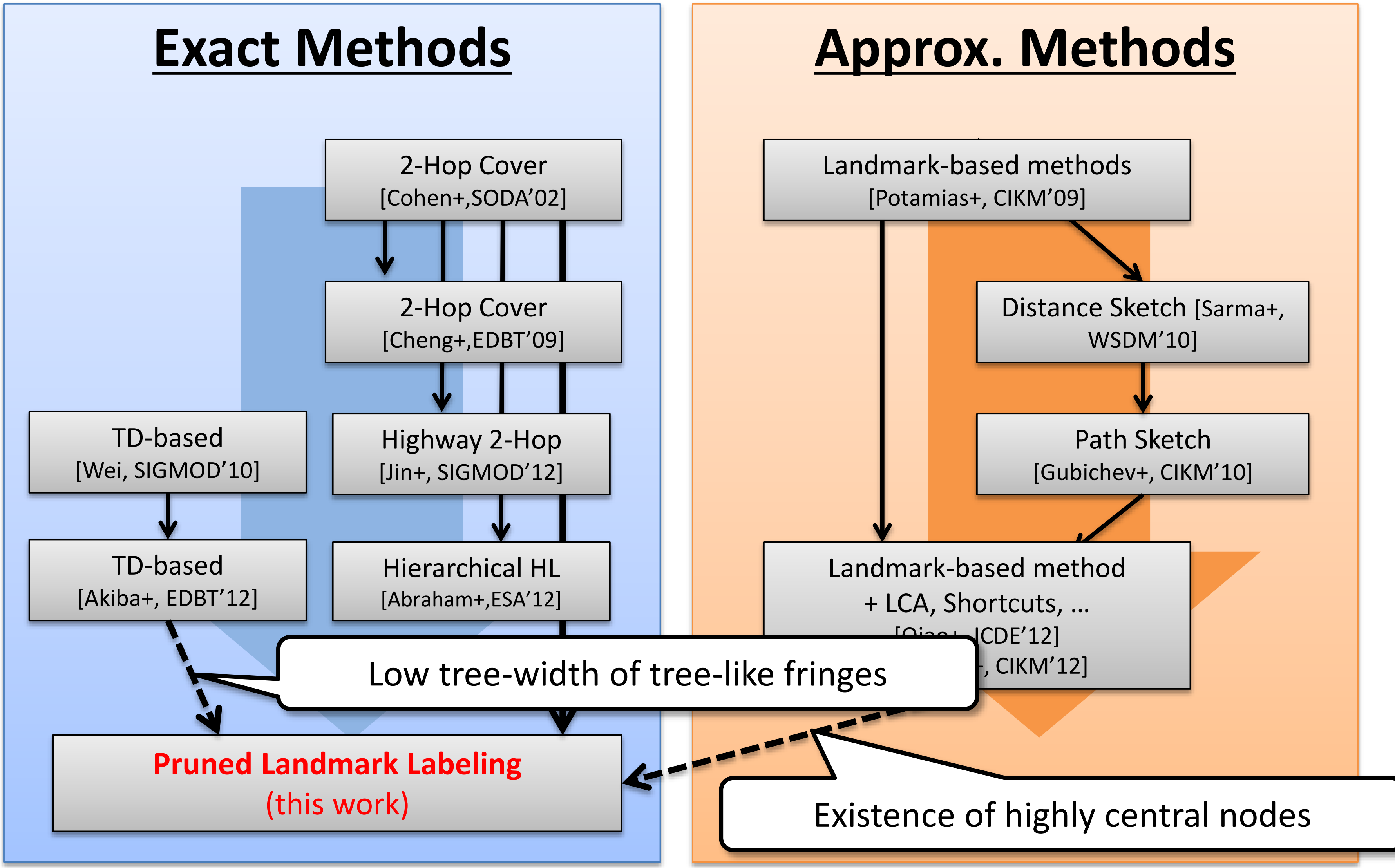


Theorem 4.4
PLL perform well on graphs with small tree-width
PLL can also exploit tree-like fringes

Combination of Advantages



Combination of Advantages



Bit-parallel Labeling

(or *broadword* labeling)

Two Labeling Schemes

Further improve the performance of pruned labeling

- In the beginning, pruning does not work much.
- Therefore, we ignore pruning in the beginning.
  - To skip the overhead of vain pruning testes
- We apply another labeling scheme here

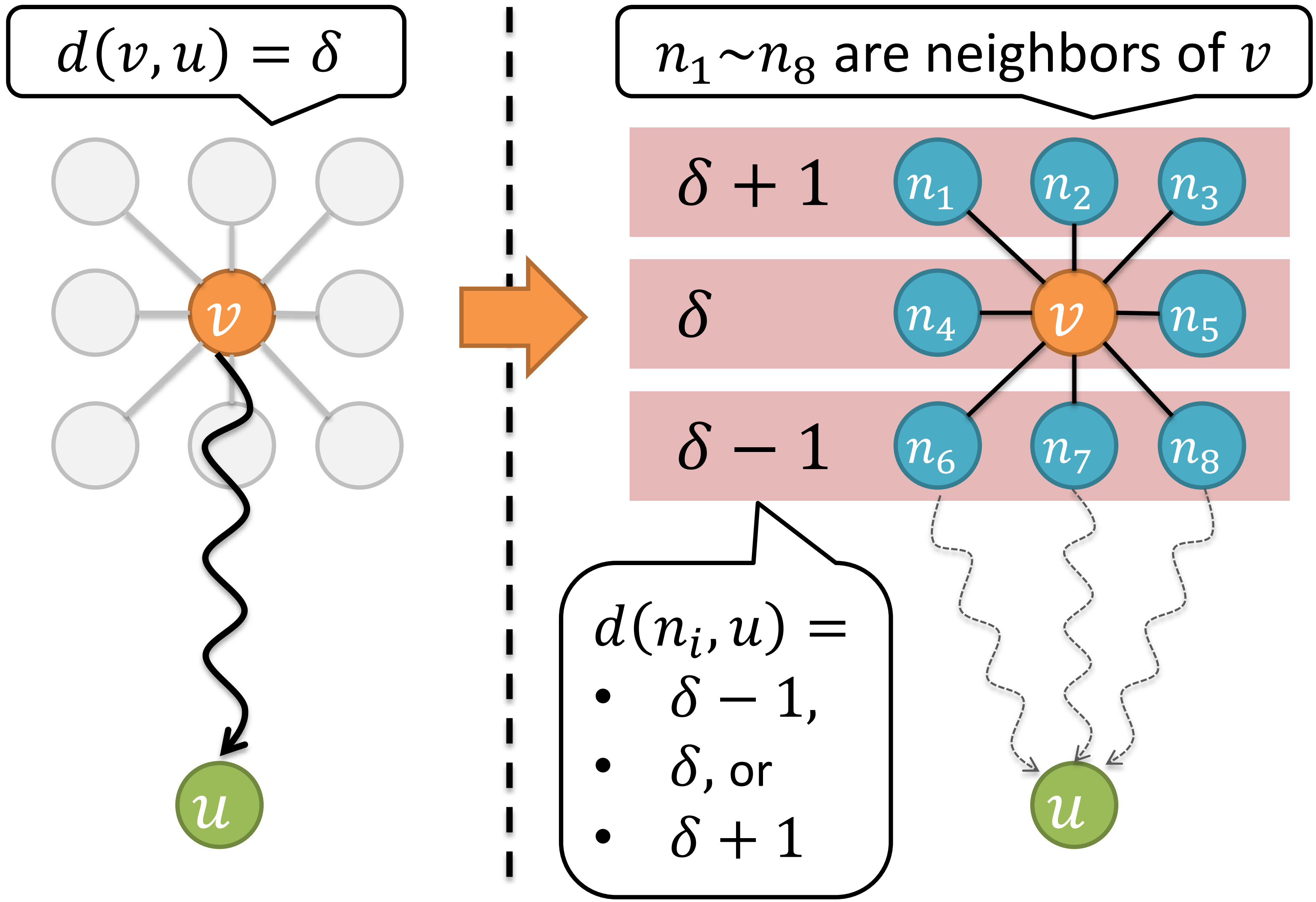
1. Bit-parallel Labeling
   - t times (typically <100)
2. Pruned Labeling
   - For the rest

(works only on unweighted graphs)

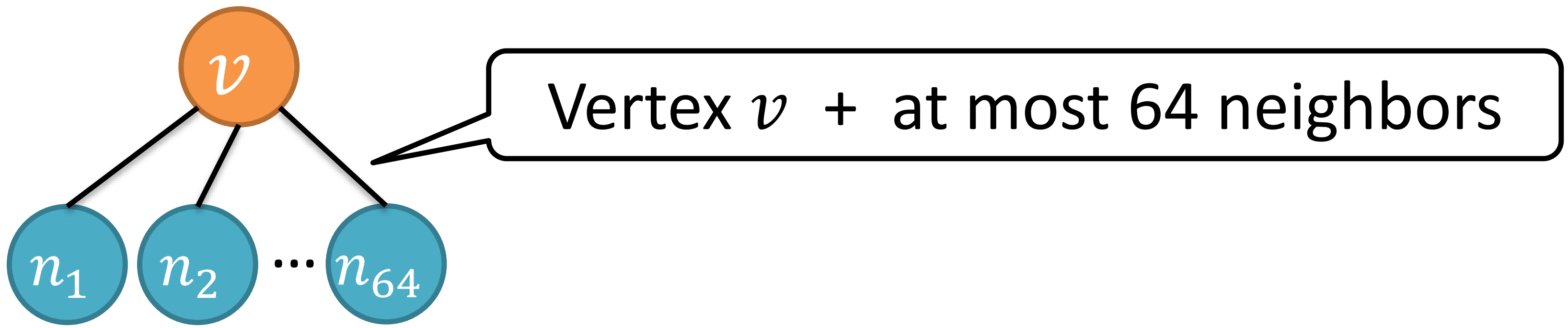
Naïve Landmark Labeling

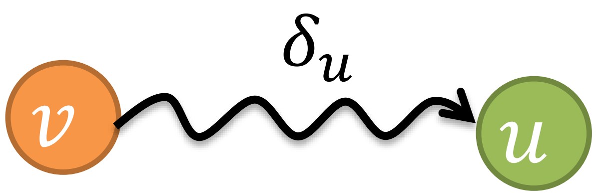
1. $L_0 \leftarrow$ an empty index
2. For each vertex $v_1, v_2, \ldots, v_n$
   - Conduct a BFS from v_i
   - Label all the visited vertices
     - $L_i(u) = L_{i-1}(u) \cup \left(v_i, d_G(u, v_i)\right)$

Key Insight: Distances of neighbors



Bit-parallel Labeling: 65 BFSs at once





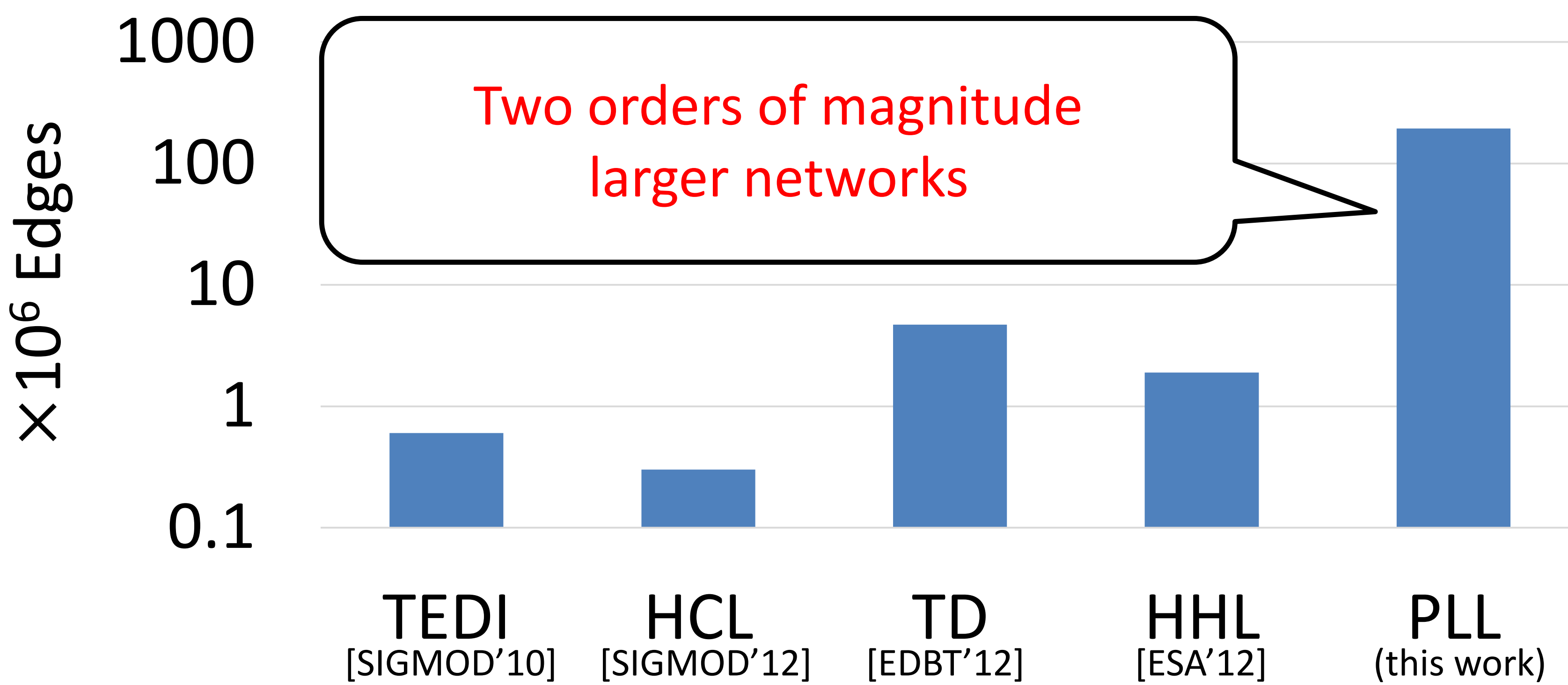
For each u, we compute:

- Distance δ_u *(BFS)*
- 64bit bitset $\times$ 2 *(Dynamic Programming)*
  - Neighbors with distance $\delta_u - 1$
  - Neighbors with distance δ_u
  - (Neighbors with distance $\delta_u + 1$)

Experiments

Scalability

Largest networks used in experiments *(indexing time: <1day)*



Summary of Experimental Results

- **Scalability:** much better
- **Query time and index size**: comparable

The paper contains detailed comparison and analysis

Conclusion

- **Distance querying**
- **Proposed method: *Pruned Landmark Labeling***
  - 2-hop cover
  - pruned BFSs + bit-parallel BFSs
- **Experiments**
  - Scalable
  - Fast
  - (easy to implement)

Software available: http://git.io/pll