Hierarchical Cut Labelling - Scaling Up Distance Queries on Road Networks

Muhammad Farhan[1], Henning Koehler[2], Robert Ohms[1], Qing Wang[1]

[1]Graph Research Lab, School of Computing, Australian National University

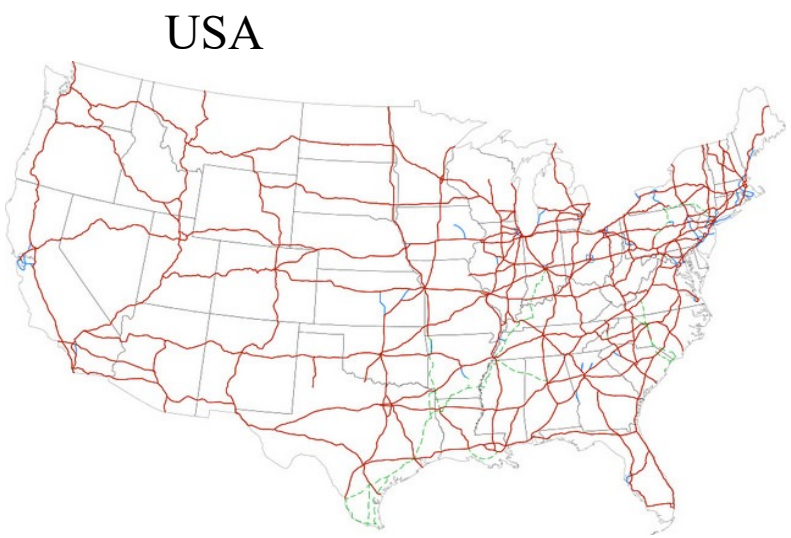
[2]School of Mathematical and Computational Sciences, Massey University

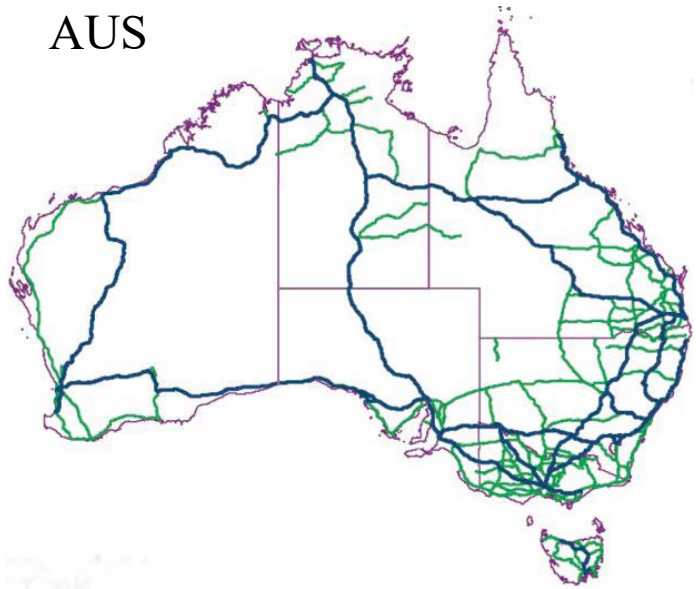
Australian National University

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

Road Networks

- **Road Network** is a low node degree and high diameter graph.

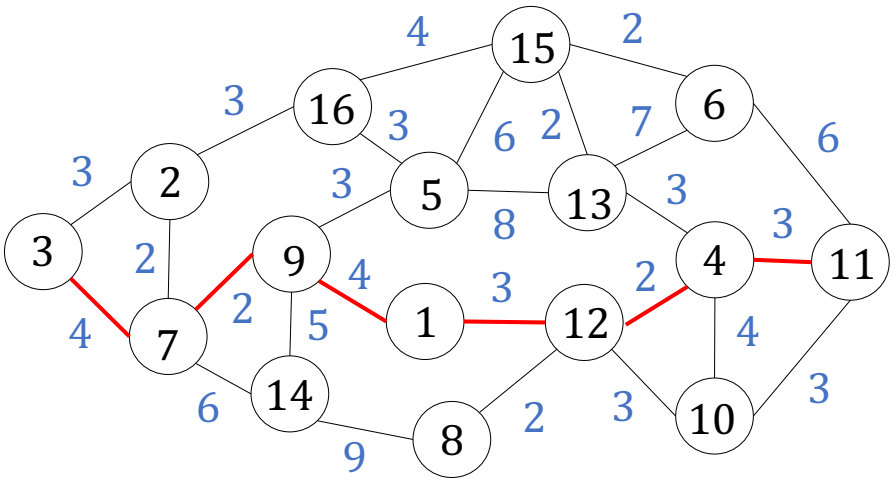




- Applications: GPS navigation, route planning, traffic monitoring, point-of-interest recommendation, etc.

Distance Query Problem

- Given a weighted graph $G = (V, E, \omega)$, how to compute the distance of a shortest path between two vertices?



– A shortest path between 3 and 11:

$\langle 3, 7, 9, 1, 12, 4, 11 \rangle$

– The distance between 3 and 11:

$d_G(3, 11) = 18$

2-Hop Cover Property

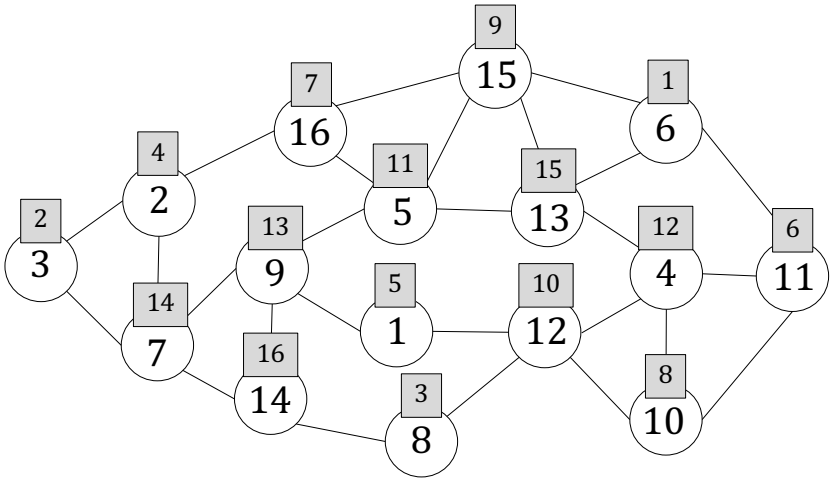
- The shortest-path distance between two vertices s and t can be computed via their labels $L(s)$ and $L(t)$:

$$d_G(s,t) = min\{\delta_{su} + \delta_{tu} : (u, \delta_{su}) \in L(s), (u, \delta_{tu}) \in L(t)\}.$$

- Three main labelling techniques that exploit hierarchical structure under 2-hop cover property:
  1. Hierarchical hub-based Labelling
  2. Highway-based Labelling
  3. Tree-Decomposition Labelling

Hierarchical Hub-based Labelling

- **Hierarchical hub-based labelling** defines a vertex hierarchy.



Vertex Hierarchy

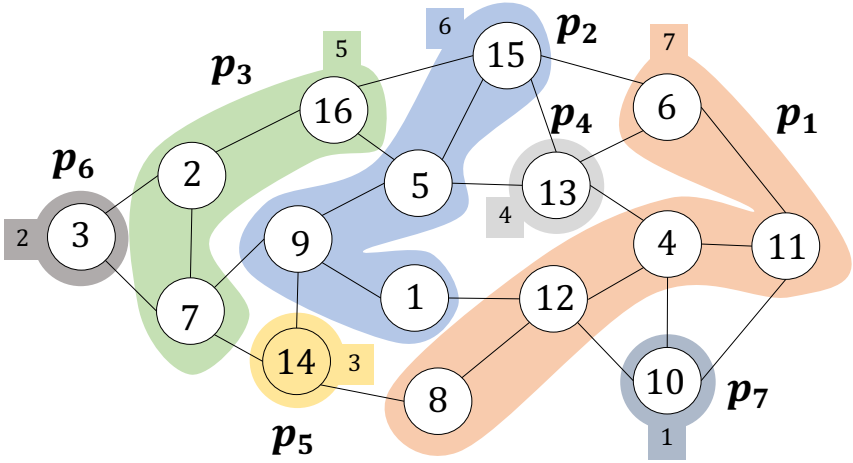
Label	Distance Entries
L(14)	(14, 0)
L(13)	(14 3) (13 0)
L(7)	(14 1) (13 3) (7 0)
L(9)	(14 1) (13 2) (7 1) (9 0)
L(4)	(14 3) (13 1) (4 0)
L(5)	(14 2) (13 1) (7 2) (9 1) (5 0)
L(12)	(14 2) (13 2) (9 2) (4 1) (12 0)
L(15)	(14 3) (13 1) (7 3) (9 2) (5 1) (15 0)
L(10)	(14 3) (13 2) (9 3) (4 1) (12 1) (10 0)
L(16)	(14 3) (13 2) (7 2) (9 2) (5 1) (15 1) (16 0)
L(11)	(14 4) (13 2) (4 1) (15 2) (10 1) (11 0)
L(1)	(14 2) (13 3) (7 2) (9 1) (4 2) (12 1) (1 0)
L(2)	(14 2) (13 3) (7 1) (5 2) (15 2) (16 1) (2 0)
L(8)	(14 1) (13 3) (4 2) (12 1) (8 0)
L(3)	(14 2) (13 4) (7 1) (15 3) (16 2) (2 1) (3 0)
L(6)	(14 4) (13 1) (15 1) (10 2) (11 1) (6 0)

Hierarchical Labelling

- **Limitations:**
  - Search over all entries in labels $L(s)$ & $L(t)$ for each (s, t)
  - Difficult to find a good vertex hierarchy

Highway-based Labelling

- **Highway-based labelling** defines a highway structure among vertices.



Highway Hierarchy

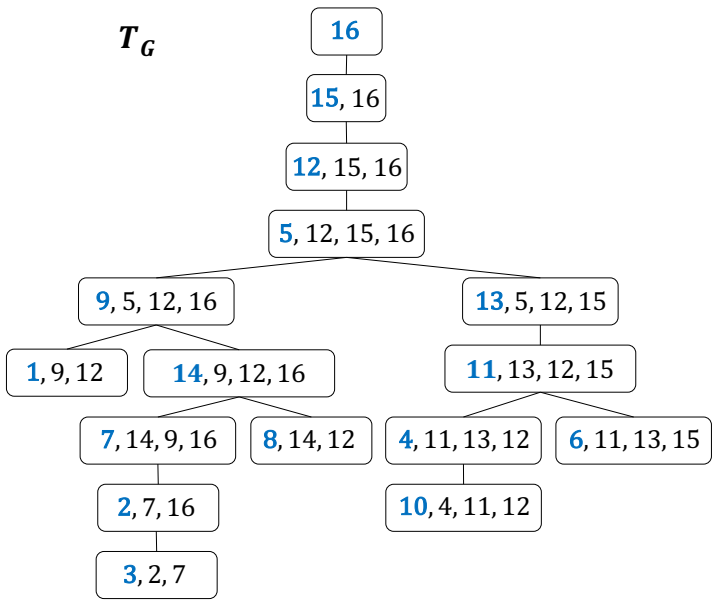
Label	Distance Entries
L(6)	(1, 6, 0)
L(11)	(1, 6, 1) (1, 11, 0)
L(4)	(1, 6, 2) (1, 11, 1) (1, 4, 0)
L(12)	(1, 6, 3) (1, 11, 2) (1, 4, 1) (1, 12, 0)
L(8)	(1, 6, 4) (1, 11, 3) (1, 4, 2) (1, 12, 1) (1, 8, 0)
L(1)	(1, 6, 4) (1, 11, 3) (1, 4, 2) (1, 12, 1) (2, 1, 0)
L(9)	(1, 6, 3) (1, 4, 3) (1, 9, 2) (1, 8, 2) (2, 1, 1) (2, 9, 0)
L(5)	(1, 6, 2) (1, 4, 2) (1, 8, 3) (2, 1, 2) (2, 9, 1) (2, 5, 0)
L(15)	(1, 6, 1) (1, 4, 2) (2, 1, 3) (2, 9, 2) (2, 5, 1) (2, 15, 0)
L(7)	(1, 6, 4) (1, 4, 4) (1, 12, 3) (1, 8, 2) (2, 9, 1) (3, 7, 0)
L(2)	(1, 6, 3) (1, 4, 4) (1, 12, 4) (1, 8, 3) (2, 1, 3) (2, 9, 2) (2, 5, 2) (2, 15, 2) (3, 7, 1) (3, 2, 0)
L(16)	(1, 6, 2) (1, 4, 3) (1, 8, 4) (2, 1, 3) (2, 9, 2) (2, 5, 1) (2, 15, 1) (3, 7, 2) (3, 2, 1) (3, 16, 0)
L(13)	(1, 6, 1) (1, 4, 1) (2, 9, 2) (2, 5, 1) (2, 15, 1) (4, 13, 0)
L(14)	(1, 6, 4) (1, 11, 4) (1, 4, 3) (1, 12, 2) (1, 8, 1) (2, 1, 2) (2, 9, 1) (3, 7, 1) (5, 14, 0)
L(3)	(1, 6, 4) (1, 4, 5) (1, 12, 4) (1, 8, 3) (2, 1, 3) (2, 9, 2) (2, 15, 3) (3, 7, 1) (3, 2, 1) (6, 3, 0)
L(10)	(1, 6, 2) (1, 11, 1) (1, 4, 1) (1, 12, 1) (7, 10, 0)

Pruned Highway Labelling

- **Limitations:**
  - Search over all entries in labels $L(s)$ & $L(t)$ for each (s, t)
  - Difficult to find a good highway hierarchy

Tree-Decomposition Labelling

- **Tree-decomposition labelling** defines a tree decomposition.



Tree decomposition T_G

Label	Position	Distance
L(1)	[3 5 6]	[3 3 1 2 1 0]
L(2)	[1 7 8]	[1 2 4 2 2 2 1 0]
L(3)	[7 8 9]	[2 3 4 3 2 2 1 1 0]
L(4)	[3 5 6 7]	[3 2 1 2 1 1 0]
L(5)	[1 2 3 4]	[1 1 3 0]
L(6)	[2 5 6 7]	[2 1 3 2 1 1 0]
L(7)	[1 5 6 7]	[2 3 3 2 1 1 0]
L(8)	[3 6 7]	[4 4 1 3 2 1 0]
L(9)	[1 3 4 5]	[2 2 2 1 0]
L(10)	[3 6 7 8]	[4 3 1 3 2 1 1 0]
L(11)	[2 3 5 6]	[3 2 2 3 2 0]
L(12)	[1 2 3]	[3 3 0]
L(13)	[2 3 4 5]	[2 1 2 1 0]
L(14)	[1 3 5 6]	[3 3 2 2 1 0]
L(15)	[1 2]	[1 0]
L(16)	[1]	[0]

H2H-Index

- **Limitations:**
  - Existing algorithms may yield very large tree width and height
  - Additional computational cost to compute LCA(s,t)

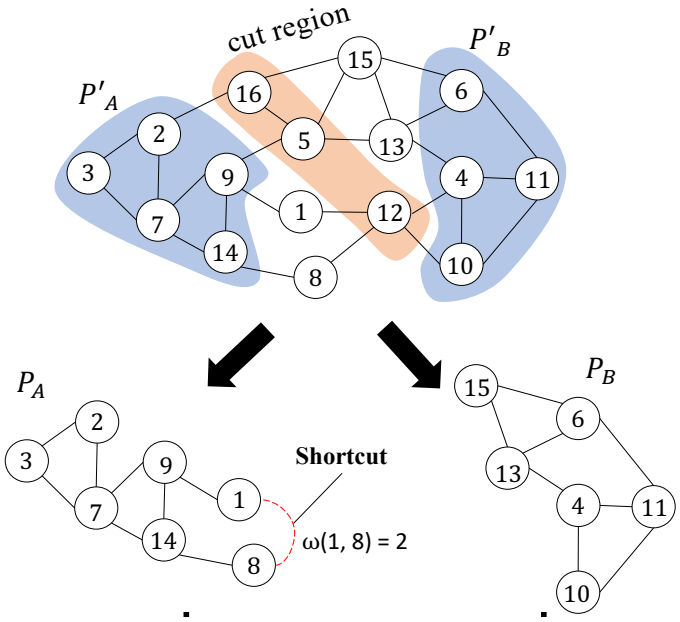
Can We Do Better?

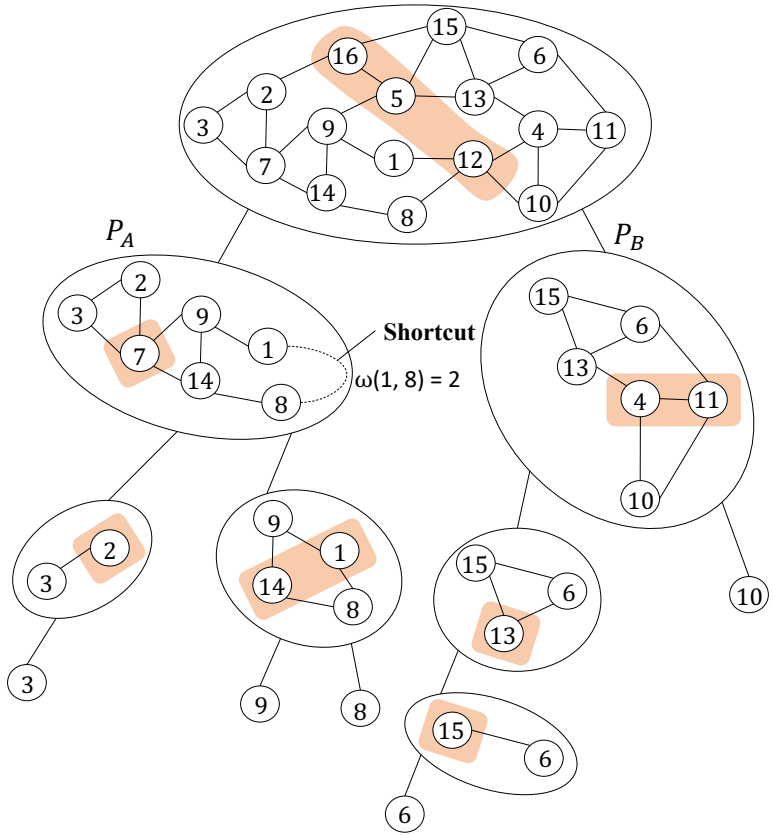
- **Some simple observations**:
  - 2-hop labelling needs to be remained for efficiency
  - Binary trees are efficient data structure for searching
  - A road network can be easily partitioned (due to low node degrees)
  - Bi-partitioning yields a vertex cut between two subsets of vertices
- **Some challenges**:
  - How to obtain a *balanced* binary tree with vertex cuts as nodes?
    $\hookrightarrow$ Devise an efficient algorithm for *hierarchical balanced cuts*
  - How to preserve distances partitioning?
    $\hookrightarrow$ Add *shortcuts* to ensure distance-preserving property
  - How to eliminate unnecessary distance entries in labels?
    $\hookrightarrow$ Apply *tail pruning strategy* to reduce labelling size

Our Solution - Hierarchical Balanced Cuts

(a) Hierarchical balanced cuts

- *Balanced partitioning*
- *Minimal vertex cuts*

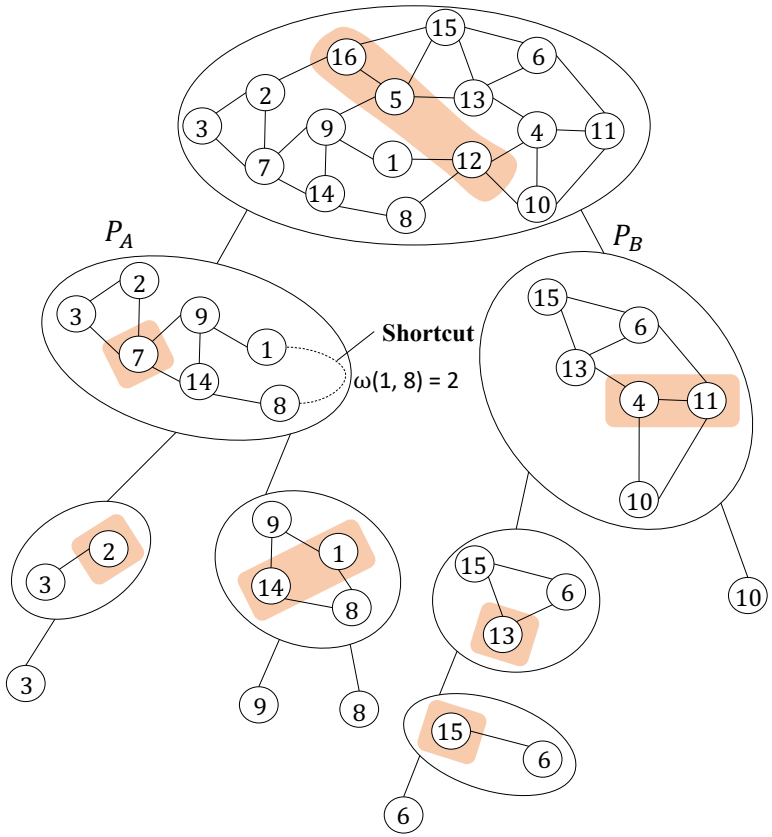


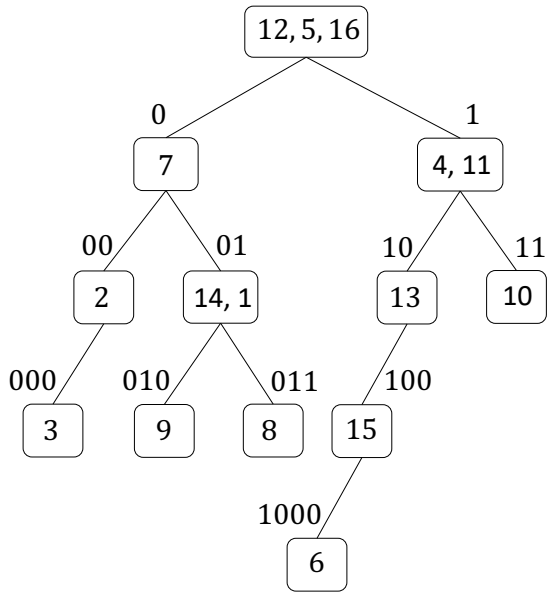


Our Solution - Balanced Tree Hierarchy

(b) Balanced tree hierarchy H_G

- A balanced binary tree, parameterized by a balancing parameter β
- At least one vertex in $LCA(s, t)$ on a shortest path between s and t

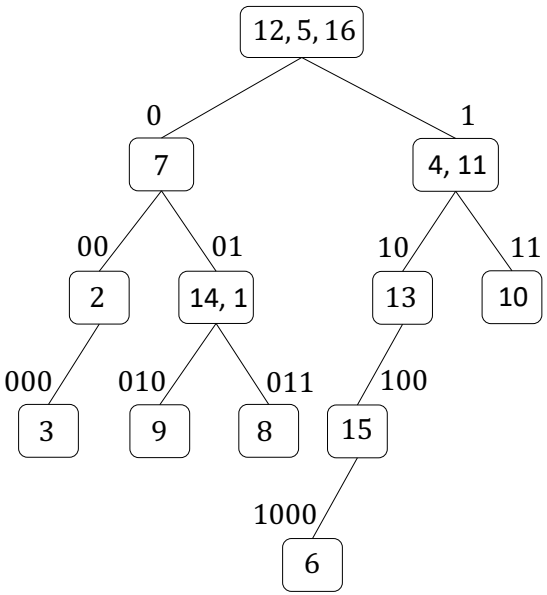




Our Solution - Hierarchical Cut 2-Hop Labelling

(c) Hierarchical cut 2-hop labelling

- Hierarchical labelling w.r.t. the *vertex quasi-order* decided by H_G
- 2-hop labelling s.t. a hub $r \in LCA(s, t)$ exists for any $\{s, t\} \subseteq V$



Label	Distance Entries
L(1)	[1 2], [2], [0]
L(2)	[4 2 1], [1], [0]
L(3)	[4 3 2], [1], [1], [0]
L(4)	[1 2], [0]
L(5)	[3 0]
L(6)	[3 2 2], [2 1], [1], [1], [0]
L(7)	[3 2 2], [0]
L(8)	[1 3], [2], [2 1], [0]
L(9)	[2 1], [1], [1 1], [0]
L(10)	[1 3], [1 1], [0]
L(11)	[2 3 3], [1 0]
L(12)	[0]
L(13)	[2 1], [1], [0]
L(14)	[2 2], [1], [2 0]
L(15)	[3 1 1], [2 2], [1], [0]
L(16)	[4 1 0]

Empirical Evaluation

- **Datasets**

Dataset	Region	$\|V\|$	$\|E\|$	*diam.*	Memory
NY	New York City	264,346	733,846	720	17 MB
BAY	San Francisco	321,270	800,172	721	18 MB
COL	Colorado	435,666	1,057,066	1,245	24 MB
FLA	Florida	1,070,376	2,712,798	2,058	62 MB
CAL	California	1,890,815	4,657,742	2,315	107 MB
E	Eastern USA	3,598,623	8,778,114	4,461	201 MB
W	Western USA	6,262,104	15,248,146	4,420	349 MB
CTR	Central USA	14,081,816	34,292,496	5,533	785 MB
USA	United States	23,947,347	58,333,344	8,440	1.30 GB
EUR	Western Europe	18,010,173	42,560,279	5,175	974 MB

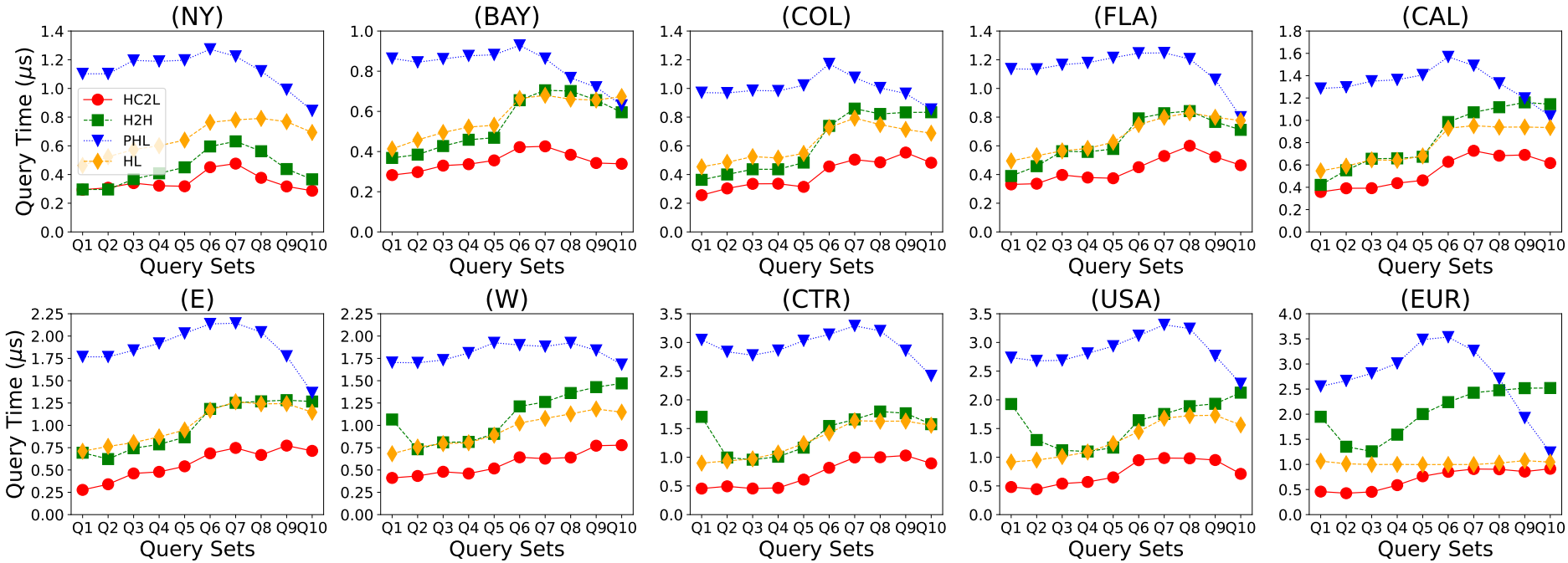
- **Baseline algorithms**
  - Hub Labelling (HL)
  - Pruned Highway Labelling (PHL)
  - Hierarchical 2-Hop Labelling (H2H)
  - Projected Vertex Separator Based 2-Hop Labelling (P2H)

Empirical Evaluation

Dataset	Query Time [μs]				Labelling Size				Construction Time [s]				
	HC2L	H2H	PHL	HL	HC2L	H2H	PHL	HL	HC2L	HC2L^p	H2H	PHL	HL
NY	0.225	0.432	0.983	0.765	144 MB	341 MB	320 MB	233 MB	15	6	16	34	32
BAY	0.220	0.563	0.707	0.665	113 MB	339 MB	235 MB	219 MB	12	4	12	18	27
COL	0.351	0.750	0.909	0.720	236 MB	217 MB	403 MB	341 MB	27	12	21	38	45
FLA	0.371	0.754	0.965	0.827	487 MB	1.25 GB	1.14 GB	907 MB	68	23	46	121	137
CAL	0.442	1.125	1.106	0.958	1.24 GB	3.87 GB	2.58 GB	1.78 GB	215	57	146	327	318
E	0.555	1.241	1.671	1.218	3.37 GB	9.81 GB	8.44 GB	4.74 GB	654	163	409	1,578	1,149
W	0.583	1.382	1.661	1.163	5.71 GB	18.3 GB	13.5 GB	7.50 GB	1,197	261	702	2,314	1,654
CTR	0.760	1.630	2.503	1.613	24.4 GB	73.9 GB	55.9 GB	25.5 GB	6,203	1,658	4,029	15,882	7,591
USA	0.737	1.940	2.389	1.663	45.1 GB	155 GB	95.6 GB	44.7 GB	11,203	1,977	7,737	26,515	13,157
EUR	0.922	2.414	2.239	1.673	44.1 GB	160 GB	70.9 GB	34.1 GB	12,242	3,083	9,194	20,466	8,728

- *How does our solution perform, compared with STOA (e.g., H2H)?*
  - Query time: reduced to less 50%
  - Labelling size: up to 60% smaller
  - Construction time: parallelized to be about 3 times faster

Empirical Evaluation



- *How does our solution perform for answering distance queries with varying distances?*
  - Our solution (red) consistently outperforms existing solutions.

Concluding Remarks

- Recursive partitioning algorithm is built upon existing ideas

  Ideas are not new, but used from a new perspective.

- Balanced tree hierarchy is a simple structure

  Simple is often better: efficient to store and search.

- Hierarchical cut 2-hop labelling is scalable on large road network

  The power lies in "balanced tree hierarchy".

Questions

Thank You