LIP6 25 septembre 2015

COMPACT ROUTING

Main results and techniques

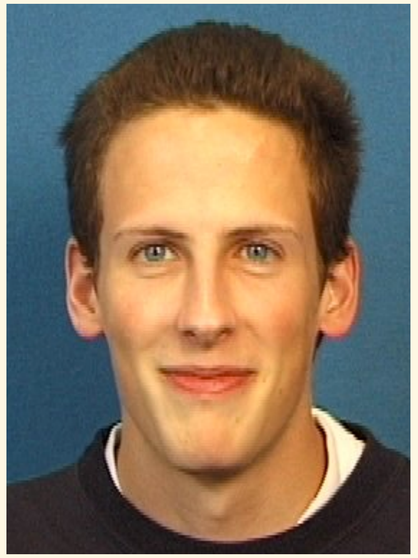
LaBRI - Université de Bordeaux - CNRS

CO-AUTHORS

Cyril Gavoille

Nicolas Hanusse

David Ilcinkas



Road map

- Models and results.
- How to build a compact routing scheme.
- Compact routing in internet-like graphs.

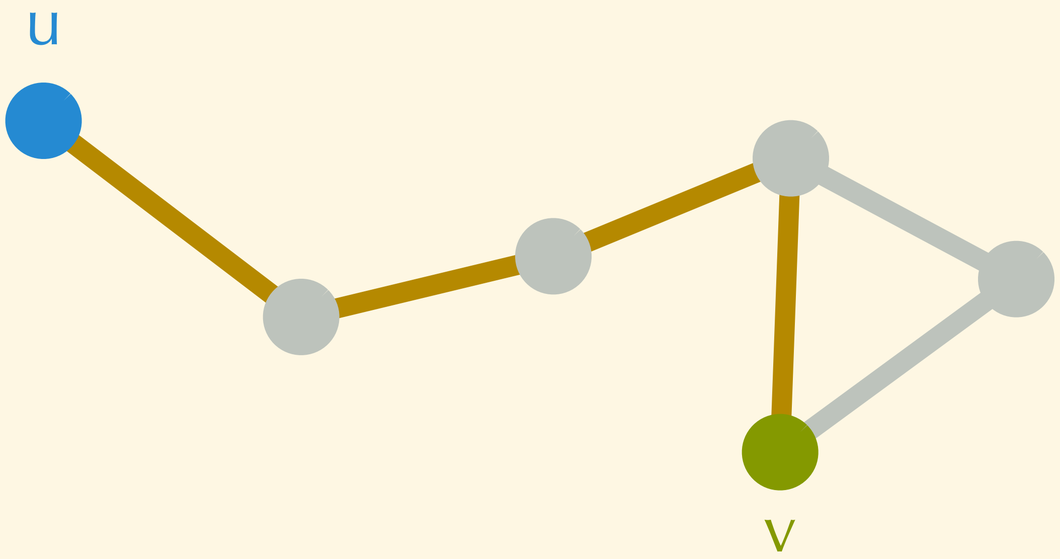
COMPACT ROUTING

What is compact routing, why does it exist ?

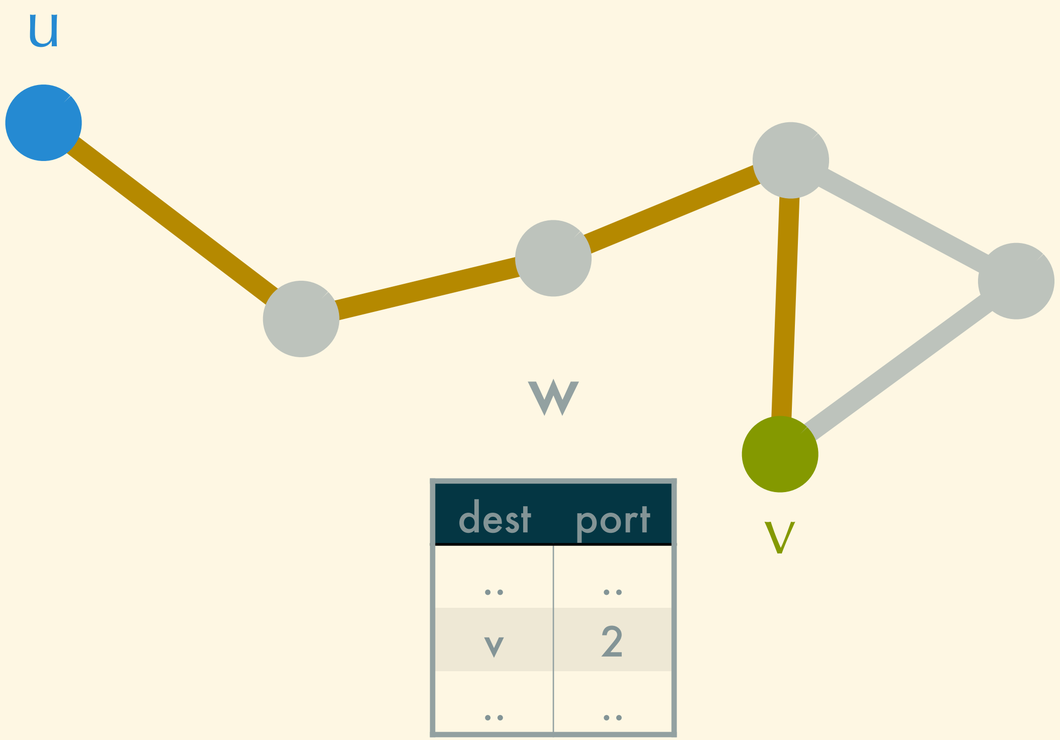
Motivations for compact routing

- Routing in large scale networks
  - having better scaling capabilities
  - forward packets faster
  - maintain routing tables up to date efficiently
- May help to save energy

Memory-Stretch Tradeoff



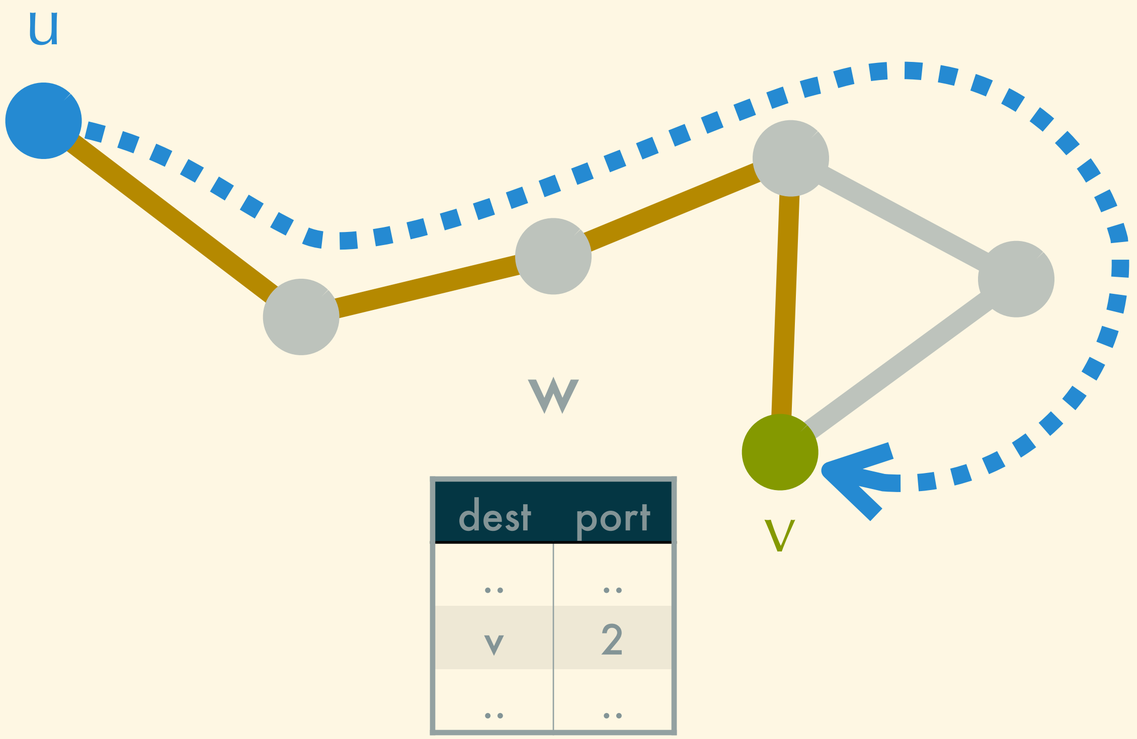
MEMORY-STRETCH TRADEOFF



MEMORY

entries stored per node

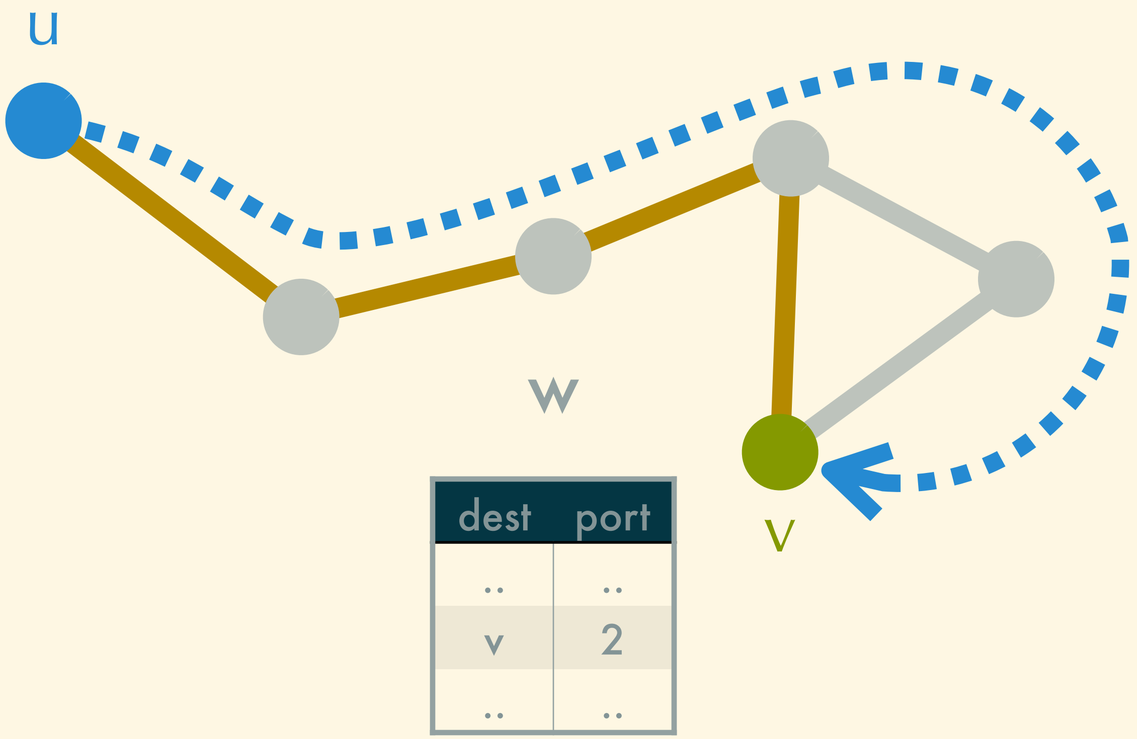
Memory-Stretch Tradeoff



Memory

entries stored per node

Memory-Stretch Tradeoff



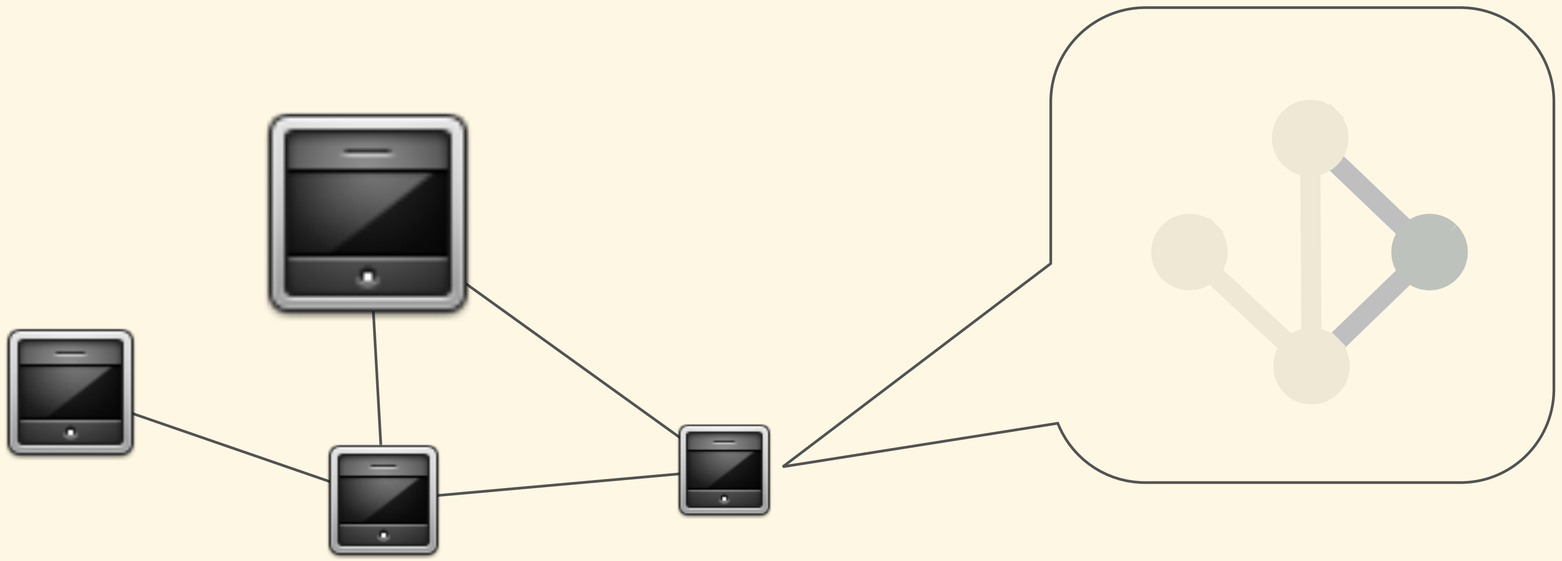
Route length (u,v) = 5
Distance (u,v) = 4

Stretch (u,v) = 5/4

Memory
entries stored per node

Stretch
maximal stretch over all pairs

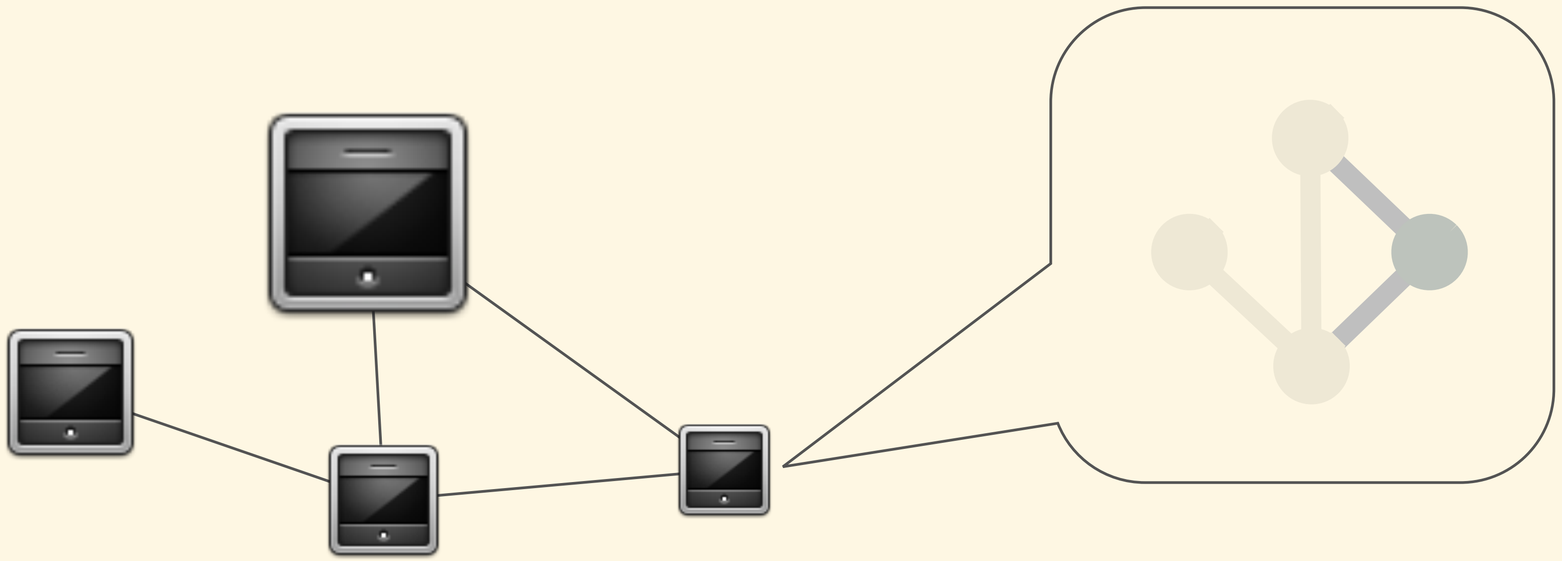
DISTRIBUTED ROUTING



Partial knowledge of
the graph (local)

⇒ need of communication
to compute routing tables

DISTRIBUTED ROUTING



Partial knowledge of
the graph (local)

⇒ need of communication
to compute routing tables

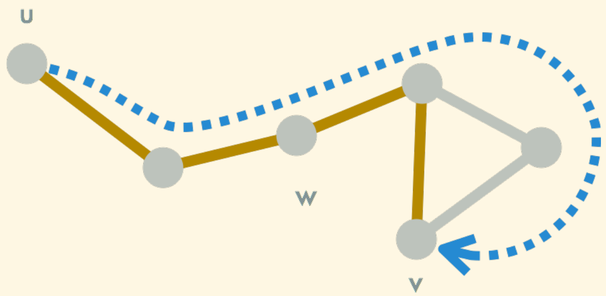
COMMUNICATION COST

of *small** messages exchanged

*polylogarithmic in the size of the network

Compact Routing

Stretch



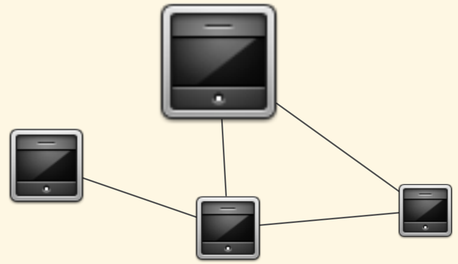
- Increased latency
- Increased energy consumption

Memory

dest	port
..	..
w	2
..	..

- Forwarding packets faster
- Saving energy

Communication cost



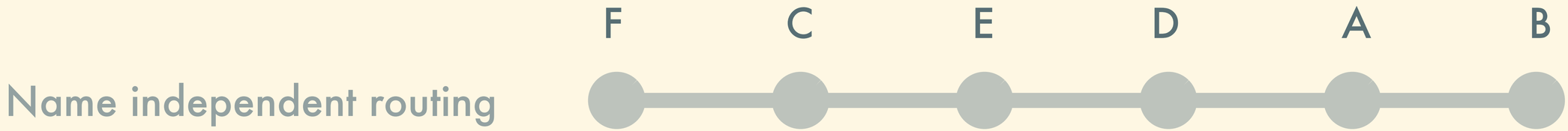
- Cost efficient maintenance
- Saving energy

Name-independent vs. Labeled

- Two models are considered for nodes naming (addresses)
  - Name-independent routing: use arbitrary routing addresses
  - Labeled routing: the designer chooses nodes' names

Name-independent vs. Labeled

The short answer



Name-independent vs. Labeled

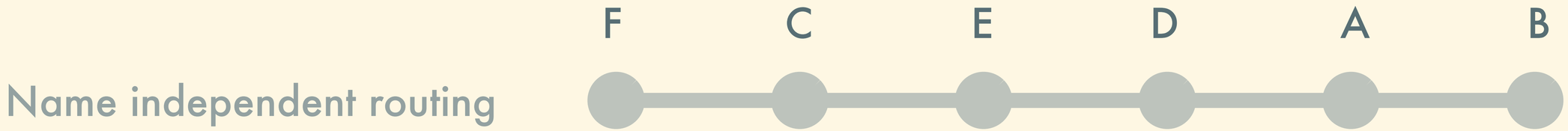
The short answer

Name independent routing

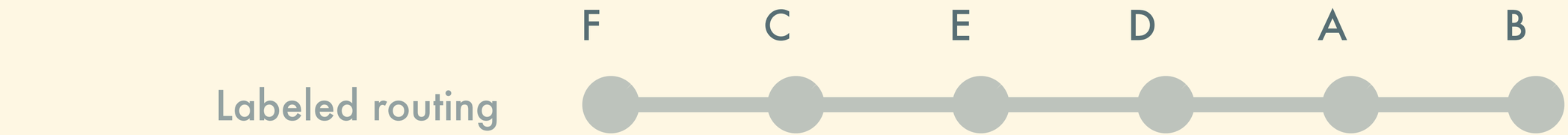
Query route to « A »

Name-independent vs. Labeled

The short answer

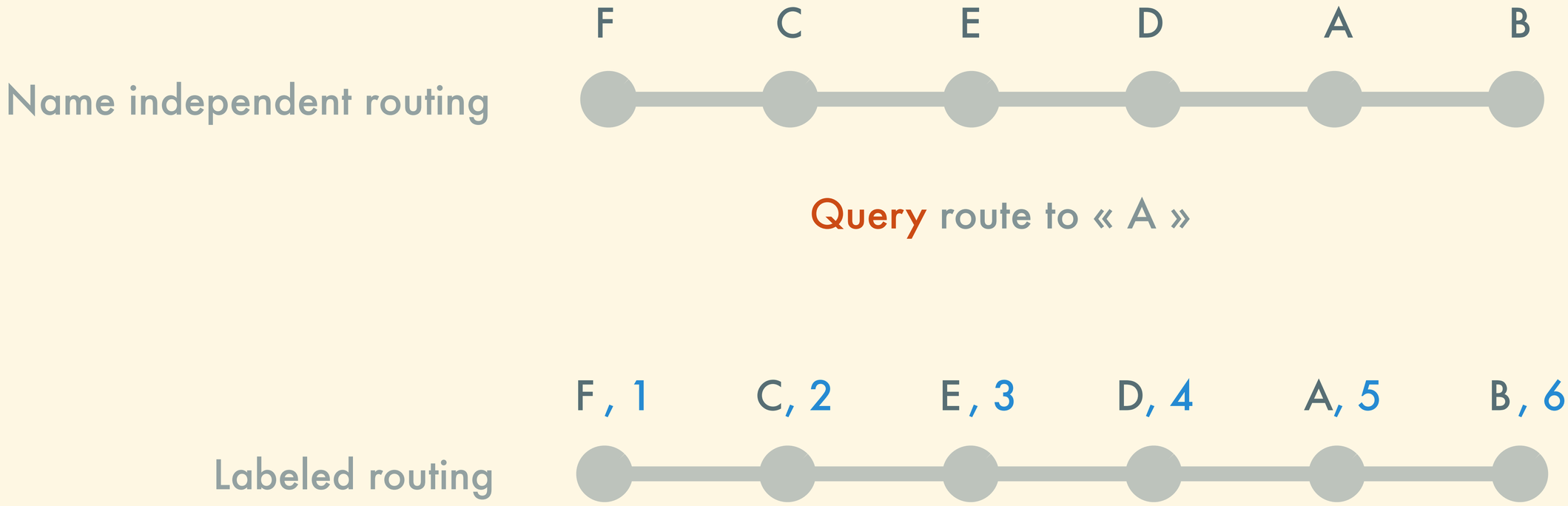


Query route to « A »



Name-independent vs. Labeled

The short answer



Name-independent vs. Labeled

The short answer

Name independent routing

Query route to « A »

Labeled routing

Query route to « A, 5 »

Tradeoffs (Name-independent)

lower bounds

	Stretch	Memory	Communication cost
[AGM'06a]	$< 2k+1$	$\Omega(n^{1/k})$	any

TRADEOFFS (NAME-INDEPENDENT)

lower bounds

	Stretch	Memory	Communication cost
[AGM'06a]	$< 2k+1$	$\Omega(n^{1/k})$	any
[GGHI'13]	1	any	$\Omega(n^2)$

TRADEOFFS (NAME-INDEPENDENT)

lower bounds

upper bounds

	Stretch	Memory	Communication cost
[AGM'06a]	$< 2k+1$	$\Omega(n^{1/k})$	any
[GGHI'13]	1	any	$\Omega(n^2)$
BFS-tree	1	$\tilde{O}(n)$	$\tilde{O}(nm)$

TRADEOFFS (NAME-INDEPENDENT)

	Stretch	Memory	Communication cost
lower bounds			
[AGM'06a]	$< 2k+1$ < 3	$\Omega(n^{1/k})$ $\Omega(n)$	any
[GGHI'13]	1	any	$\Omega(n^2)$
upper bounds			
BFS-tree	1 optimal	$\tilde{O}(n)$	$\tilde{O}(nm)$

TRADEOFFS (NAME-INDEPENDENT)

	Stretch	Memory	Communication cost
lower bounds			
[AGM'06a]	$< 2k+1$ < 3	$\Omega(n^{1/k})$ $\Omega(n)$	any
[GGHI'13]	1	any	$\Omega(n^2)$
upper bounds			
BFS-tree	1 optimal	$\tilde{O}(n)$	$\tilde{O}(nm)$
[AR'93]	1	$\tilde{O}(n)$	$\tilde{O}(n^2)$

Tradeoffs (Name-independent)

	Stretch	Memory	Communication cost
lower bounds			
[AGM'06a]	$< 2k+1$ < 3	$\Omega(n^{1/k})$ $\Omega(n)$	any
[GGHI'13]	1	any	$\Omega(n^2)$
upper bounds			
BFS-tree	1 optimal	$\tilde{O}(n)$	$\tilde{O}(nm)$
[AR'93]	1	$\tilde{O}(n)$	$\tilde{O}(n^2)$
[AGM'06b]	$O(k)$	$\tilde{O}(n^{1/k})$	centralised

Tradeoffs (Name-independent)

		Stretch	Memory	Communication cost
lower bounds	[AGM'06a]	$< 2k+1$ < 3	$\Omega(n^{1/k})$ $\Omega(n)$	any
	[GGHI'13]	1	any	$\Omega(n^2)$
upper bounds	BFS-tree	1 optimal	$\tilde{O}(n)$	$\tilde{O}(nm)$
	[AR'93]	1	$\tilde{O}(n)$	$\tilde{O}(n^2)$
	[AGM'06b]	$O(k)$	$\tilde{O}(n^{1/k})$	centralised
	[AGM'08]	3	$\tilde{O}(\sqrt{n})$	centralised

Tradeoffs (Name-independent)

	Stretch	Memory	Communication cost
lower bounds			
[AGM'06a]	$< 2k+1$; < 3 ; < 5	$\Omega(n^{1/k})$; $\Omega(n)$; $\Omega(\sqrt{n})$	any
[GGHI'13]	1	any	$\Omega(n^2)$
upper bounds			
BFS-tree	1 optimal	$\tilde{O}(n)$	$\tilde{O}(nm)$
[AR'93]	1	$\tilde{O}(n)$	$\tilde{O}(n^2)$
[AGM'06b]	$O(k)$	$\tilde{O}(n^{1/k})$	centralised
[AGM'08]	3 optimal	$\tilde{O}(\sqrt{n})$	centralised

Tradeoffs (Name-independent)

	Stretch	Memory	Communication cost
lower bounds	[AGM'06a]	$< 2k+1$ (< 3, < 5)	$\Omega(n^{1/k})$ ($\Omega(n)$, $\Omega(\sqrt{n})$)	any
	[GGHI'13]	1	any	$\Omega(n^2)$
upper bounds	BFS-tree	1	$\tilde{O}(n)$	$\tilde{O}(nm)$
	[AR'93]	1 (optimal)	$\tilde{O}(n)$	$\tilde{O}(n^2)$
	[AGM'06b]	$O(k)$	$\tilde{O}(n^{1/k})$	centralised
	[AGM'08]	3 (optimal)	$\tilde{O}(\sqrt{n})$	centralised
	[SGF+10]	7	$\tilde{O}(\sqrt{n})$	NC.

Tradeoffs (Name-independent)

		Stretch		Memory		Communication cost
lower bounds	[AGM'06a]	$< 2k+1$	< 3 < 5	$\Omega(n^{1/k})$	$\Omega(n)$ $\Omega(\sqrt{n})$	any
	[GGHI'13]	1		any		$\Omega(n^2)$
upper bounds	BFS-tree	1	optimal	$\tilde{O}(n)$		$\tilde{O}(nm)$
	[AR'93]	1		$\tilde{O}(n)$		$\tilde{O}(n^2)$
	[AGM'06b]	$O(k)$		$\tilde{O}(n^{1/k})$		centralised
	[AGM'08]	3	optimal	$\tilde{O}(\sqrt{n})$		centralised
	[SGF+10]	7		$\tilde{O}(\sqrt{n})$		NC.
	[GGHI'13]	7		$\tilde{O}(\sqrt{n})$		$O(m\sqrt{n})$

Tradeoffs (Name-independent)

		Stretch		Memory		Communication cost
lower bounds	[AGM'06a]	$< 2k+1$	< 3 < 5	$\Omega(n^{1/k})$	$\Omega(n)$ $\Omega(\sqrt{n})$	any
	[GGHI'13]	1		any		$\Omega(n^2)$
upper bounds	BFS-tree	1	optimal	$\tilde{O}(n)$		$\tilde{O}(nm)$
	[AR'93]	1		$\tilde{O}(n)$		$\tilde{O}(n^2)$
	[AGM'06b]	$O(k)$		$\tilde{O}(n^{1/k})$		centralised
	[AGM'08]	3	optimal	$\tilde{O}(\sqrt{n})$		centralised
	[SGF+10]	7		$\tilde{O}(\sqrt{n})$		NC.
	[GGHI'13]	7		$\tilde{O}(\sqrt{n})$		$O(m\sqrt{n})$
	my thesis	5		$\tilde{O}(\sqrt{n})$		$O(m\sqrt{n})$

TRADEOFFS (LABELED)

lower bounds

upper bounds

	Stretch	Memory	Communication cost
[AGM'06]	$< 2k+1$	$\Omega(n^{1/k})$	any

TRADEOFFS (LABELED)

lower bounds

upper bounds

	Stretch	Memory	Communication cost
[AGM'06]	$< 2k+1$	$\Omega(n^{1/k})$	any
[GGHI'13]	1	any	$\Omega(n^2)$

TRADEOFFS (LABELED)

lower bounds

upper bounds

	Stretch	Memory	Communication cost
[AGM'06]	$< 2k+1$	$\Omega(n^{1/k})$	any
[GGHI'13]	1	any	$\Omega(n^2)$
All name-independent results hold,			

TRADEOFFS (LABELED)

	Stretch	Memory	Communication cost
lower bounds			
[AGM'06]	$< 2k+1$	$\Omega(n^{1/k})$	any
[GGHI'13]	1	any	$\Omega(n^2)$
All name-independent results hold,			
upper bounds			
[TZ'01]	$4k-5$	$\tilde{O}(n^{1/k})$	centralised

TRADEOFFS (LABELED)

	Stretch	Memory	Communication cost
lower bounds			
[AGM'06]	$< 2k+1$	$\Omega(n^{1/k})$	any
[GGHI'13]	1	any	$\Omega(n^2)$
All name-independent results hold,			
upper bounds			
[TZ'01]	$4k-5$	$\tilde{O}(n^{1/k})$	centralised
[TZ'01]	3	$\tilde{O}(\sqrt{n})$	centralised

HOW TO BUILD A ROUTING SCHEME

PART I : Labeled Compact Routing in Trees

Labeled compact routing in trees

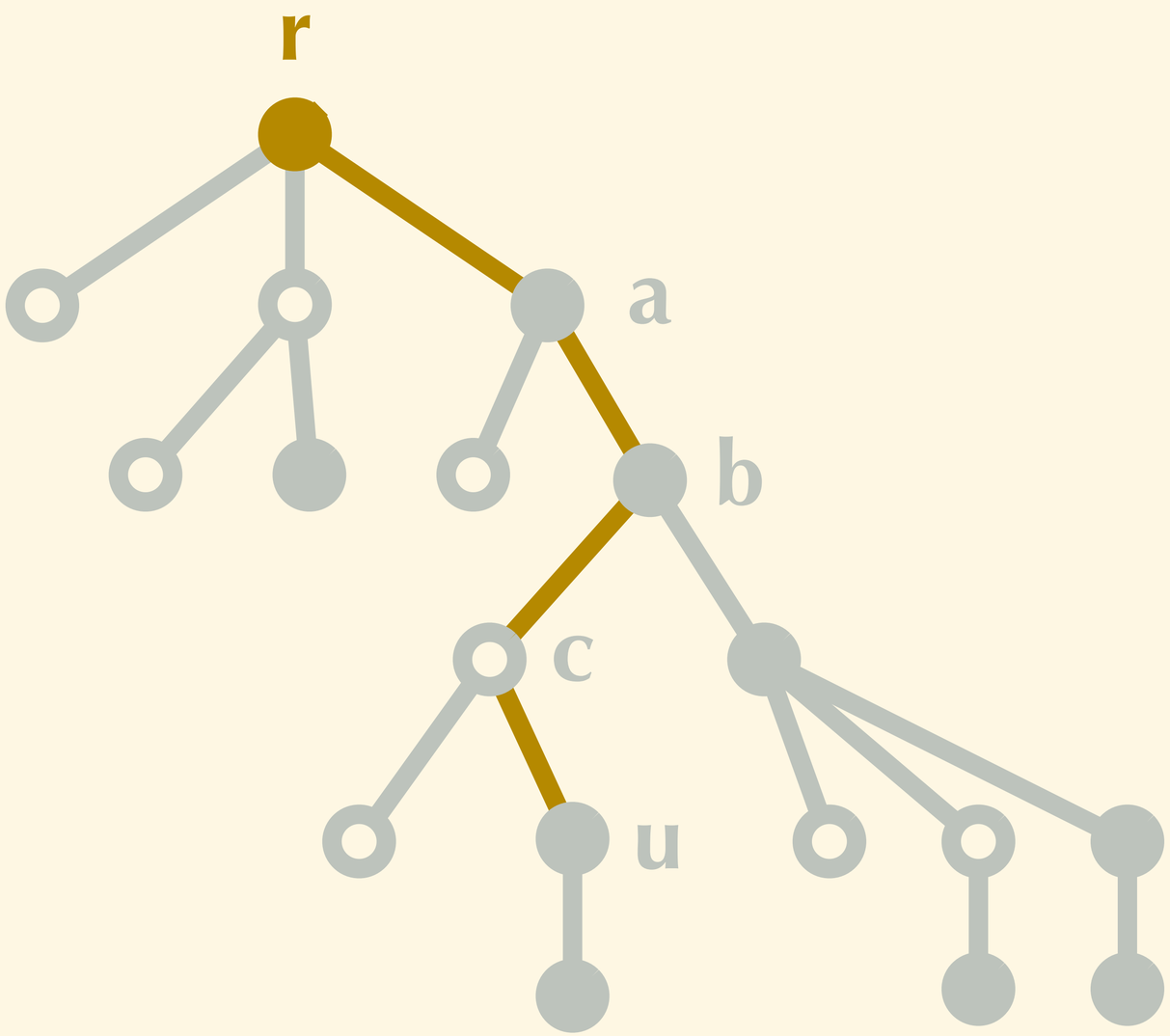
Two candidates

- I will not talk about the difference, they have very similar performances:
  - "Thorup-Zwick" [TZ'01]
  - "Fraigniaud-Gavoille" [FG'01]
- Achieve labeled compact routing in rooted trees with:
  - Memory space: $O(\log^2 n)$*
  - Stretch: 1
  - (Constant query time)

*[FG'01] actually have a $O(\log^2 n/\text{loglog } n)$ memory space

LABELED COMPACT ROUTING IN TREES

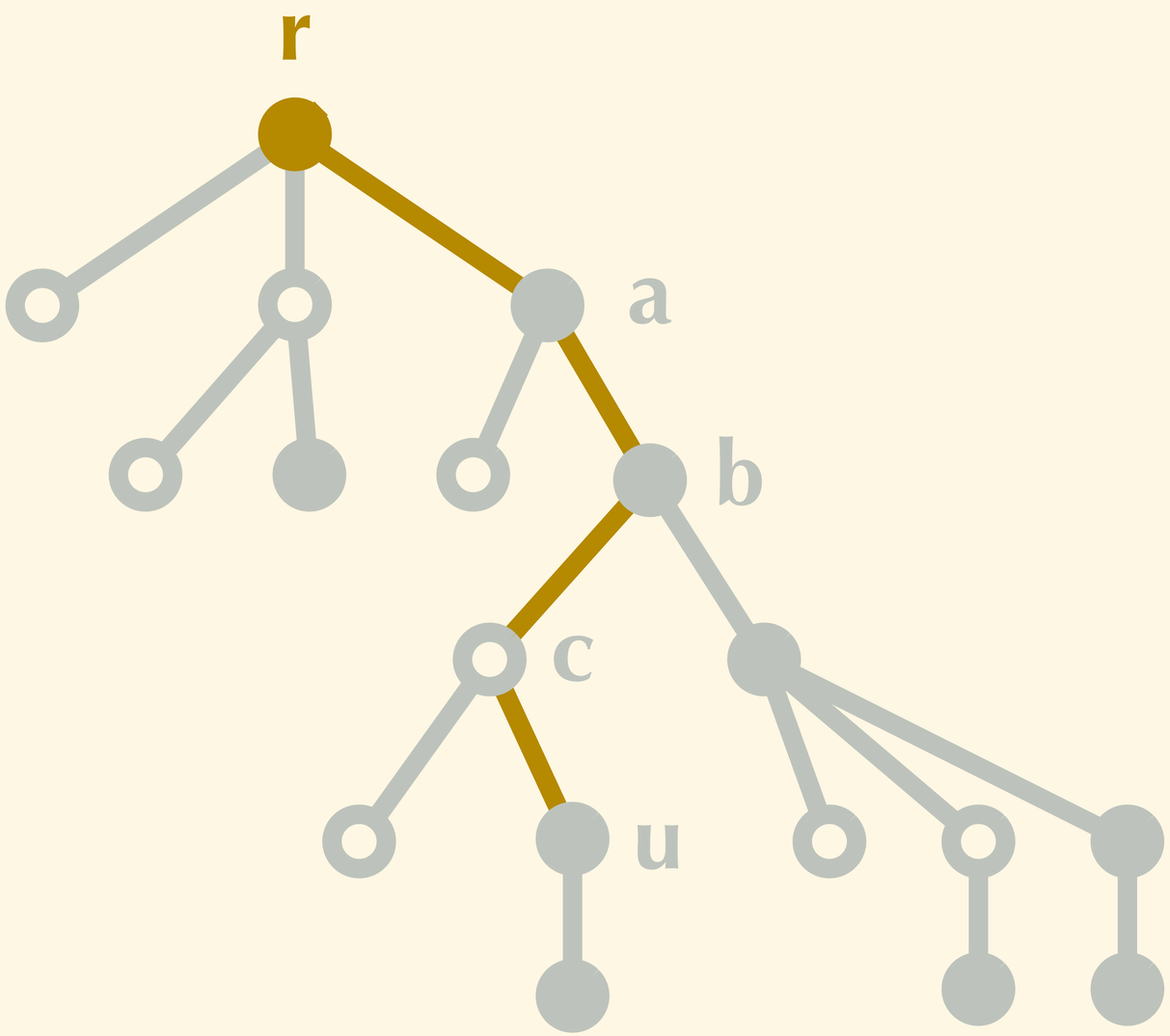
Fraignaud-Gavoille - Heavy child



- GOAL, for every node u:
  - compute the compact routing label $\ell(u)$ which represent path from u to r in the tree.
- First step, heavy children nodes:
  - in the example the heavy child is always the rightmost one.

Labeled compact routing in trees

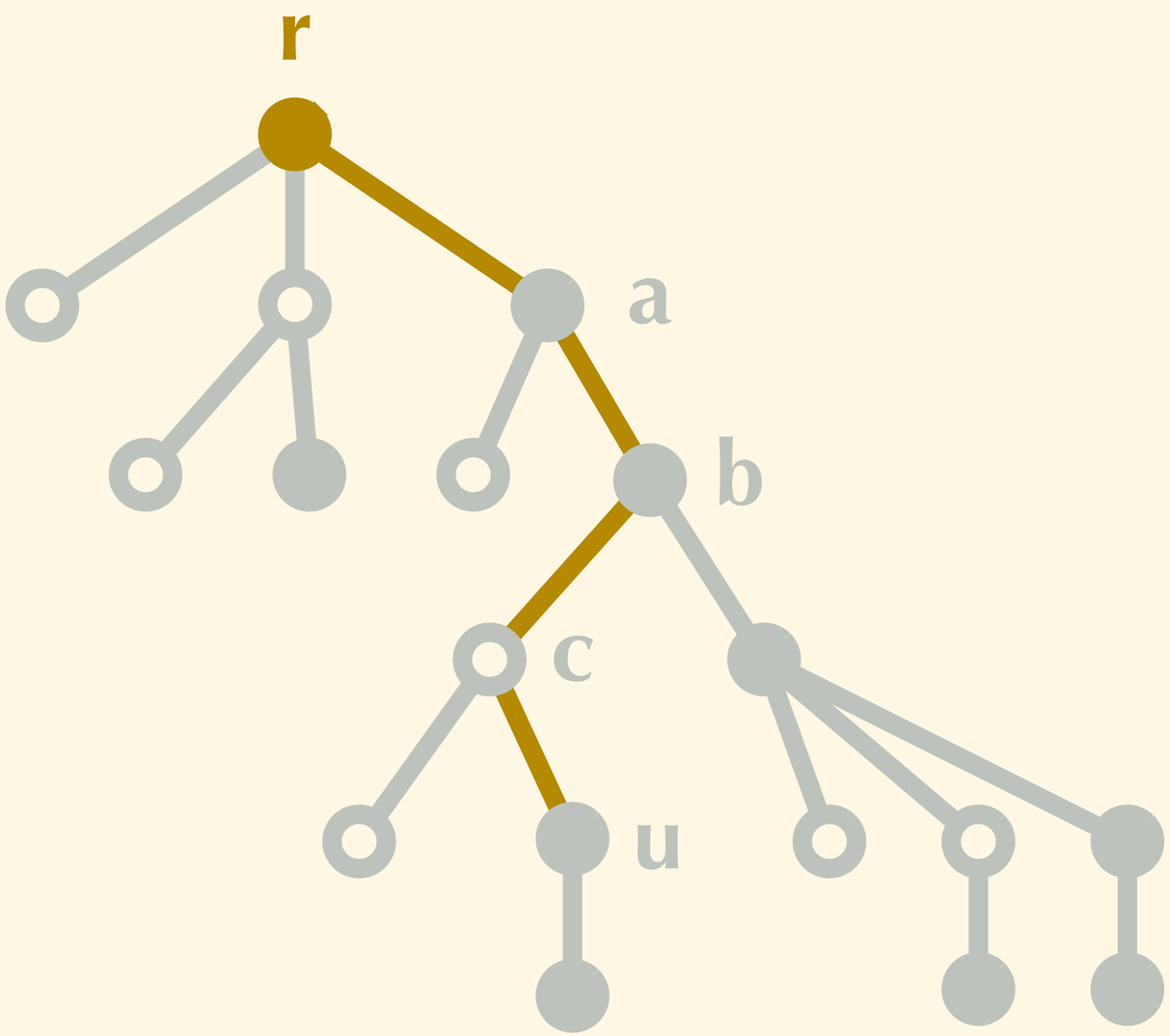
Fraignaud-Gavoille - Nodes names



path	r	a	b	c	u

Labeled compact routing in trees

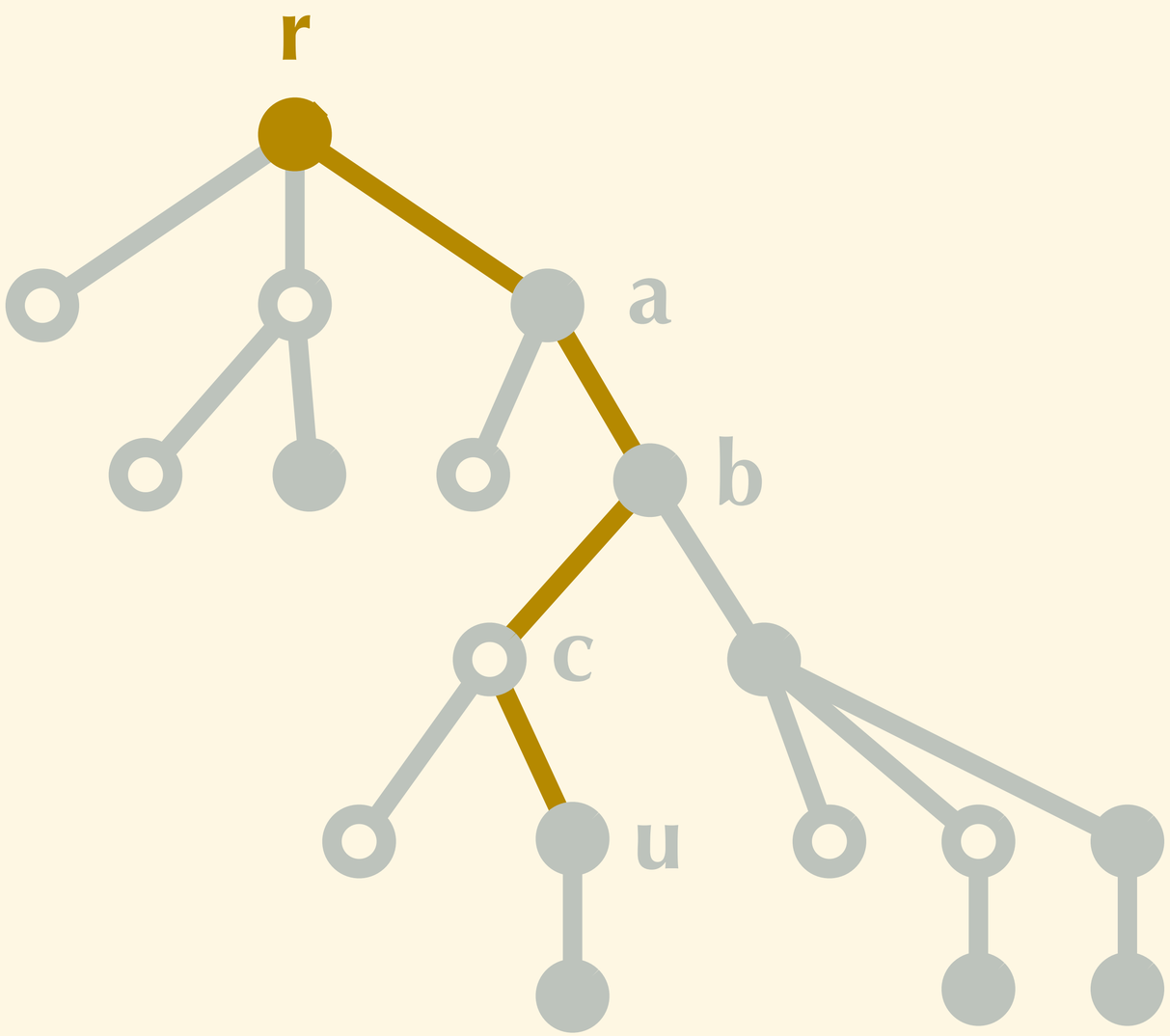
Fraignaud-Gavoille - Nodes names



path	r	a	b	c	u
path*	r	*	*	c	*

Labeled compact routing in trees

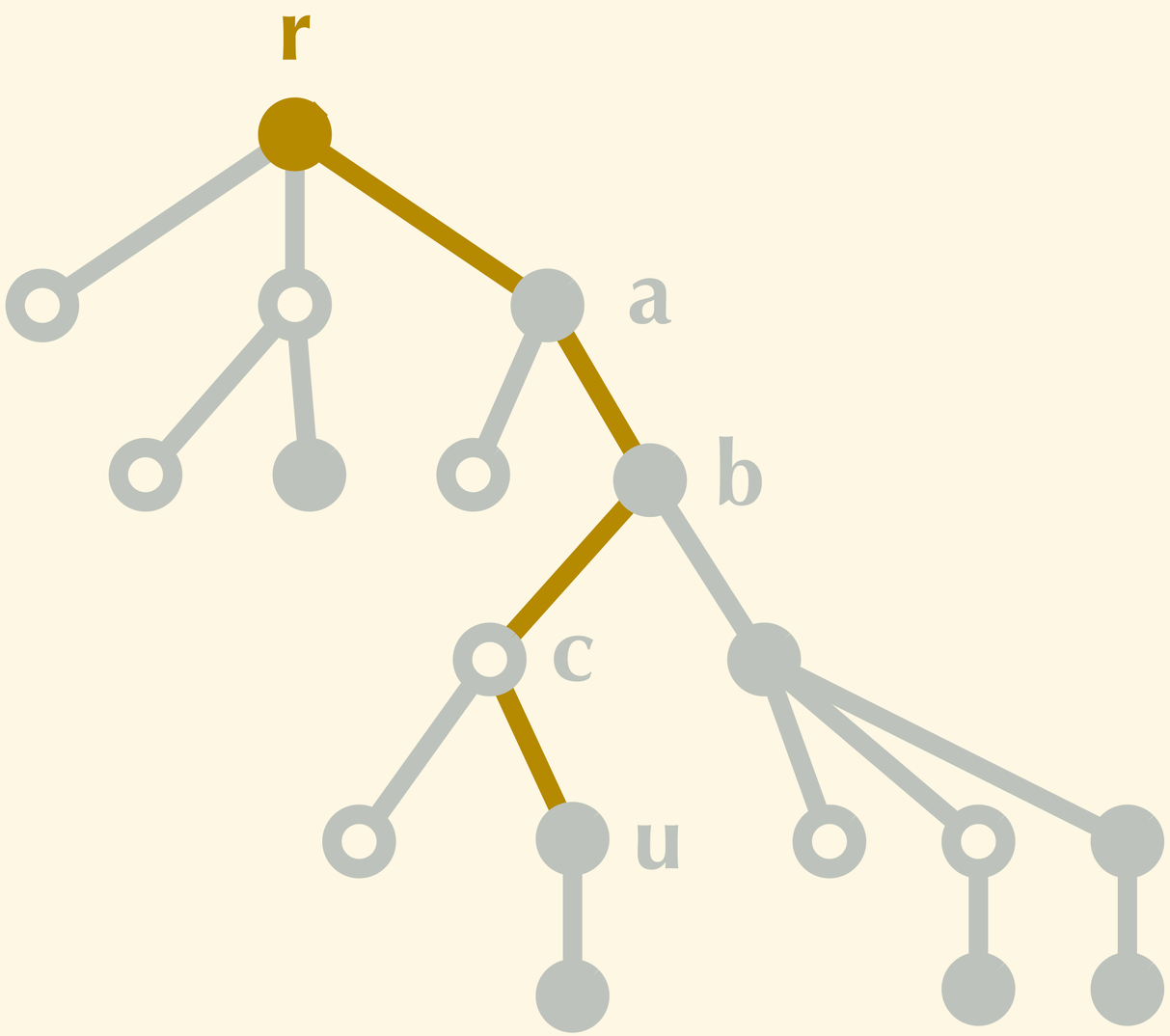
Fraignaud-Gavoille - Nodes names



path	r	a	b	c	u
path*	r	*	*	c	*
cpath_u	r		2	c	1

Labeled compact routing in trees

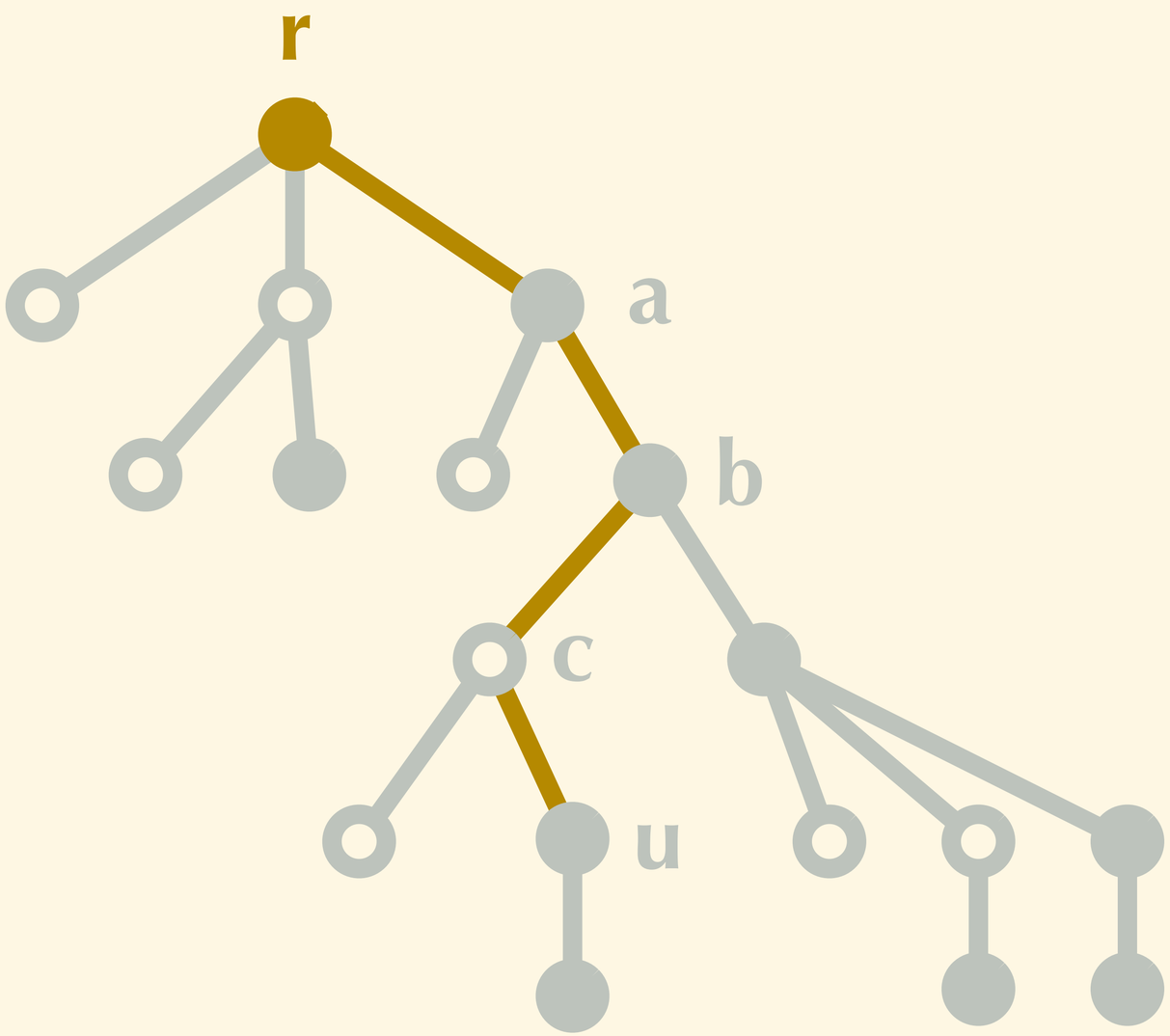
Fraignaud-Gavoille - Nodes names



path	r	a	b	c	u
path*	r	*	*	c	*
cpath_u	r		2	c	1
b_u	1		0	1	0

Labeled compact routing in trees

Fraignaud-Gavoille - Nodes names



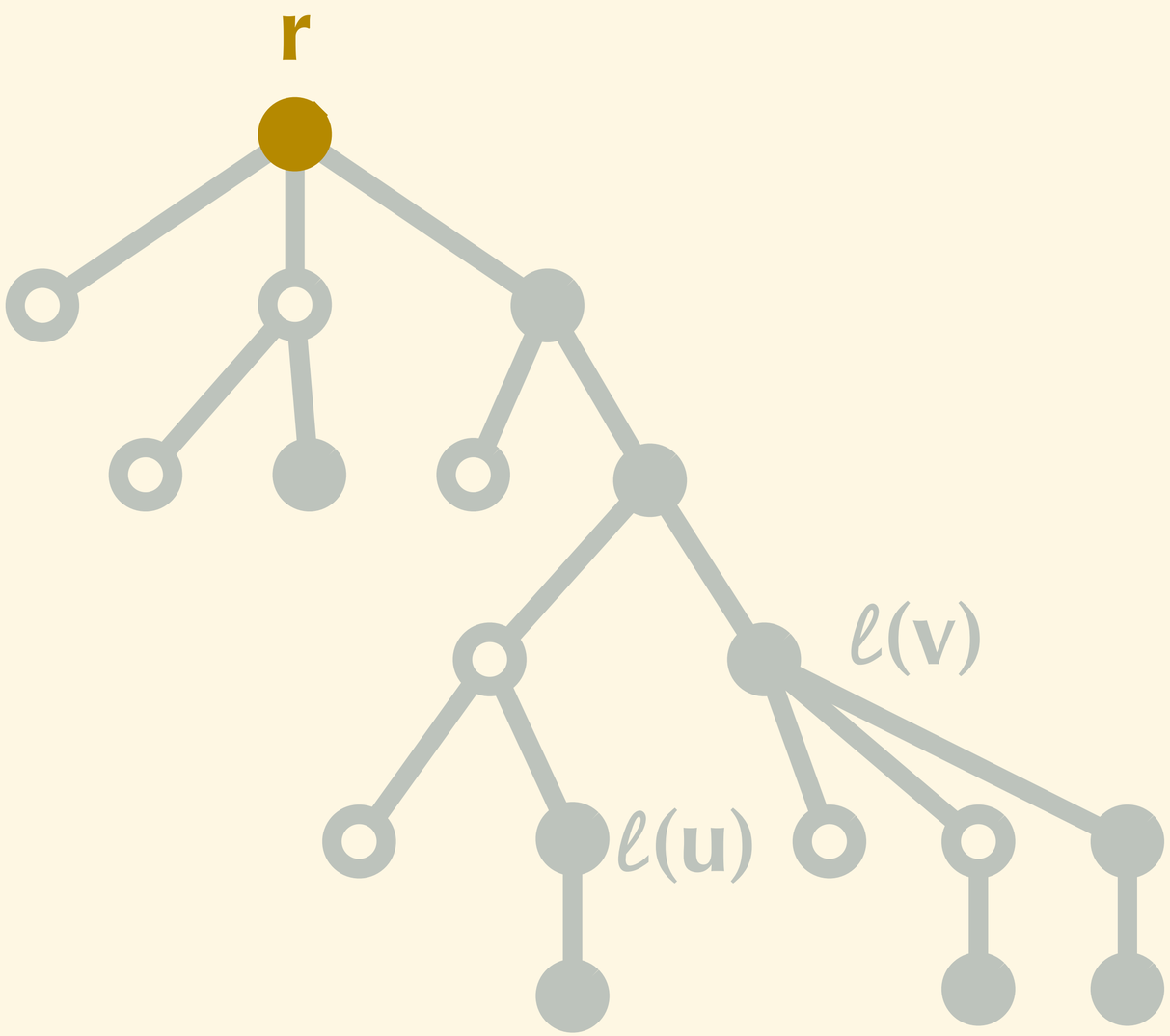
path	r	a	b	c	u
path*	r	*	*	c	*
cpath_u	r		2	c	1
b_u	1		0	1	0

Finally we set the name of node u to

$\ell(u) = (u, cpath\_u, b\_u)$

LABELED COMPACT ROUTING IN TREES

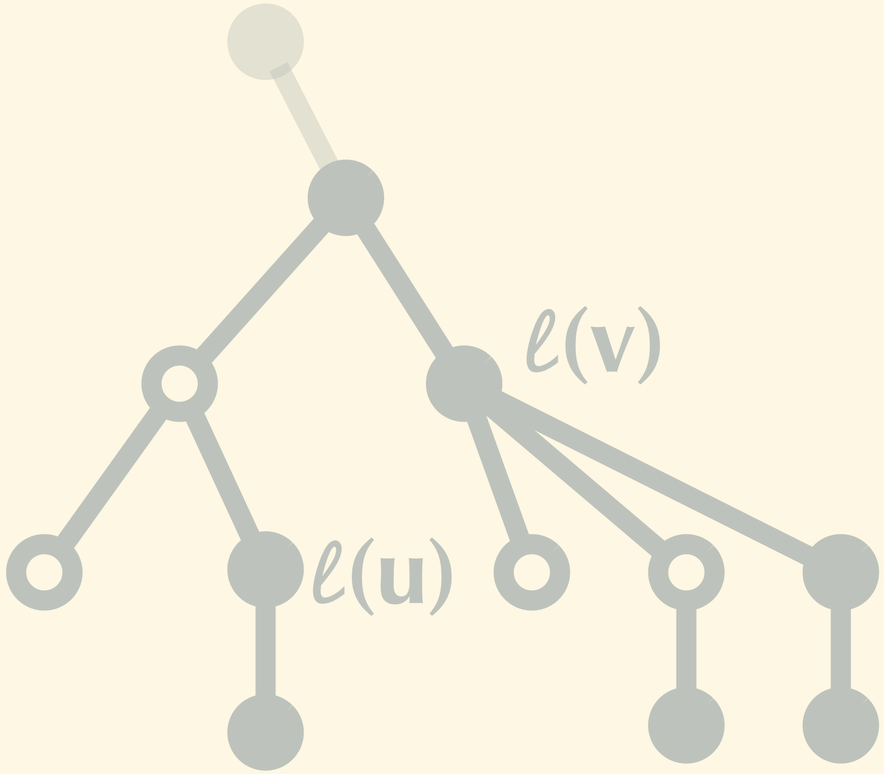
Fraignaud-Gavoille - Routing tables



- Routing tables are very small, every node stores two entries:
  - link to the parent
  - link to the heavy child
- Routing is based on "prefix comparisons" of the source and destination labels.

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Routing algorithm (roughly)



$\ell(u)$

cpath_u	r	2	c	1
b_u	1	0	1	0

$\ell(v)$

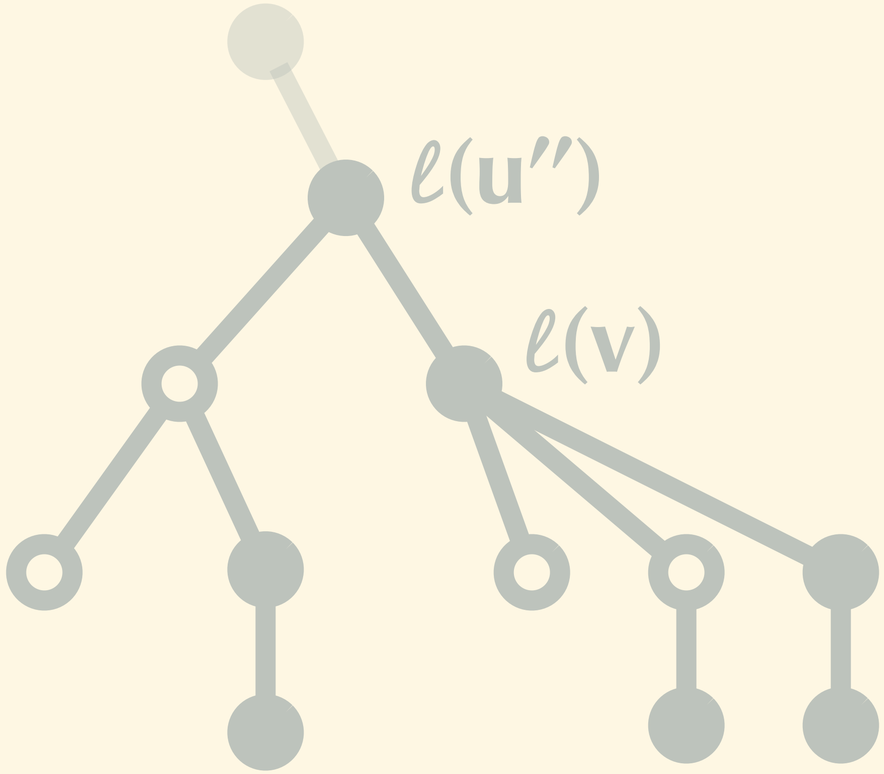
cpath_v	r	3
b_v	1	0

Routing:

- b_u **is not a prefix of** b_v ⇒ $\ell(v)$ **is not a descendent**
- **route toward parent of** $\ell(u)$ **(info. in routing table)**

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Routing algorithm (roughly)



ℓ(u″)

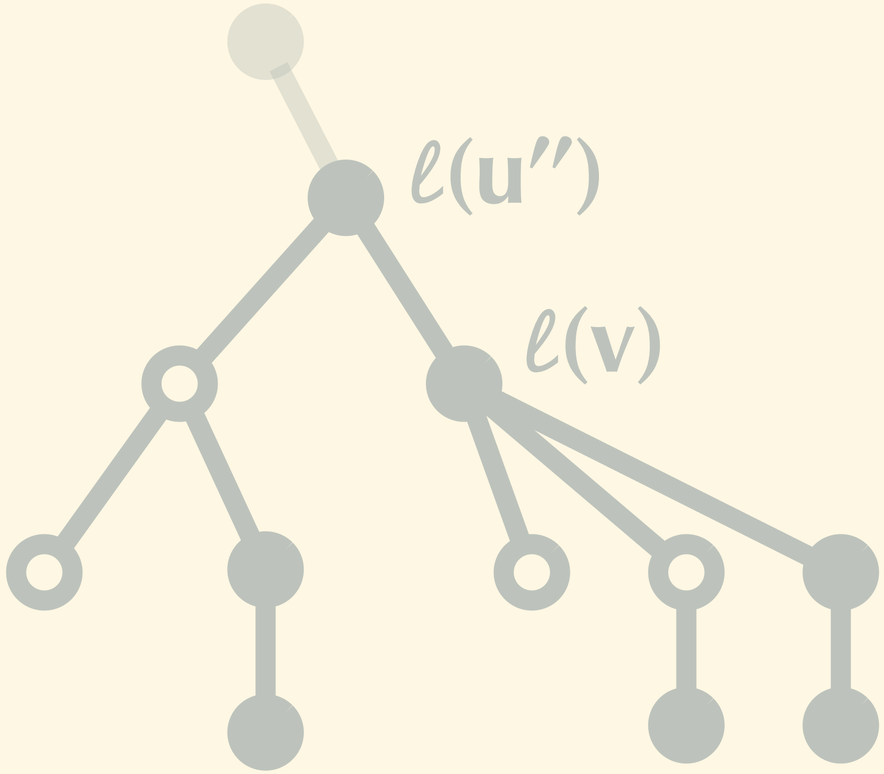
cpath_u″	r	2
b_u″	1	0

ℓ(v)

cpath_v	r	3
b_v	1	0

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Routing algorithm (roughly)



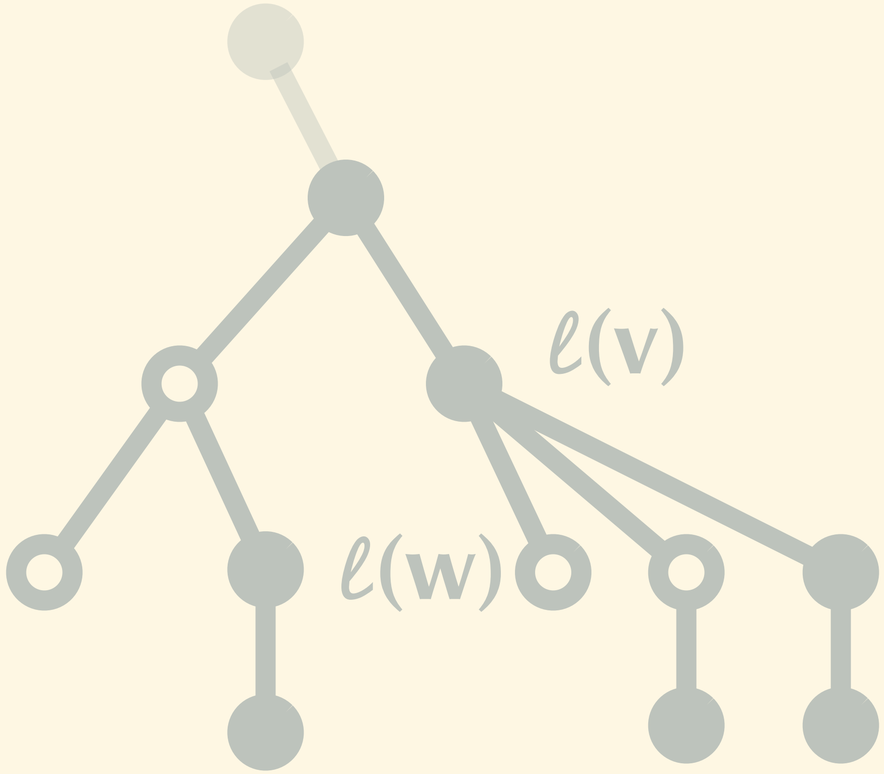
ℓ(u'')	cpath_u''	r	2
	b_u''	1	0
ℓ(v)	cpath_v	r	3
	b_v	1	0

Routing:

- $\ell(u'')$ **is almost a prefix of** $\ell(v)$
- cpath_u''[-1] < cpath_v[-1]
  therefore $\ell(v)$ **is a descendent of** $\ell(u'')$
- **since** b_v **ends with 0,** $\ell(v)$ **is in the heavy side,**
- **route using the routing table of** $\ell(u'')$

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Routing algorithm (roughly)



ℓ(v)

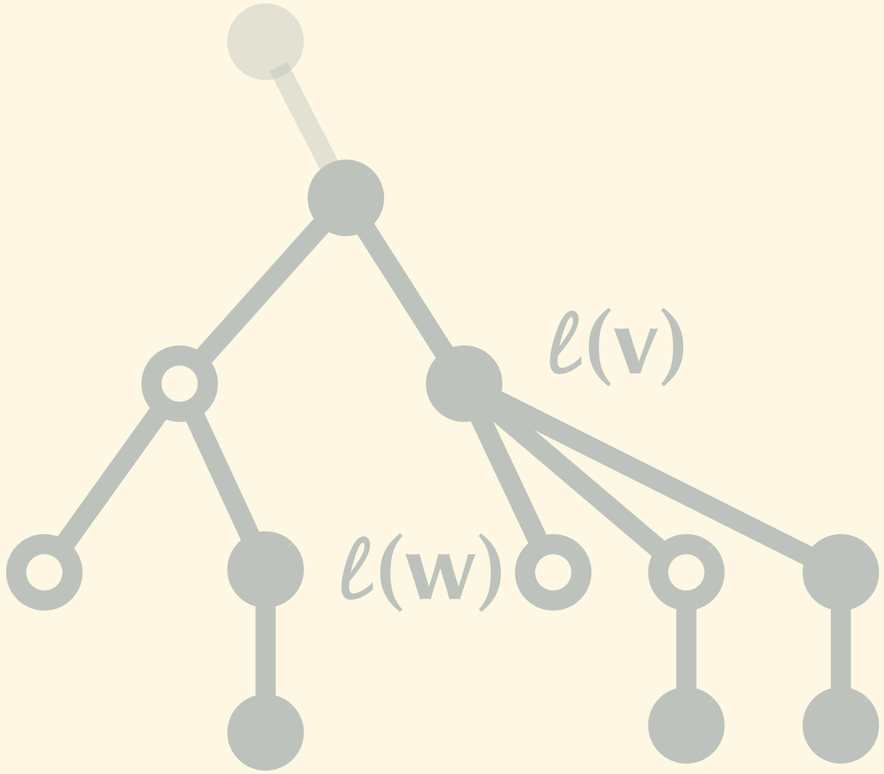
cpath_w	r	3
b_w	1	0

ℓ(w)

cpath_v	r	3	w
b_v	1	0	1

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Routing algorithm (roughly)



$\ell(v)$

cpath_w	r	3	
b_w	1	0	

$\ell(w)$

cpath_v	r	3	w
b_v	1	0	1

Routing:

- similarly as before, but in this case the routing information is retrieved in the cpath of $\ell(v)$.

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Labels size

- Now we know that routing can be achieved giving nodes labels, with two routing entries per node.
- But what about the nodes names (labels) sizes?

Labeled compact routing in trees

Fraignaud-Gavoille - Labels size

path	r	a	b	c	u
path*	r	*	*	c	*
cpath_u	r		2	c	1
b_u	1		0	1	0

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Labels size

path	r	a	b	c	u
path*	r	*	*	c	*
cpath_u	r		2	c	1
b_u	1		0	1	0

- Size of path can be n.
- What about cpath size?

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Labels size

path	r	a	b	c	u
path*	r	*	*	c	*
cpath_u	r	2		c	1
b_u	1	0		1	0

- Size of path can be n.
- What about cpath size?
  - What is the maximum number of 1's in b_u? #of non-heavy children? (clue: the worst-case-tree is the binary one, why?)

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Labels size

path	r	a	b	c	u
path*	r	*	*	c	*
cpath_u	r	2		c	1
b_u	1	0		1	0

- Size of path can be n.
- What about cpath size?
  - What is the maximum number of 1's in b_u? #of non-heavy children? (clue: the worst-case-tree is the binary one, why?)
  - By definition, path* stars are grouped if they are consecutive, therefore the number of 0's in b_u is also bounded by log n.

LABELED COMPACT ROUTING IN TREES

Fraignaud-Gavoille - Labels size

path	r	a	b	c	u
path*	r	*	*	c	*
cpath_u	r	2		c	1
b_u	1	0		1	0

- Size of path can be n.
- What about cpath size?
  - What is the maximum number of 1's in b_u? #of non-heavy children? (clue: the worst-case-tree is the binary one, why?)
  - By definition, path* stars are grouped if they are consecutive, therefore the number of 0's in b_u is also bounded by log n.
- Every item in cpath is of size O(log n) (either a length or a node id), therefore, the total size of $\ell(u)$ is $O(\log^2 n)$.

Labeled compact routing in trees

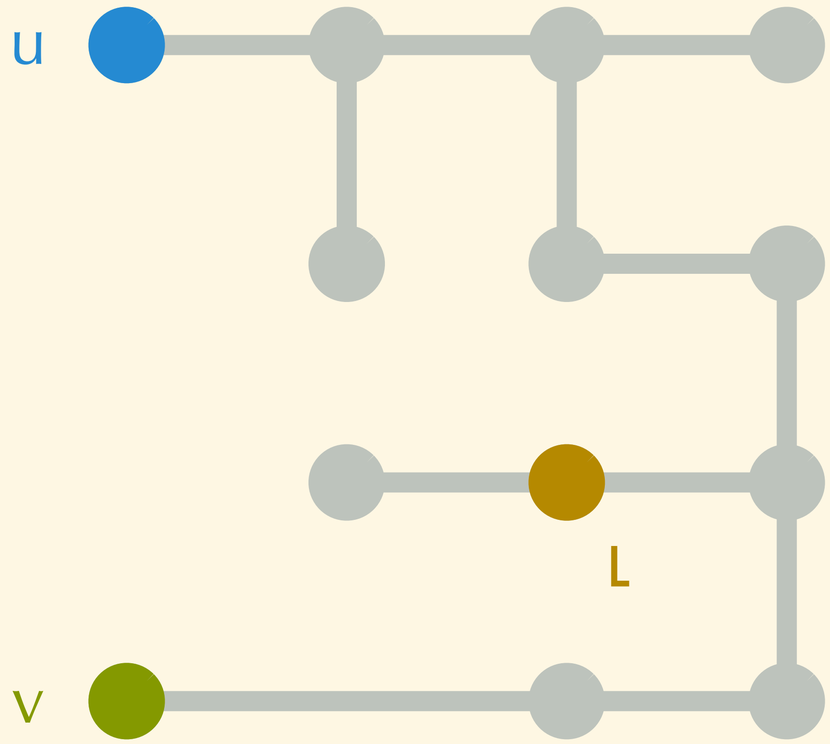
Two candidates

- Performances of [FG'01] for rooted trees:
  - Memory space: $O(\log^2 n)$
  - Stretch: 1 (shortest path)
  - Labeled scheme

HOW TO BUILD A ROUTING SCHEME

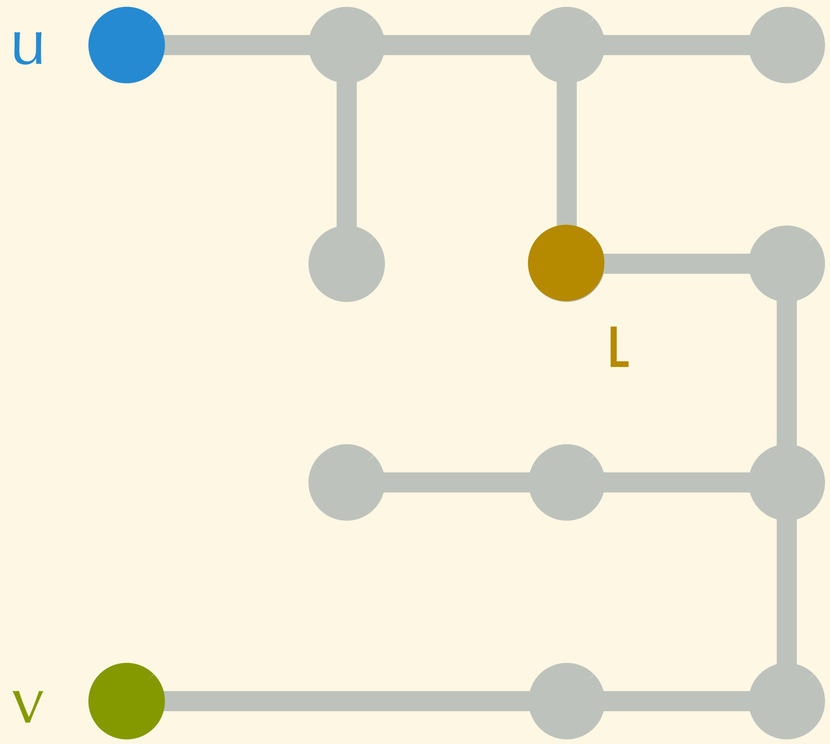
PART II : Labeled compact routing for general graphs

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES



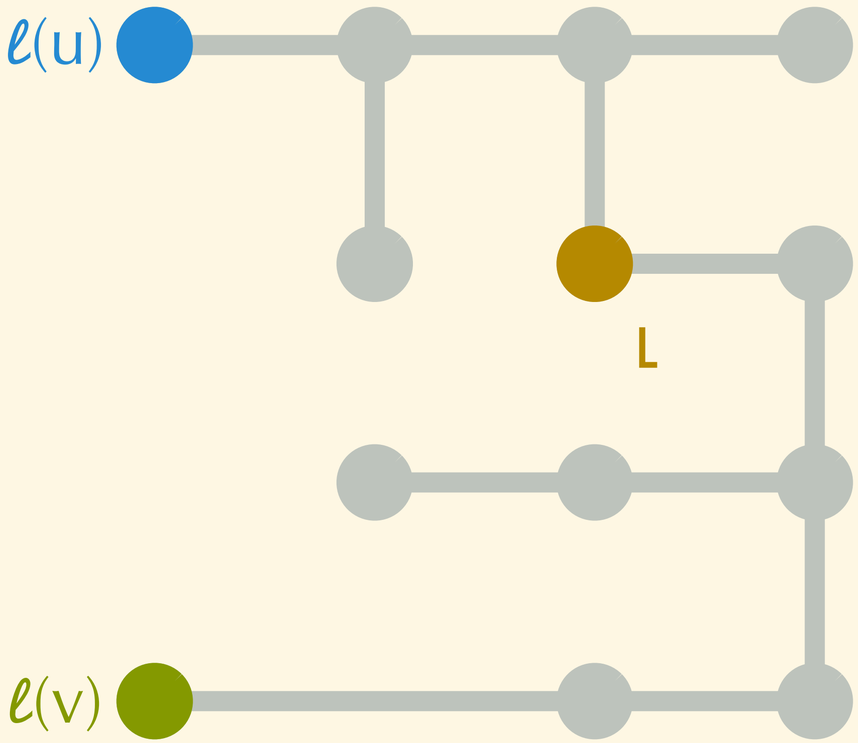
- Using "reference node(s)"
  - we call them, Landmark(s)

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES



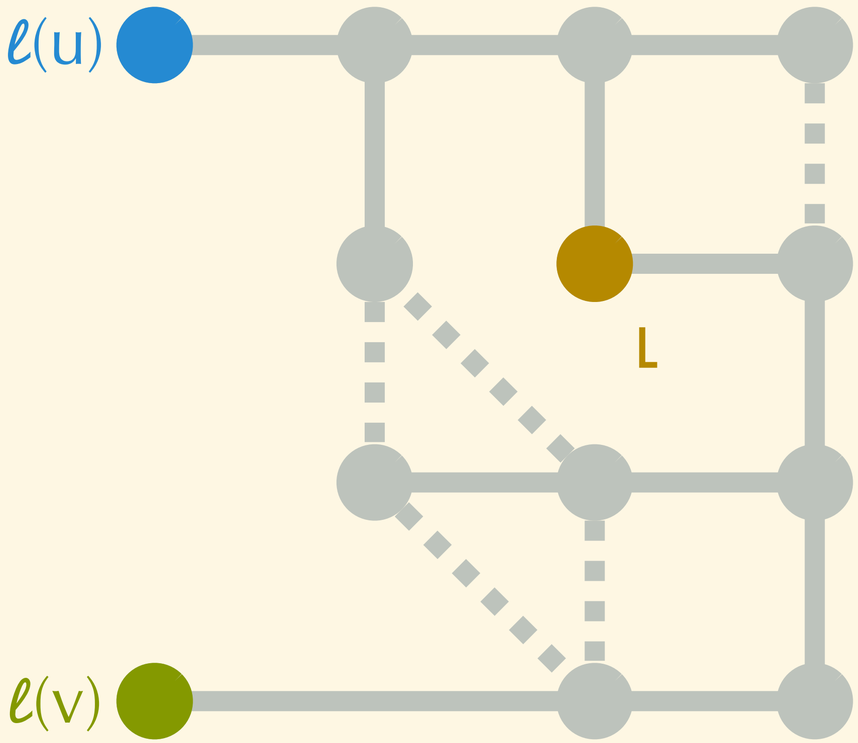
- Using "reference node(s)"
  - we call them, Landmark(s)

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES



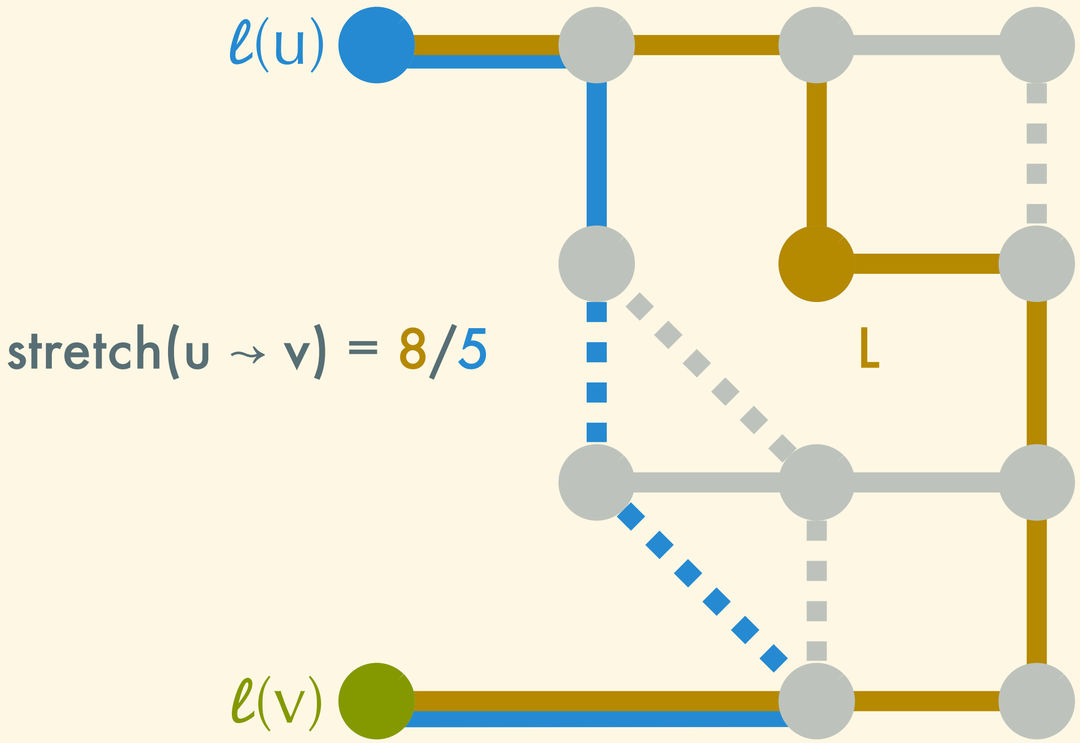
- In a tree, with [FG'01], we can do shortest path with constant memory in the labeled model.

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES



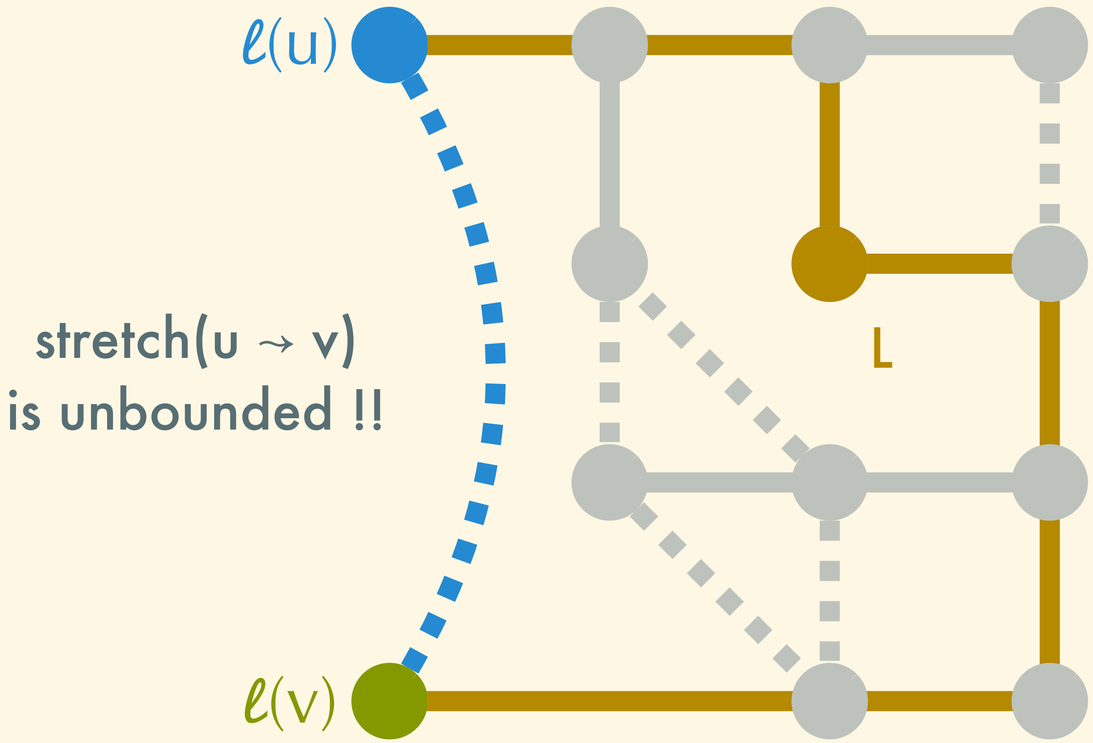
- In a tree, with [FG'01], we can do shortest path with constant memory in the labeled model.
- But not every graph is a tree …

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES



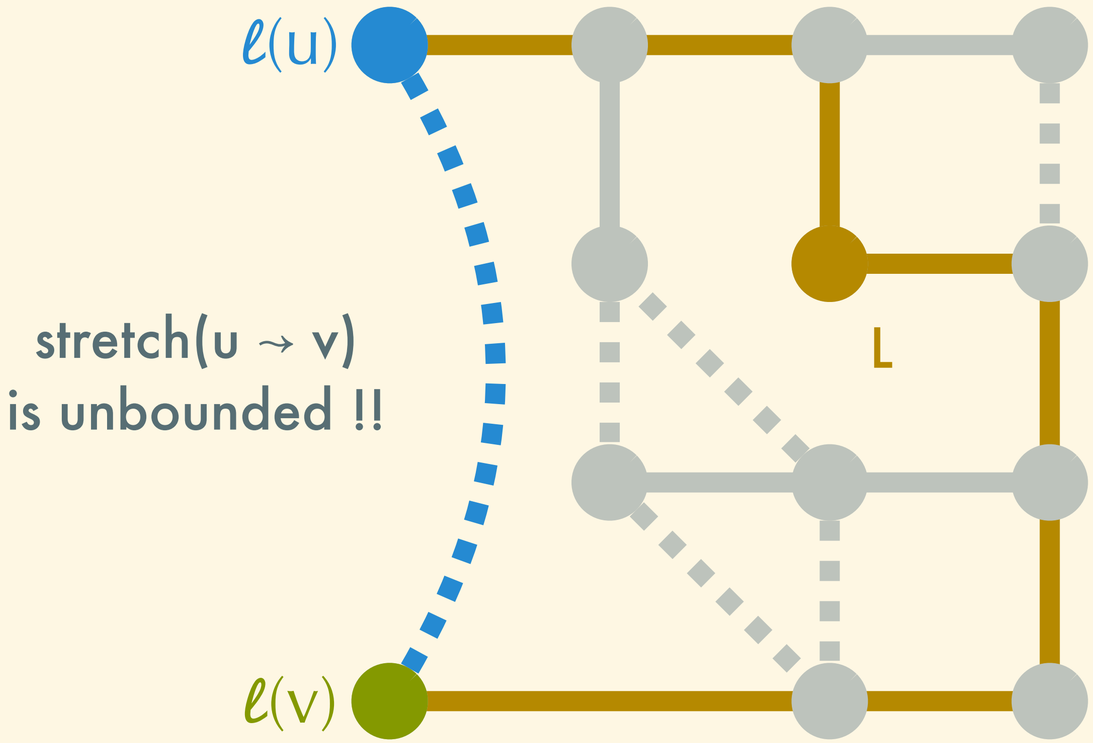
- In a tree, with [FG'01], we can do shortest path with constant memory in the labeled model.
- But not every graph is a tree …

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES



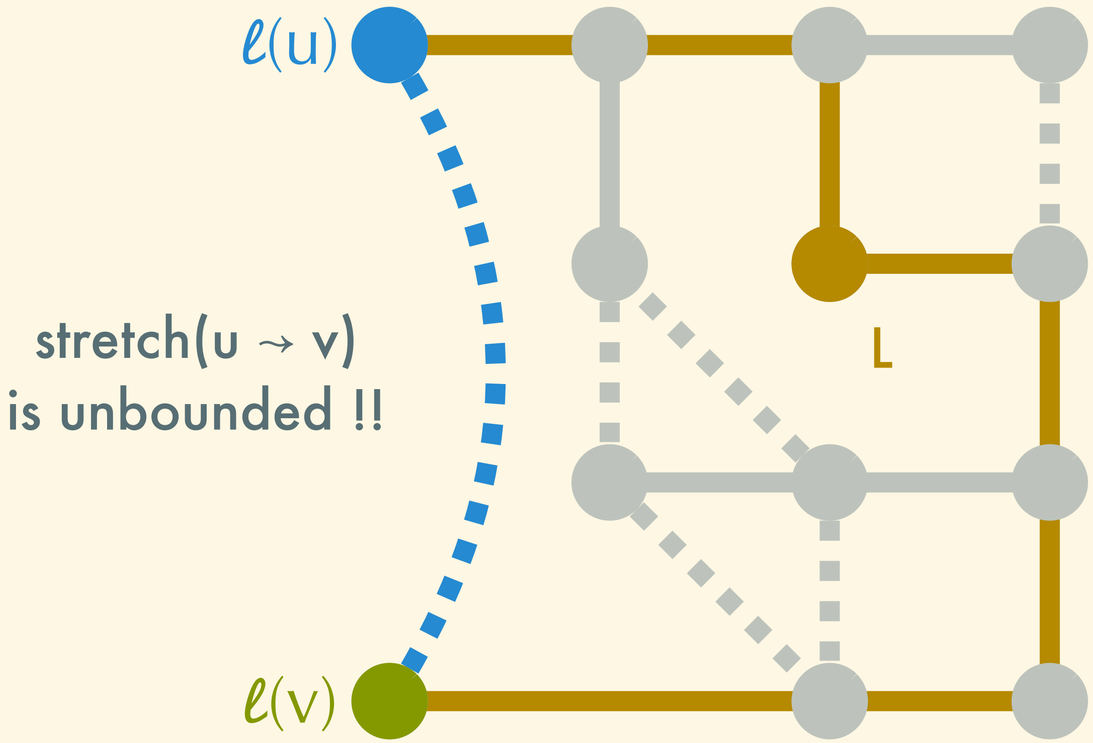
- In a tree, with [FG'01], we can do shortest path with constant memory in the labeled model.
- But not every graph is a tree …

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES



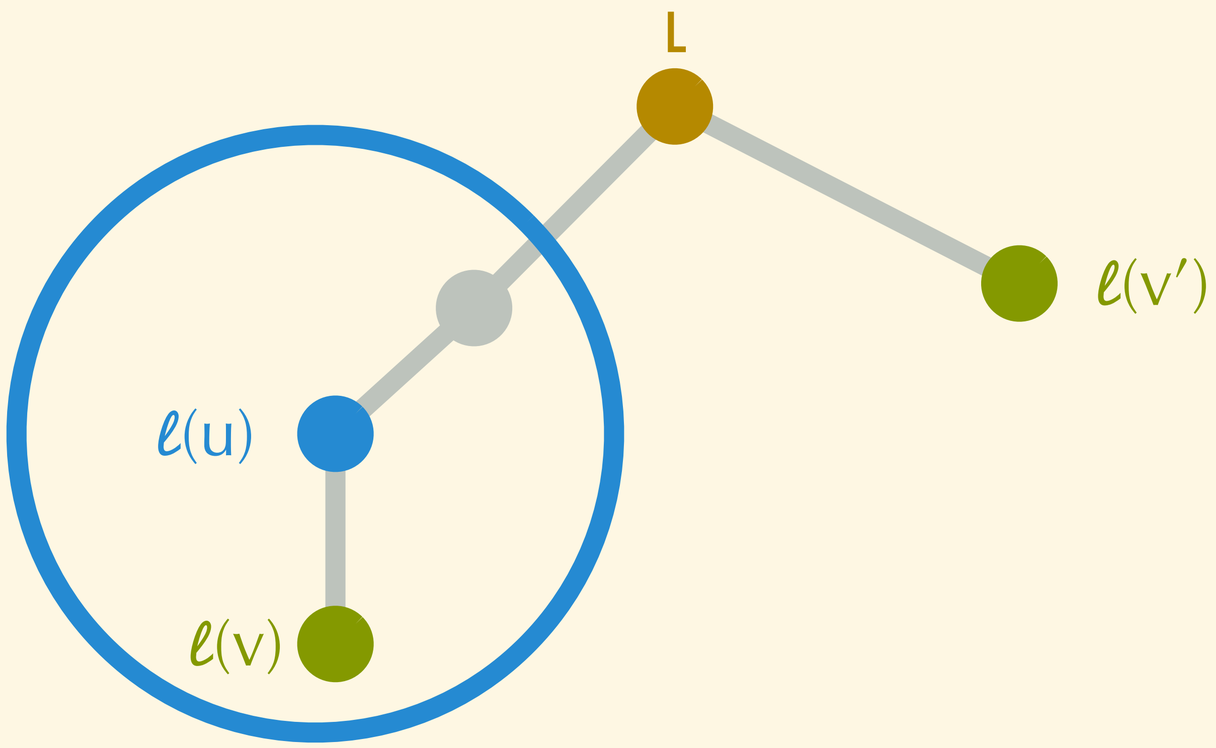
- In a tree, with [FG'01], we can do shortest path with constant memory in the labeled model.
- But not every graph is a tree …
- The stretch can be unbounded.

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES



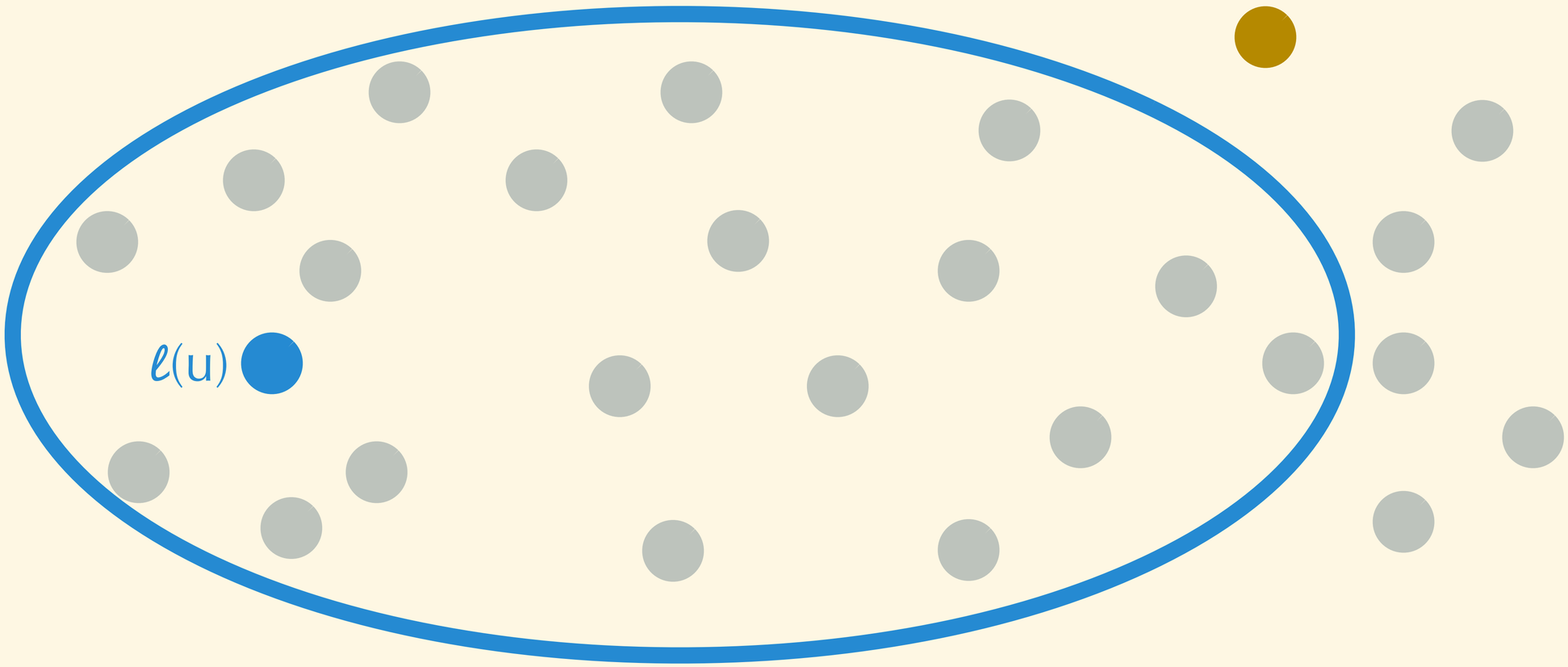
- In a tree, with [FG'01], we can do shortest path with constant memory in the labeled model.
- But not every graph is a tree …
- The stretch can be unbounded.
- Solution: use *"vicinity balls"*

USING [FG'01] TO DESIGN UNIVERSAL COMPACT ROUTING SCHEMES

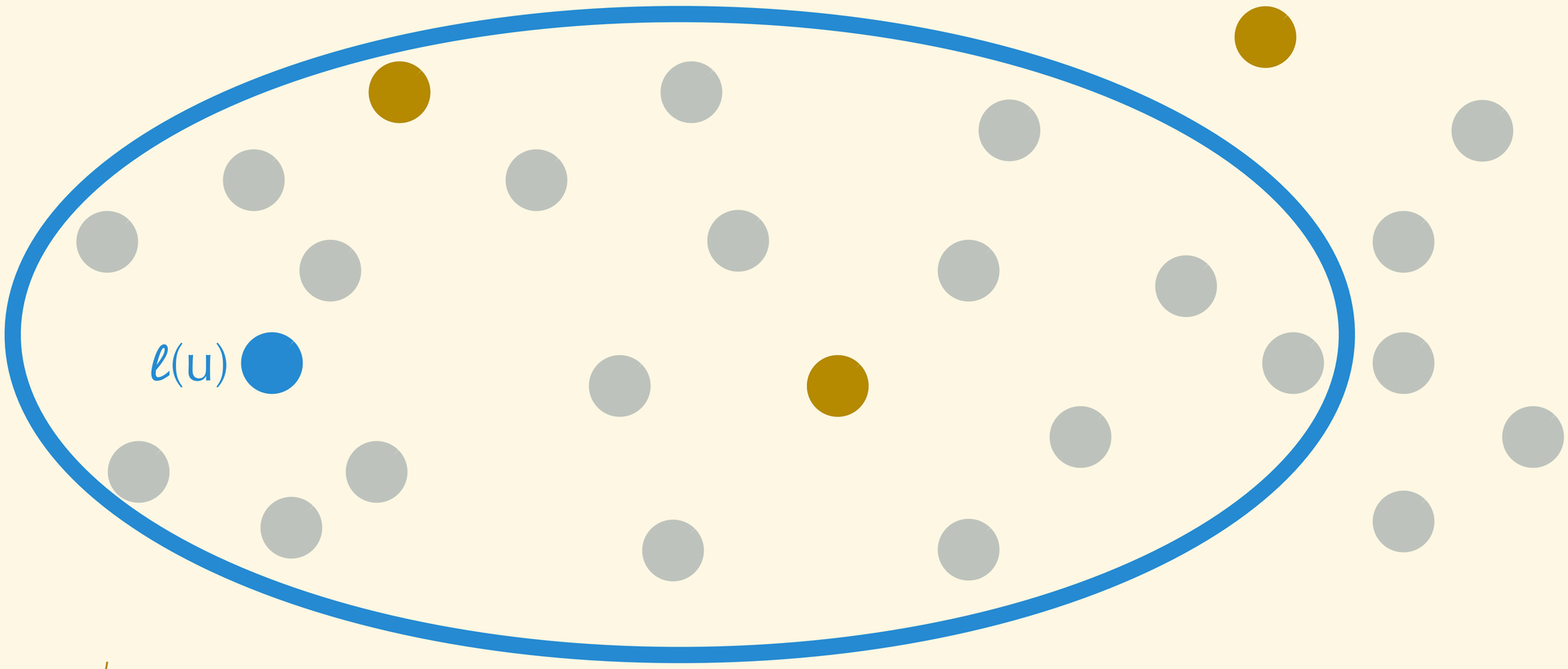


- For nodes closer than L e.g., ℓ(v), use classical shortest path routing
- For others e.g., ℓ(v'), use [FG'01] routing
- This way the stretch is bounded by 3.
- But … vicinity balls can be huge, and so as the routing tables size.

Labeled compact routing scheme

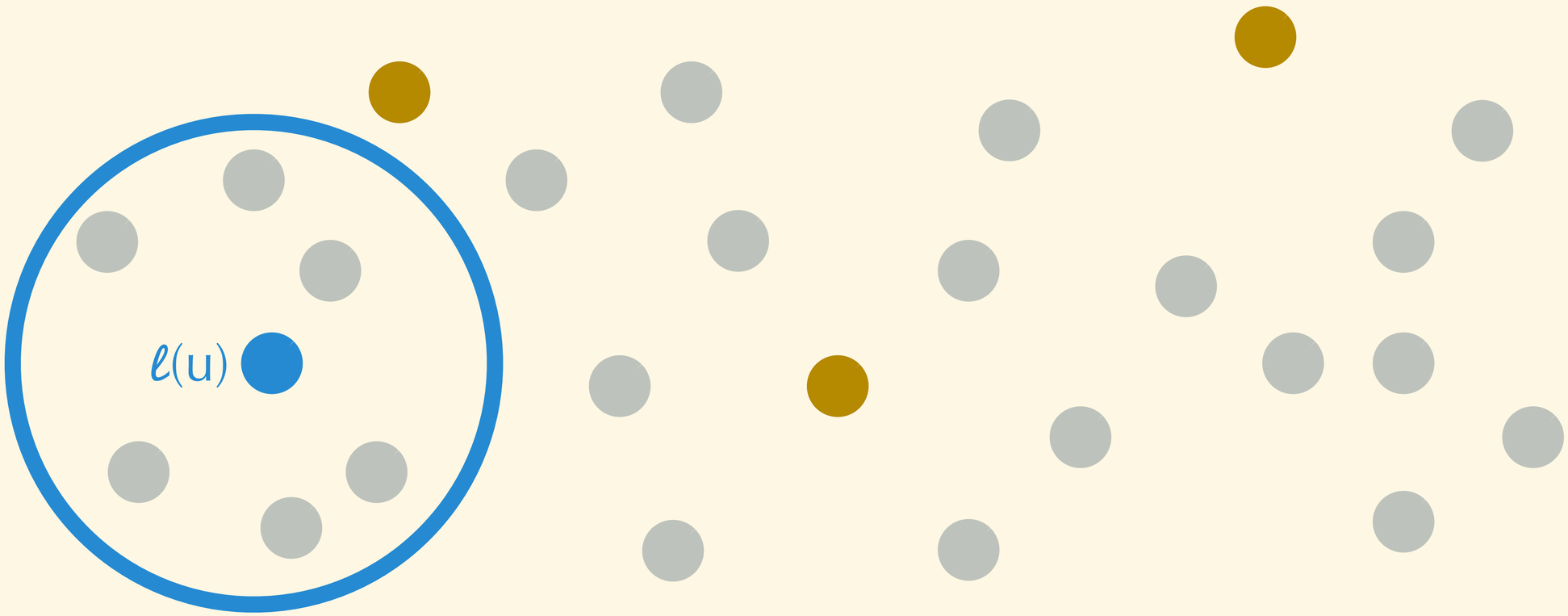


Labeled compact routing scheme



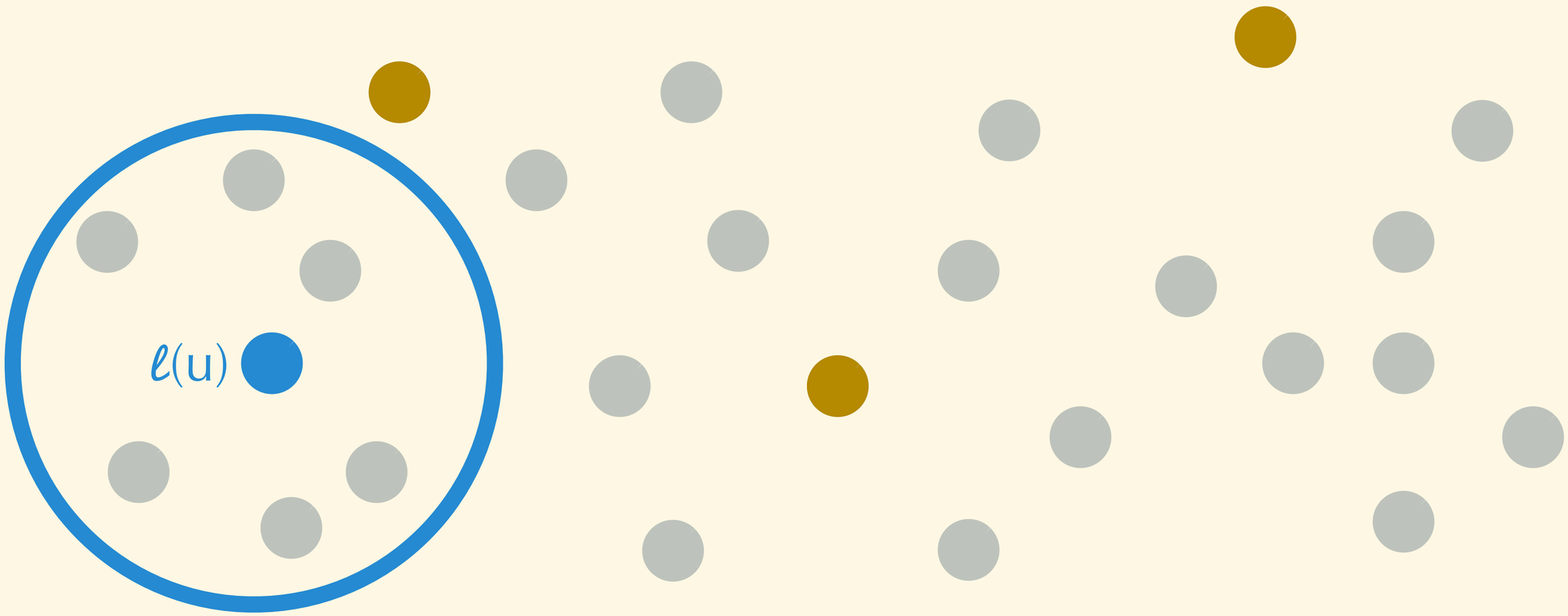
With $\sqrt{n}$ landmarks

LABELED COMPACT ROUTING SCHEME



With $\sqrt{n}$ landmarks
vicinity balls have
size $\tilde{O}(\sqrt{n})$

Labeled compact routing scheme

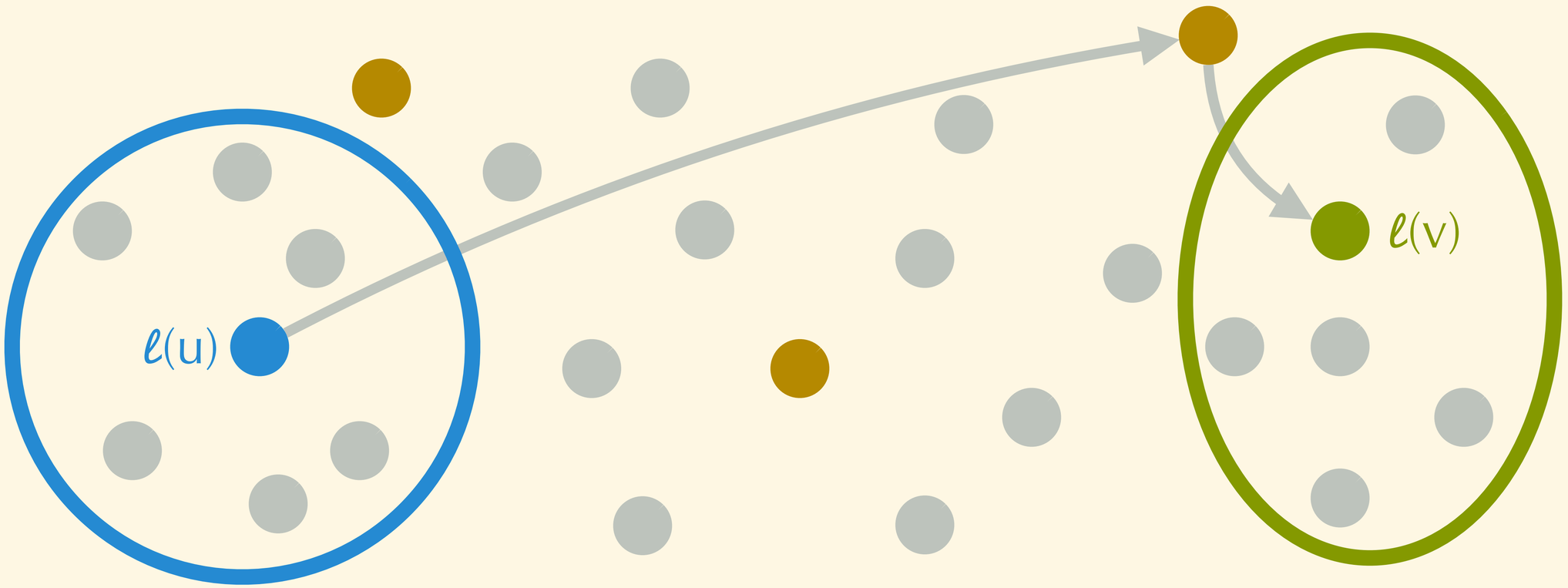


With $\sqrt{n}$ landmarks vicinity balls have size $\tilde{O}(\sqrt{n})$

For $\sqrt{n}$ landmarks nodes:

- Memory: $\tilde{O}(\sqrt{n})$
- Stretch: 3 at most
- Similar to the scheme by Thorup & Zwick [TZ'01]

Labeled compact routing scheme



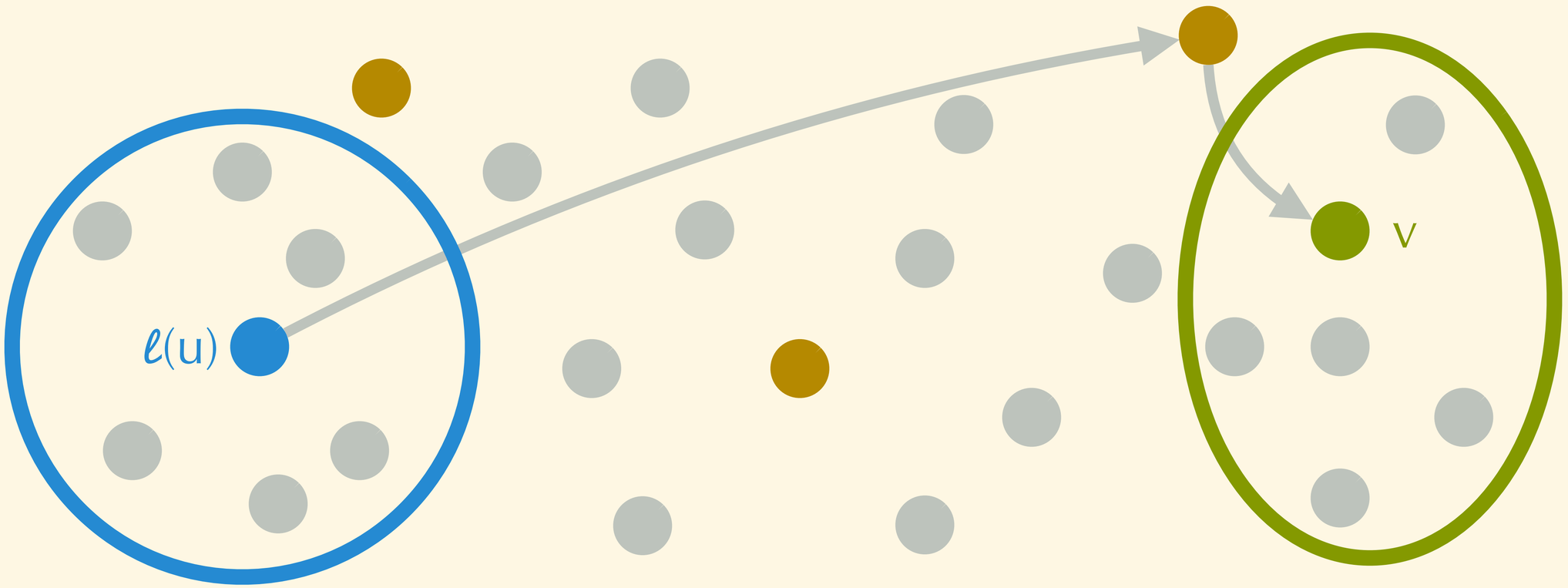
For $\sqrt{n}$ landmarks nodes:

- Memory: $\tilde{O}(\sqrt{n})$
- Stretch: 3 at most
- Similar to the scheme by Thorup & Zwick [TZ'01]

HOW TO BUILD A ROUTING SCHEME

PART III : From Labeled to Name-independent routing

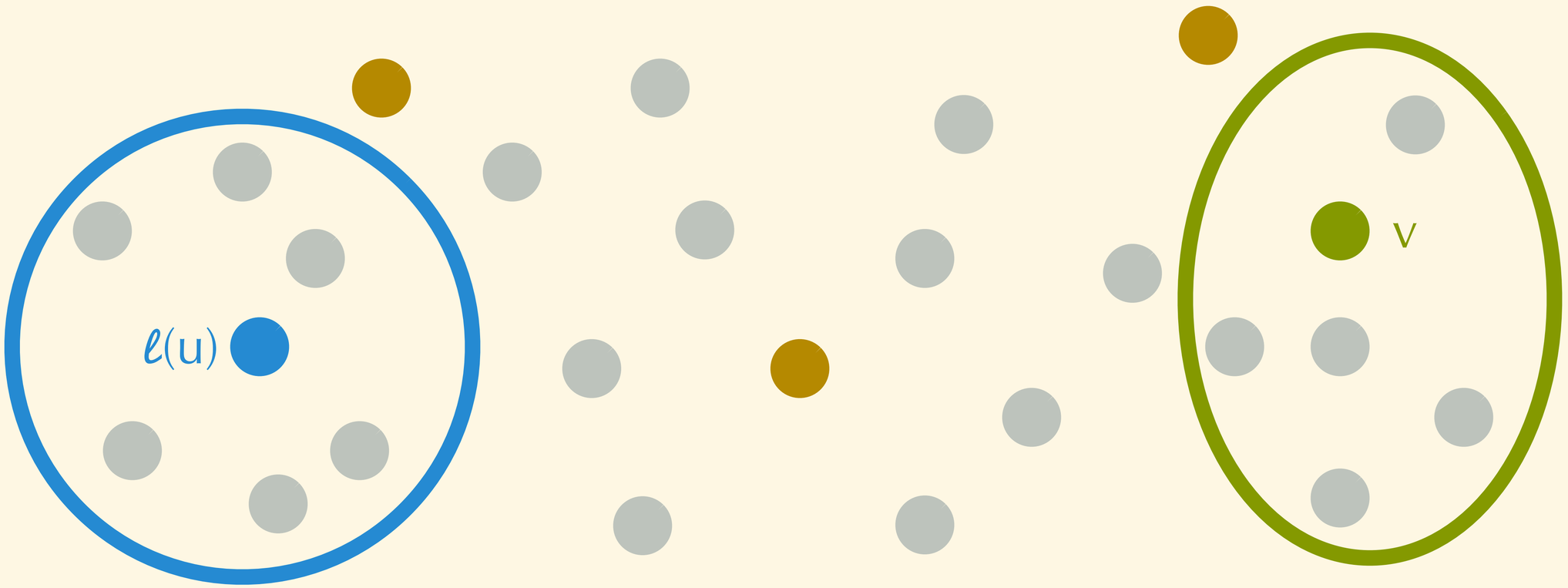
Labeled compact routing scheme



For $\sqrt{n}$ landmarks nodes:

- Memory: $\tilde{O}(\sqrt{n})$
- Stretch: 3 at most
- Similar to the scheme by Thorup & Zwick [TZ'01]

Labeled compact routing scheme



For $\sqrt{n}$ landmarks nodes:

- Memory: $\tilde{O}(\sqrt{n})$
- Stretch: 3 at most
- Similar to the scheme by Thorup & Zwick [TZ'01]

Name-independent vs. Labeled

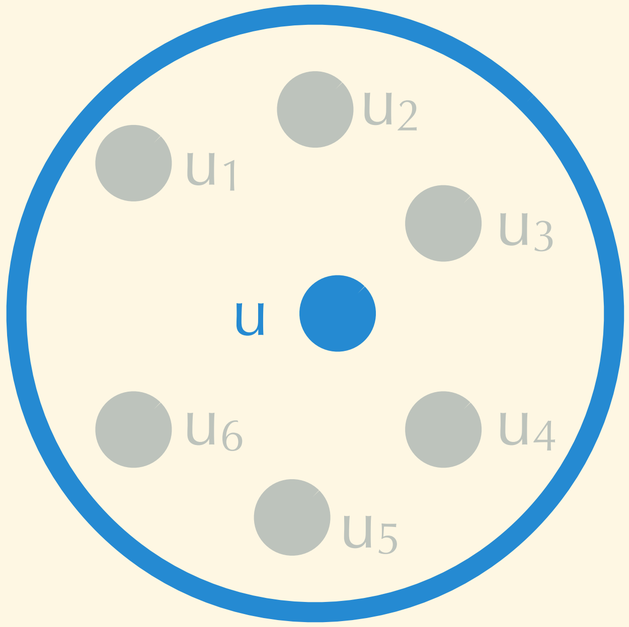
How to adapt this algorithm in the name-independent model ?

- When routing from u to v, if $\ell(u)$ and $\ell(v)$ can be retrieved then we are done.
  1. every node u could store its own label $\ell(u)$ (constant memory)
  2. every node u cannot store every destination label otherwise memory would me $O(n)$.

- The routing label $\ell(v)$ can be retrieved using local collaboration amongst vicinity nodes.

Name-independent vs. Labeled

Vicinity label-sharing

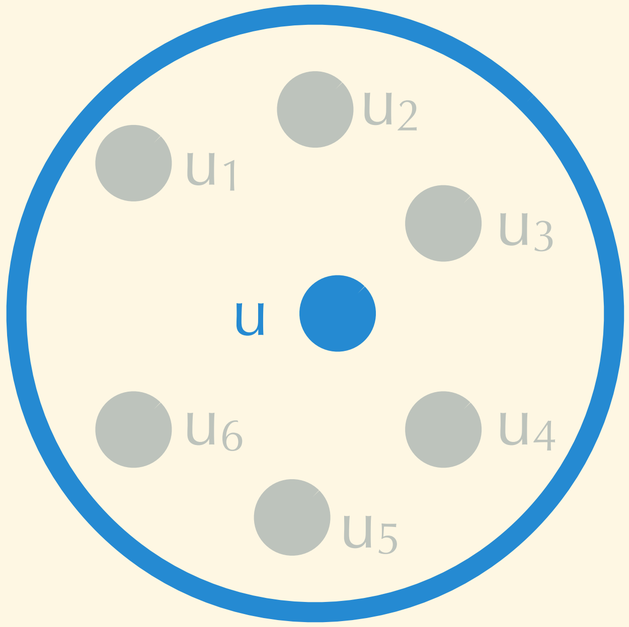


all n labels

Identifier	compact-routing-label
v_1	$\ell(v_1)$
v_2	$\ell(v_2)$
v_3	$\ell(v_3)$
v_4	$\ell(v_4)$
…	…
$v_i = v$	$\ell(v)$
…	…
v_n	$\ell(v_n)$

Name-independent vs. Labeled

Vicinity label-sharing



all n labels

Identifier	compact-routing-label
v_1	$\ell(v_1)$
v_2	$\ell(v_2)$
v_3	$\ell(v_3)$
v_4	$\ell(v_4)$
…	…
$v_i = v$	$\ell(v)$
…	…
v_n	$\ell(v_n)$

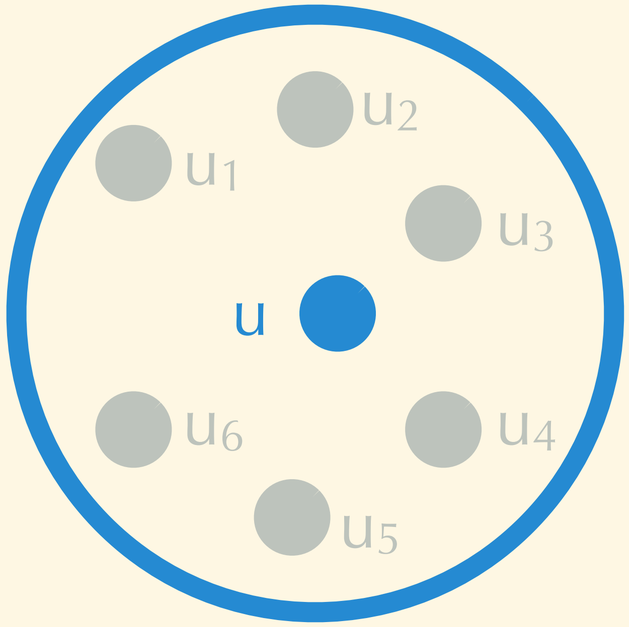
hash function $n \rightarrow \sqrt{n}$

Identifier	compact-routing-label
v_2	$\ell(v_2)$
v_3	$\ell(v_3)$
v_i	$\ell(v_i)$

Identifier	compact-routing-label
v_1	$\ell(v_1)$
v_4	$\ell(v_4)$
v_9	$\ell(v_9)$

Name-independent vs. Labeled

Vicinity label-sharing



all n labels

Identifier	compact-routing-label
v_1	$\ell(v_1)$
v_2	$\ell(v_2)$
v_3	$\ell(v_3)$
v_4	$\ell(v_4)$
…	…
$v_i = v$	$\ell(v)$
…	…
v_n	$\ell(v_n)$

hash function
$n \rightarrow \sqrt{n}$

u_6

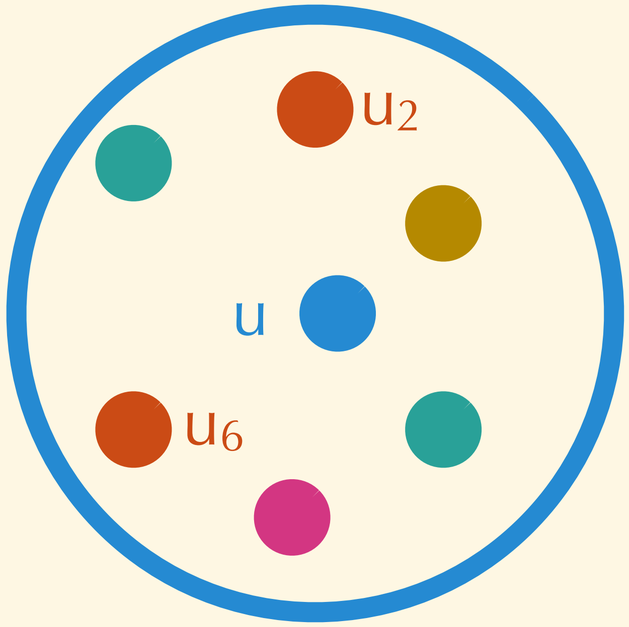
Identifier	compact-routing-label
v_2	$\ell(v_2)$
v_3	$\ell(v_3)$
v_i	$\ell(v_i)$

u

Identifier	compact-routing-label
v_1	$\ell(v_1)$
v_4	$\ell(v_4)$
v_9	$\ell(v_9)$

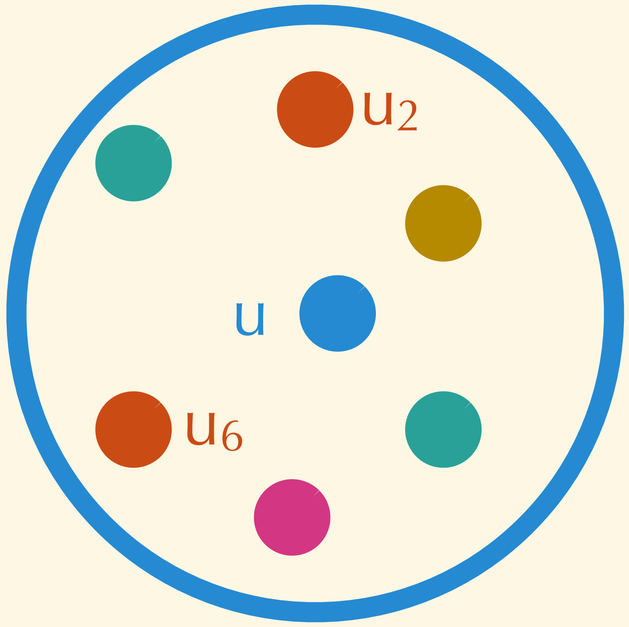
Name-independent vs. Labeled

How to adapt this algorithm in the name-independent model ?



Name-independent vs. Labeled

How to adapt this algorithm in the name-independent model ?

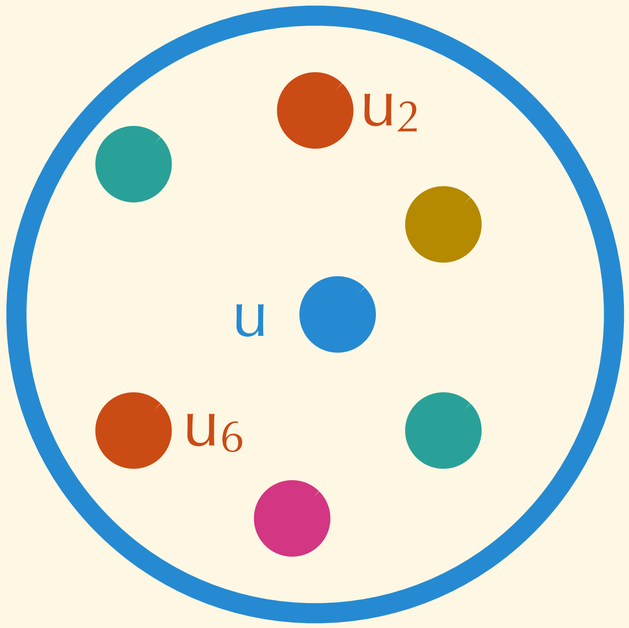


all labels

Identifier	compact-routing-label
v_1	$\ell(v_1)$
v_2	$\ell(v_2)$
v_3	$\ell(v_3)$
v_4	$\ell(v_4)$
…	…
$v_i = v$	$\ell(v)$
…	…
v_n	$\ell(v_n)$

Name-independent vs. Labeled

How to adapt this algorithm in the name-independent model ?



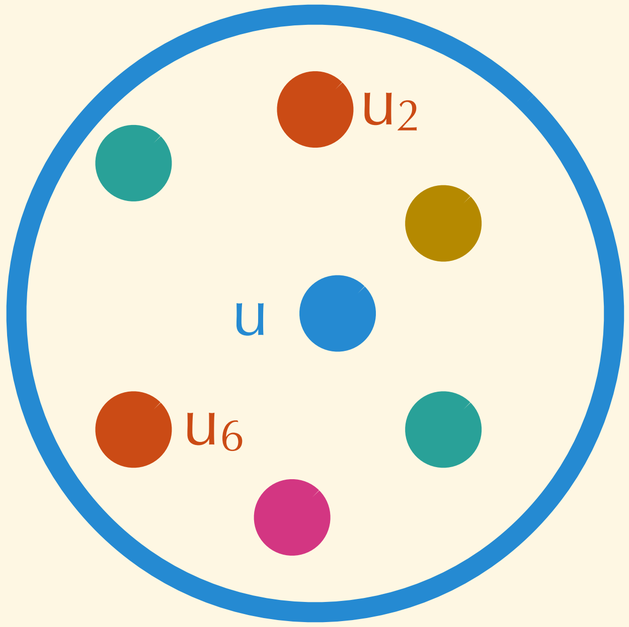
all labels

Identifier	compact-routing-label
v_1	$\ell(v_1)$
v_2	$\ell(v_2)$
v_3	$\ell(v_3)$
v_4	$\ell(v_4)$
…	…
$v_i = v$	$\ell(v)$
…	…
v_n	$\ell(v_n)$

- Moreover, in its vicinity routing table, node u stores the colours of the nodes.

Name-independent vs. Labeled

How to adapt this algorithm in the name-independent model ?



all labels

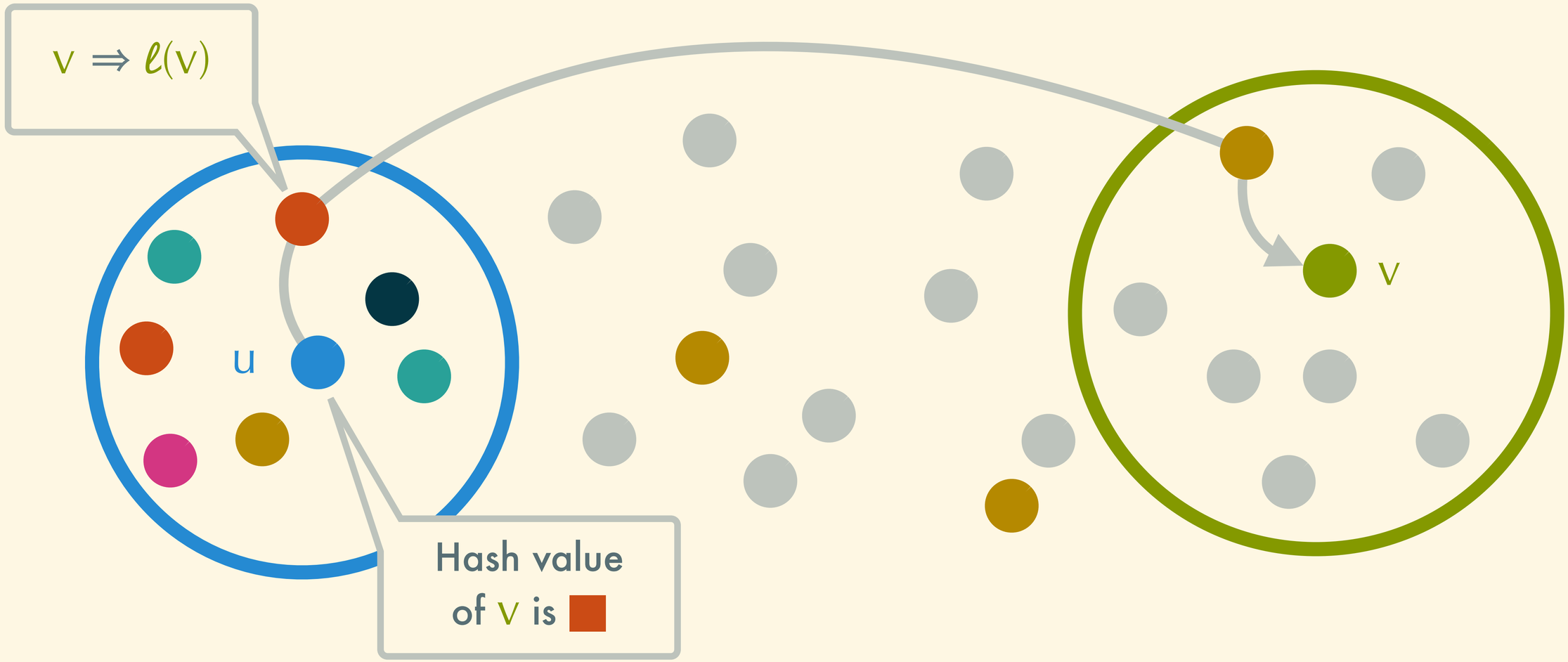
Identifier	compact-routing-label
v_1	$\ell(v_1)$
v_2	$\ell(v_2)$
v_3	$\ell(v_3)$
v_4	$\ell(v_4)$
…	…
$v_i = v$	$\ell(v)$
…	…
v_n	$\ell(v_n)$

- Moreover, in its vicinity routing table, node u stores the colours of the nodes.
- The memory used for label sharing is #labels/#colors, for example it can be:

$$n/\sqrt{n} = \sqrt{n}$$

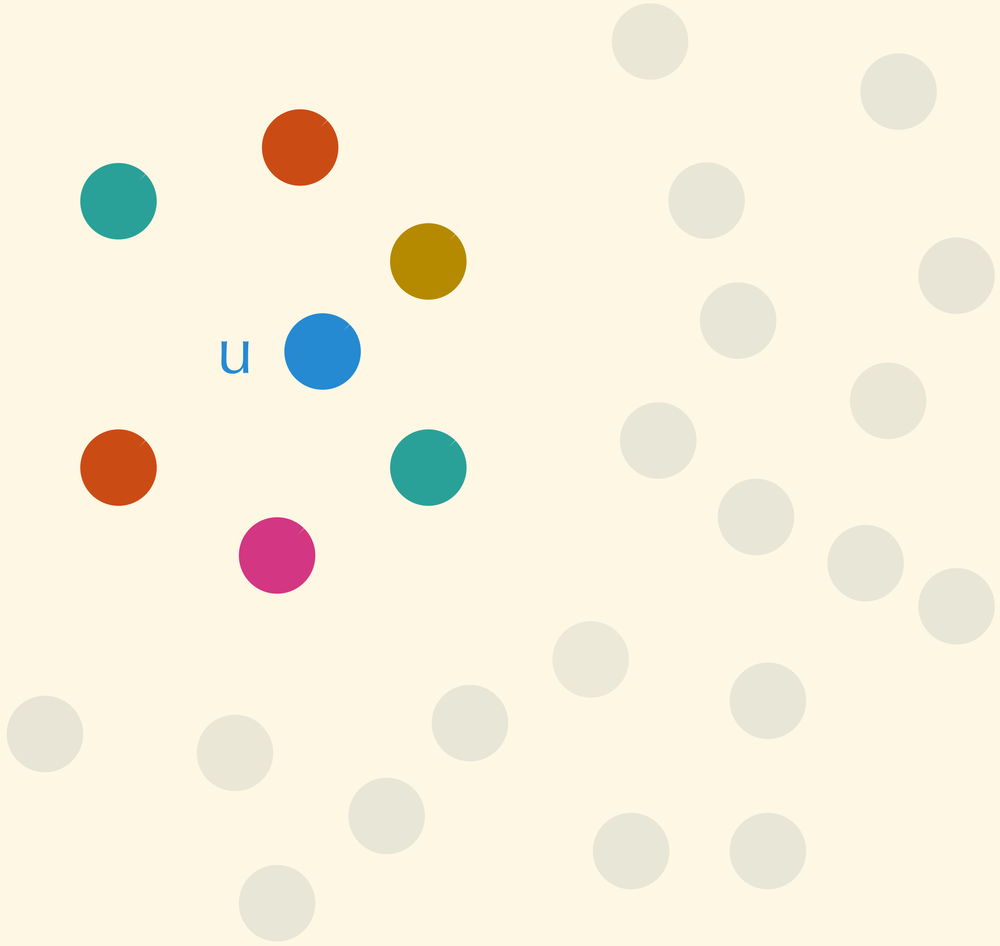
Name-independent vs. Labeled

Routing



Name-independent vs. Labeled

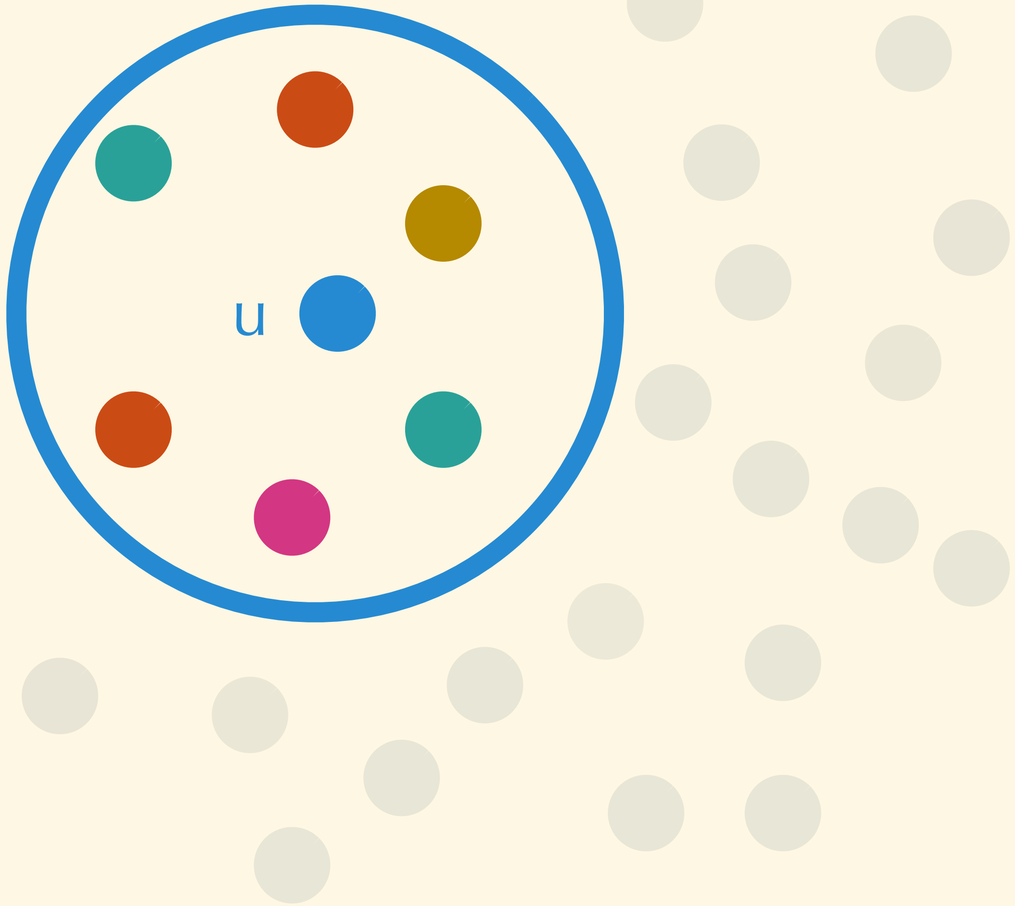
Memory



- Every node u stores two types of entries:

Name-independent vs. Labeled

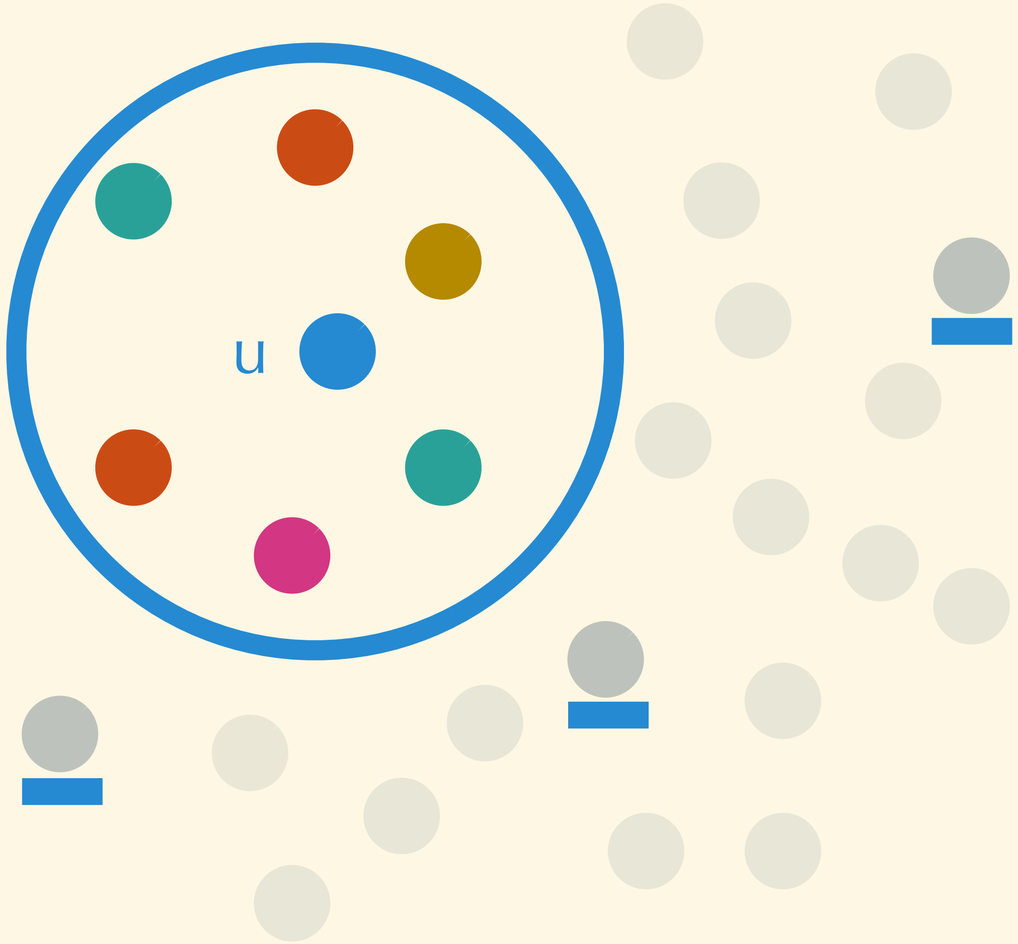
Memory



- Every node u stores two types of entries:
  - Next-hop and color of nodes in its vicinity ball, ie., $\tilde{O}(\sqrt{n})$ entries.

Name-independent vs. Labeled

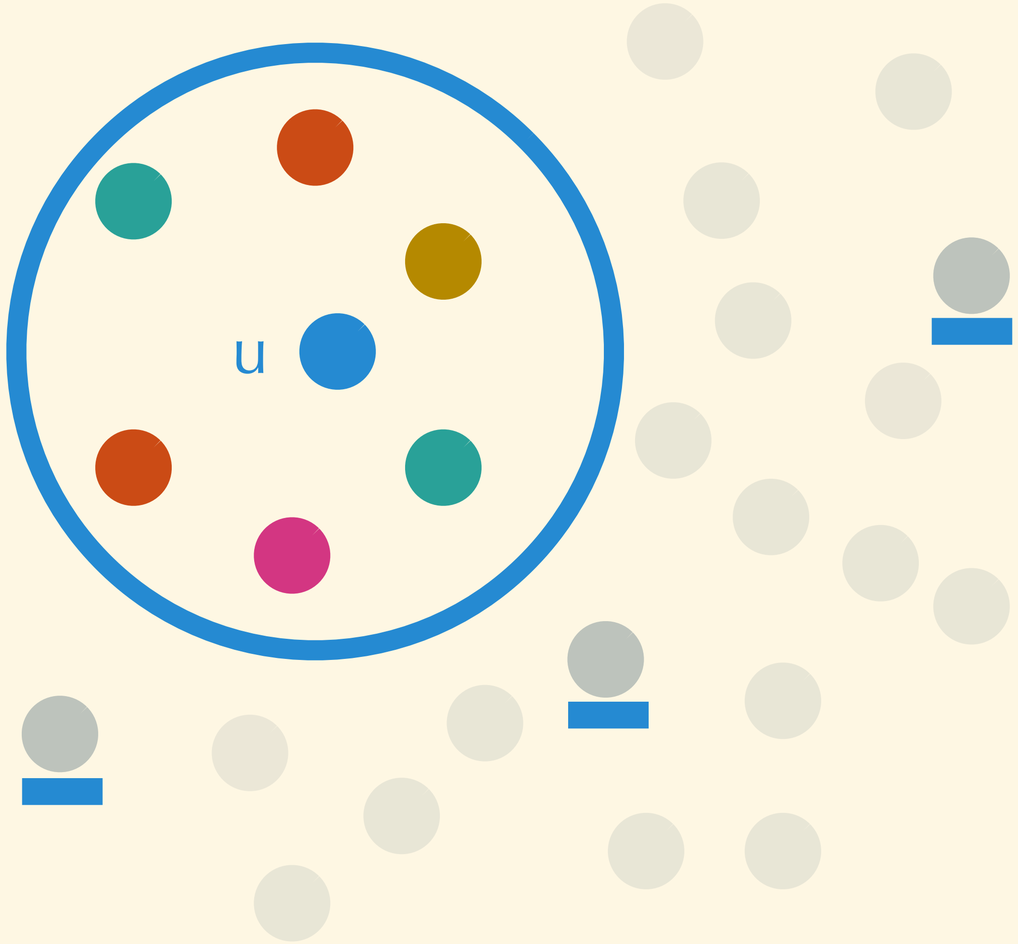
Memory



- Every node u stores two types of entries:
  - Next-hop and color of nodes in its vicinity ball, ie., $\tilde{O}(\sqrt{n})$ entries.
  - Compact routing labels of nodes whose hash value is blue, ie., $\tilde{O}(\sqrt{n})$ entries.

Name-independent vs. Labeled

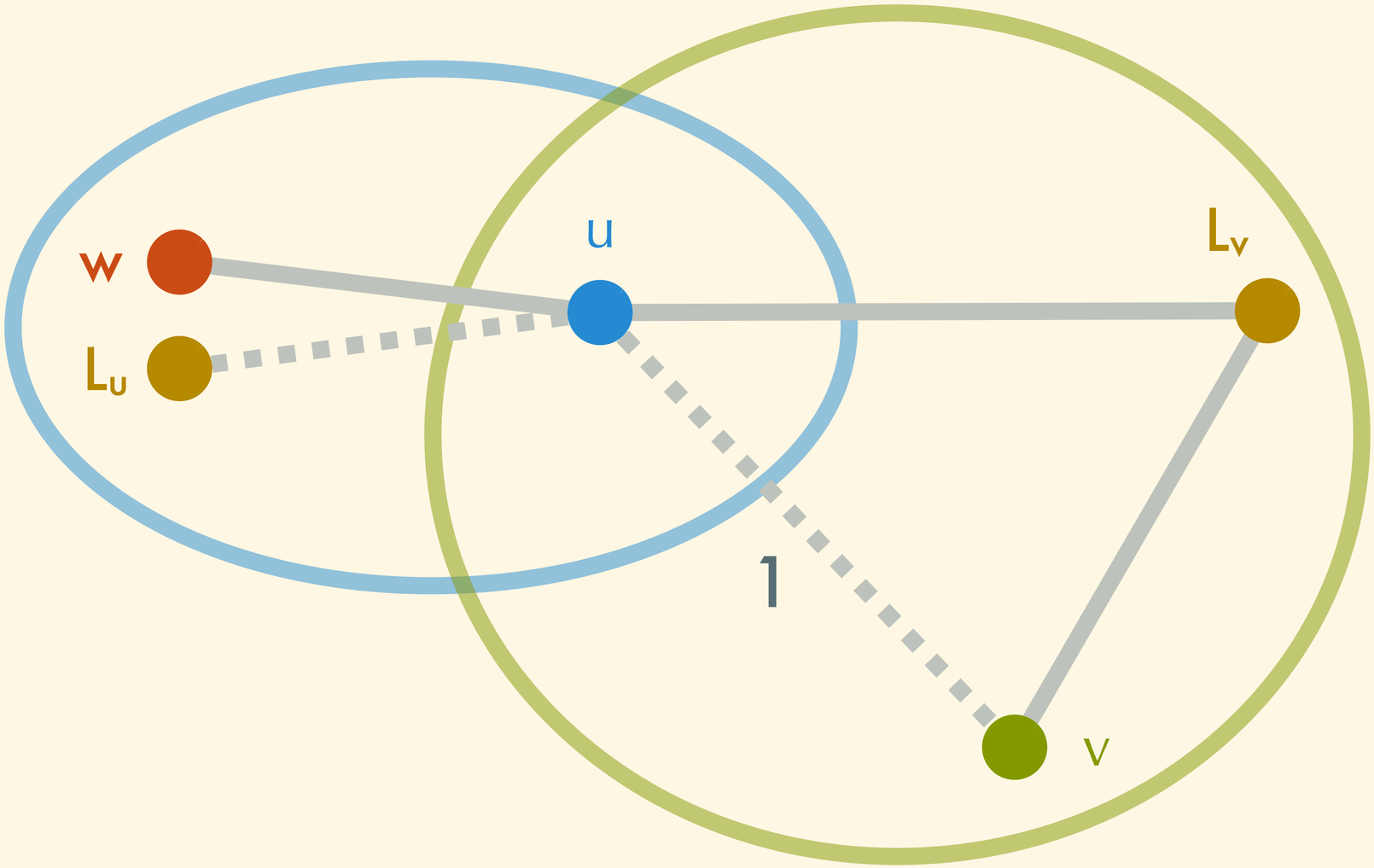
Memory



- Every node u stores two types of entries:
  - Next-hop and color of nodes in its vicinity ball, ie., $\tilde{O}(\sqrt{n})$ entries.
  - Compact routing labels of nodes whose hash value is blue, ie., $\tilde{O}(\sqrt{n})$ entries.
- Therefore, the total memory per node is $\tilde{O}(\sqrt{n})$

Name-independent vs. Labeled

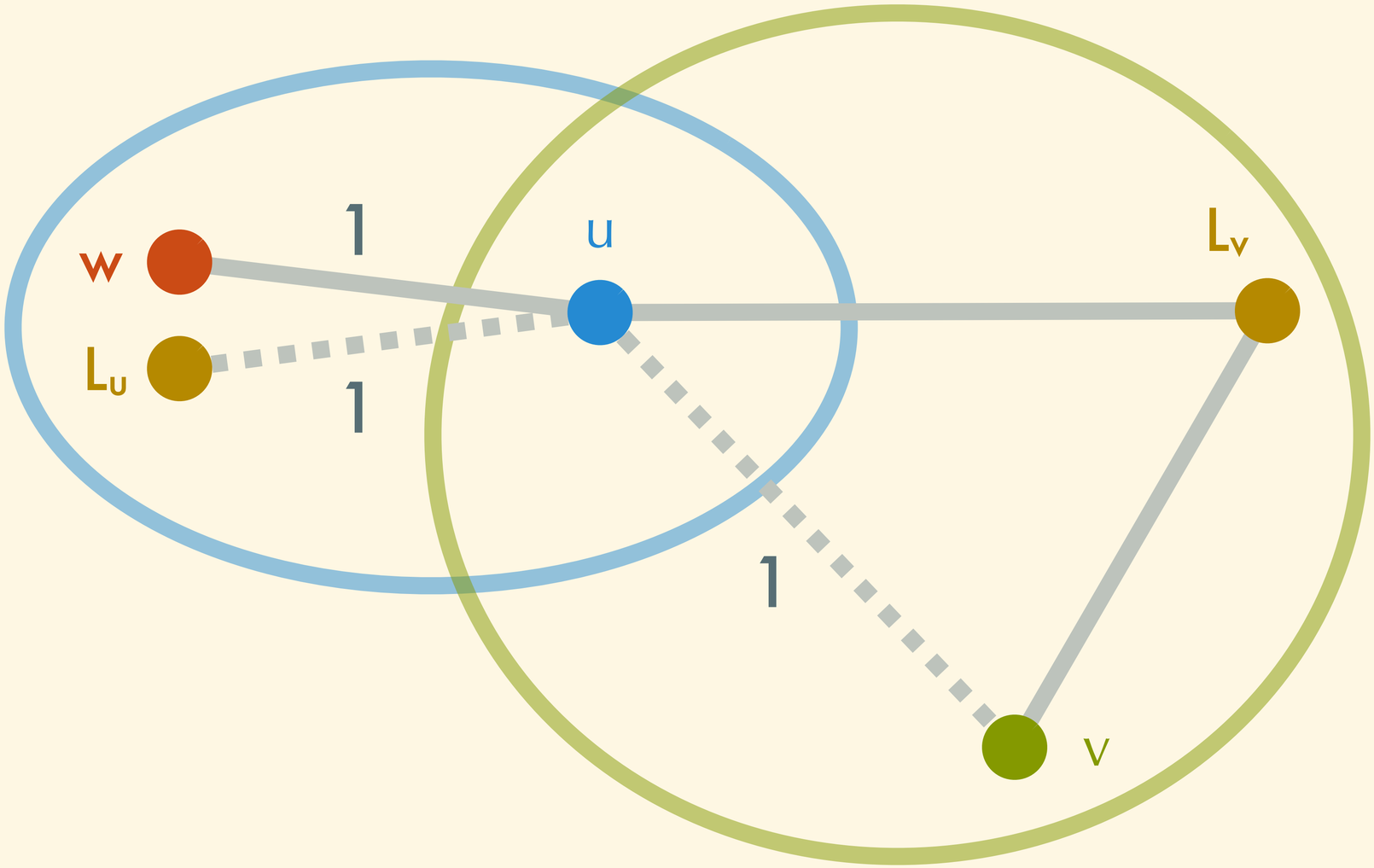
Stretch factor



- Hyp.: d(u,v) = 1

Name-independent vs. Labeled

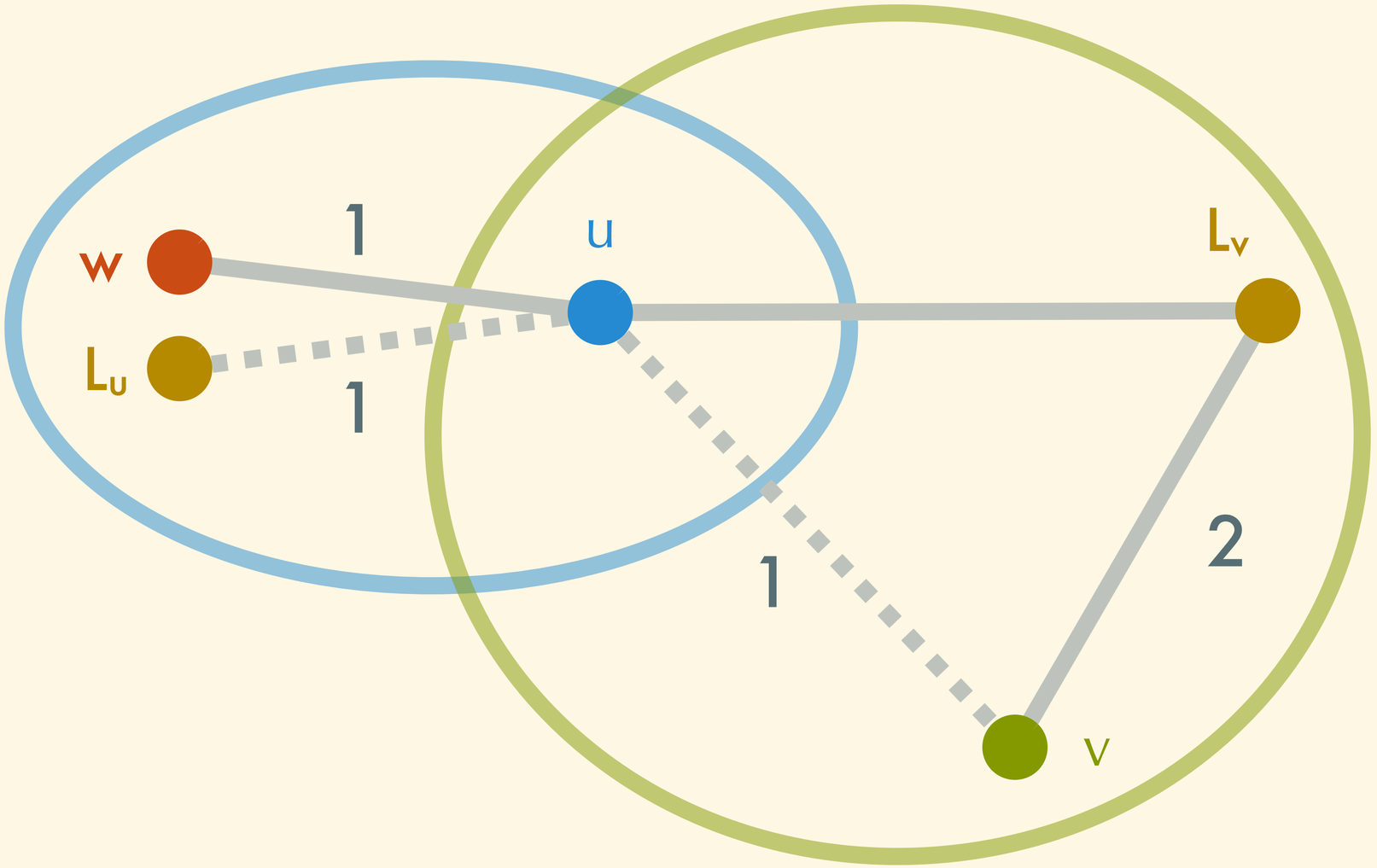
Stretch factor



- Hyp.: d(u,v) = 1
- ⇒ Radius ≤ 1

Name-independent vs. Labeled

Stretch factor

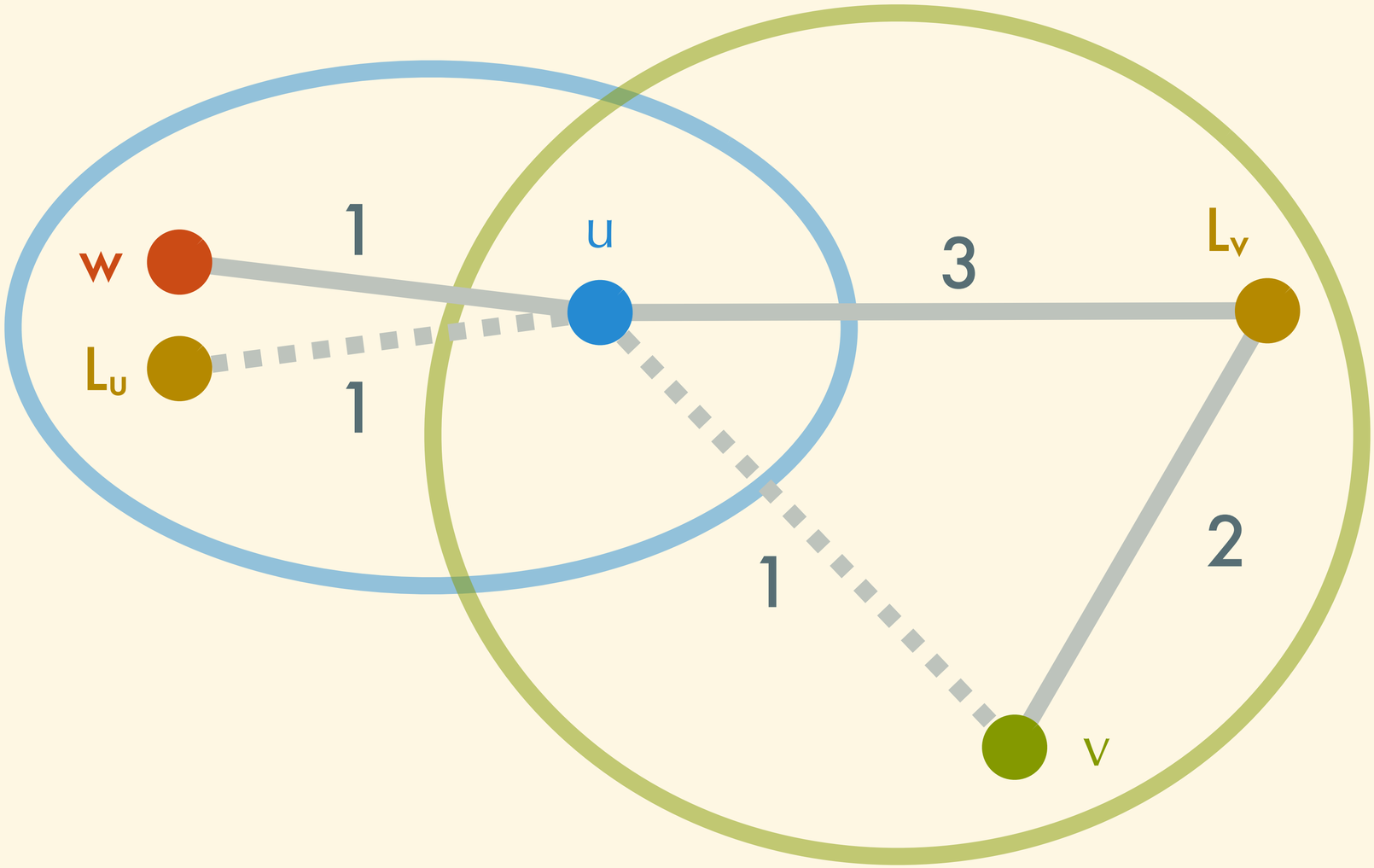


- Hyp.: d(u,v) = 1
- ⇒ Radius ≤ 1

- $d(v,L_v) \leq d(v,L_u) \leq 2$

Name-independent vs. Labeled

Stretch factor

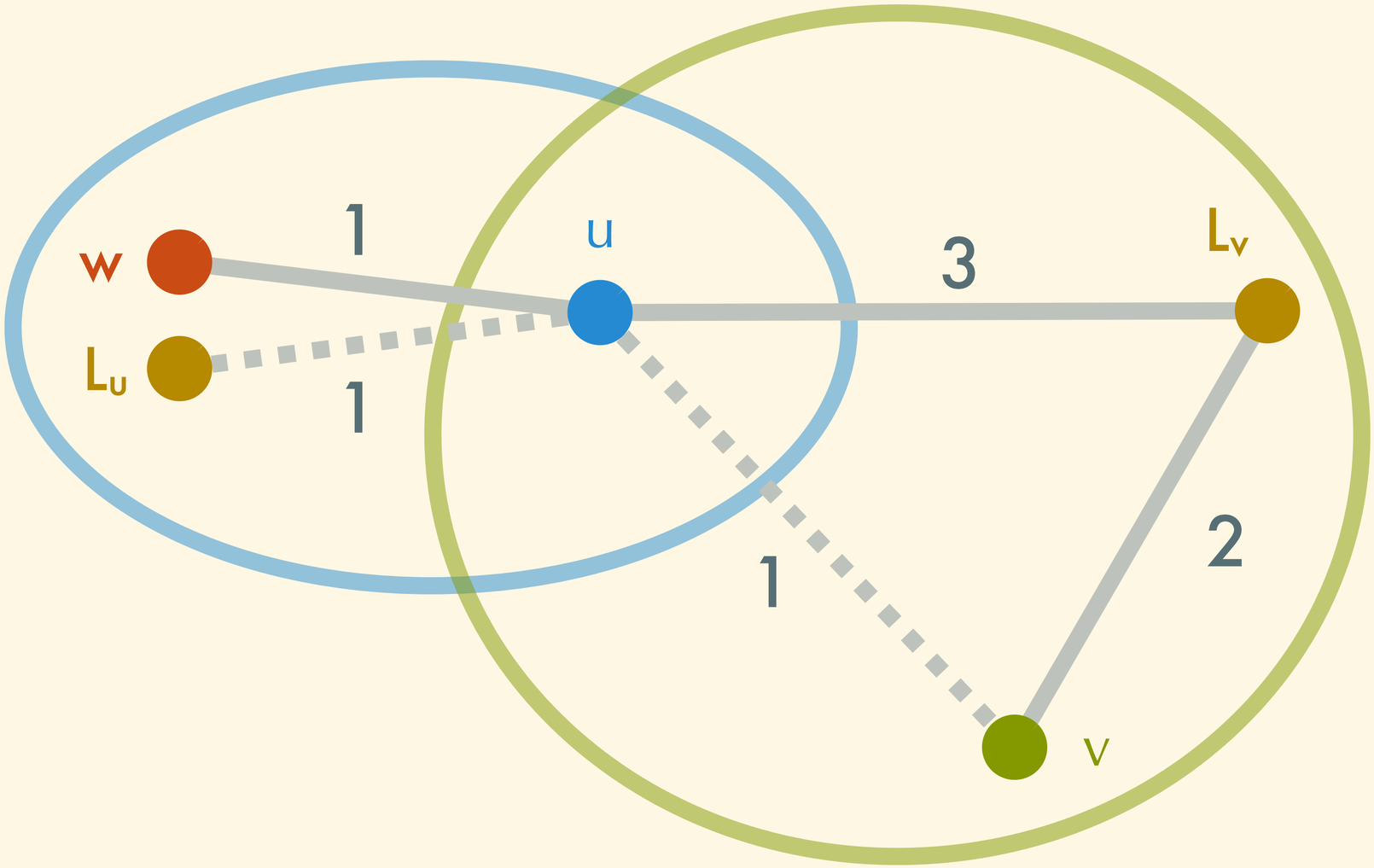


- Hyp.: d(u,v) = 1
- ⇒ Radius ≤ 1

- $d(v,L_v) \le d(v,L_u) \le 2$
- $d(u,Lv) \le d(v,Lv)+d(u,v) \le 3$

Name-independent vs. Labeled

Stretch factor



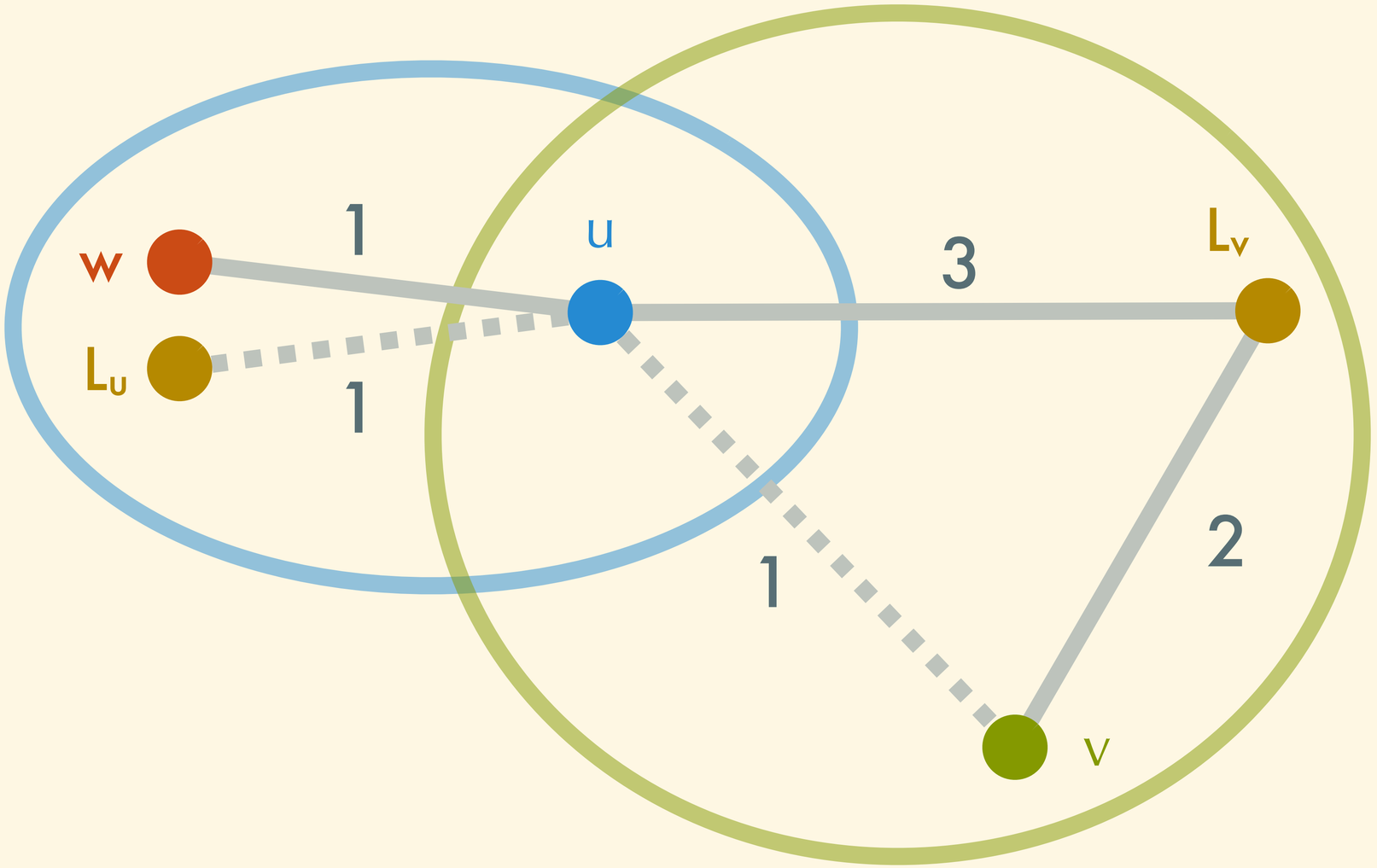
- Hyp.: d(u,v) = 1
- ⇒ Radius ≤ 1

- $d(v,L_v) \le d(v,L_u) \le 2$
- $d(u,Lv) \le d(v,Lv)+d(u,v) \le 3$

- "u ⇝ w ⇝ u ⇝ Lv ⇝ v ≤ 7"

Name-independent vs. Labeled

Stretch factor



- Hyp.: d(u,v) = 1
- ⇒ Radius ≤ 1

- $d(v,L_v) \le d(v,L_u) \le 2$
- $d(u,Lv) \le d(v,Lv)+d(u,v) \le 3$

- "u ⇝ w ⇝ u ⇝ Lv ⇝ v ≤ 7"
- Therefore the stretch is ≤ 7.

PERFORMANCES

	Stretch		Memory	Communication cost
[AGM'08]	3	optimal	$\tilde{O}(\sqrt{n})$	centralised
→ [GGHI'13]	7		$\tilde{O}(\sqrt{n})$	$O(m\sqrt{n})$
my thesis	5		$\tilde{O}(\sqrt{n})$	$O(m\sqrt{n})$
To be done (hard)	3		$\tilde{O}(\sqrt{n})$	$o(mn)$

- If the route use the best landmark from $\{L_u, L_v\}$ the stretch can be improved to 5.
- This can even be done in a distributed way without any impact on the asymptotic complexities.

COMPACT ROUTING FOR INTERNET-LIKE GRAPHS

Internet-like graphs

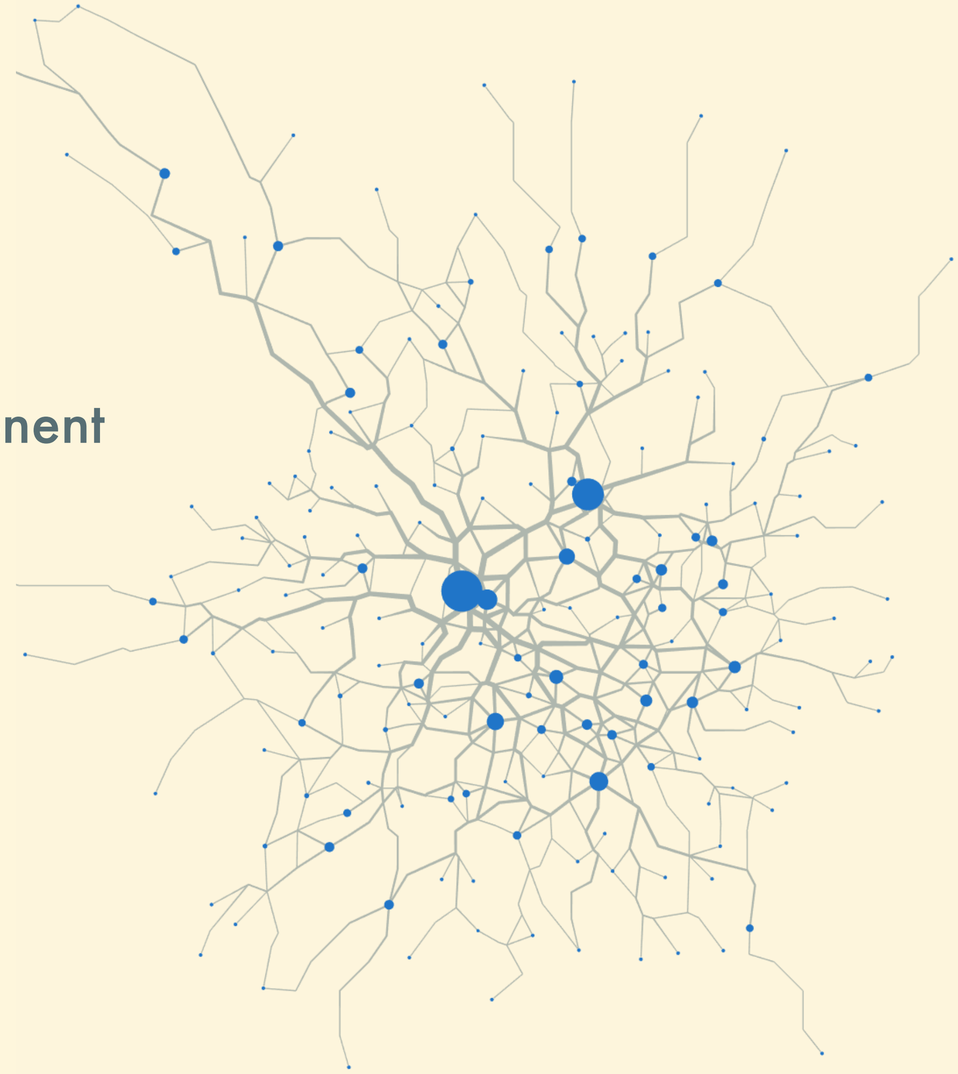
- Power-law degree distribution
  - Sparse graph (logarithmic average degree)
  - Low diameter (also logarithmic)
- Looks like a tree with a dense cluster as a root
- Mimicked by a synthetic graph model, called RPLG:
  - Random Power Law Graph
  - Parametrised by the power-law exponent, t

INTERNET-LIKE GRAPHS

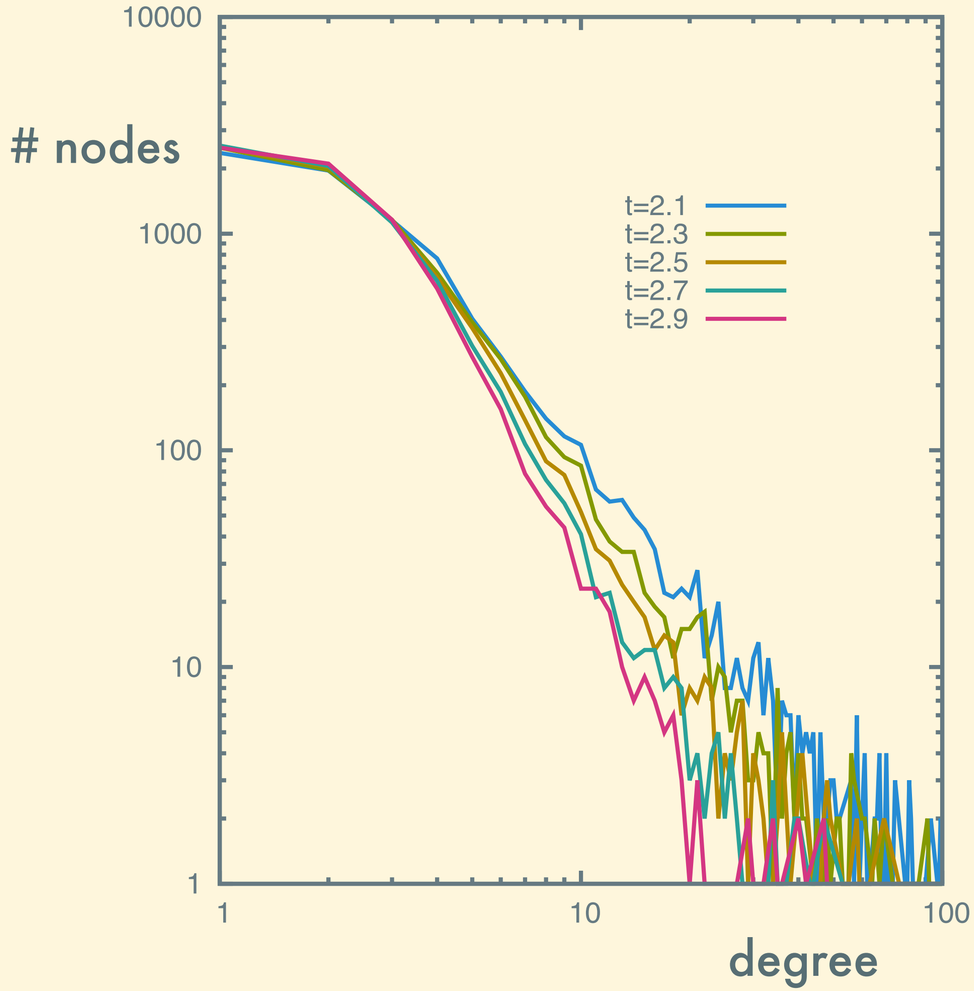
(RPLG)

Power-law exponent

t = 2.1

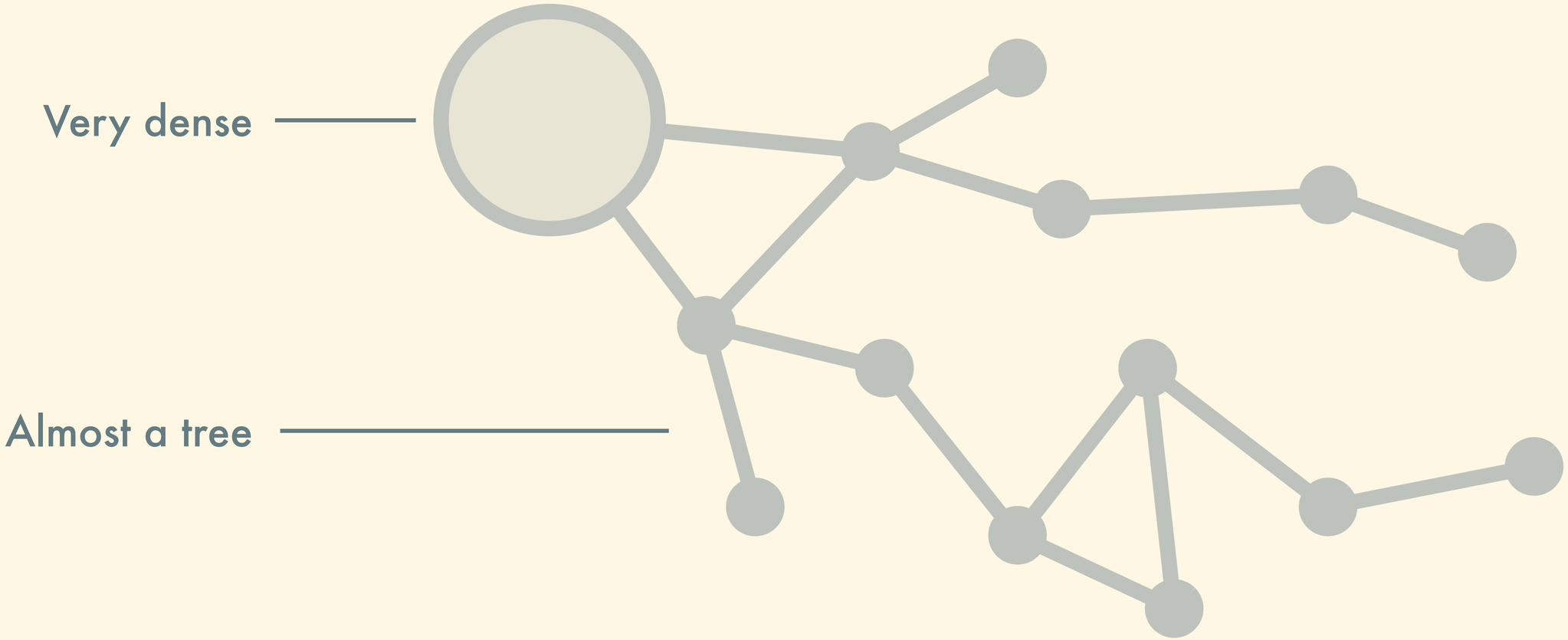


degree distribution



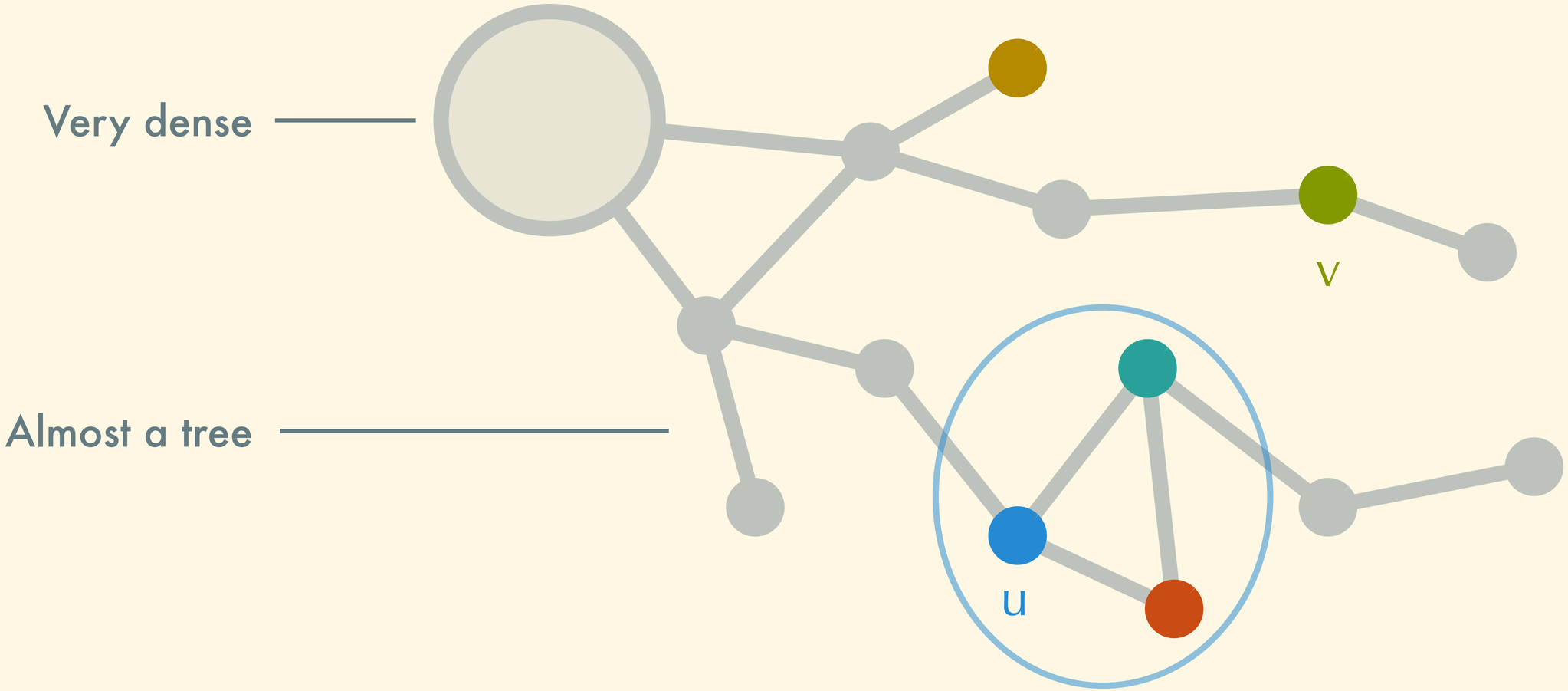
Improving memory performances

How to use the structure to save some entries?



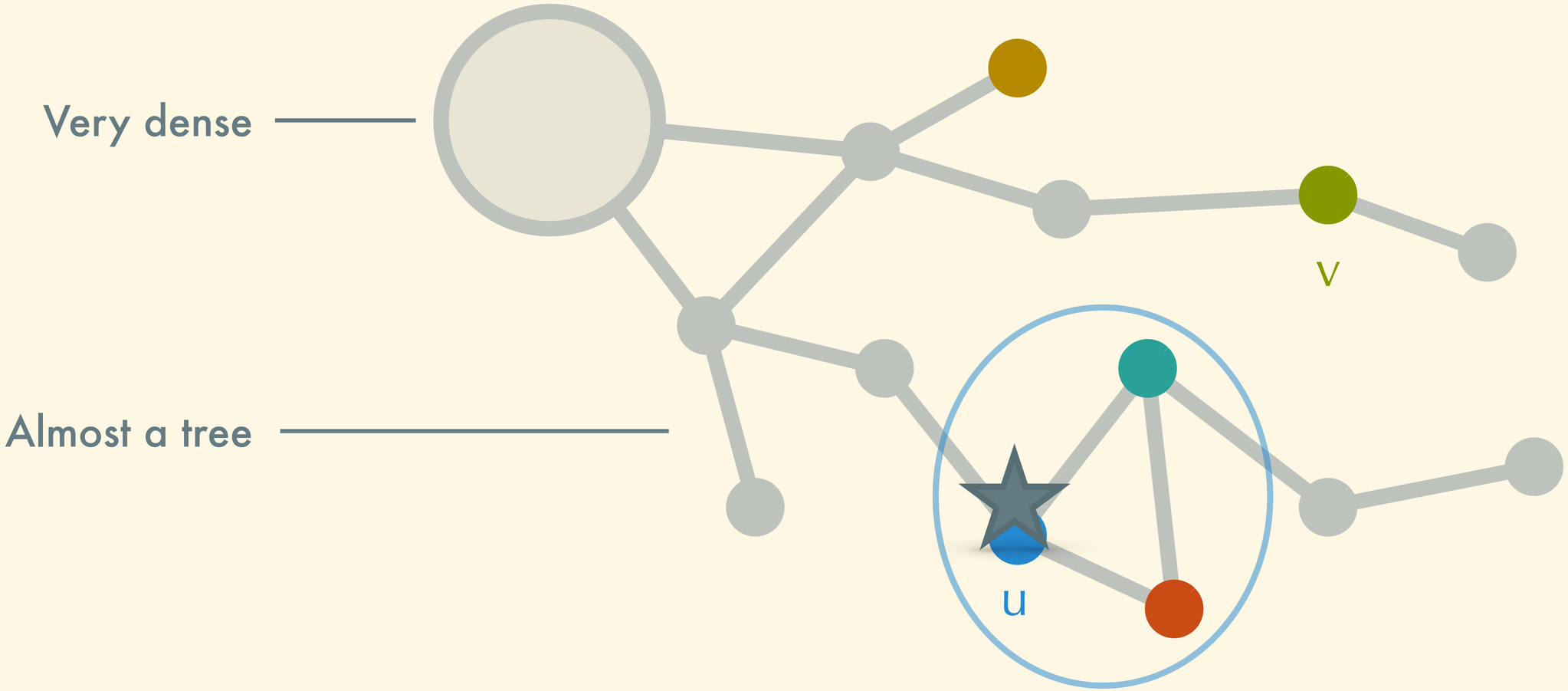
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



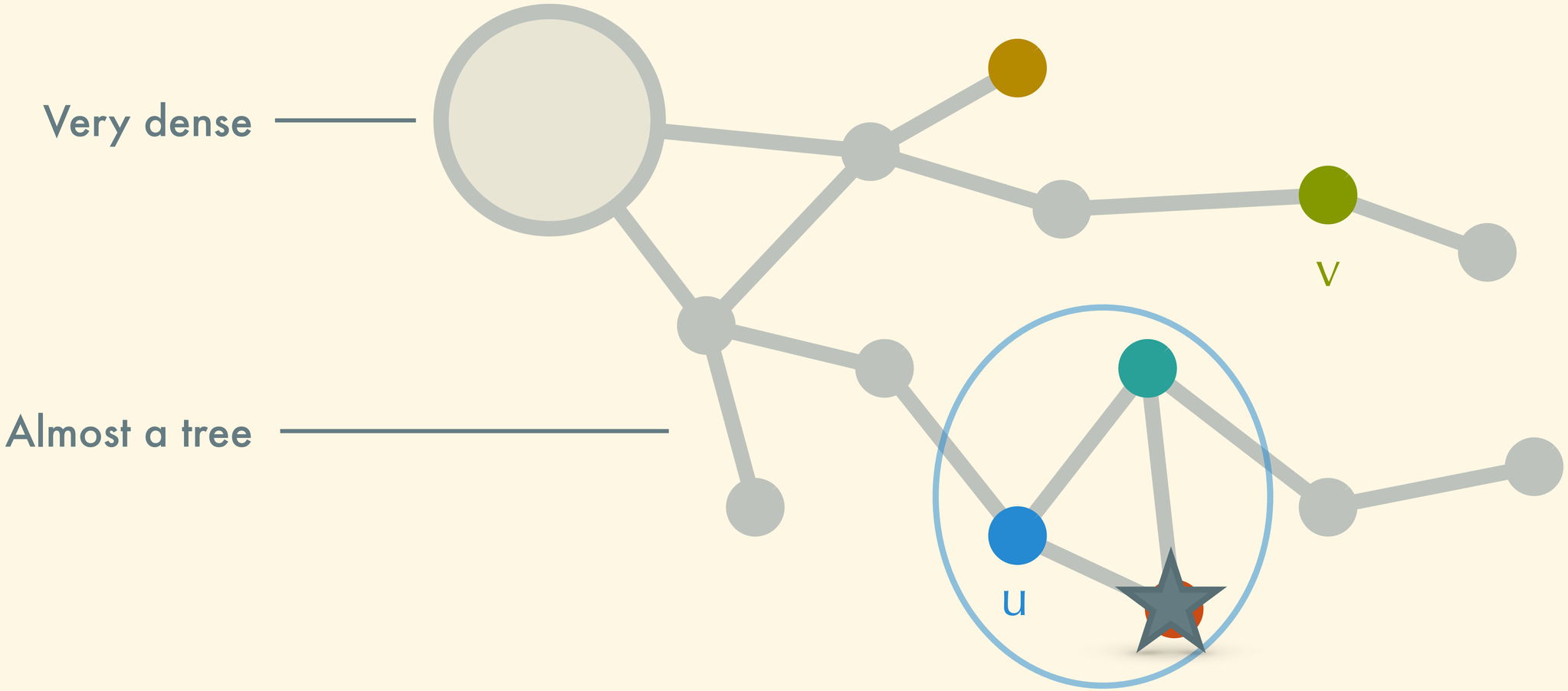
Improving memory performances

How to use the structure to save some entries?



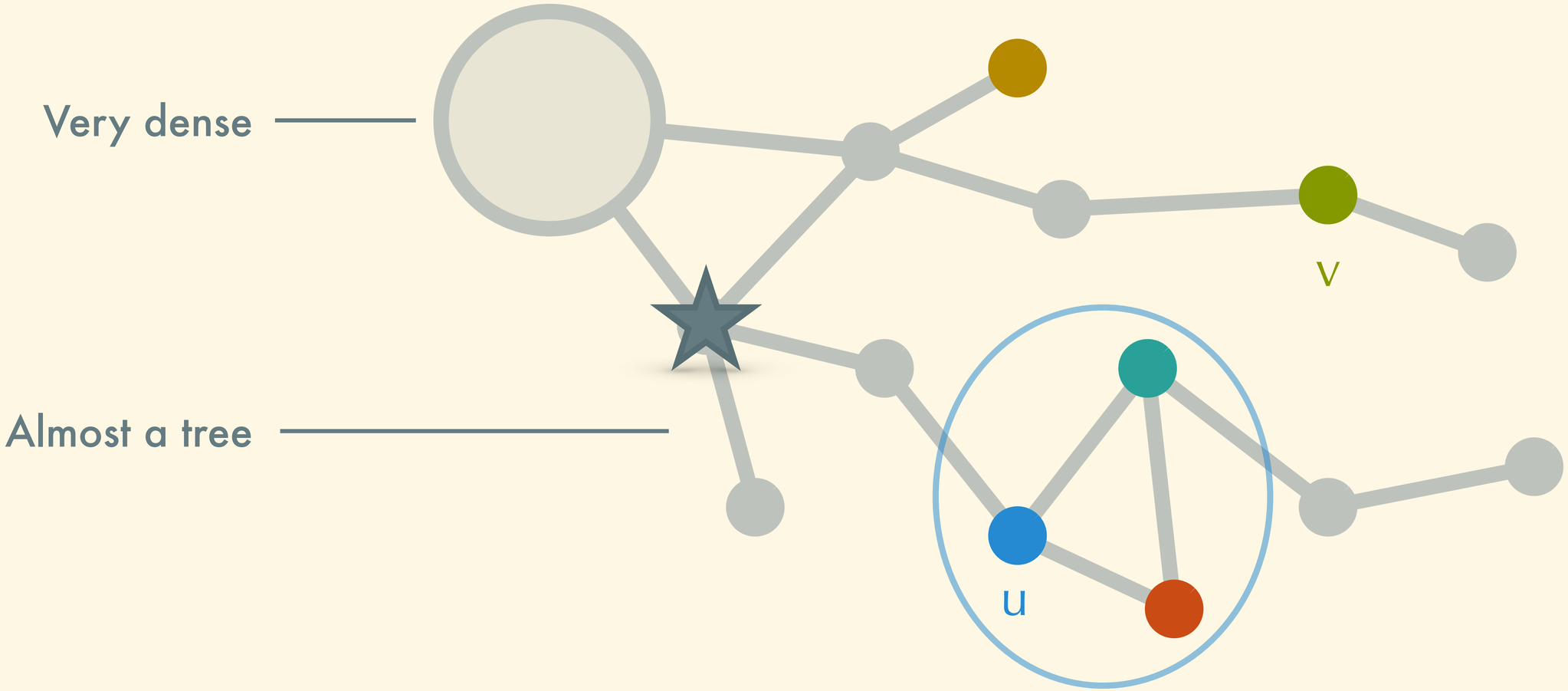
Improving memory performances

How to use the structure to save some entries?



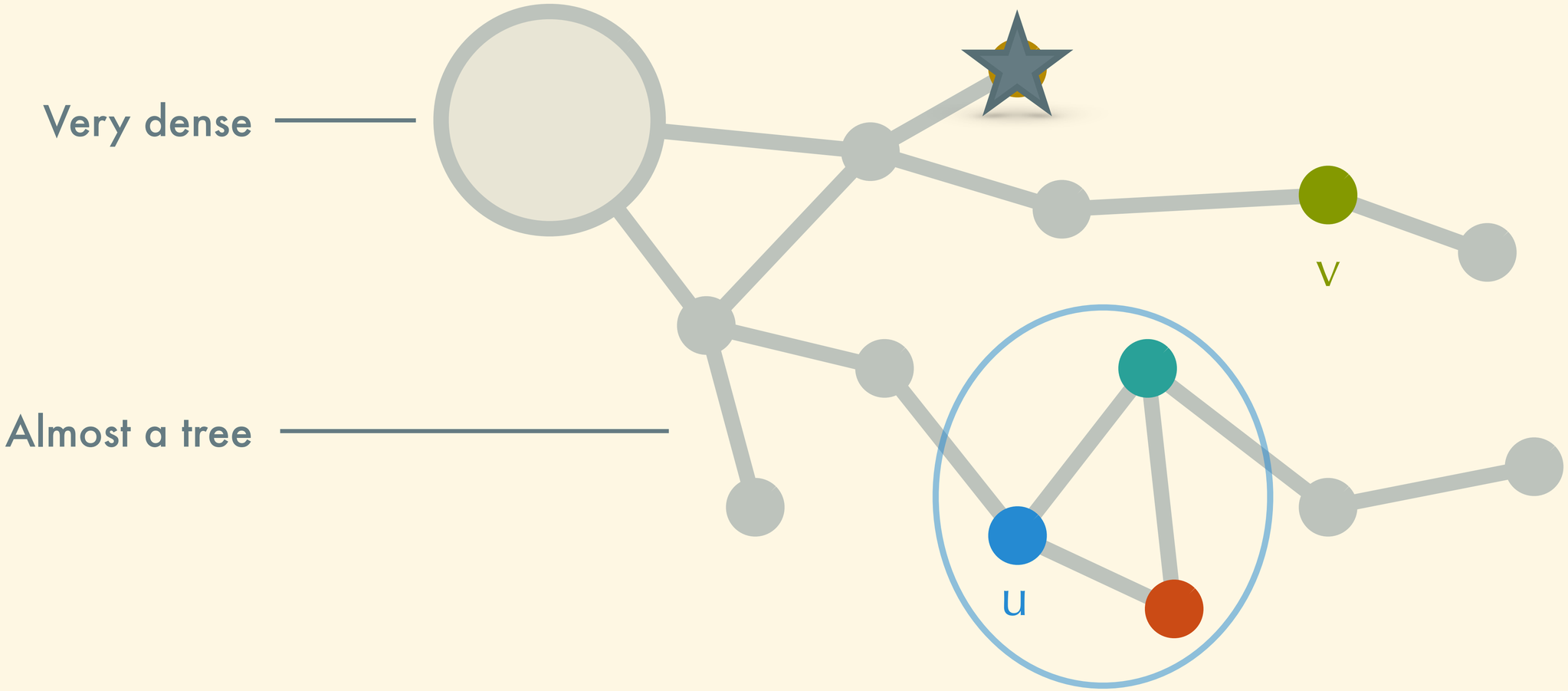
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



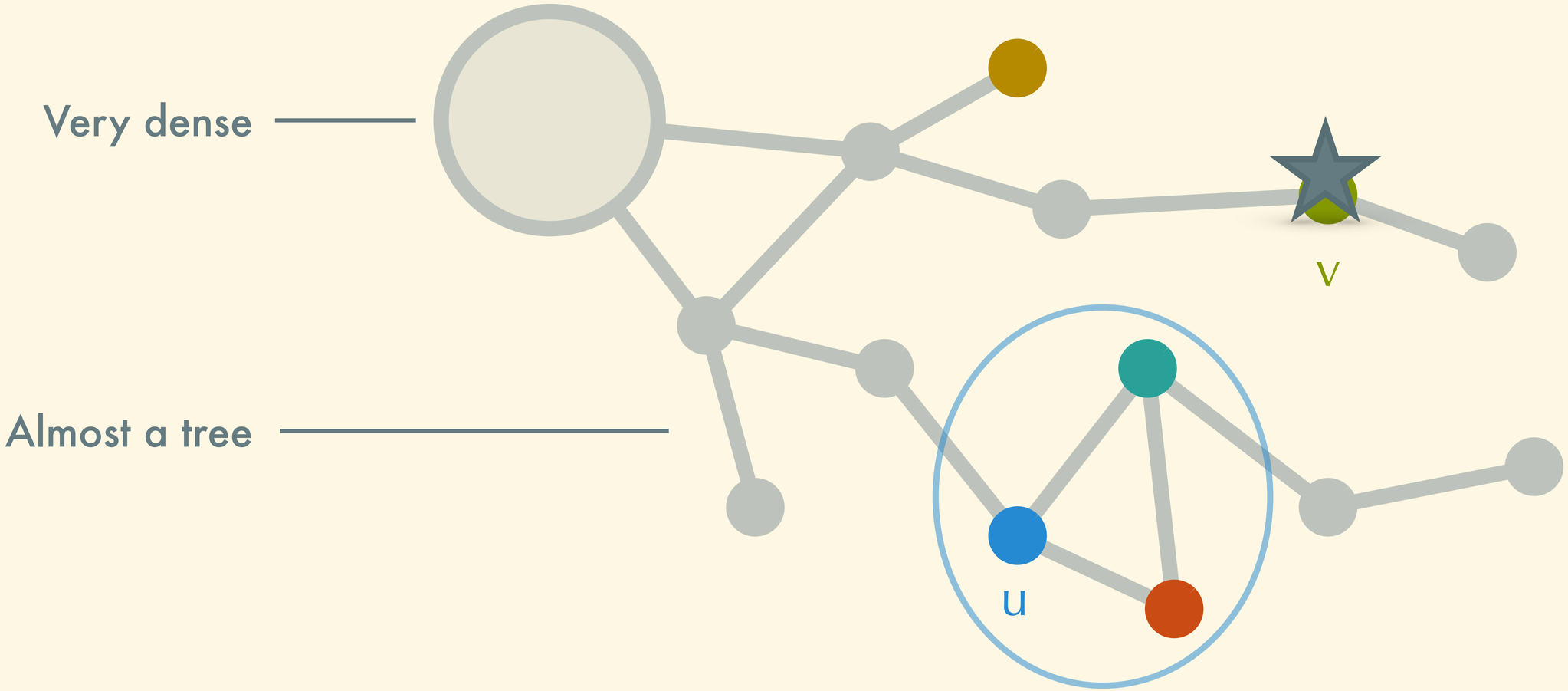
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



Improving memory performances

How to use the structure to save some entries?

- We are actually very used in routing with these settings, lets think about this
  - Low memory
  - Low stretch
  - Dense center
  - Very sparse outside the center
  - Most of the destinations are outside the center

Improving memory performances

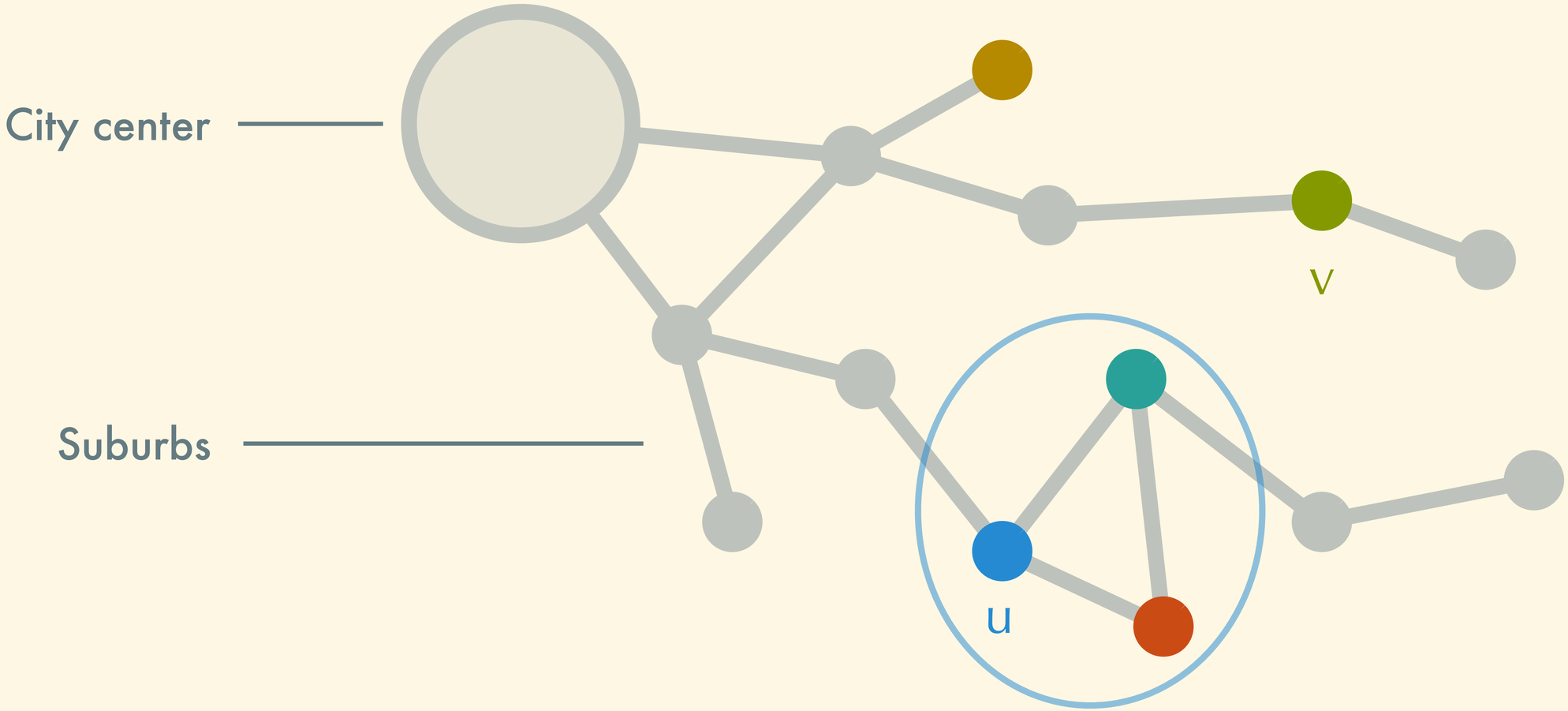
How to use the structure to save some entries?

- We are actually very used in routing with these settings, lets think about Public transportation
  - Low memory - New city “there’s no one around and your phone is dead”*
  - Low stretch - Its holidays, you can’t wait in transports
  - Dense center - City center is easily accessible
  - Very sparse outside the center - Suburban public transportation …
  - Most of the destinations are outside the center - Every analogy limps …

*Rob Cantor

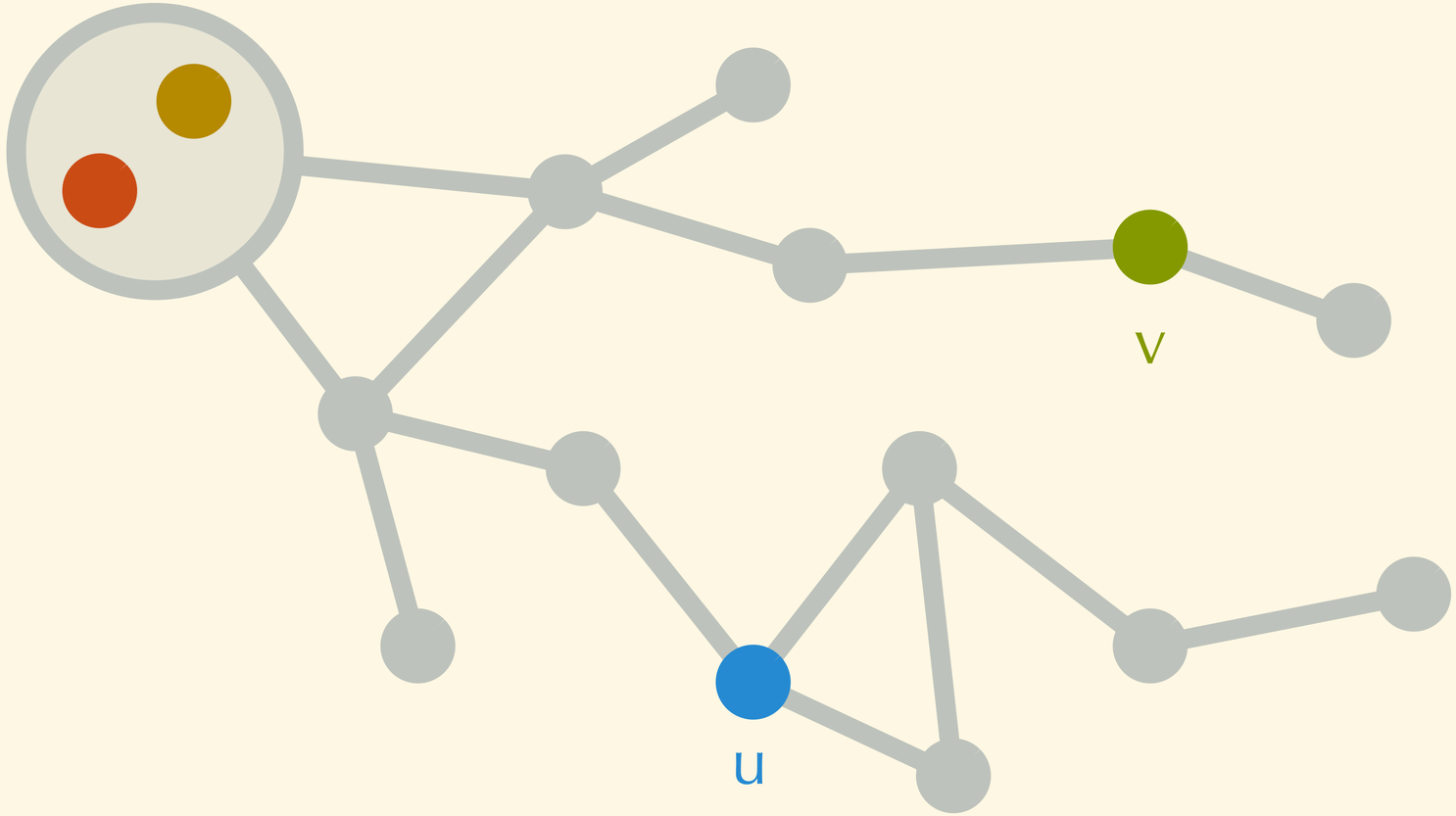
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



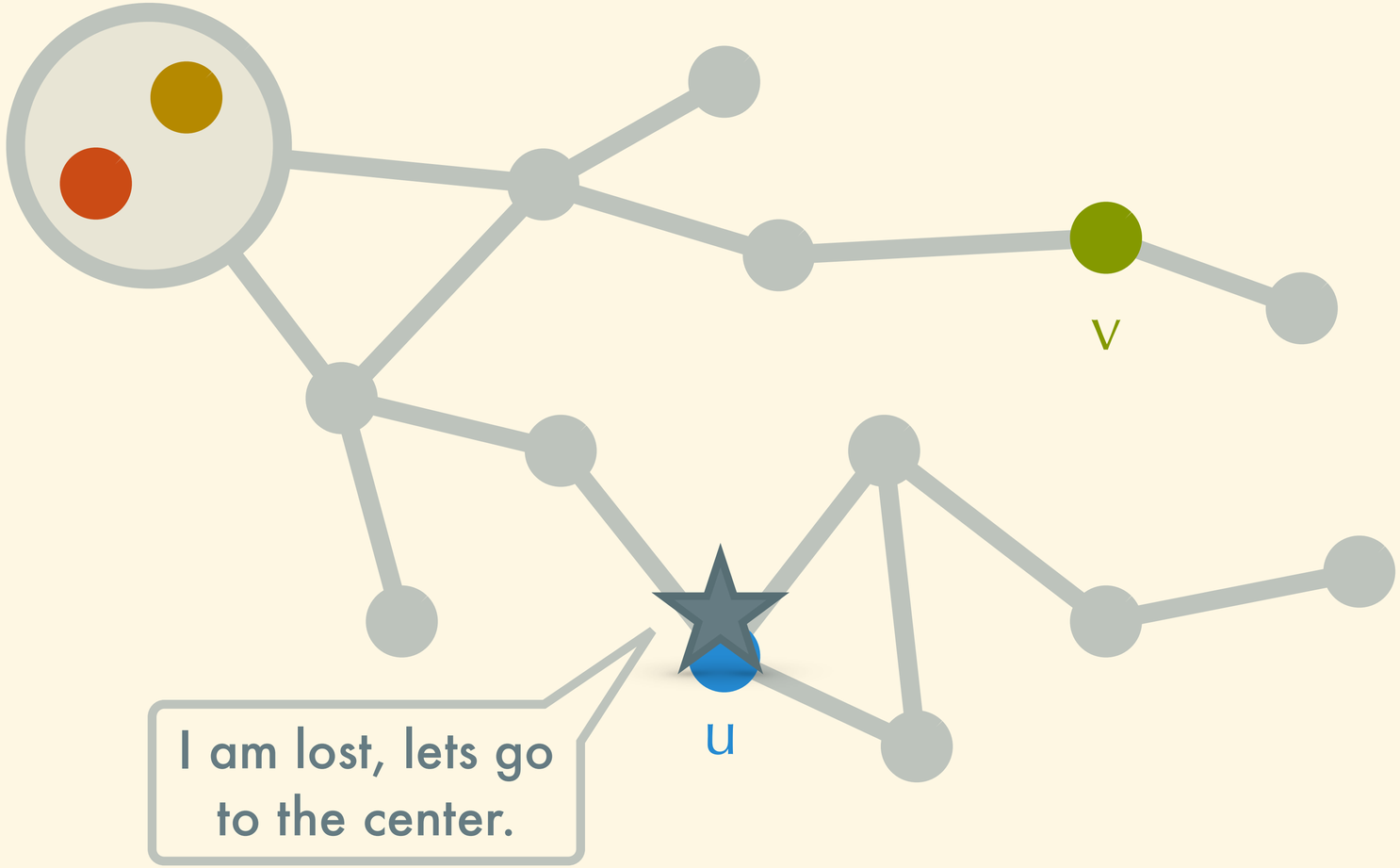
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



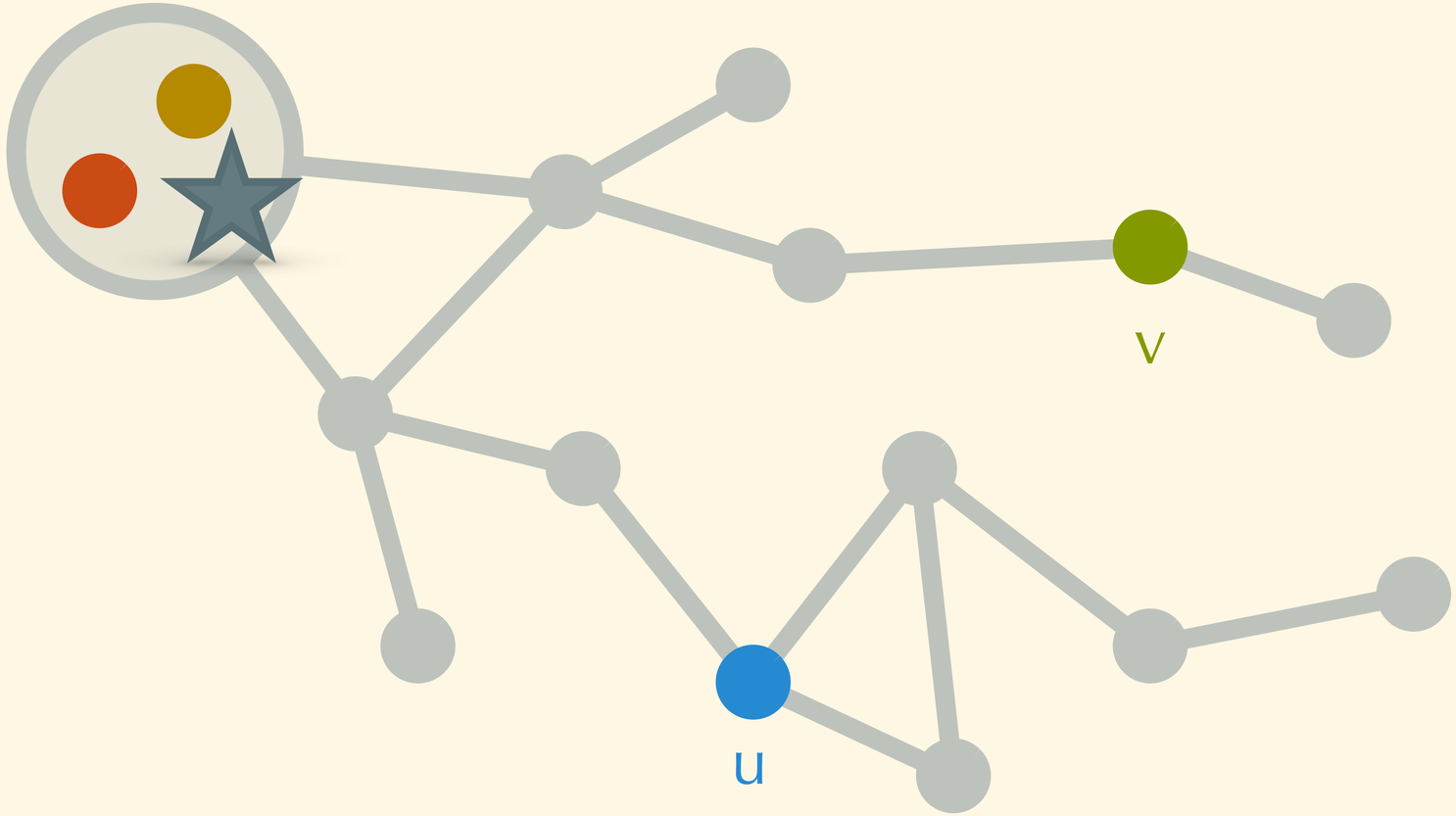
Improving memory performances

How to use the structure to save some entries?



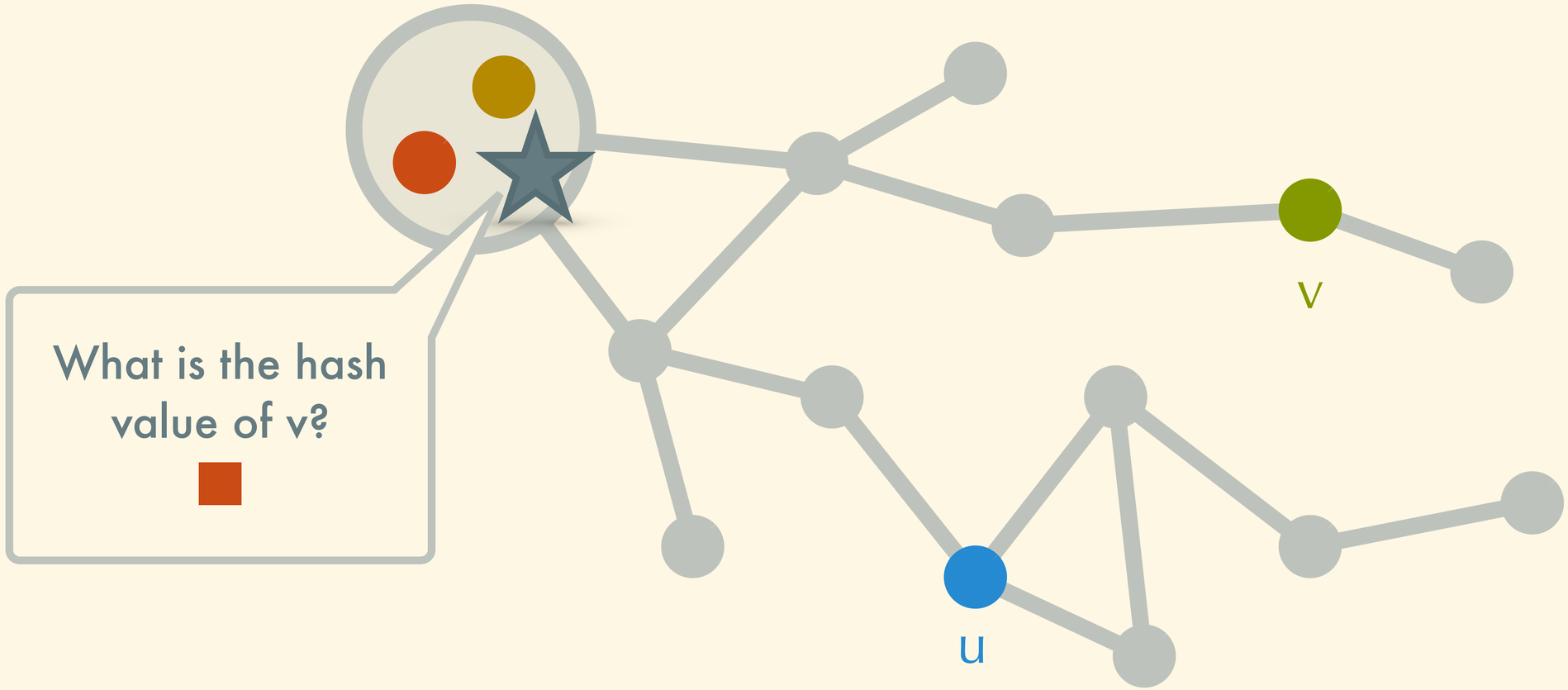
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



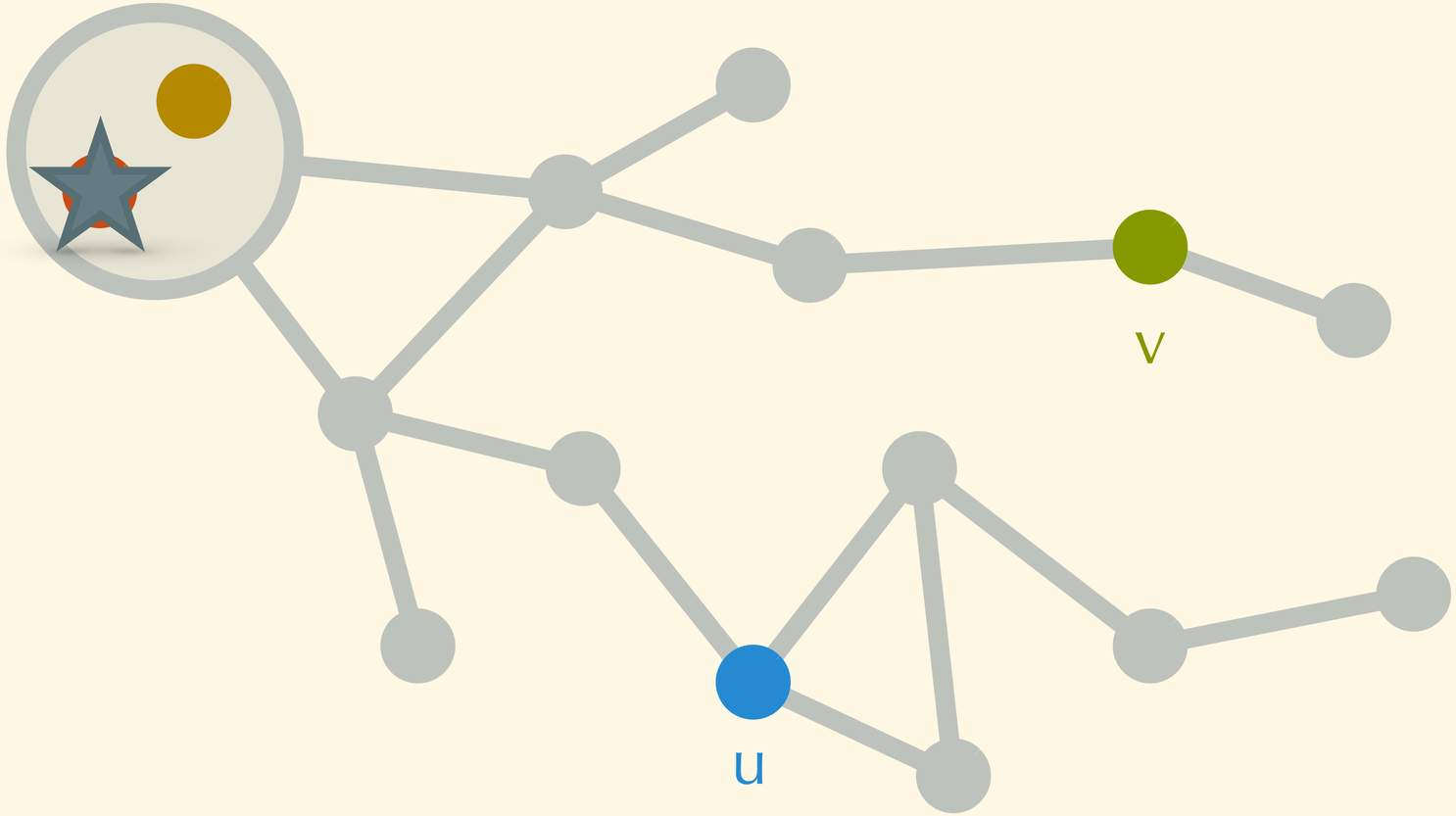
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



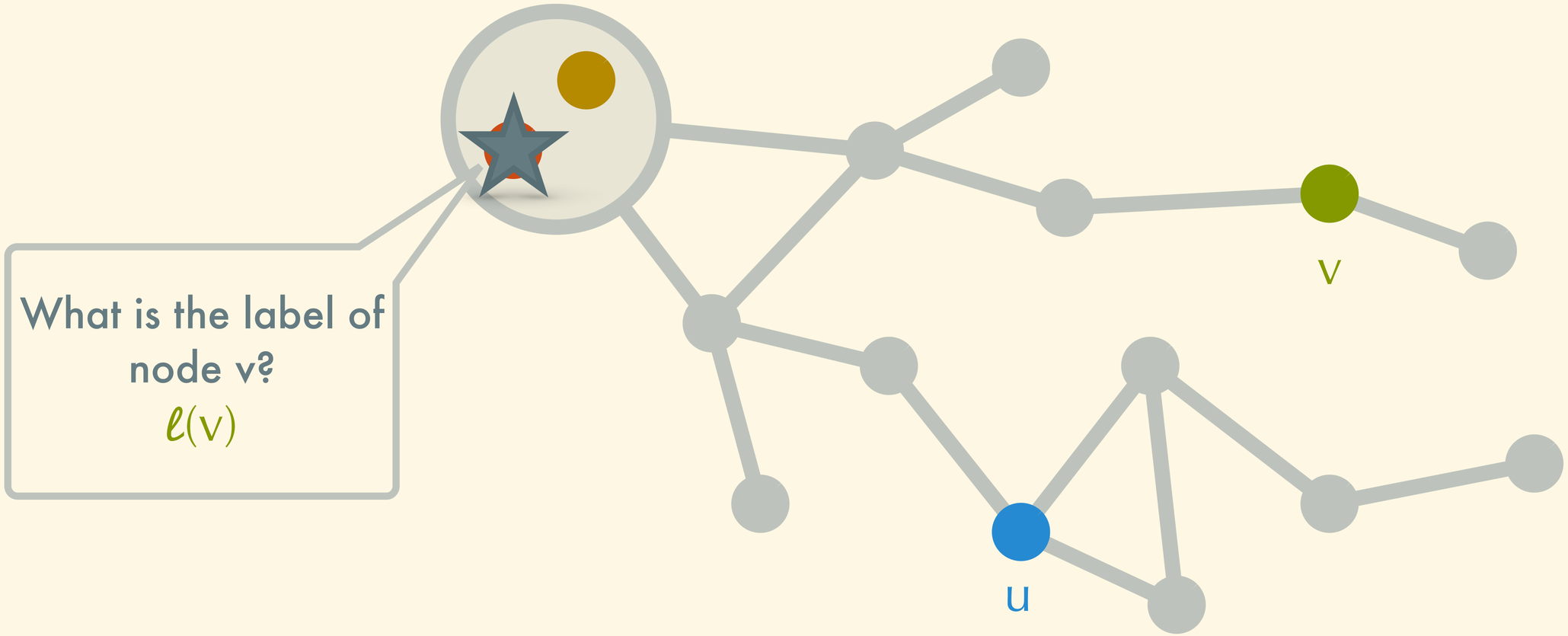
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



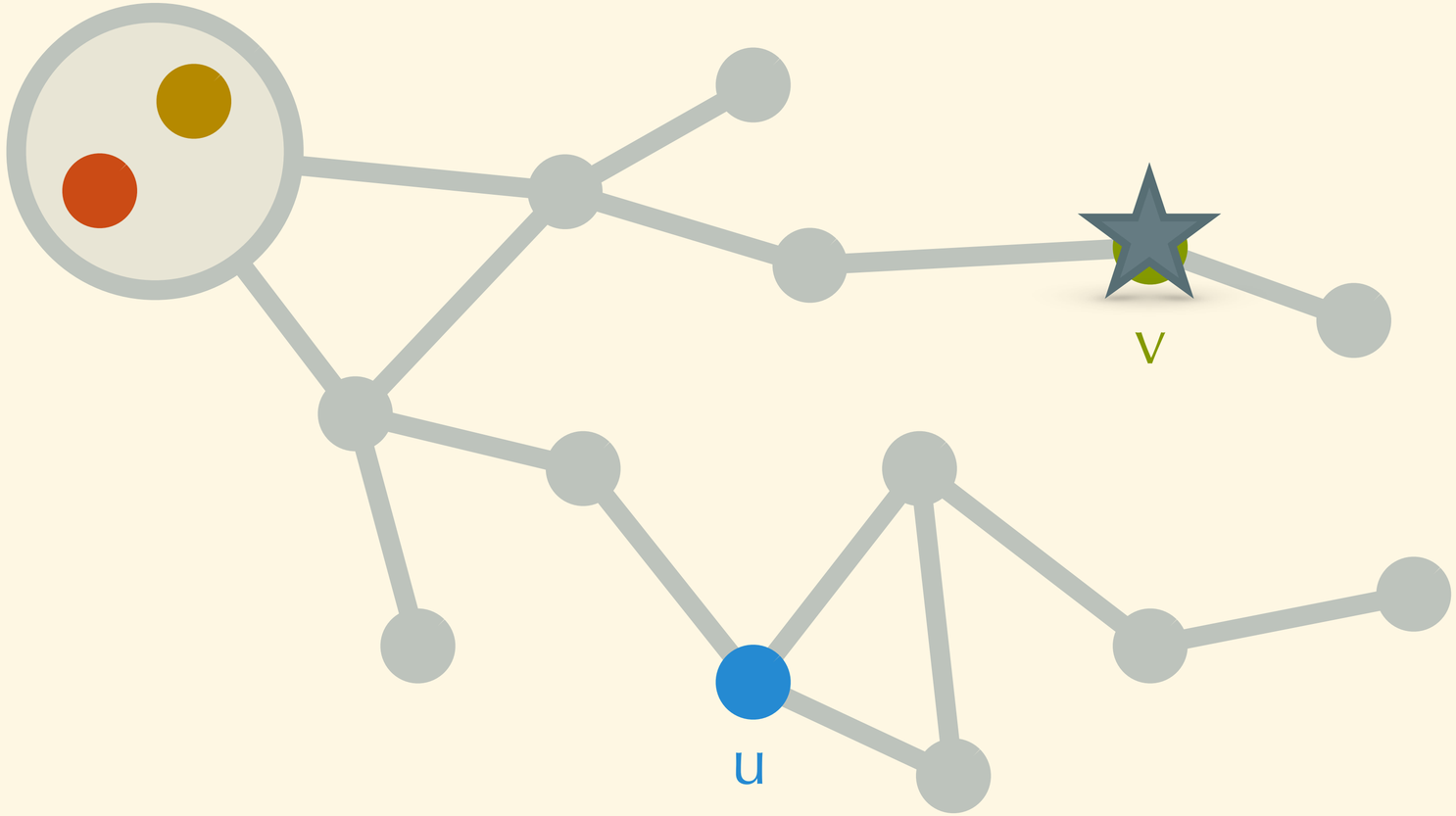
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



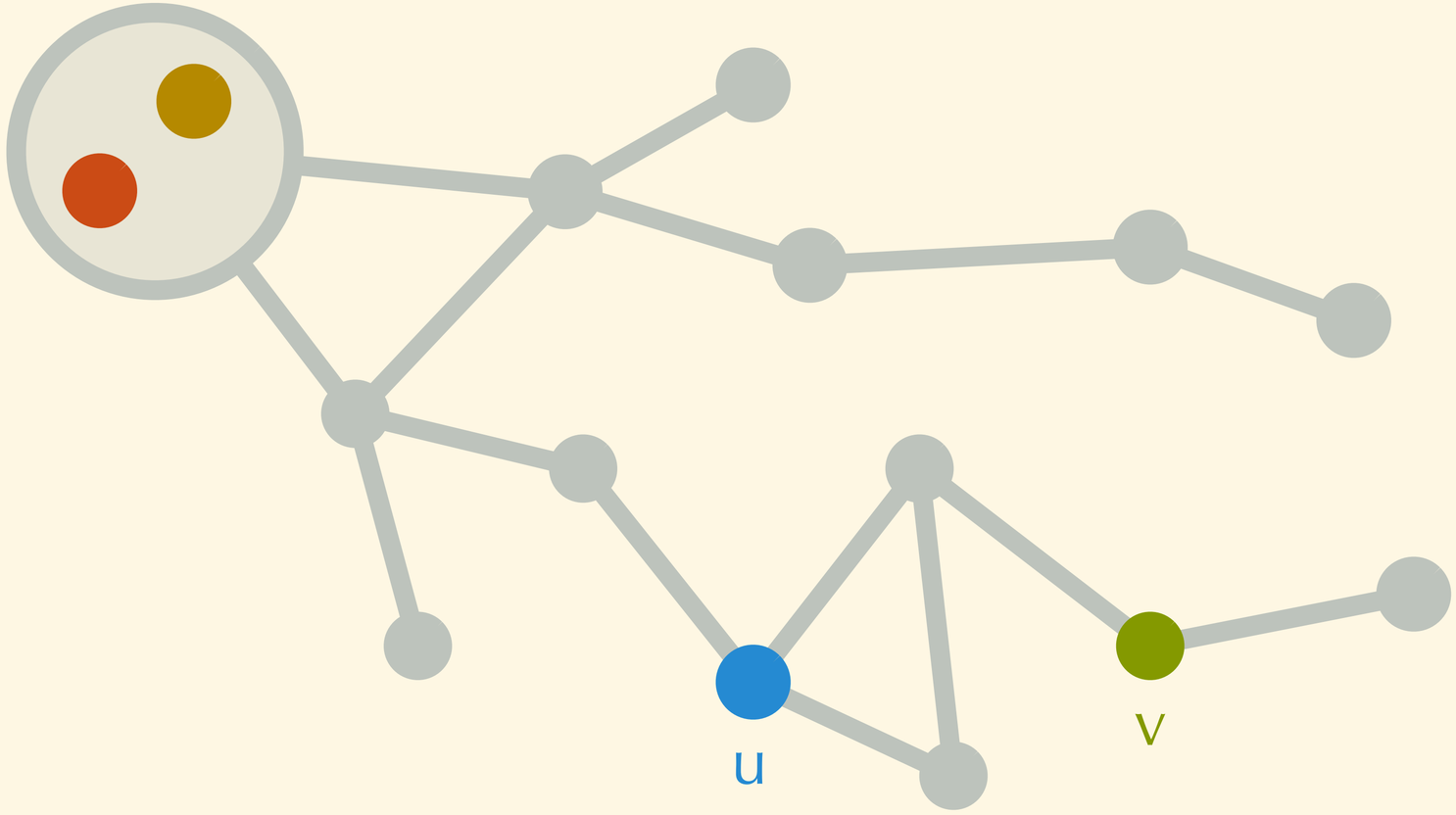
IMPROVING MEMORY PERFORMANCES

How to use the structure to save some entries?



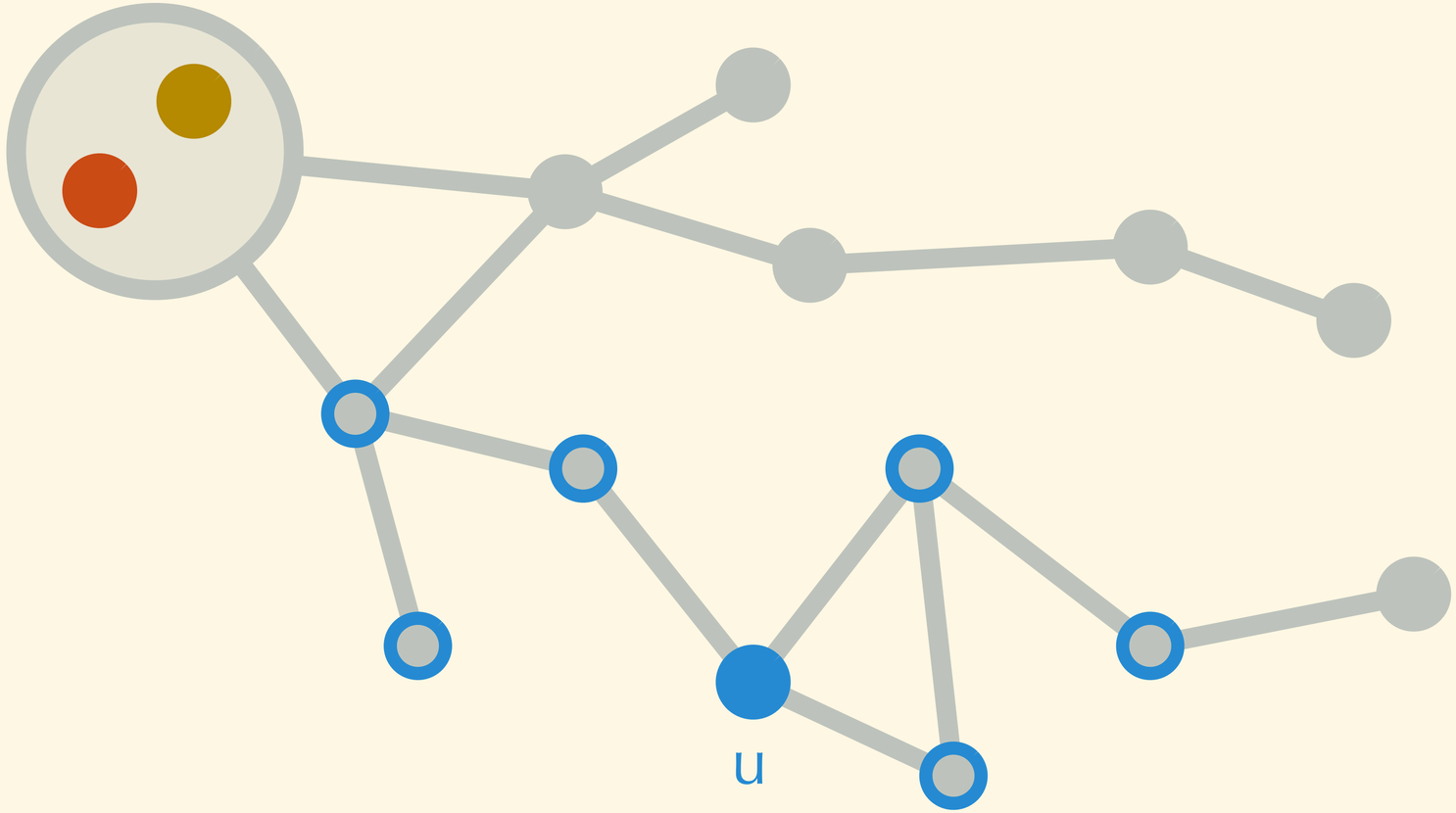
Improving memory performances

How to use the structure to save some entries?



Improving memory performances

How to use the structure to save some entries?



THEORETICAL RESULTS

Compared to older results for similar settings on RPLG

	Stretch	Memory	
		average	maximum
[GGHI'15]	5	$O(n^{1/(t-1)+t-3})$ $O(n^{1/110})$	$O(n^{1-1/(t-1)})$ $O(n^{1/11})$
Labeled [CSTW'12]	5	$O(n^{(t-2)/(2t-3)})$ $O(n^{1/12})$	
[TZLL'13]	≥ 6	$O(\sqrt{n})$	$O(n)$

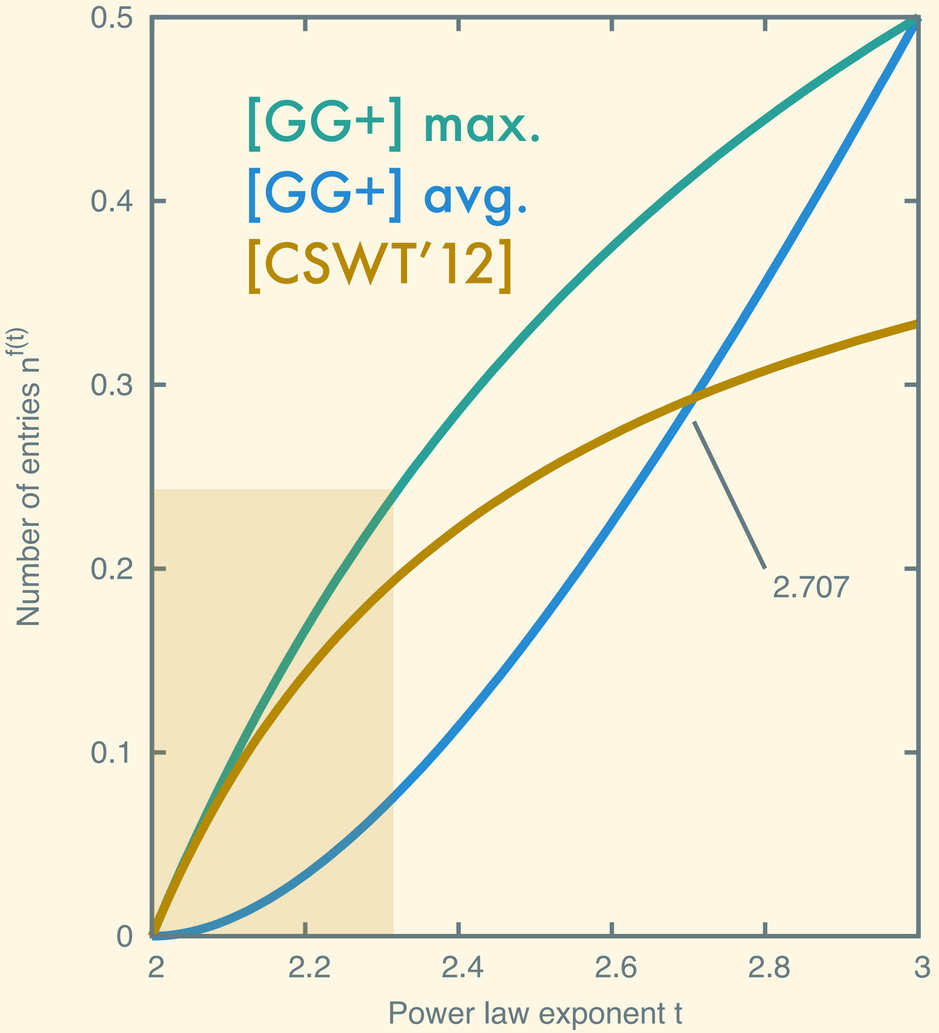
Upper bounds for $t = 2.1$

In depth [CSWT'12] vs. [GGHI'15]

Theoretical comparison for various exponential values

- when $t < 2.707$ [GGHI'15] avg. memory is smaller
- [CSTW'12]:
  - has a smaller theoretical max. memory
  - but is a labeled scheme!
- For small values of t, hidden constants may play an important role

What are the performances in practice?



EXPERIMENTAL RESULTS

For an RPLG with t=2.1

		avg stretch	memory	
			average	maximum
	[AGMNT'08]	1.56	396	1 143
Labeled	[CSTW'12]	1.30	55.2	580
	[TZLL'13]	1.24	404	1 877
	[GGHI'15]	1.75	6.47	228

EXPERIMENTAL RESULTS

For an AS-graph 16K nodes

		avg stretch	memory	
			average	maximum
	[AGMNT'08]	1.74	465	1 261
Labeled	[CSTW'12]	1.18	24	687
	[TZLL'13]	1.52	106	2 324
	[GGHI'15]	1.59	**4.05**	415

Future Works

- In sparse networks, for $q=1/2$, routing tables of size $\tilde{O}(n^q)$
  - communication cost is $\tilde{O}(n^{1+q})$
  - can that be done for any q?

- Labeled = Name-independent? (True for stretch 3)*
  - ie., name-independent routing with stretch $2k+1$ and memory $O(n^{1/k})$

- Very compact routing schemes for internet-like graphs:
  - "logarithmic memory is required for infinite scaling"

*Disclaimer: No prize is offered for solving this.

MERCI