Scaling Distance Labeling on Small-World Networks

Wentao Li[1], Miao Qiao[2], Lu Qin[1], Ying Zhang[1], Lijun Chang[3], Xuemin Lin[4]

[1]University of Technology Sydney

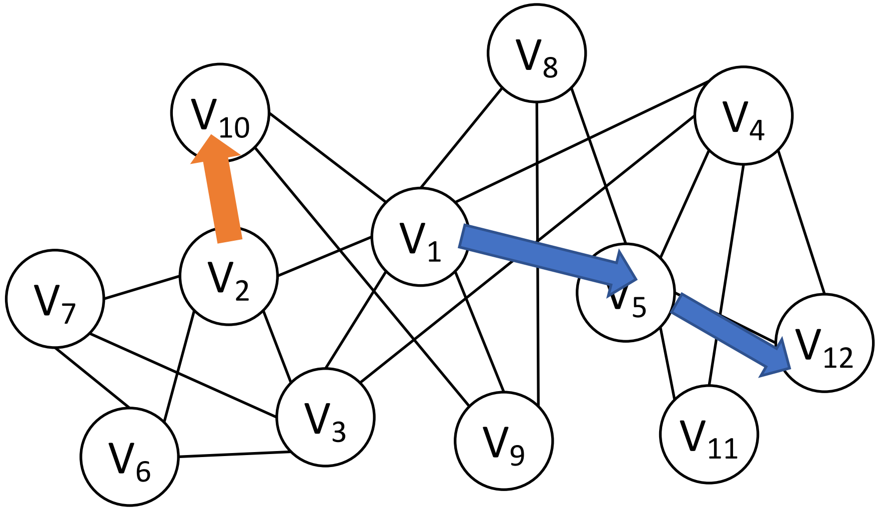
[2]University of Auckland

[3]The University of Sydney

[4]The University of New South

Problem

- ***shortest distance query*** Q(s,t) for any node pair s, t



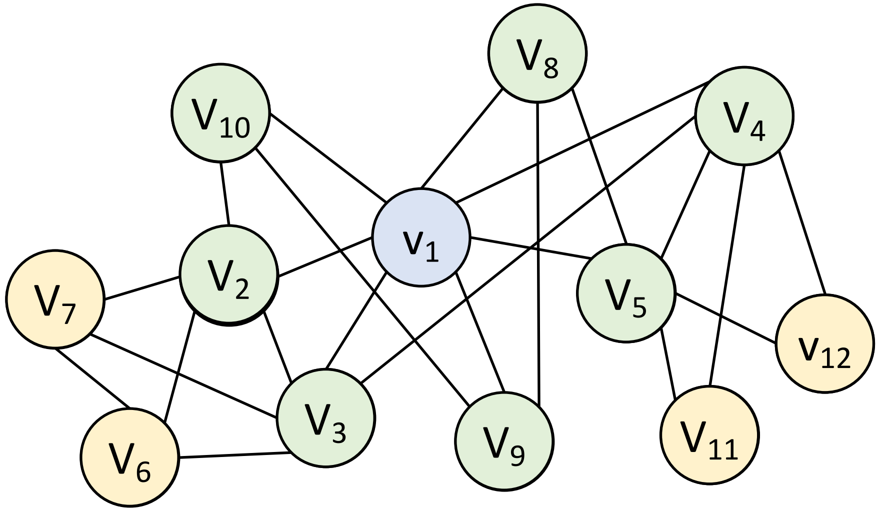
$Q(v_2,v_{10})=dist(v_2,v_{10})=1$

$Q(v_1,v_{12})=dist(v_1,v_{12})=2$

Solutions

- Answer the ***shortest distance query*** Q(s,t)
  - Online search (breadth-first search)
  - Offline index (2-hop labeling)

$s=v_1$, $t=v_{12}$



2-hop Labeling: What

- Index
  - Each node v has a label set L(v)
    - we will tell how to determine L(v) in the sequel

distance between v_2 and v_1 is 1

distance between v_2 and v_2 is 0

$L(v_2)$	$(v_1,1)$	$(v_2,0)$		
$L(v_{10})$	$(v_1,1)$	$(v_2,1)$	$(v_9,1)$	$(v_{10},0)$

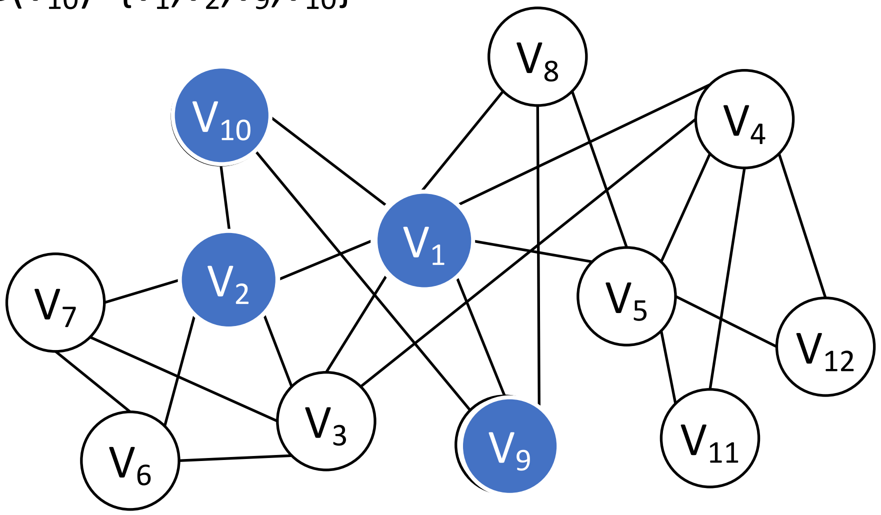
hub nodes of v_2: $C(v_2)=\{v_1,v_2\}$

hub nodes of v_{10}: $C(v_{10})=\{v_1,v_2,v_9,v_{10}\}$

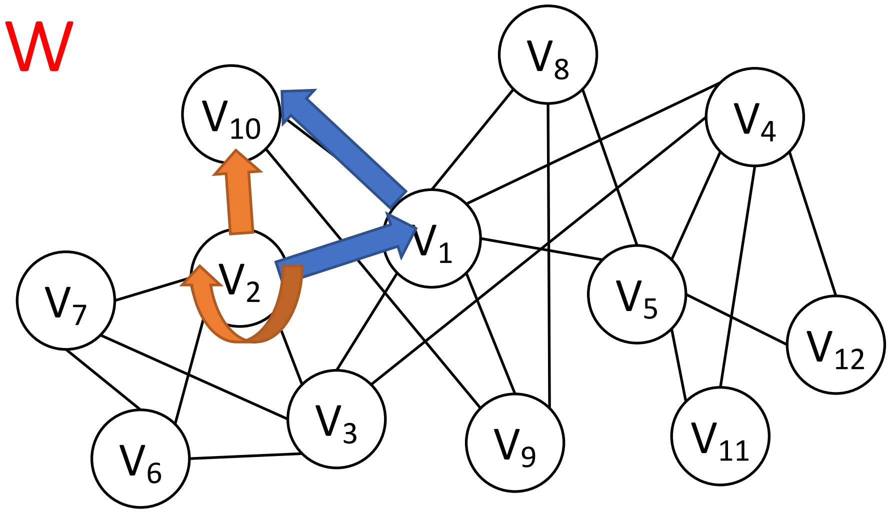
  - Label size $|L(v)|$ for each node v

$|L(v_2)|=2$

$|L(v_{10})|=4$



2-hop Labeling: How



– Distance Query Q(s,t,L)

$s=v_2, t=v_{10}$ Q(s,t,L) = 1

L(v_2)	(v_1,1)	(v_2,0)		
L(v_{10})	(v_1,1)	(v_2,1)	(v_9,1)	(v_{10},0)

$d(v_2,v_1)+d(v_1,v_{10})$:1+1=2

$d(v_2,v_2)+d(v_2,v_{10})$:0+1=1

common hub nodes of v_2 and v_{10} is $\{v_1, v_2\}$

the cost is |L(s)|+|L(t)|

– Why the distance query Q(s,t) is correct?
  - at least one common node of L(s) and L(t) must on the s-t path
  - dist(s,w)+dist(w,t) reports the correct distance (the path s-w-t)

2-hop Labeling Algorithms

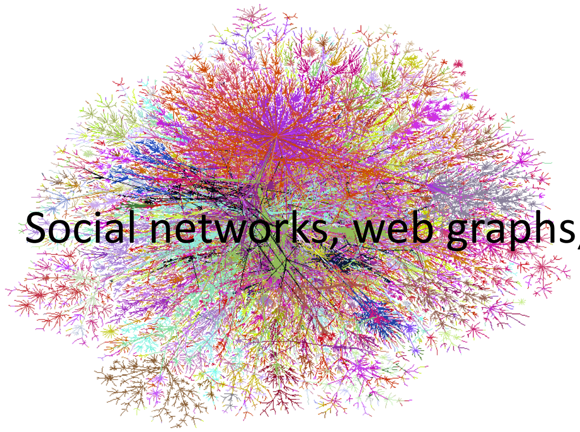
- How to build the index
  - 2-hop algorithms

Prune Landmark Labeling

Road Network

the planarity and hierarchical ***structure***: small label size

Small-world Network

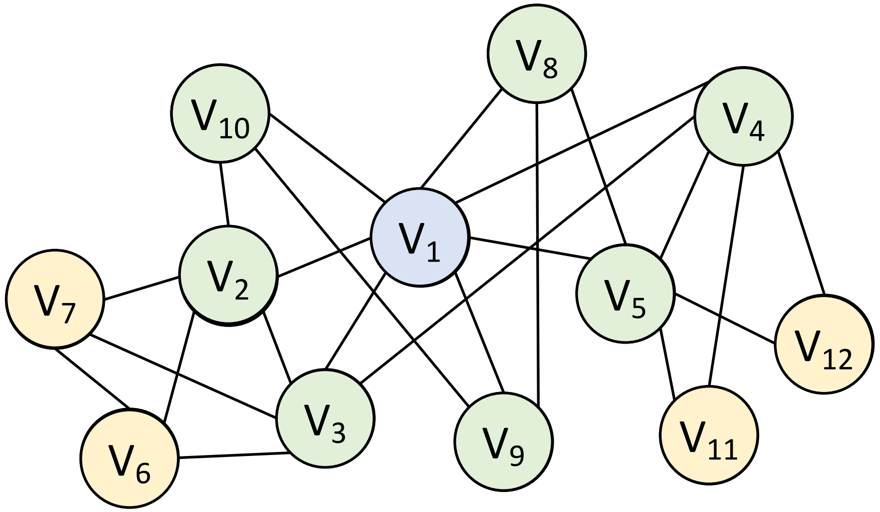


Social networks, web graphs, …

Label size would be large

Prune Landmark Labeling

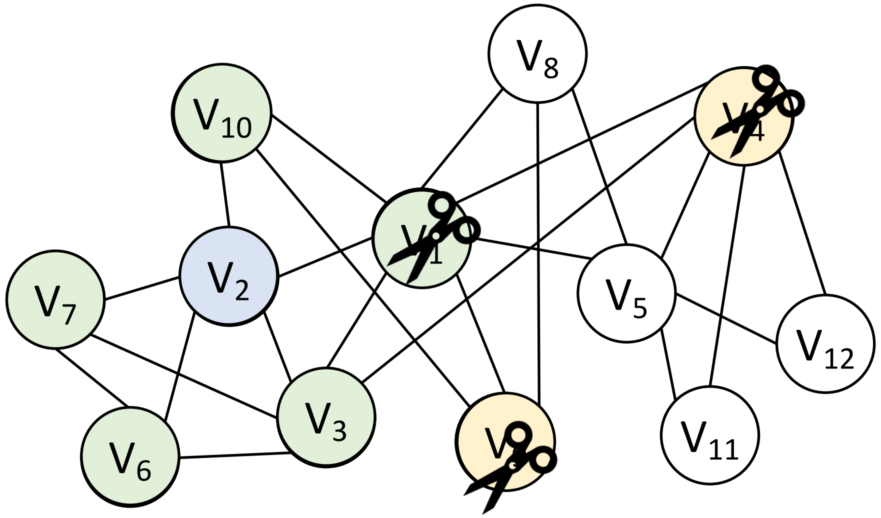
- Predefine a total order on all nodes
  - $r(v_1)>r(v_2)>r(v_3)\ldots>r(v_{12})$ by degrees
- Perform pruned BFS from v in the order
  - when scanning w, add (v,dist(v,w)) to L(w)



	v_1
$L(v_1)$	$(v_1,0)$
$L(v_2)$	$(v_1,1)$
$L(v_3)$	$(v_1,1)$
$L(v_4)$	$(v_1,1)$
$L(v_5)$	$(v_1,1)$
$L(v_6)$	$(v_1,2)$
$L(v_7)$	$(v_1,2)$
$L(v_8)$	$(v_1,1)$
$L(v_9)$	$(v_1,1)$
$L(v_{10})$	$(v_1,1)$
$L(v_{11})$	$(v_1,2)$
$L(v_{12})$	$(v_1,2)$

- Perform pruned BFS from v in the order
  - when scanning w

    if Q(v,w,L) > dist(v,w) add (v,dist(v,w)) to L(w); otherwise stopping scanning w



Partial Labels

0+1=1

$\leq d(v_2,v_1)$

1+1=2

$> d(v_2,v_3)$

	v_1	v_2
$L(v_1)$	$(v_1,0)$	—
$L(v_2)$	$(v_1,1)$	$(v_2,0)$
$L(v_3)$	$(v_1,1)$	$(v_2,1)$
$L(v_4)$	$(v_1,1)$	—
$L(v_5)$	$(v_1,1)$	
$L(v_6)$	$(v_1,2)$	$(v_2,1)$
$L(v_7)$	$(v_1,2)$	$(v_2,1)$
$L(v_8)$	$(v_1,1)$	
$L(v_9)$	$(v_1,1)$	—
$L(v_{10})$	$(v_1,1)$	$(v_2,1)$
$L(v_{11})$	$(v_1,2)$	
$L(v_{12})$	$(v_1,2)$	

Summary of PLL

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
L(v_1)	(v_1,0)											
L(v_2)	(v_1,1)	(v_2,0)										
L(v_3)	(v_1,1)	(v_2,1)	(v_3,0)									
L(v_4)	(v_1,1)		(v_3,1)	(v_4,0)								
L(v_5)	(v_1,1)			(v_4,1)	(v_5,0)							
L(v_6)	(v_1,2)	(v_2,1)	(v_3,1)			(v_6,0)						
L(v_7)	(v_1,2)	(v_2,1)	(v_3,1)			(v_6,1)	(v_7,0)					
L(v_8)	(v_1,1)				(v_5,1)			(v_8,0)				
L(v_9)	(v_1,1)							(v_8,1)	(v_9,0)			
L(v_{10})	(v_1,1)	(v_2,1)							(v_9,1)	(v_{10},0)		
L(v_{11})	(v_1,2)		(v_3,2)	(v_4,1)	(v_5,1)						(v_{11},0)	
L(v_{12})	(v_1,2)		(v_3,2)	(v_4,1)	(v_5,1)							(v_{12},0)

Pros.

PLL exploits previous labels to ***avoid roducing redundant labels***

Once a node order is predefined, the resulting labels are ***minimal***

Cons.

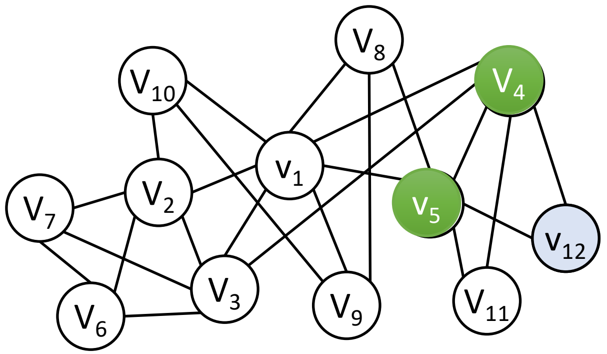
“previous labels” means: depends on the node order (hard to be parallelized strong sequential nature: ***index time***)

Index size is massive for large graphs

Index Time Reduction

Parallelized Shortest Distance Labeling

Reorganize PLL



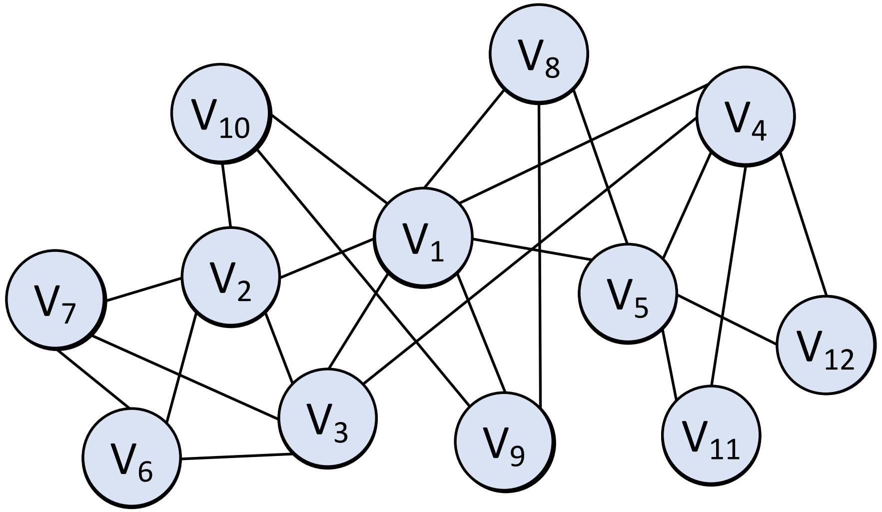
Insights: ***labels with distance from d to v*** can be derived from ***labels with distance d-1 to a node in N(v)*** (superset)

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
$L(v_1)$	$(v_1,0)$											
$L(v_2)$	$(v_1,1)$	$(v_2,0)$										
$L(v_3)$	$(v_1,1)$	$(v_2,1)$	$(v_3,0)$									
$L(v_4)$	$(v_1,1)$		$(v_3,1)$	$(v_4,0)$								
$L(v_5)$	$(v_1,1)$			$(v_4,1)$	$(v_5,0)$							
$L(v_6)$	$(v_1,2)$	$(v_2,1)$	$(v_3,1)$			$(v_6,0)$						
$L(v_7)$	$(v_1,2)$	$(v_2,1)$	$(v_3,1)$			$(v_6,1)$	$(v_7,0)$					
$L(v_8)$	$(v_1,1)$				$(v_5,1)$			$(v_8,0)$				
$L(v_9)$	$(v_1,1)$							$(v_8,1)$	$(v_9,0)$			
$L(v_{10})$	$(v_1,1)$	$(v_2,1)$							$(v_9,1)$	$(v_{10},0)$		
$L(v_{11})$	$(v_1,2)$		$(v_3,2)$	$(v_4,1)$	$(v_5,1)$						$(v_{11},0)$	
$L(v_{12})$	$(v_1,2)$		$(v_3,2)$	$(v_4,1)$	$(v_5,1)$							$(v_{12},0)$

d=0	d=1	d=2
$(v_1,0)$		
$(v_2,0)$	$(v_1,1)$	
$(v_3,0)$	$(v_1,1)$ $(v_2,1)$	
$(v_4,0)$	$(v_1,1)$ $(v_3,1)$	
$(v_5,0)$	$(v_1,1)$ $(v_4,1)$	
$(v_6,0)$	$(v_2,1)$ $(v_3,1)$	$(v_1,2)$
$(v_7,0)$	$(v_2,1)$ $(v_3,1)$ $(v_6,1)$	$(v_1,2)$
$(v_8,0)$	$(v_1,1)$ $(v_5,1)$	
$(v_9,0)$	$(v_1,1)$ $(v_8,1)$	
$(v_{10},0)$	$(v_1,1)$ $(v_2,1)$ $(v_9,1)$	
$(v_{11},0)$	$(v_4,1)$ $(v_5,1)$	$(v_1,2)$ $(v_3,2)$
$(v_{12},0)$	$(v_4,1)$ $(v_5,1)$	$(v_1,2)$ $(v_3,2)$

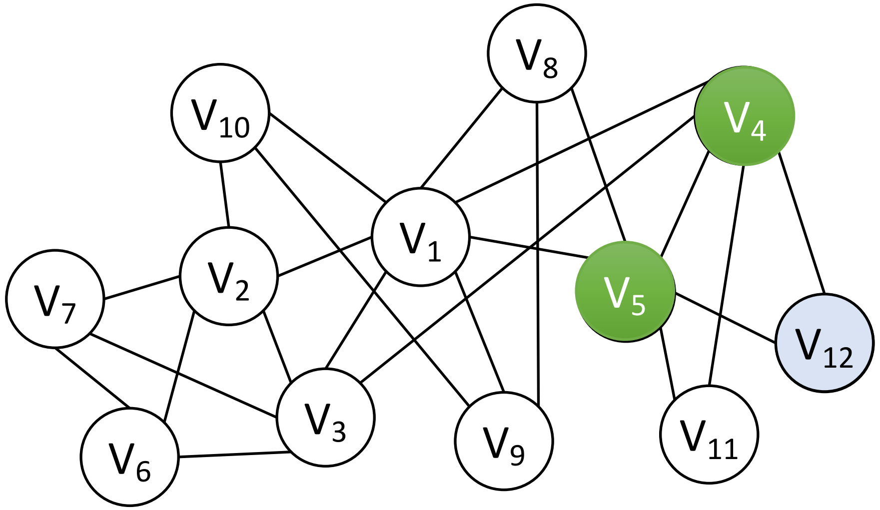
Parallel Shortest Distance Labeling (PSL)

- Initialize with d = 0
  - insert (v,0) to L(v) for all v ***concurrently***



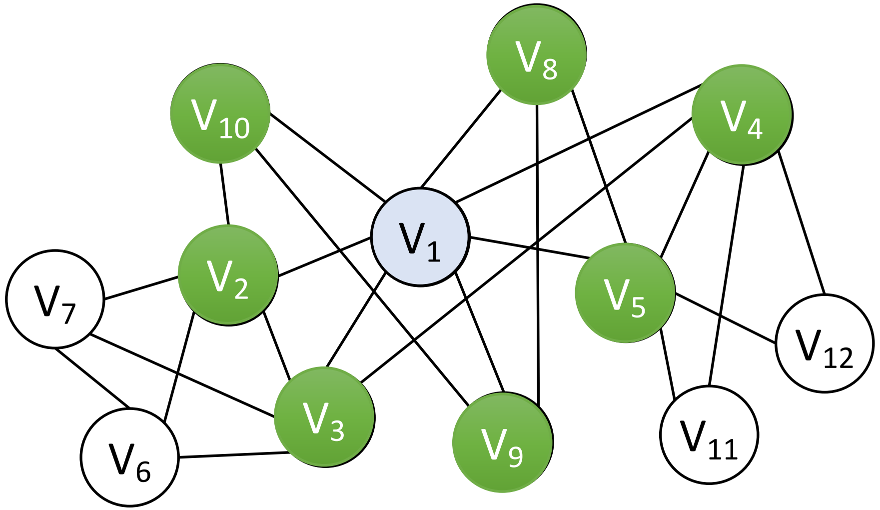
	d=0
L(v_1)	(v_1,0)
L(v_2)	(v_2,0)
L(v_3)	(v_3,0)
L(v_4)	(v_4,0)
L(v_5)	(v_5,0)
L(v_6)	(v_6,0)
L(v_7)	(v_7,0)
L(v_8)	(v_8,0)
L(v_9)	(v_9,0)
L(v_{10})	(v_{10},0)
L(v_{11})	(v_{11},0)
L(v_{12})	(v_{12},0)

- Initial with d = 0
  - insert (v,0) to L(v) for all v concurrently
- While there is a newly formed label
  - increase d by one
  - for each node v ***concurrently***
    - gather (d-1)-hubs w in N(v) as d-hubs of v



	d=0	d=1
L(v_1)	(v_1,0)	
L(v_2)	(v_2,0)	
L(v_3)	(v_3,0)	
L(v_4)	(v_4,0)	
L(v_5)	(v_5,0)	
L(v_6)	(v_6,0)	
L(v_7)	(v_7,0)	
L(v_8)	(v_8,0)	
L(v_9)	(v_9,0)	
L(v_{10})	(v_{10},0)	
L(v_{11})	(v_{11},0)	
L(v_{12})	(v_{12},0)	(v_4,1) (v_5,1)

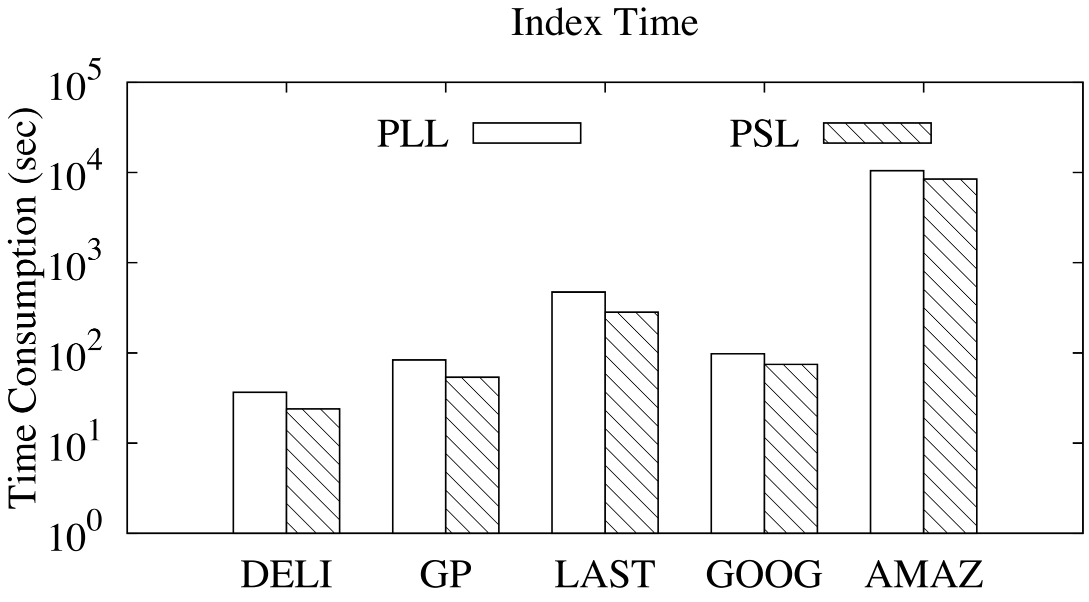
– Initialize with d = 0
  - insert (v,0) to L(v) for all v concurrently
– While there is a newly formed label
  - increase d by one
  - for each node v ***concurrently***
    - gather (d-1)-hubs w in N(v) as d-hubs of v
    - prune w if w is redundant



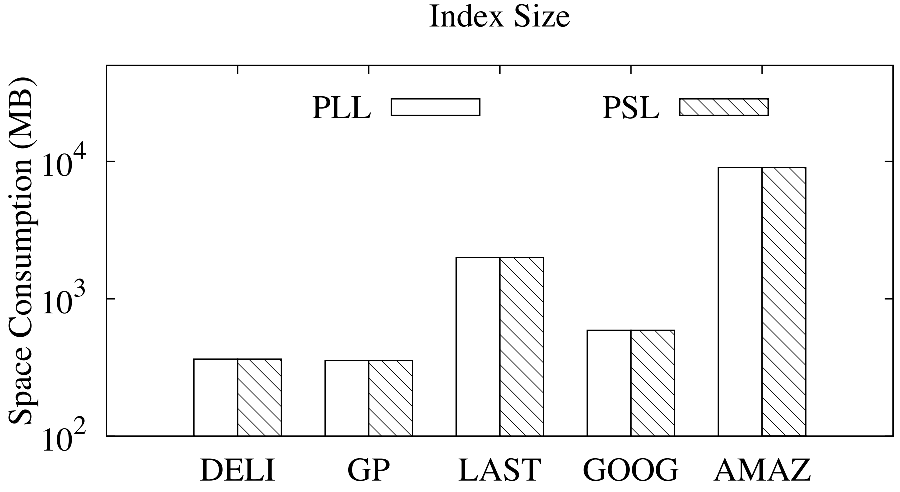
	d=0	d=1
L(v_1)	(v_1,0)	~~(v_2,1) (v_3,1)(v_4,1) (v_5,1) (v_8,1) (v_9,1) (v_{10},1)~~
L(v_2)	(v_2,0)	(v_1,1)
L(v_3)	(v_3,0)	(v_1,1) (v_2,1)
L(v_4)	(v_4,0)	(v_1,1) (v_3,1)
L(v_5)	(v_5,0)	(v_1,1) (v_4,1)
L(v_6)	(v_6,0)	(v_2,1) (v_3,1)
L(v_7)	(v_7,0)	(v_2,1) (v_3,1) (v_6,1)
L(v_8)	(v_8,0)	(v_1,1) (v_5,1)
L(v_9)	(v_9,0)	(v_1,1) (v_8,1)
L(v_{10})	(v_{10},0)	(v_1,1) (v_2,1) (v_9,1)
L(v_{11})	(v_{11},0)	(v_4,1) (v_5,1)
L(v_{12})	(v_{12},0)	(v_4,1) (v_5,1)

PSL achieves the identical index with PLL

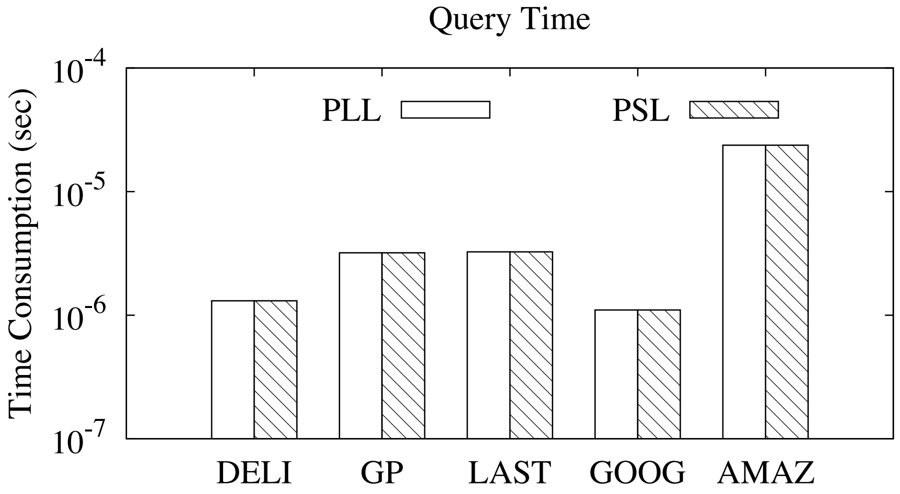
Exp 1: PSL vs PLL on One Core



PSL has an index time comparable to PLL

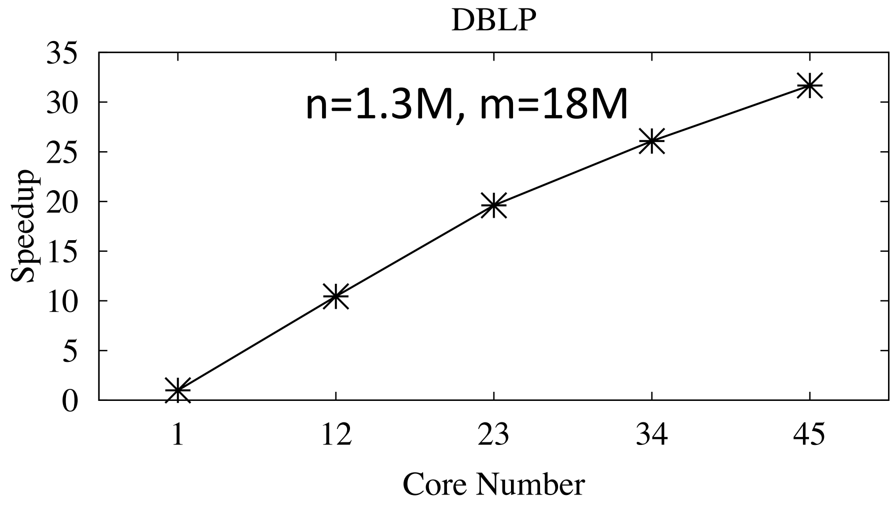


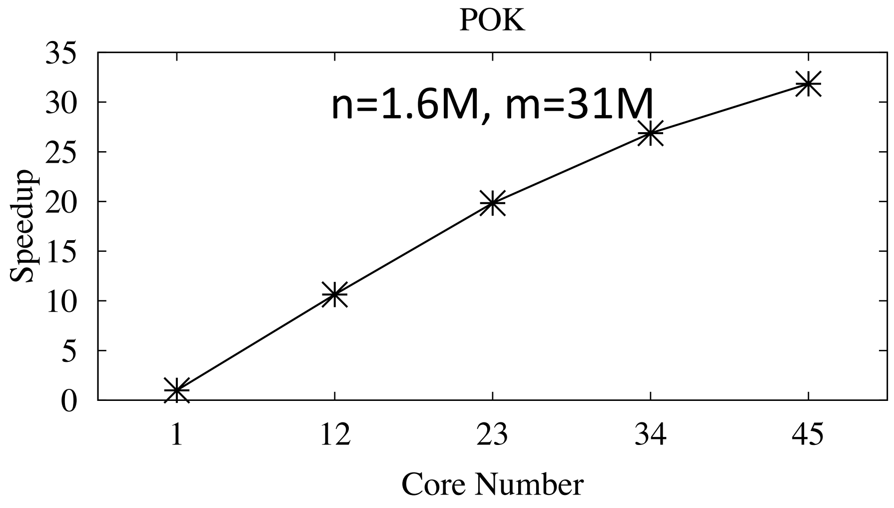
The label size of PLL and PSL is the same



The query time of PLL and PSL is the same

Exp 2: Near-Linear Speedup





Near-linear speedup of our algorithms in a multi-core environment

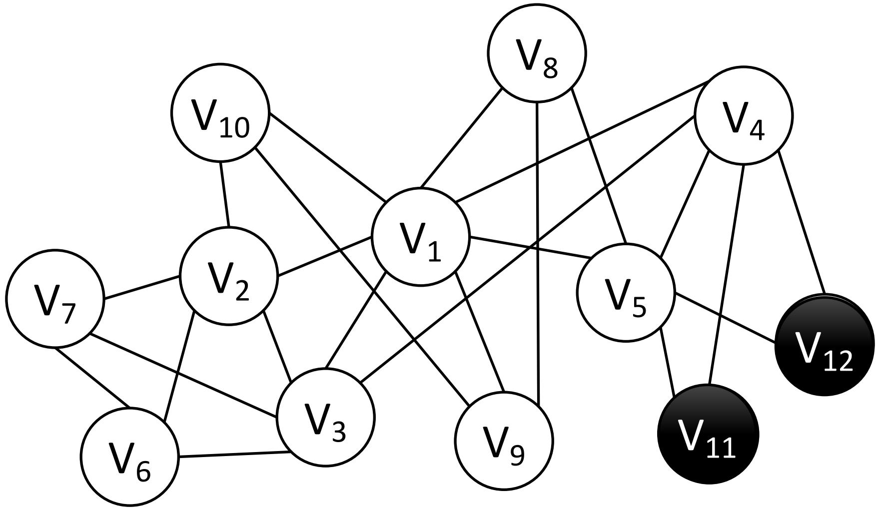
Index Size Reduction

Equivalence Relation Reduction

– Technique 1: two nodes with identical neighbor set have identical label structure

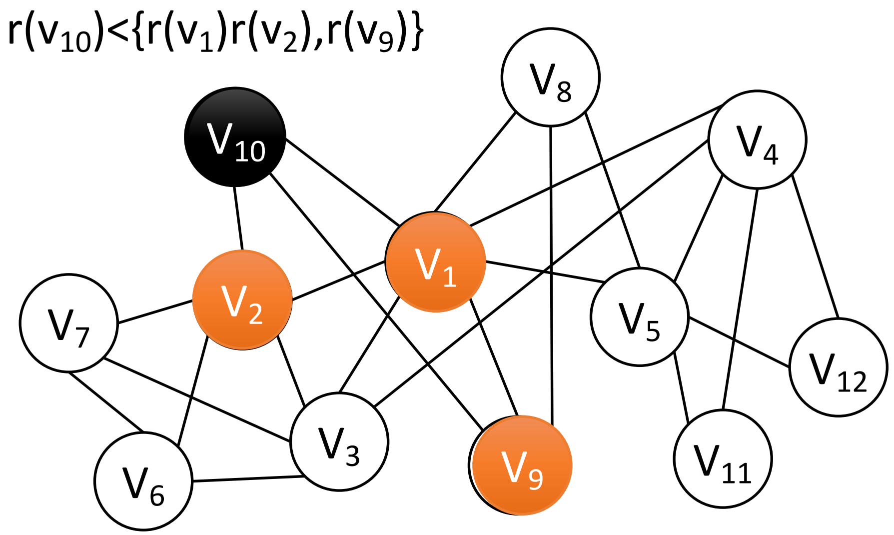
$L(v_{11})$	$(v_1,2)$	$(v_3,2)$	$(v_4,1)$	$(v_5,0)$	$(v_{11},0)$
$L(v_{12})$	$(v_1,1)$	$(v_2,1)$	$(v_9,1)$	$(v_{10},0)$	$(v_{12},0)$

the same structure



Equivalence Relation Reduction

- Technique 1: two nodes with identical neighbor set have basically identical label structure
- Technique 2: a node v with the lowest ranking among N(v) never appears in the labels of other nodes, and thus can be removed safely



Exp : Index Size

Index Size Reduction

	n=|V|	m=|E|	Original	T1	T1+T2
ABRA	22M	640M	146GB	60GB (41%)	35GB (24%)
SK	51M	1.9B	190GB	86GB (45%)	55GB (29%)

Summary

Speedup the index process

Reduce the index size