Voting in Networks

Network Theory Reading Group

Théo Delemazure

Nokia Bell Labs

December 9, 2020

Motivation

In 2009, thousands of outraged users of Facebook accused the social network of claiming too many right over the user-generated content.

In response, Facebook organized a vote to choose the new terms of use of the website. It was also announced that at least 30% of the active users would have to vote for the results to be binding.

The new rules were approved by a large majority (74.4%). However, only 0.3% of the users actually voted.

$\Rightarrow$ Voting require **time** and **dedication**. Can we use the structure of the network to make things easier ?

Motivation

In 2009, thousands of outraged users of Facebook accused the social network of claiming too many right over the user-generated content.

In response, Facebook organized a vote to choose the new terms of use of the website. It was also announced that at least 30% of the active users would have to vote for the results to be binding.

The new rules were approved by a large majority (74.4%). However, only 0.3% of the users actually voted.

$\Rightarrow$ Voting require **time** and **dedication**. Can we use the structure of the network to make things easier ?

Motivation

In 2009, thousands of outraged users of Facebook accused the social network of claiming too many right over the user-generated content.

In response, Facebook organized a vote to choose the new terms of use of the website. It was also announced that at least 30% of the active users would have to vote for the results to be binding.

The new rules were approved by a large majority (74.4%). However, only 0.3% of the users actually voted.

$\Rightarrow$ Voting require **time** and **dedication**. Can we use the structure of the network to make things easier ?

Motivation

In 2009, thousands of outraged users of Facebook accused the social network of claiming too many right over the user-generated content.

In response, Facebook organized a vote to choose the new terms of use of the website. It was also announced that at least 30% of the active users would have to vote for the results to be binding.

The new rules were approved by a large majority (74.4%). However, only 0.3% of the users actually voted.

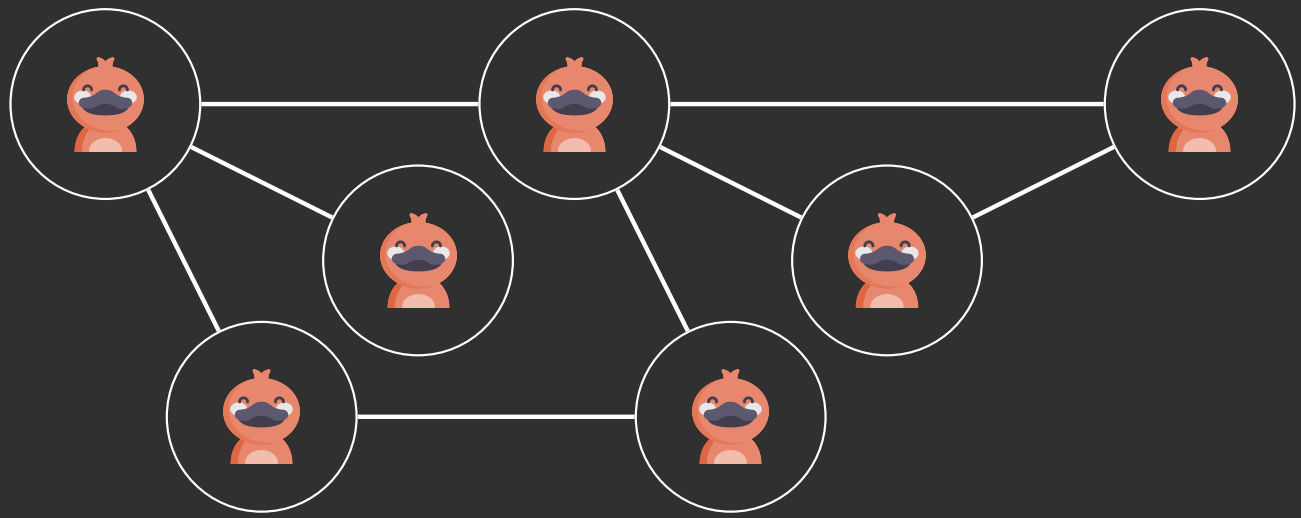
$\Rightarrow$ Voting require **time** and **dedication**. Can we use the structure of the network to make things easier ?

Outline

Outline

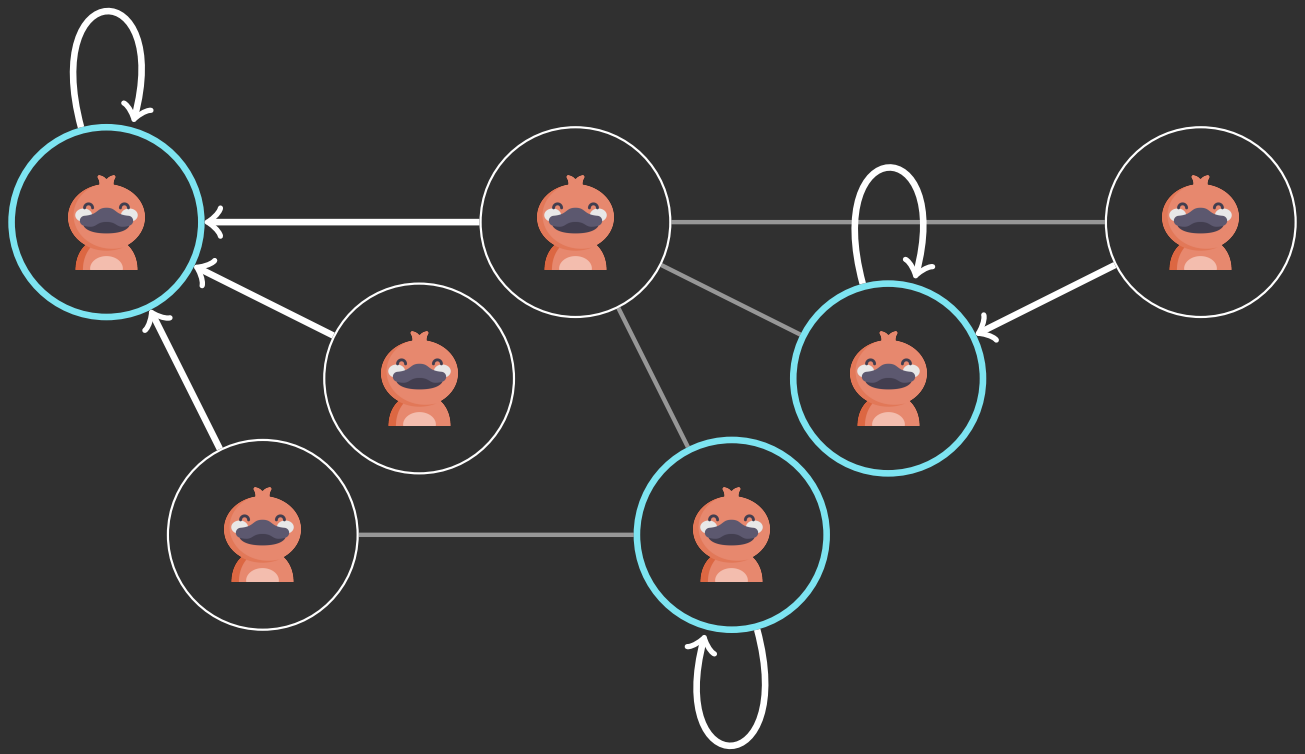
Classic proxy voting

In classic proxy voting settings, a voter can transfer his voting power to a representative.



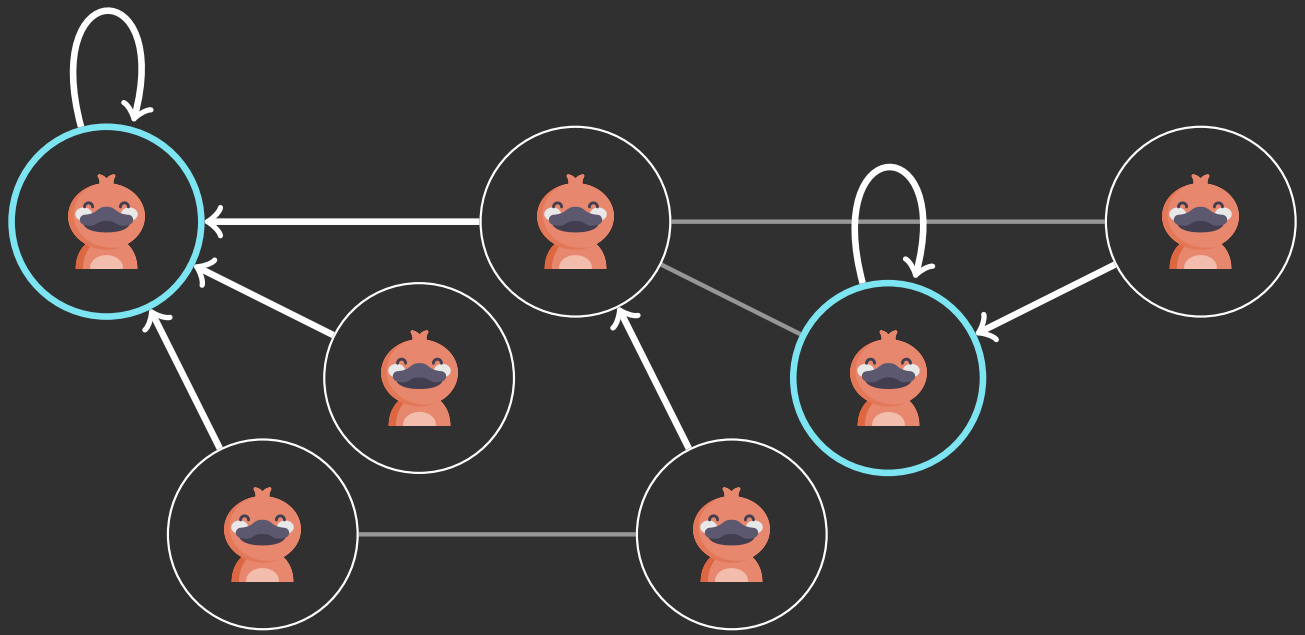
Classic proxy voting

In classic proxy voting settings, a voter can transfer his voting power to a representative.



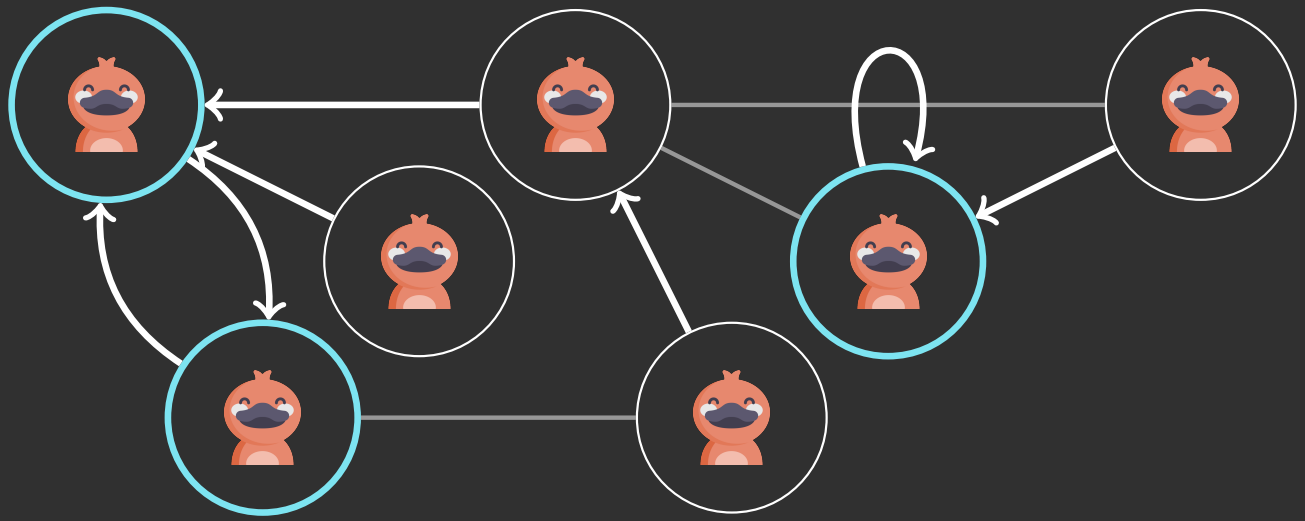
Transferable proxy voting

In transferable proxy voting settings, a voter can transfer his voting power to a representative, even his *proxy* voting power.



Transferable proxy voting

In transferable proxy voting settings, a voter can transfer his voting power to a representative, even his *proxy* voting power.



Outline

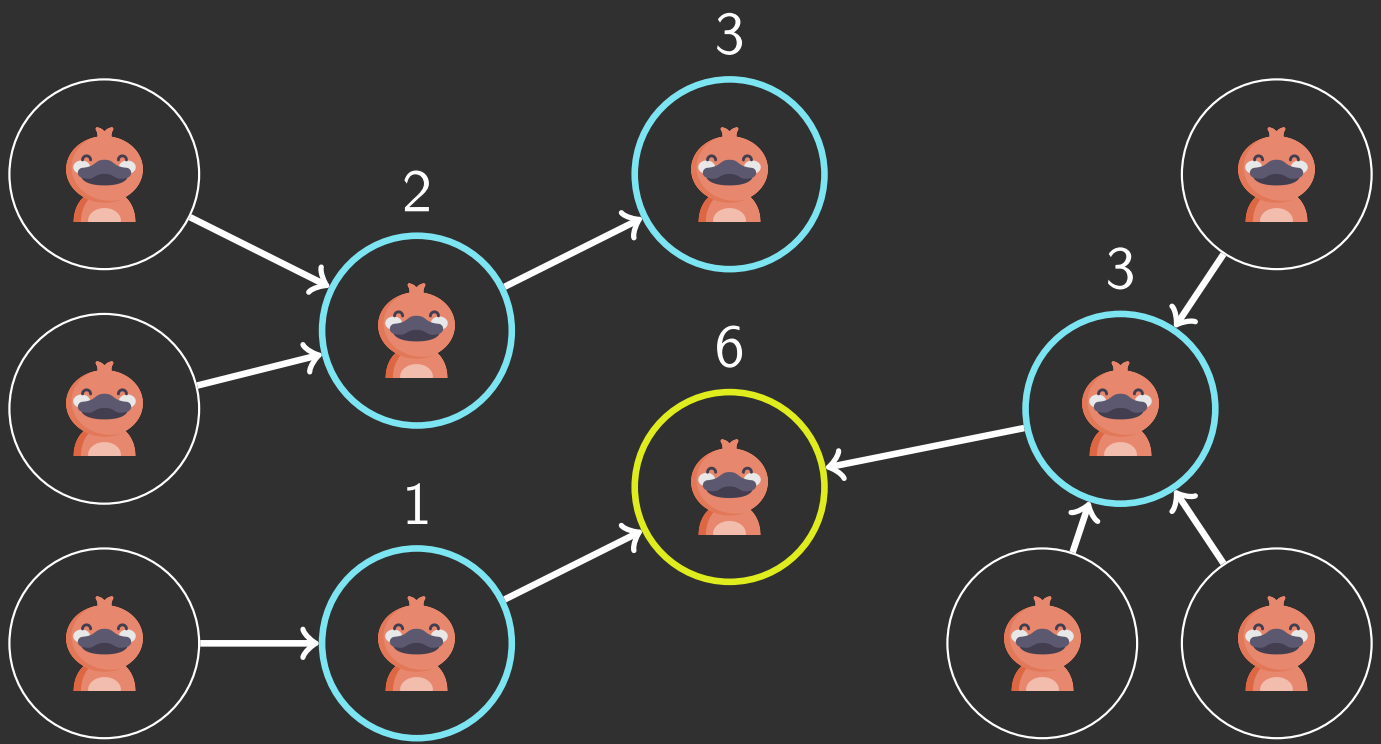
Electing representatives

Electing representatives

"Voting in Social Network"
(P. Boldi et al., 2010)

Electing representatives

$\Rightarrow$ The score of every node is the number of directed paths going to this node.



Viscious democracy

How much do you trust your friends ?
The friends of your friends ?
The friends of the friends of your friends ?

$\Rightarrow$ *Trust decays with distance*

Viscious democracy

How much do you trust your friends ?
The friends of your friends ?
The friends of the friends of your friends ?

$\Rightarrow$ *Trust decays with distance*

Dumping factor

$\alpha \in [0, 1]$ is the dumping factor

$$\text{score}(j) \propto \sum_{\pi \in \text{Path}(\to j)} \alpha^{|\pi|-1}$$

A large dumping factor indicates more trust in your connections.

Dumping factor

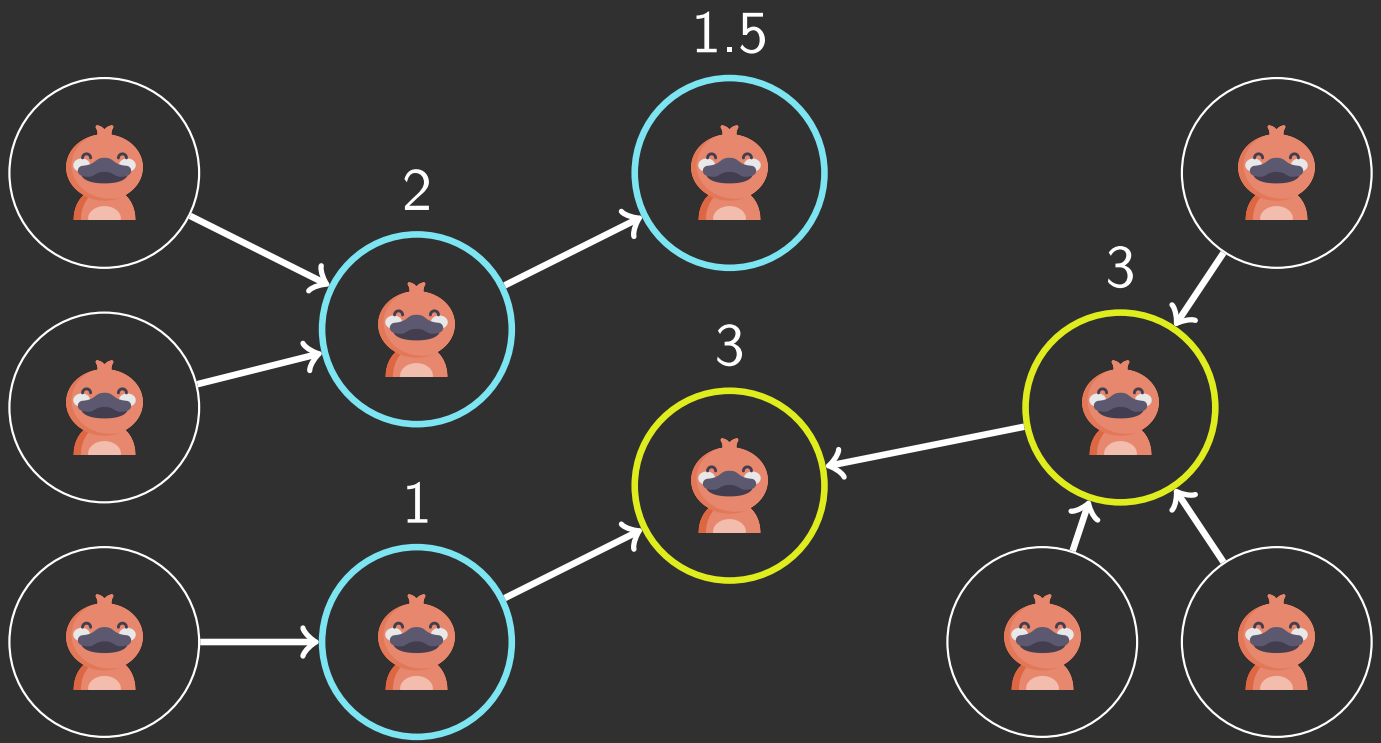
$\alpha \in [0, 1]$ is the dumping factor

$$\text{score}(j) \propto \sum_{\pi \in \text{Path}(\to j)} \alpha^{|\pi|-1}$$

A large dumping factor indicates more trust in your connections.

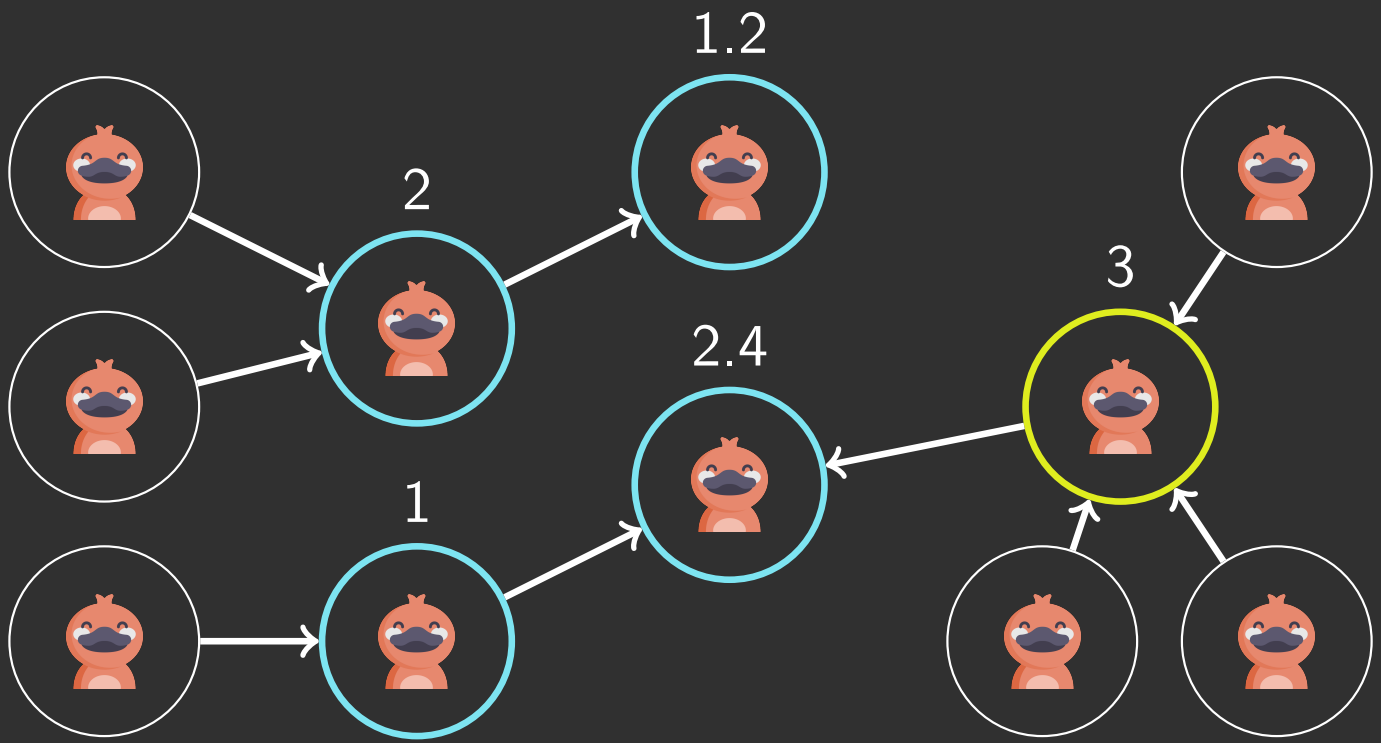
Dumping factor : Example

With $\alpha = \frac{1}{4}$:



Dumping factor : Example

With $\alpha = \frac{1}{10}$:



Wait... I've seen this already

It is actually a special case of the **PageRank** algorithm. Indeed, in a directed graph $D = (V, E)$, the *PageRank* of a node x is computed :

$$PR(x) = \frac{1-\alpha}{|V|} \sum_{\pi \in \mathsf{Path}(\to j)} \alpha^{|\pi|} \mathsf{br}(\pi)$$

With the *branching* of a path defined as follow :

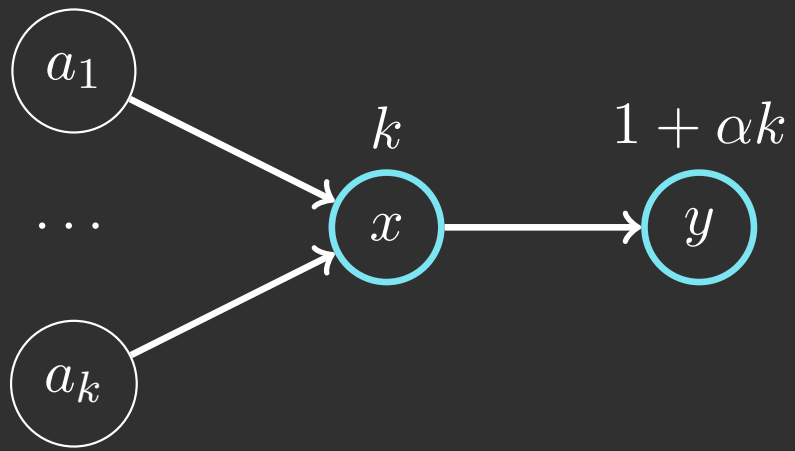
$$\mathsf{br}(x_1, \cdots, x_{k+1}) = \frac{1}{|N_{x_1}|} \times \cdots \times \frac{1}{|N_{x_k}|}$$

$\Rightarrow$ In **PageRank**, a node can vote for several friends and divide its voting power equally among all its out-neighbors.

Large dumping case ($\alpha \to 0$)

Property 1

For sufficiently small α, the vote of every node x can only influence the victory of its *guru*.

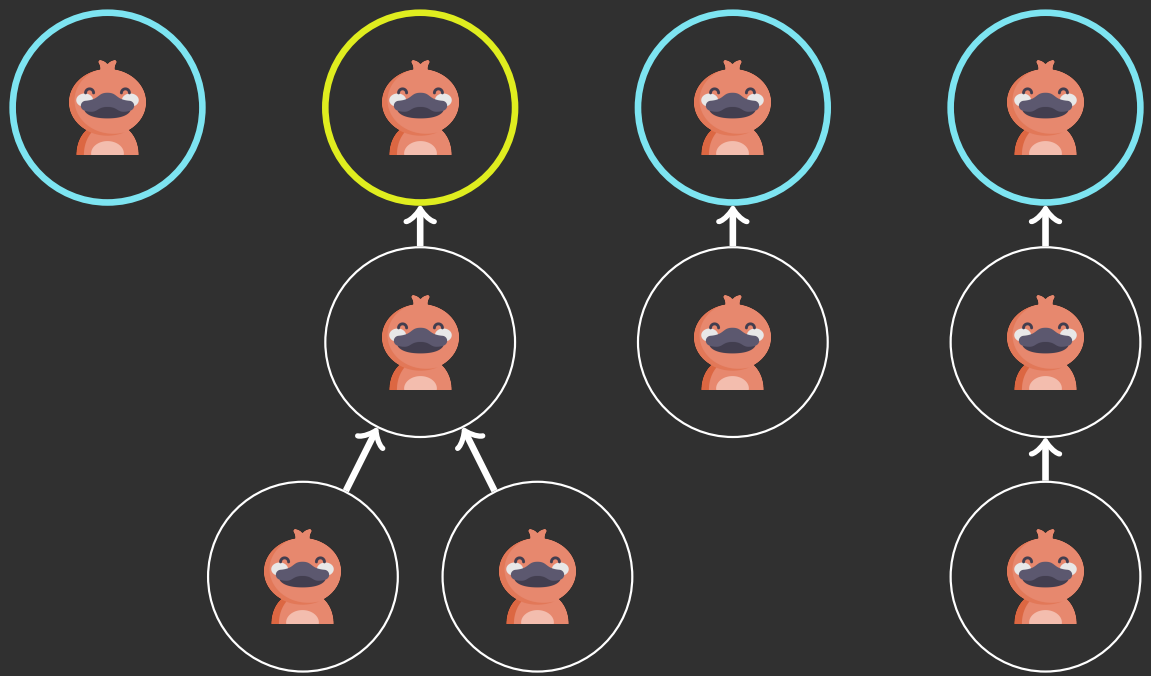


$\Rightarrow$ y win only if $\alpha > 1 - \frac{1}{k}$

Small dumping case ($\alpha \to 1$)

Property 2

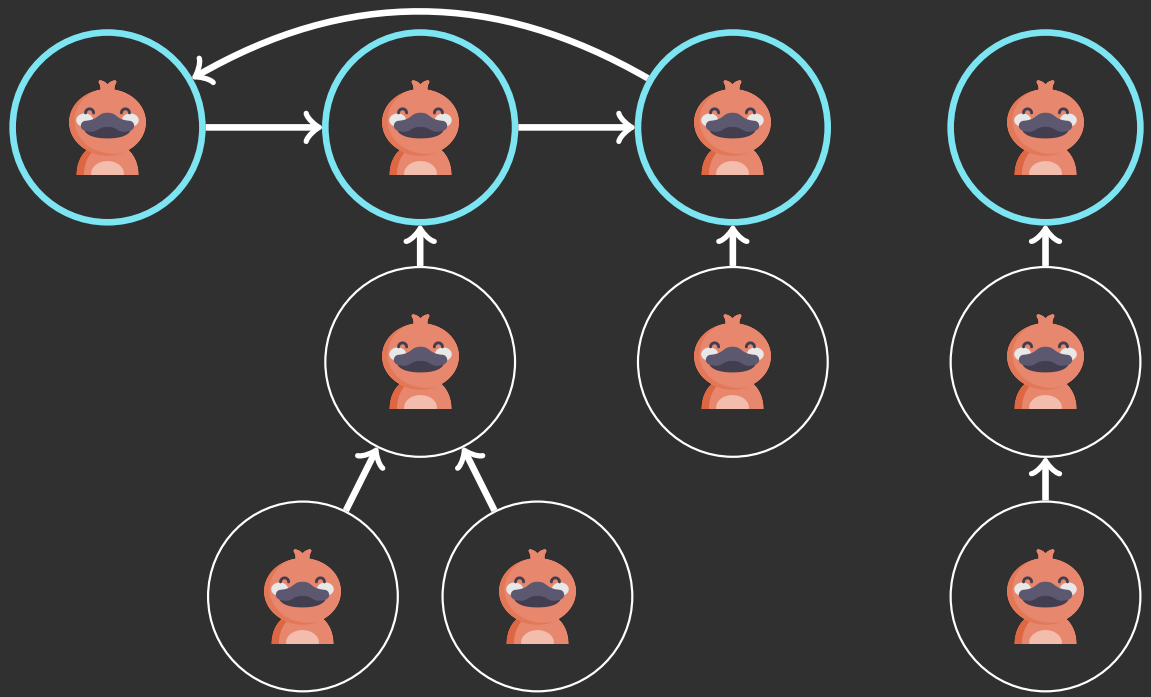
For sufficiently large α, all winners are *non-transient*.



Small dumping case ($\alpha \to 1$)

Property 2

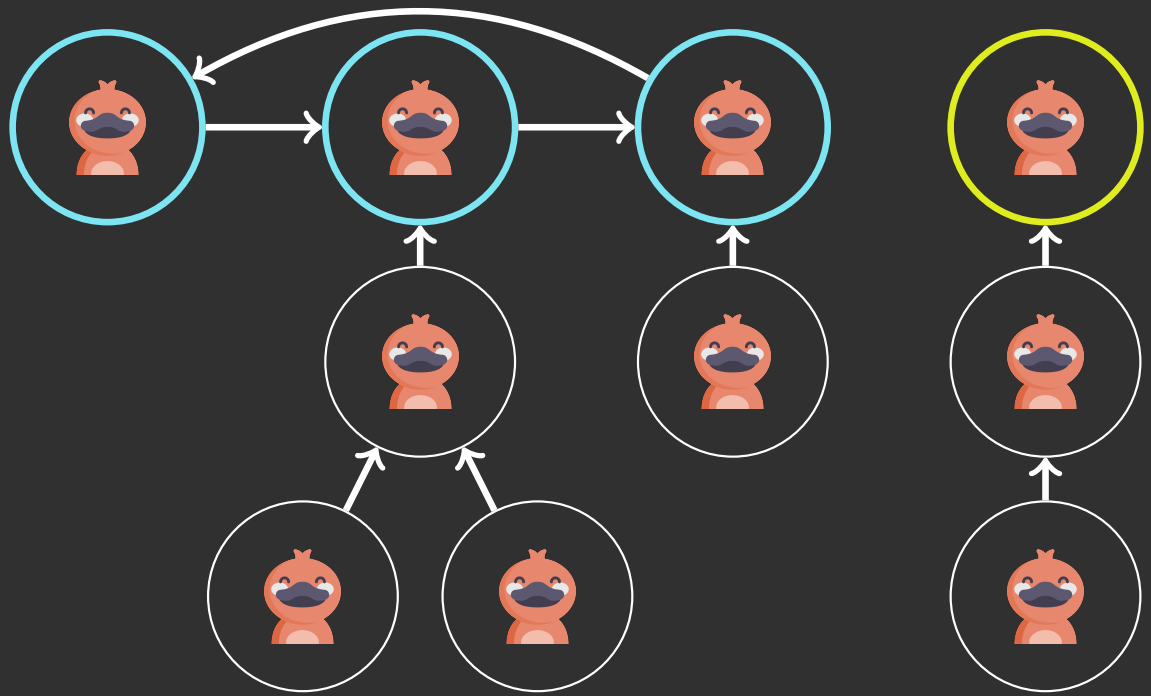
For sufficiently large α, all winners are *non-transient*.



Small dumping case ($\alpha \to 1$)

Property 3

When $\alpha \to 1$, the winners are the nodes on the cycle belonging to the component with the largest average tree size.



Outline

Judgement aggregation

Judgement aggregation with Liquid Democracy

"Binary Voting with Delegable Proxy : An Analysis of Liquid Democracy"
(Z. Christoff and D. Grossi, 2017)

What is judgement aggregation ?

$\mathcal{P} = (P_1, \cdots, P_k)$ a set of propositions. Each voter approve or disapprove a proposition. We associate to each user i a function :

$$\phi_i : \mathcal{P} \to \{0, 1, *\}$$

Then, we use an aggregation rule, for instance the majority rule.

Question : Do you approve the new terms of uses ?

Yes	No	Didn't vote
$45,563$	$34,052$	$1,230,231$

What is judgement aggregation ?

$\mathcal{P} = (P_1, \cdots, P_k)$ a set of propositions. Each voter approve or disapprove a proposition. We associate to each user i a function :

$$\phi_i : \mathcal{P} \to \{0, 1, *\}$$

Then, we use an aggregation rule, for instance the majority rule.

Question : Do you approve the new terms of uses ?

Yes	No	Didn't vote
$45,563$	$34,052$	$1,256,234$

What is judgement aggregation ?

$\mathcal{P} = (P_1, \cdots, P_k)$ a set of propositions. Each voter approve or disapprove a proposition. We associate to each user i a function :

$$\phi_i : \mathcal{P} \to \{0, 1, *\}$$

Then, we use an aggregation rule, for instance the majority rule.

Question : Do you approve the new terms of uses ?

Yes	No	Didn't vote
$45,563$	$34,052$	$1,256,234$

Back to liquid democracy

For each proposition :

1. I vote myself
2. OR I delegate my vote, maybe to someone who is specialized on the question

For each proposition, every user choose a *guru*, who can be herself. This gives us a delegation graph. The users on top of each trees are called *great gurus*.

Then, we propagate the vote of gurus to their followers, and we apply a judgement aggregation rule, for instance *the majority rule*.

Back to liquid democracy

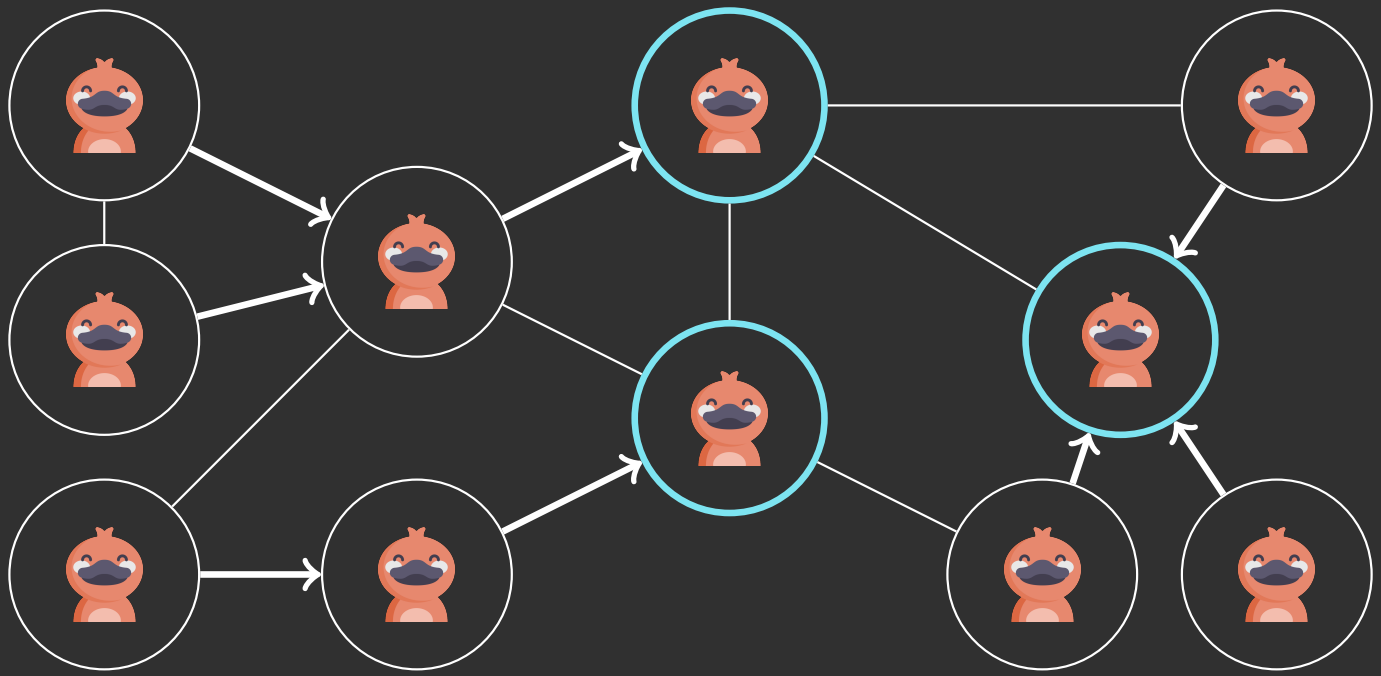
For each proposition :

1. I vote myself
2. OR I delegate my vote, maybe to someone who is specialized on the question

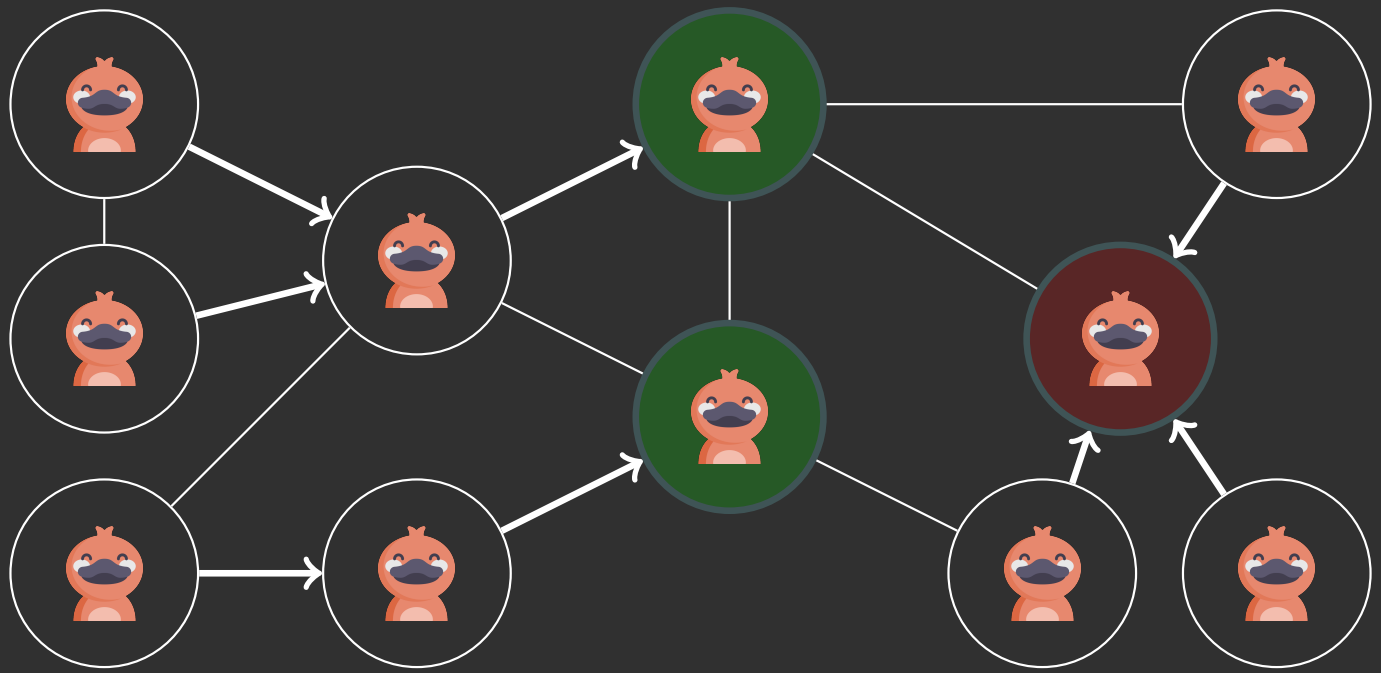
For each proposition, every user choose a *guru*, who can be herself. This gives us a delegation graph. The users on top of each trees are called *great gurus*.

Then, we propagate the vote of gurus to their followers, and we apply a judgement aggregation rule, for instance *the majority rule*.

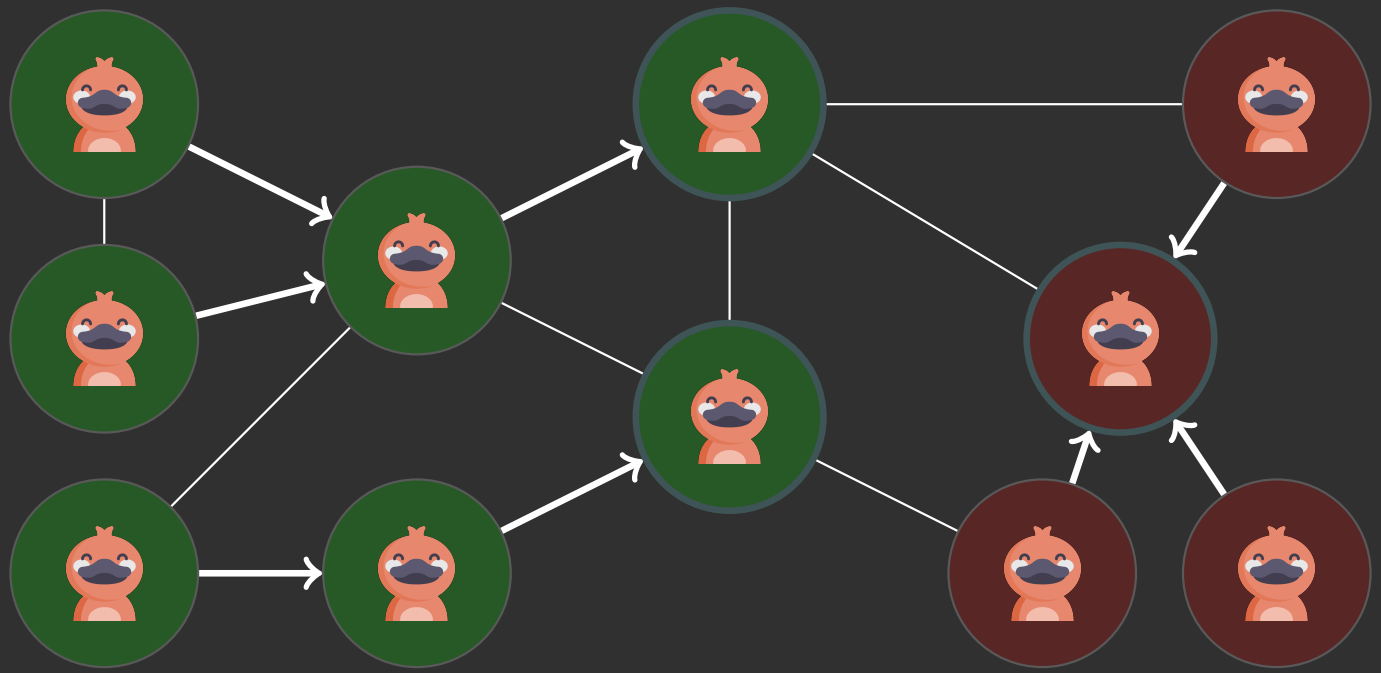
Example in action



Example in action



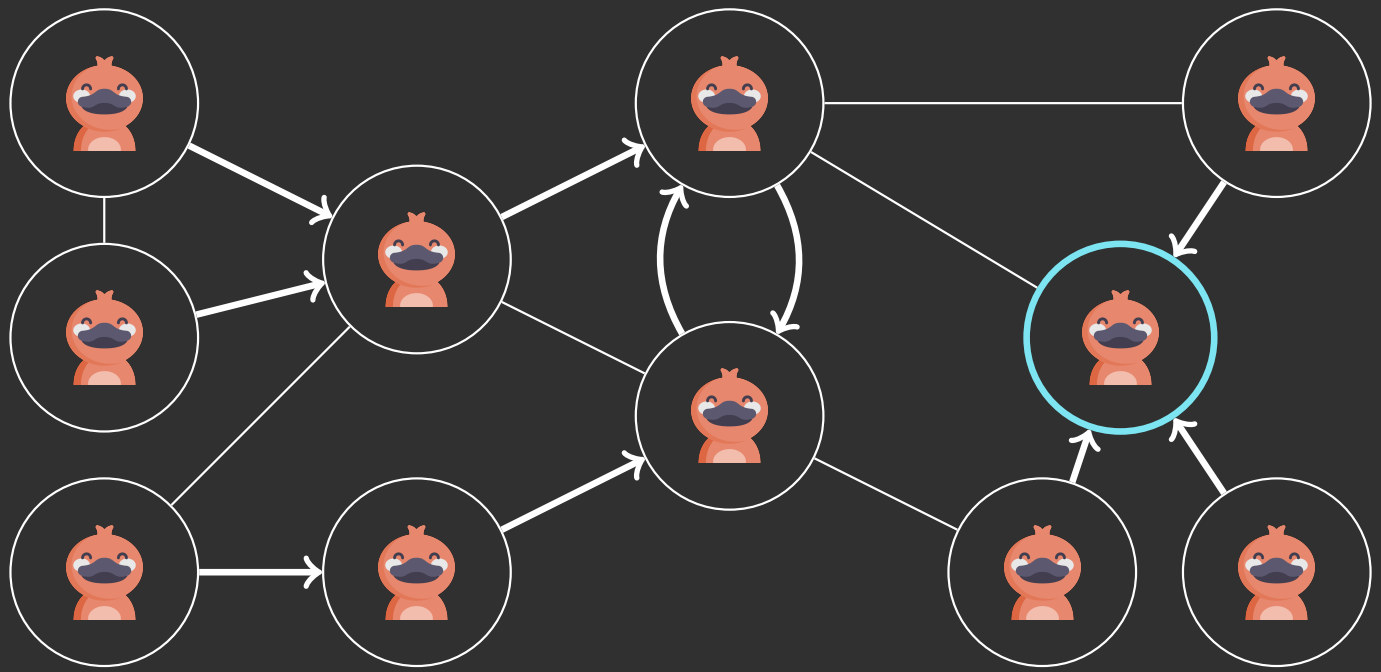
Example in action



Some examples

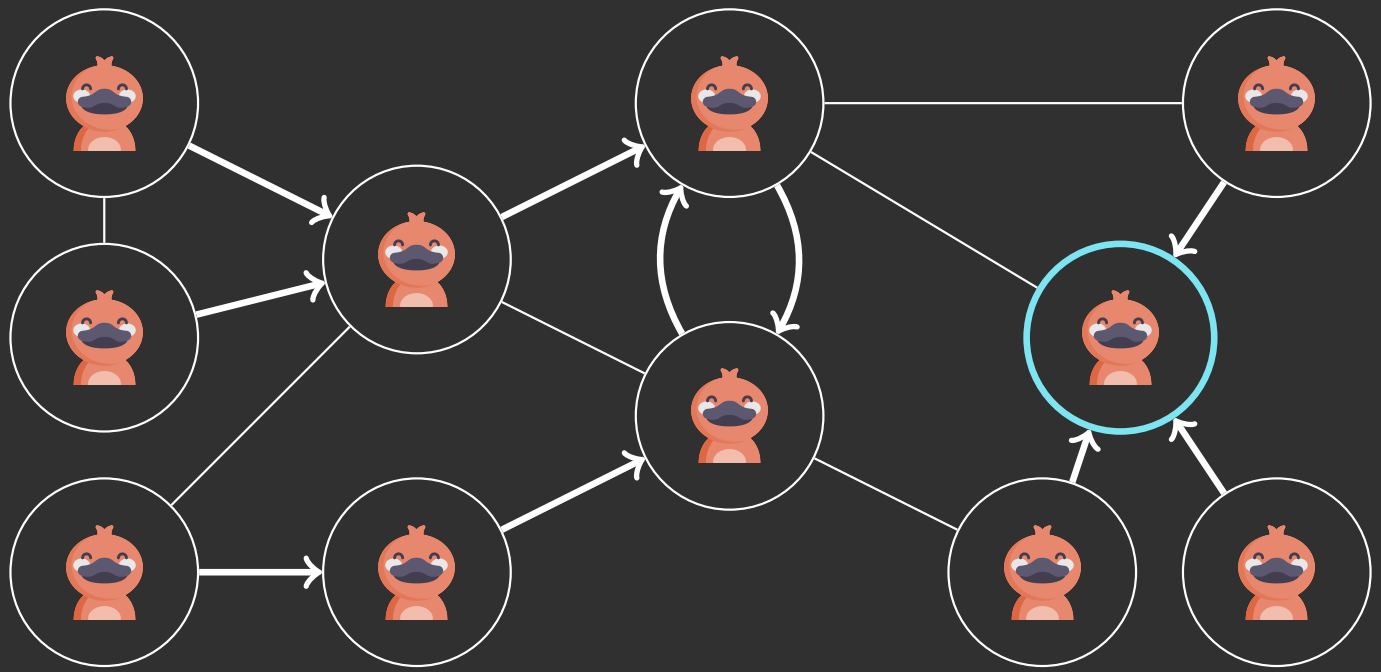
- LiquidFeedback is a software that is used for political opinion formation and decision making.
- Pirate parties in Germany, Italy, Austria, Norway, France and the Netherlands use it.
- Some experiments like Google Votes and LiquidFriesland

Cyclic delegation strikes again !



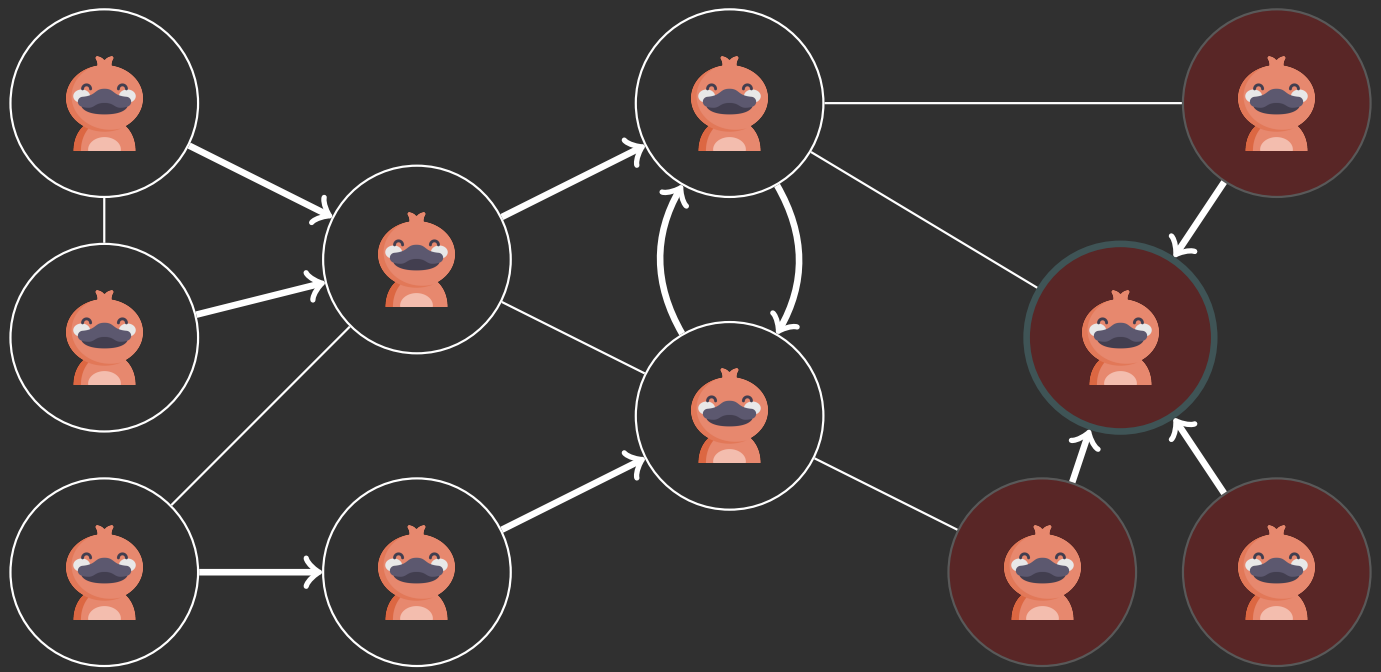
$\Rightarrow$ We **ignore** these voters !

Cyclic delegation strikes again !



$\Rightarrow$ We **ignore** these voters !

Cyclic delegation strikes again !



$\Rightarrow$ We **ignore** these voters !

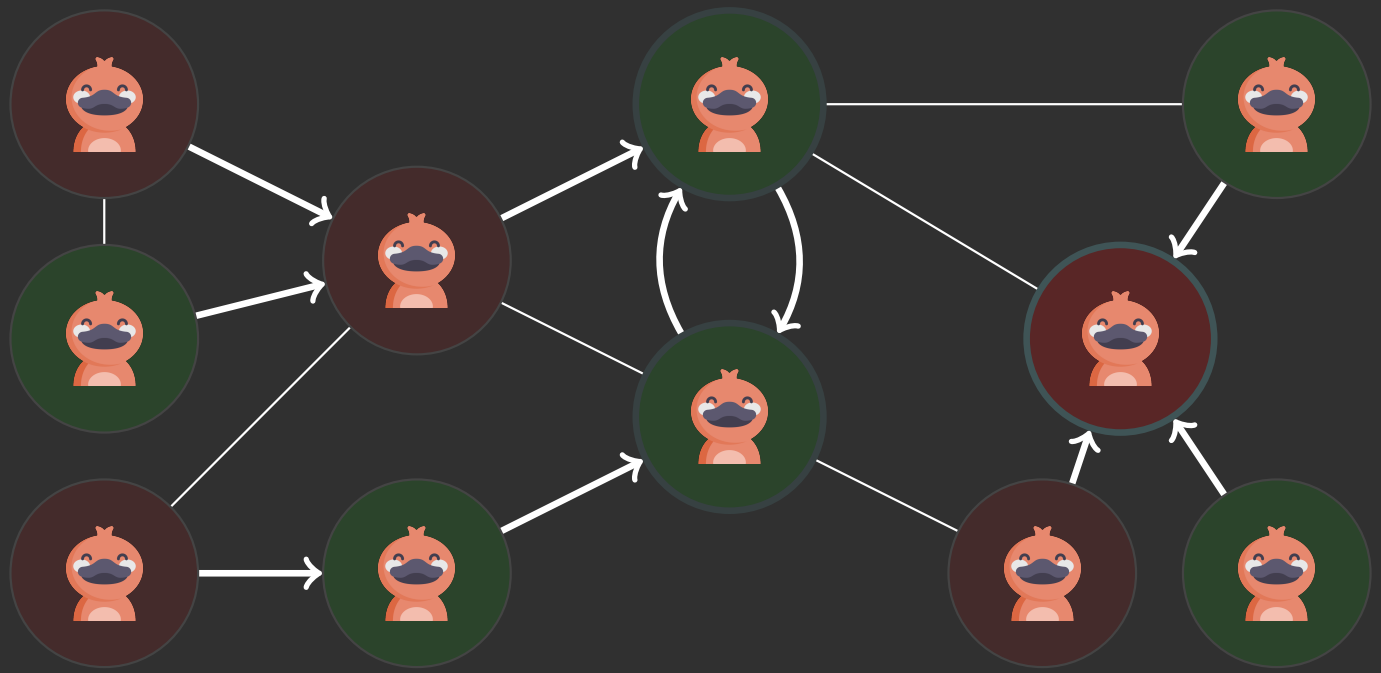
Cyclic delegation strikes again !

Christoff and Grossi highlight that by ignoring these voters, we drop the nice one-man-one-vote property. Moreover, they show that, if each proxy profile is equally probable, the probability that every voter abstain is not 0.

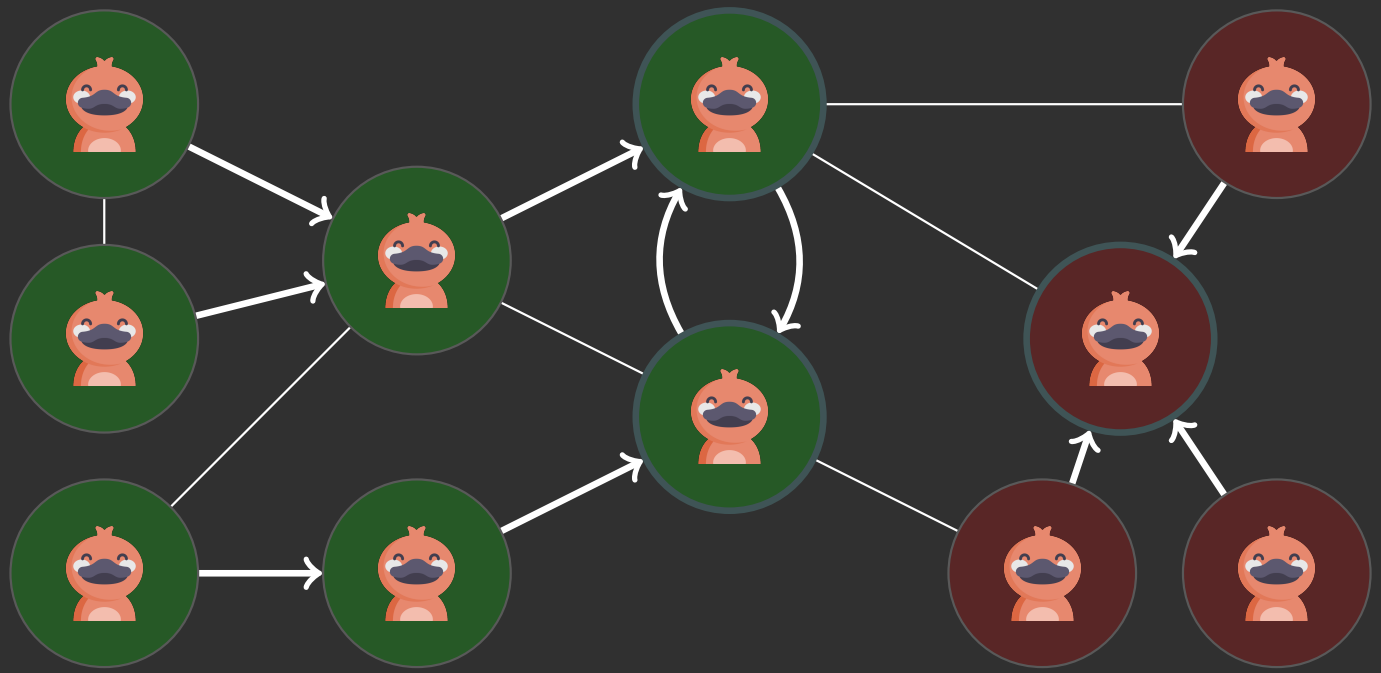
They suggest a slight modification of the process in which every voter casts a trustee and a substantive opinion.

Then, for each *guru cycle*, we take the majority opinion among the cycle.

Cyclic delegation strikes again !



Cyclic delegation strikes again !



Outline

Propagation of votes in a decentralized network

Let's assume each node can only communicate with its neighbors

At the time of the election, great gurus voted, and we want to propagate their votes to their followers.

Christoff and Grossi suggest a very simple process, called Boolean DeGroot Process. If we note $O_i(p)^t$ the opinion of voter i on proposition p at time t, then :

- $O_i(p)^0 = v_i$, the *"default"* vote of the node.
- $O_i(p)^{t+1} = O_j(p)^t$, where j is i's guru.

Propagation of votes in a decentralized network

Let's assume each node can only communicate with its neighbors

At the time of the election, great gurus voted, and we want to propagate their votes to their followers.

Christoff and Grossi suggest a very simple process, called Boolean DeGroot Process. If we note $O_i(p)^t$ the opinion of voter i on proposition p at time t, then :

- $O_i(p)^0 = v_i$, the *"default"* vote of the node.
- $O_i(p)^{t+1} = O_j(p)^t$, where j is i's guru.

Propagation of votes in a decentralized network

Let's assume each node can only communicate with its neighbors

At the time of the election, great gurus voted, and we want to propagate their votes to their followers.

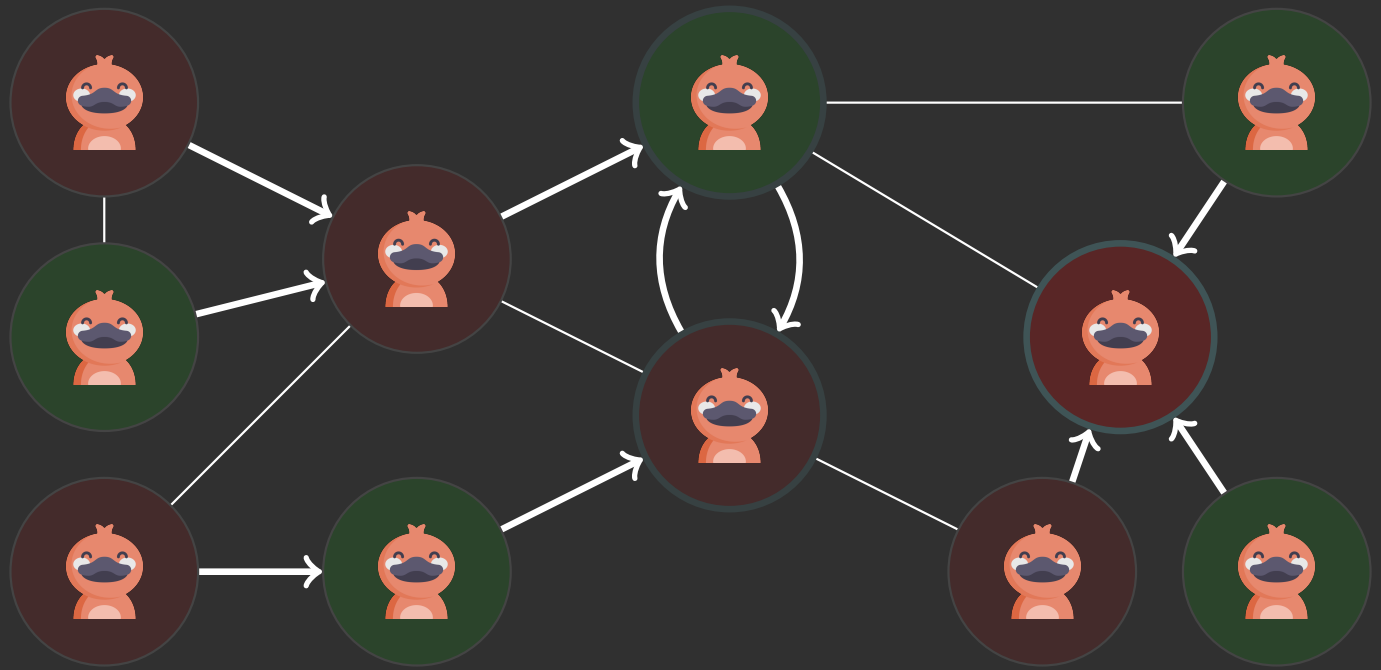
Christoff and Grossi suggest a very simple process, called Boolean DeGroot Process. If we note $O_i(p)^t$ the opinion of voter i on proposition p at time t, then :

- $O_i(p)^0 = v_i$, the *"default"* vote of the node.
- $O_i(p)^{t+1} = O_j(p)^t$, where j is i's guru.

Propagation of votes

Property 4

The *Boolean DeGroot Process* stabilizes if and only if every cycle is **unanimous**. If so, it stabilizes in less than $\mathrm{diam}(G)$ steps.



Propagation of votes

I find the unanimity constraint a bit strong, so I propose another process.

The intuition behind this process is that every *guru cycle* apply the majority rule among them to determine their vote and then propagate this vote to their followers.

Propagation of votes

At time $t = 0$, great gurus send the pair $(v_i, \star)$ to their followers where $v_i \in \{0, 1\}$ is their vote ; Other nodes send the pair (v_i, i). Each node initialize a count $c_i = v_i$.

At time t, the node i receive a message from its guru (v, x) :

- If $x = k \neq i$, then it increments $c_i \leftarrow c_i + v$.
- If $x = \star$ then v is the vote of the great guru of i and $v_i \leftarrow v$.

In these first two cases, the node transfer the same message to its followers.

- If $x = i$, that means i is in a cycle, in which the average vote is c_i/t. The node can compute the vote of the cycle v by applying majority rule. In the next step, it send the message $(v, \star)$ to its followers.

Propagation of votes

Property 5

This process always stabilize after at most $\text{diam}(G) + 1$ steps.

Property 6

This process requires to send messages of $log(|V| + 1) + 1$ bits.

$\Rightarrow$ However, since the process always stabilize, every node can stop sending and receiving messages once it received a $\star$.

Outline

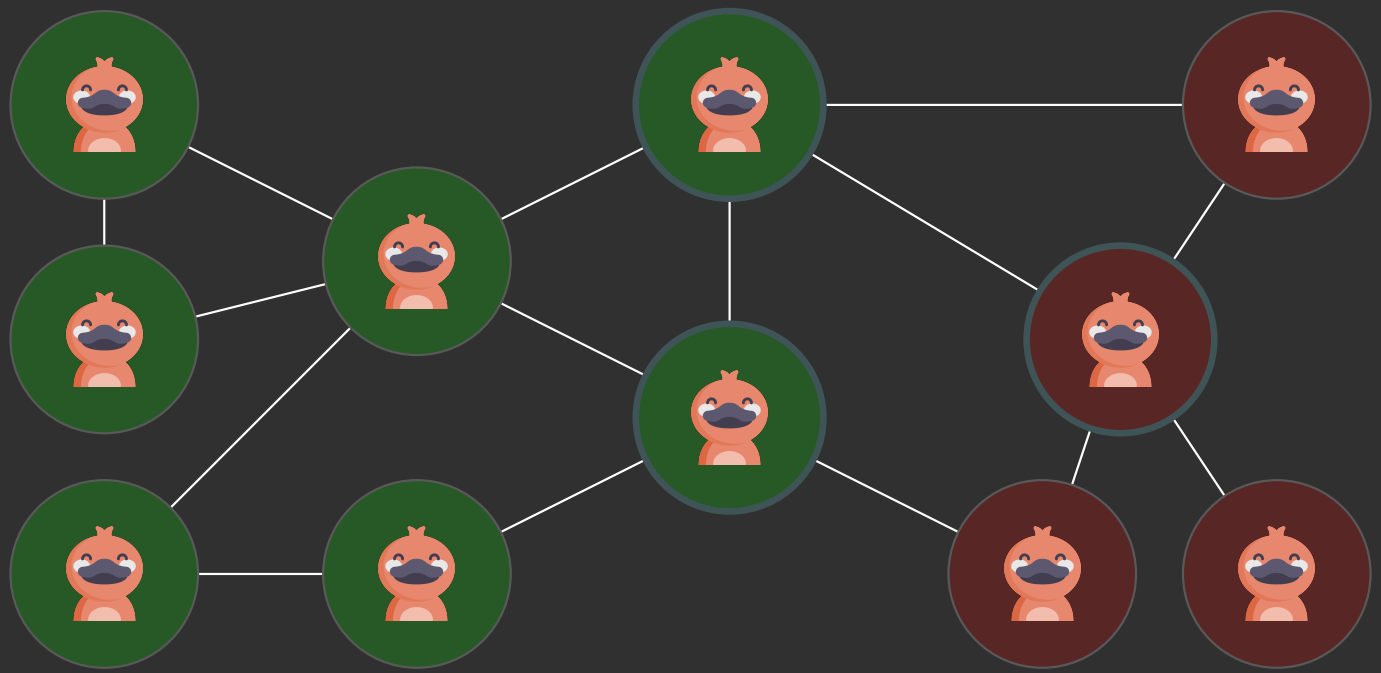
Result diffusion

Result diffusion

"Interval Consensus : From Quantized Gossip to Voting"
(F. Benezit et al., 2009)

Current state

Now that every node knows its vote (inherited from its guru), we want to compute the result and propagate it. We will focus on the majority rule here, but this can be adapted to **any binary voting rule**.



Concept

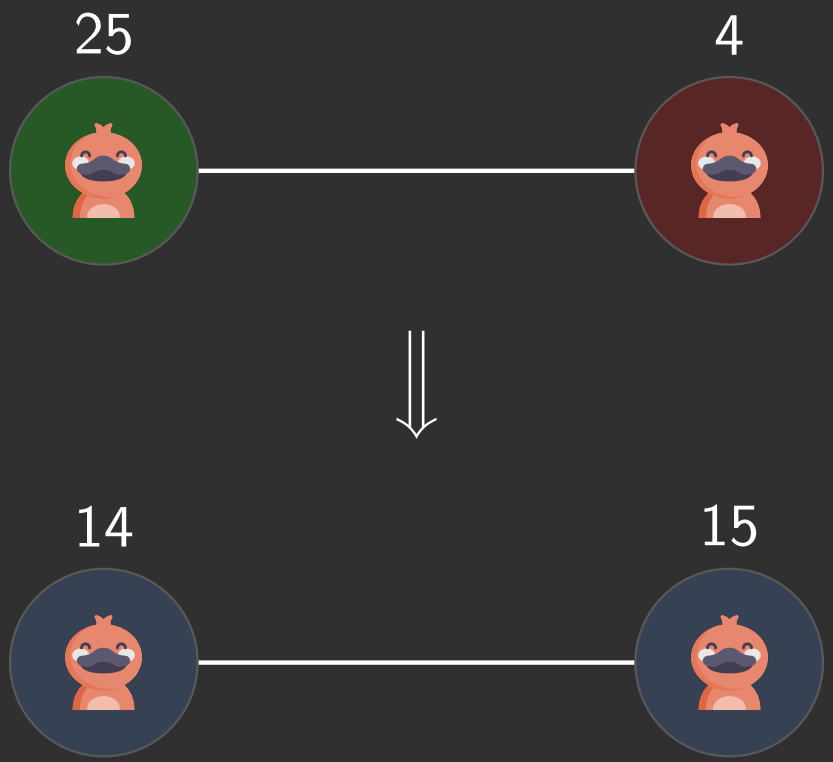
We want that the result of the vote propagate on the network. For this purpose, we use a distributed averaging algorithm.

$\Rightarrow$ *"Quantized consensus"* (2007) : We initialize nodes which approve the proposition with $s_0(i) = |V| = n$ and those which disapprove with $s_0(i) = 0$. The average is equal to the number of approving node.

Then at each time, neighboring nodes communicates and change their values to the average of the two nodes.

This ensure that $\forall t, \sum_i s_t(i) = \sum_i s_0(i)$. Ultimately, each node will know the average of all votes, and therefore the result of the voting.

Example



A gossip is all we need

In the *Quantized algorithm*, each node need to send and receive messages of $log(n)$ bits.

In 2009, F. Benezit et al. proposes in *"Interval Consensus : From Quantized Gossip to Voting"* a way to propagate the result of the voting with only messages of 2 bits.

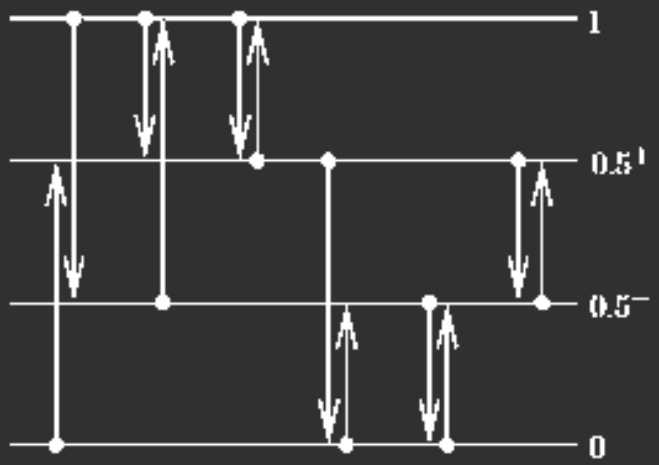
Instead of propagating the **exact average** of all votes, we propagate a gossip about the result. Indeed, we only need to know if this average is greater or less than $1/2$.

A gossip is all we need

There is 4 states : $0, 0.5^-, 0.5^+$ and 1.

Approving nodes start with 1 and disapproving nodes start with 0.

At each time t, neighbors nodes propagate the gossip and update their state according to the following figure :



A gossip is all we need

Property 7

If the algorithm converged and the graph is connected, every node have state 0 or 0.5^- if the proposition is rejected, 0.5^+ or 1 if the proposition is accepted and 0.5^- or 0.5^+ in case of ex-aequo.

Theorem 1

Let T be the first time the algorithm has converged. Then $P[T < \infty] = 1$.

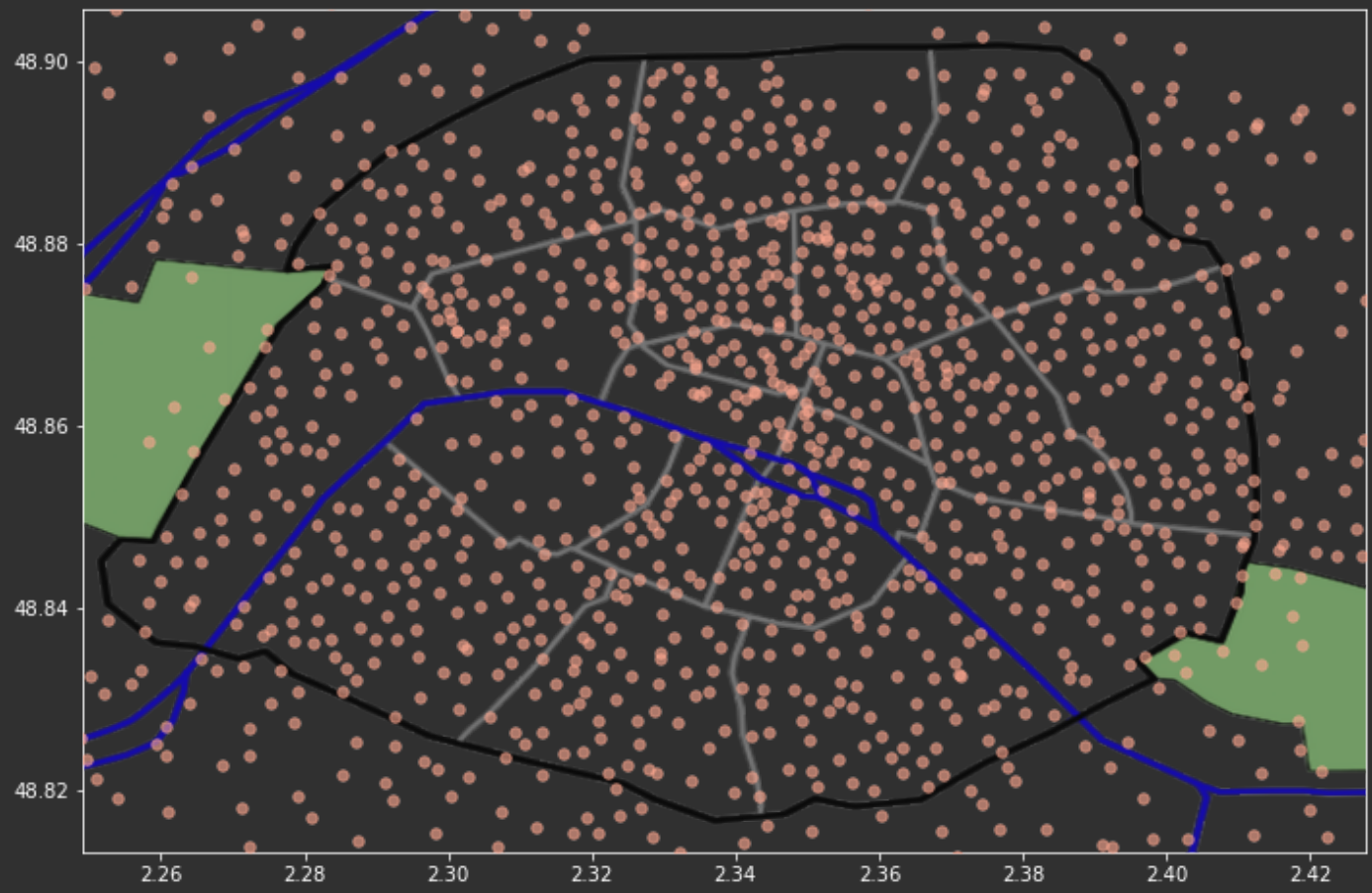
Outline

Experiment

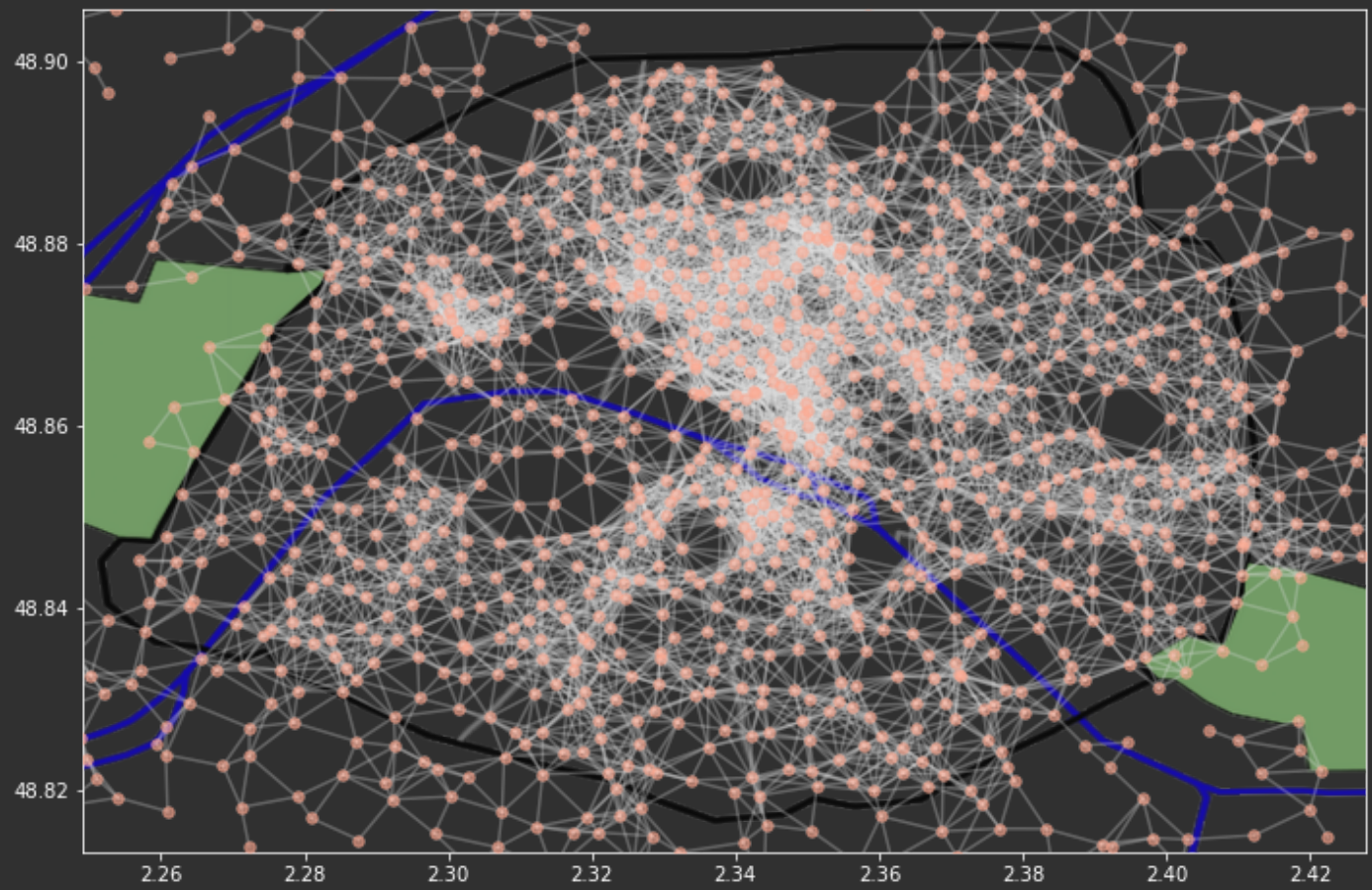
The voting procedure in a decentralized network

1. Every node choose a guru among all its neighbors. Nodes also casts a *"default vote"*.
2. We propagate votes from gurus to their followers.
3. The network use the gossip algorithm to propagate the result of the vote.

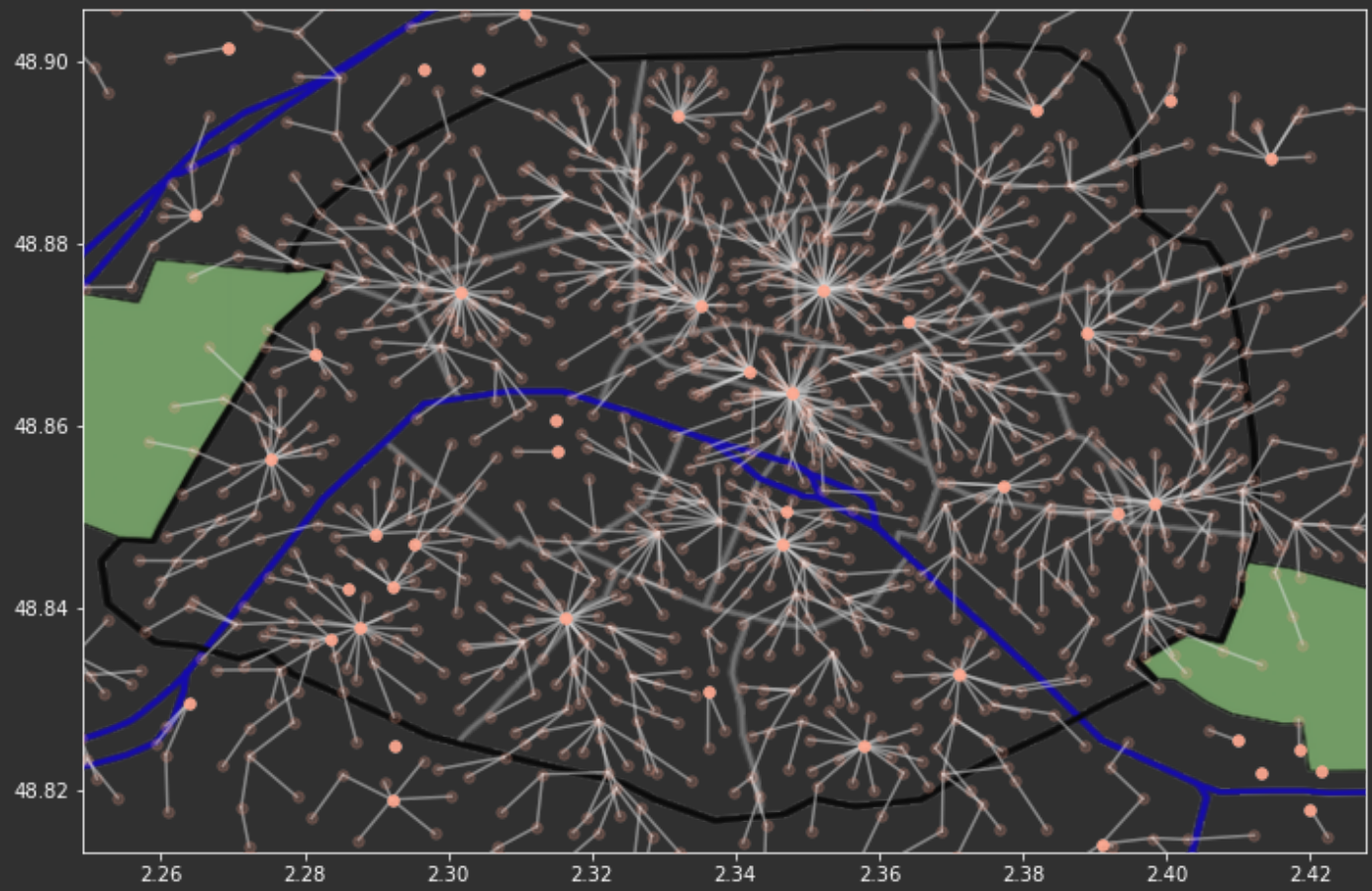
The Velib network : The nodes



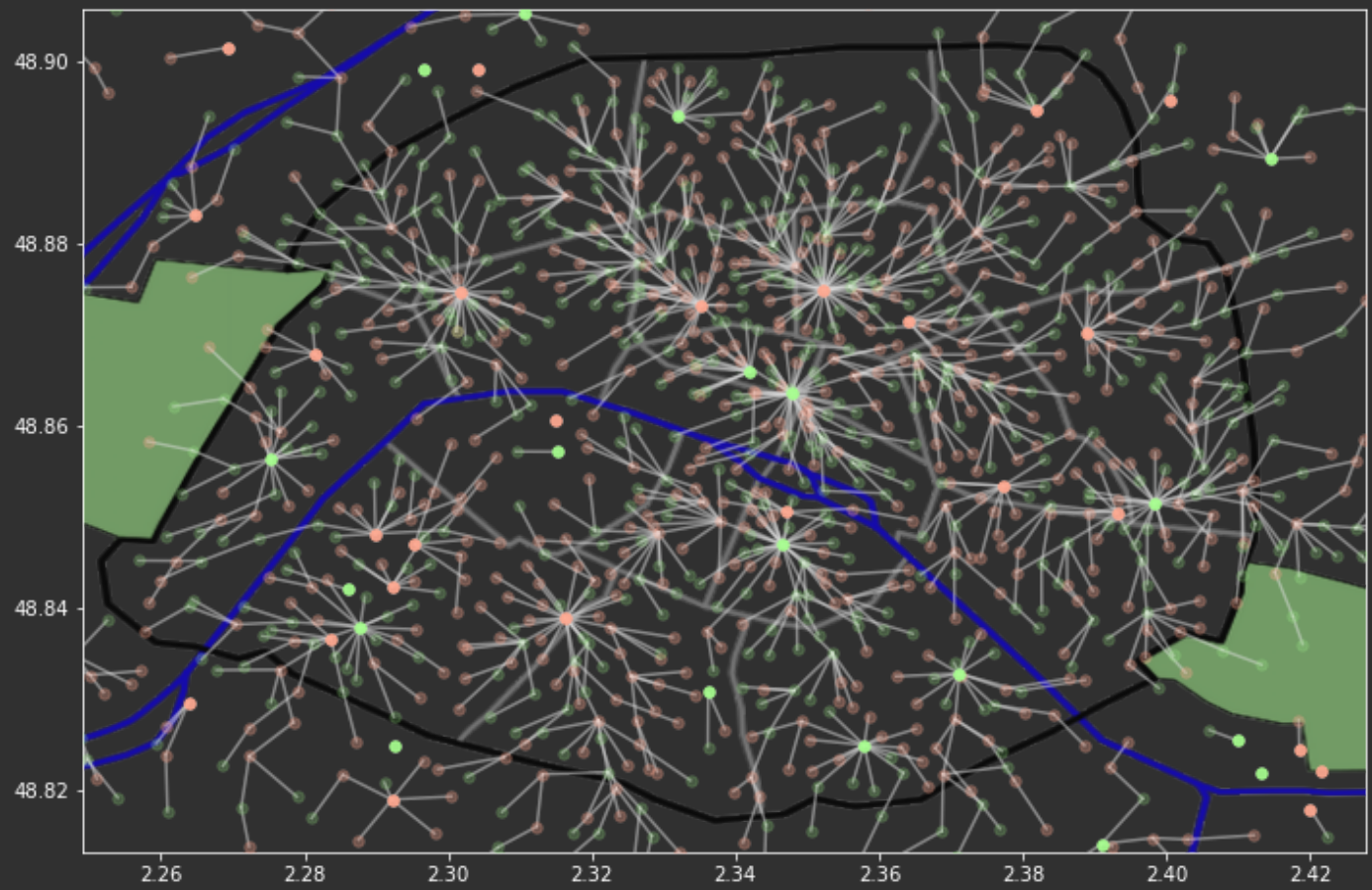
The Velib network : The edges



The delegation graph



Default vote



Propagation

We create an environment with `SimPy`, and let's go.

Thanks for your attention !