Dual-Hierarchy Labelling: Scaling Up Distance Queries on Dynamic Road Networks

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

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

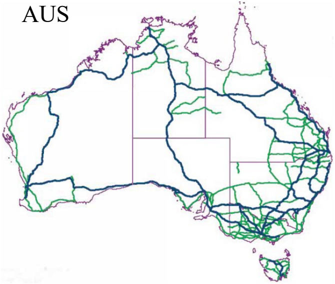
[2]School of Mathematical and Computational Sciences, Massey University

Road Networks

- **Road Networks** are low node degree and high diameter graphs.

USA

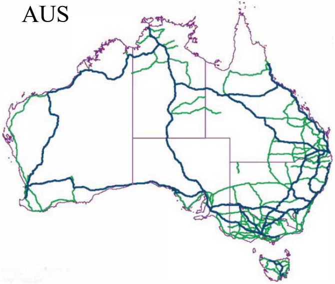
AUS



- **Road Networks** are low node degree and high diameter graphs.

USA

AUS

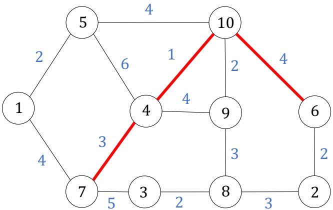


- Applications: GPS navigation, route planning, traffic monitoring, point-of-interest recommendation, etc.

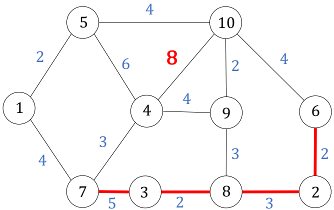
Distance Query Problem

- Given a weighted graph $G = (V, E, \omega)$ undergoing edge weight updates, compute the length of a shortest path between two vertices.

- Given a weighted graph $G = (V, E, \omega)$ undergoing edge weight updates, compute the length of a shortest path between two vertices.



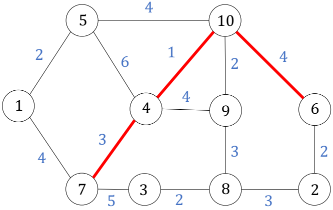
Before update



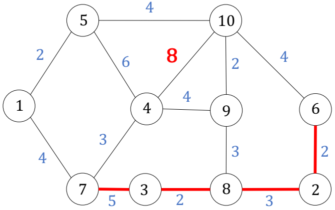
After update

Distance Query Problem

- Given a weighted graph $G = (V, E, \omega)$ undergoing edge weight updates, compute the length of a shortest path between two vertices.



Before update



After update

– A shortest path between 6 and 7:

Before update: $\langle 6, 10, 4, 7 \rangle$

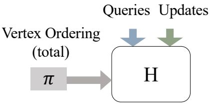
After update: $\langle 6, 2, 8, 3, 7 \rangle$

– The distance between 6 and 7:

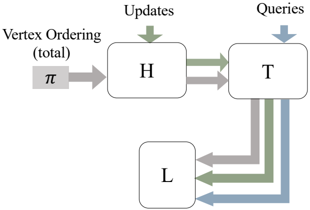
Before update: $d_G(6, 7) = 8$

After update: $d_G(6, 7) = 12$

High-level Overview

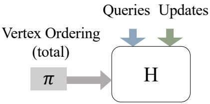


DCH

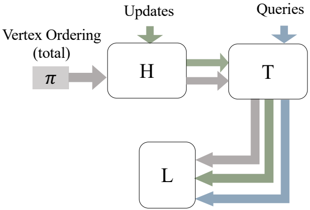


IncH2H

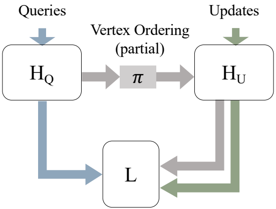
High-level Overview



DCH

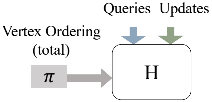


IncH2H

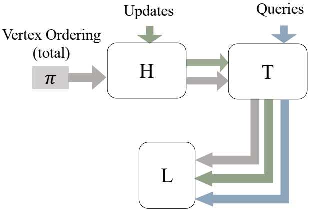


DHL (ours)

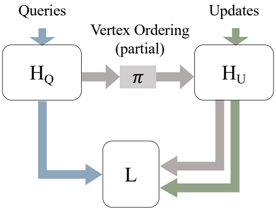
High-level Overview



DCH



IncH2H



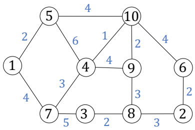
DHL (ours)

Dataset	Method	Update Time [ms]		Query Time [μs]
		Increase	Decrease	
USA	DCH	0.84	0.27	**2,915.91**
	IncH2H	**356.27**	**239.84**	3.43
	DHL	73.59	49.29	0.83
EUR	DCH	0.73	0.26	**5,440.48**
	IncH2H	**96.63**	**66.97**	3.89
	DHL	28.83	17.03	1.19

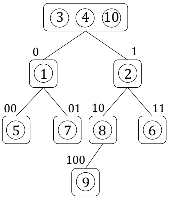
Performance Comparison

Our Solution - Dual-Hierarchy Labelling

- **Query Hierarchy** H_Q:
  - static, a balanced binary tree
  - defines vertex partial order
    $\preceq := \{3 < 4 < 10 < \ldots, 1 < 5, 1 < 7, 2 < 6, 2 < 8 < 9\}$



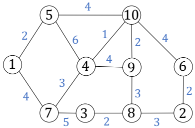
G



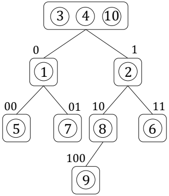
H_Q

Our Solution - Dual-Hierarchy Labelling

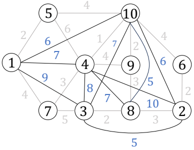
- **Query Hierarchy** H_Q:
  - static, a balanced binary tree
  - defines vertex partial order
    $\preceq := \{3 < 4 < 10 < \ldots, 1 < 5, 1 < 7, 2 < 6, 2 < 8 < 9\}$
- **Update Hierarchy** H_U:
  - dynamic, a shortcut graph constructed over G and $\preceq$
  - contains a shortcut (v, w) for every *valley path* between v and w



G



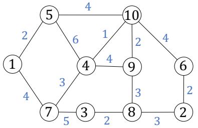
H_Q



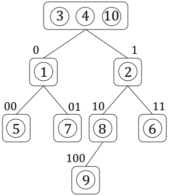
H_U

Our Solution - Dual-Hierarchy Labelling

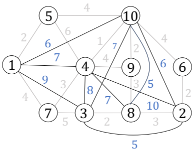
- **Query Hierarchy** H_Q:
  - static, a balanced binary tree
  - defines vertex partial order
    $\preceq := \{3 < 4 < 10 < \ldots, 1 < 5, 1 < 7, 2 < 6, 2 < 8 < 9\}$
- **Update Hierarchy** H_U:
  - dynamic, a shortcut graph constructed over G and $\preceq$
  - contains a shortcut (v, w) for every *valley path* between v and w
- **Hierarchical Labelling** L:
  - each vertex v stores distances to all ancestors in H_Q



G



H_Q



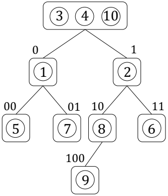
H_U

Label	Distance
L(1)	[9, 7, 6], [0]
L(2)	[5, 7, 6], [0]
L(3)	[0]
L(4)	[8, 0]
L(5)	[11, 5, 4], [2], [0]
L(6)	[7, 5, 4], [2], [0]
L(7)	[5, 3, 10], [4], [0]
L(8)	[2, 6, 5], [3], [0]
L(9)	[5, 3, 2], [6], [3], [0]
L(10)	[7, 1, 0]

L

Our Solution - Dual-Hierarchy Labelling

- **Distance Queries:**



H_Q

Label	Distance
L(1)	[9, 7, 6], [0]
L(2)	[5, 7, 6], [0]
L(3)	[0]
L(4)	[8, 0]
L(5)	[11, 5, 4], [2], [0]
L(6)	[7, 5, 4], [2], [0]
L(7)	[5, 3, 10], [4], [0]
L(8)	[2, 6, 5], [3], [0]
L(9)	[5, 3, 2], [6], [3], [0]
L(10)	[7, 1, 0]

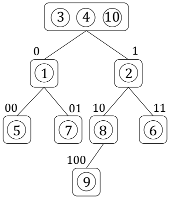
L

(a) A 2-hop query with common ancestors as hops

(b) Find LCA using bitstrings

Our Solution - Dual-Hierarchy Labelling

- **Distance Queries:**



H_Q

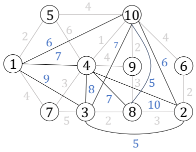
Label	Distance
L(1)	[9, 7, 6], [0]
L(2)	[5, 7, 6], [0]
L(3)	[0]
L(4)	[8, 0]
L(5)	[11, 5, 4], [2], [0]
L(6)	[7, 5, 4], [2], [0]
L(7)	[5, 3, 10], [4], [0]
L(8)	[2, 6, 5], [3], [0]
L(9)	[5, 3, 2], [6], [3], [0]
L(10)	[7, 1, 0]

L

(a) A 2-hop query with common ancestors as hops

(b) Find LCA using bitstrings

- **Weight Updates:**



H_U

Label	Distance
L(1)	[9, 7, 6], [0]
L(2)	[5, 7, 6], [0]
L(3)	[0]
L(4)	[8, 0]
L(5)	[11, 5, 4], [2], [0]
L(6)	[7, 5, 4], [2], [0]
L(7)	[5, 3, 10], [4], [0]
L(8)	[2, 6, 5], [3], [0]
L(9)	[5, 3, 2], [6], [3], [0]
L(10)	[7, 1, 0]

L

(a) Find all affected shortcuts in H_U and fix their weights

(b) Update hierarchical labelling L using affected shortcuts in H_U

Our Solution - Dual-Hierarchy Labelling

H_U **Update.**

- Neighbors in H_U are anc. or desc. in H_Q

$$up(v) = \{u \mid (v,u) \in E(H_U) \wedge u \preceq v\}$$
$$down(v) = \{u \mid (v,u) \in E(H_U) \wedge v \preceq u\}$$

- Changes propagate upwards: $\omega(v,w)$ may change if $\omega(v,x) + \omega(w,x)$ changed for some $x \in down(v) \cap down(w)$.

Our Solution - Dual-Hierarchy Labelling

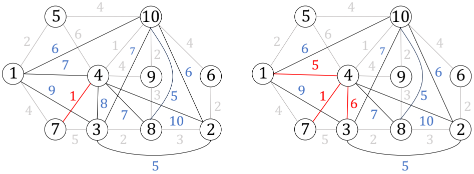
H_U **Update.**

- Neighbors in H_U are anc. or desc. in H_Q

$$up(v) = \{u \mid (v,u) \in E(H_U) \wedge u \preceq v\}$$
$$down(v) = \{u \mid (v,u) \in E(H_U) \wedge v \preceq u\}$$

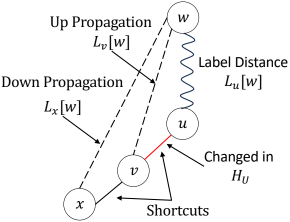
- Changes propagate upwards: $\omega(v,w)$ may change if $\omega(v,x) + \omega(w,x)$ changed for some $x \in down(v) \cap down(w)$.

Weight Decrease:



Update H_U

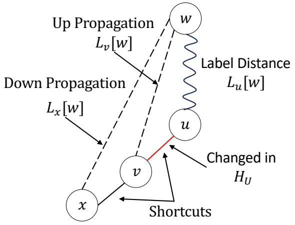
L Update.



- Distances stored in L are lengths of *upward* paths in H_U (= distances in subgraphs)
- $L_v[w] = \min_{u \in up(v)} \omega(v, u) + L_u[w]$

Our Solution - Dual-Hierarchy Labelling

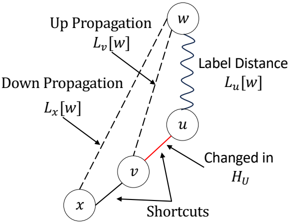
L **Update.**



- Distances stored in L are lengths of *upward* paths in H_U (= distances in subgraphs)
- $L_v[w] = \min_{u \in up(v)} \omega(v, u) + L_u[w]$
- Upward propagation: $\omega(v, u)$ change may affect $L_v[w]$ for $w \preceq u$
- Downward propagation: $L_v[w]$ change may affect $L_x[w]$ for $x \in down(v)$

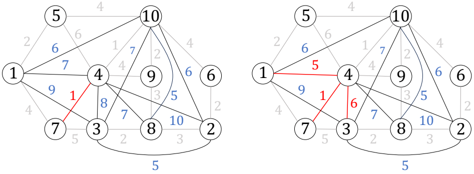
Our Solution - Dual-Hierarchy Labelling

L Update.



- Distances stored in L are lengths of *upward* paths in H_U (= distances in subgraphs)
- $L_v[w] = \min_{u \in up(v)} \ \omega(v, u) + L_u[w]$
- Upward propagation: $\omega(v, u)$ change may affect $L_v[w]$ for $w \preceq u$
- Downward propagation: $L_v[w]$ change may affect $L_x[w]$ for $x \in down(v)$

Weight Decrease:



Update H_U

Label	Distance
L(1)	[9, 5, 6], [0]
L(2)	[5, 7, 6], [0]
L(3)	[0]
L(4)	[6, 0]
L(5)	[11, 5, 4], [2], [0]
L(6)	[7, 5, 4], [2], [0]
L(7)	[5, 1, 10], [4], [0]
L(8)	[2, 6, 5], [3], [0]
L(9)	[5, 3, 2], [6], [3], [0]
L(10)	[7, 1, 0]

Update L

Empirical Evaluation

Network	Update Time - Increase [ms]				Update Time - Decrease [ms]			
	DHL_p^+	$IncH2H_p^+$	DHL^+	$IncH2H^+$	DHL_p^-	$IncH2H_p^-$	DHL^-	$IncH2H^-$
NY	0.209	0.234	0.790	2.900	0.116	0.187	0.522	2.006
BAY	0.153	0.178	0.543	2.498	0.103	0.134	0.394	1.769
COL	0.257	0.318	0.933	4.613	0.179	0.241	0.696	3.306
FLA	0.311	0.390	1.906	4.981	0.216	0.320	1.368	3.585
CAL	0.786	1.185	5.079	20.20	0.539	0.855	3.614	13.89
E	1.913	2.481	12.20	43.57	1.314	1.820	8.197	29.33
W	2.420	3.841	18.11	68.99	1.757	2.772	12.69	47.76
CTR	8.721	15.13	58.72	309.7	5.570	10.75	38.48	213.1
USA	9.321	18.20	73.59	356.3	6.004	13.06	49.29	239.8
EUR	5.634	8.283	26.83	96.63	3.273	6.969	17.03	66.97

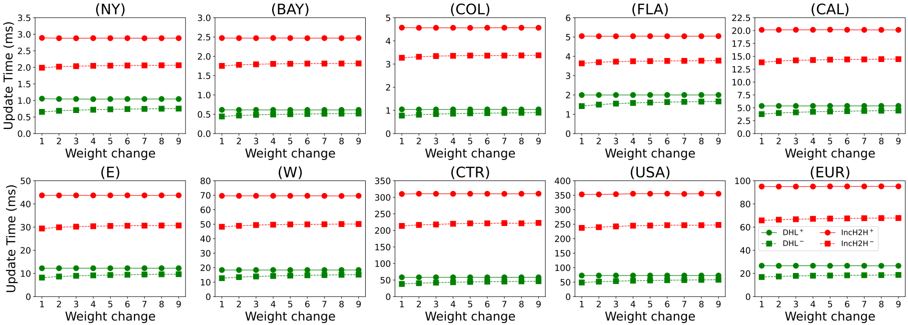
- *3-4 times faster in terms of update time compared to SOTA IncH2H*

Empirical Evaluation

Network	Query Time [μs]		Labelling Size		Shortcuts Size		Const. Time [s]	
	DHL	IncH2H	DHL	IncH2H	DHL	IncH2H	DHL	IncH2H
NY	0.287	0.913	130 MB	826 MB	15 MB	42 MB	2	4
BAY	0.299	0.841	105 MB	797 MB	12 MB	40 MB	2	3
COL	0.349	1.018	176 MB	1.35 GB	15 MB	51 MB	4	5
FLA	0.396	1.019	425 MB	2.38 GB	40 MB	129 MB	10	11
CAL	0.490	1.333	1.03 GB	8.12 GB	73 MB	233 MB	25	30
E	0.630	1.683	2.92 GB	20.5 GB	136 MB	444 MB	64	74
W	0.664	1.702	4.83 GB	36.0 GB	231 MB	758 MB	107	126
CTR	0.812	2.483	19.7 GB	177 GB	558 MB	1.77 GB	455	858
USA	0.834	3.428	35.6 GB	307 GB	931 MB	2.97 GB	710	1,081
EUR	1.185	3.888	36.4 GB	320 GB	733 MB	2.38 GB	907	1,254

- *2-4 times faster in terms of query time*
- *consuming only 10%-20% labelling space*

Empirical Evaluation



- *How does our solution perform for updating labellings with varying weights?*

Concluding Remarks

- Both query and update hierarchies are built upon existing ideas

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

Concluding Remarks

- Both query and update hierarchies are built upon existing ideas

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

- Both hierarchies are simple in structure

  Simple is often better: efficient to process both queries and updates.

Concluding Remarks

- Both query and update hierarchies are built upon existing ideas

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

- Both hierarchies are simple in structure

  Simple is often better: efficient to process both queries and updates.

- Our solution is scalable on large and dynamic road networks

  The power lies in "query hierarchy".

Questions

Thank You