Resolving multi-proxy transitive vote delegation

Jonas Degrave[1]

Abstract
Solving a delegation graph for transitive votes is already a non-trivial task for many programmers. When extending the current main paradigm, where each voter can only appoint a single transitive delegation, to a system where each vote can be separated over multiple delegations, solving the delegation graph becomes even harder. This article presents a solution of an example graph, and a non-formal proof of why this algorithm works.

Keywords
Liquid democracy, voting system, delegation, graph, multiple proxies, transitive

Introduction

In the area of voting systems, there is a growing interest into the topic of liquid democracy, in which votes can be delegated to others in order to deal with the complexity of the topic at hand. In these voting systems, many currently use a single proxy system, in which you delegate your vote to one other user. This other user can in its turn further delegate his and your vote in a transitive chain. However, when the user who receives the delegated vote decides not to use it (possibly unintentionally), the vote goes to waste.

However, there is no mathematical need for a user to only delegate to a single other user. In this paper, I propose a multi-proxy delegation system. This system fulfills the following requirements:

- Every user has one vote, which can be delegated.
- The delegation is transitive, meaning that it can be further delegated.
- This vote is distributed equally between all the user's direct delegates who actually vote, potentially through further delegations.

This approach can be used in addition to vote counting systems, as long as each vote in the counting system can be weighted with the total number of votes each user received.

Preparing the example

Suppose we have the following delegation graph for 25 people, as presented in Figure 1. These people either vote themselves, and are shown as green nodes, or they did not vote, and are shown as blue nodes. We chose this example to show all relevant difficulties encountered in solving the delegation graph.

In order to tackle the problem using a flow graph, there are still multiple problems which need fixing. Therefore, we need the following initial steps:

- Remove all edges starting from people who vote themselves.
- Descend the tree starting from the people who vote themselves, and collect all nodes encountered this way. Remove all nodes not encountered from this descend, as their vote cannot be cast in a meaningful way.

After this process, we end up with the graph depicted in Figure 2.

Solving the delegation graph: the first method

In this section, I will introduce a first approach to solving this delegation graph, based on solving a system of linear equations. As you can see, D receives one vote from C in addition to the vote he already had. Therefore:

$$D = 1 + C.$$

M started with a vote, and receives one third of a vote from I, and half of a vote from J:

$$M = 1 + \frac{1}{3}I + \frac{1}{2}J$$

If we do this for all votes, we end up with the following set of linear equations.

$$\begin{aligned}
A &= 1 \\
C &= 1 \\
D &= 1 + C \\
E &= 1 \\
F &= 1 \\
G &= 1 \\
H &= 1 + G \\
I &= 1 + \frac{1}{2}H \\
J &= 1 + \frac{1}{2}H
\end{aligned}$$

[1]Electronics and Information Systems, Ghent University, Belgium

Corresponding author:
Jonas Degrave, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium
Email: Jonas.Degrave@UGent.be

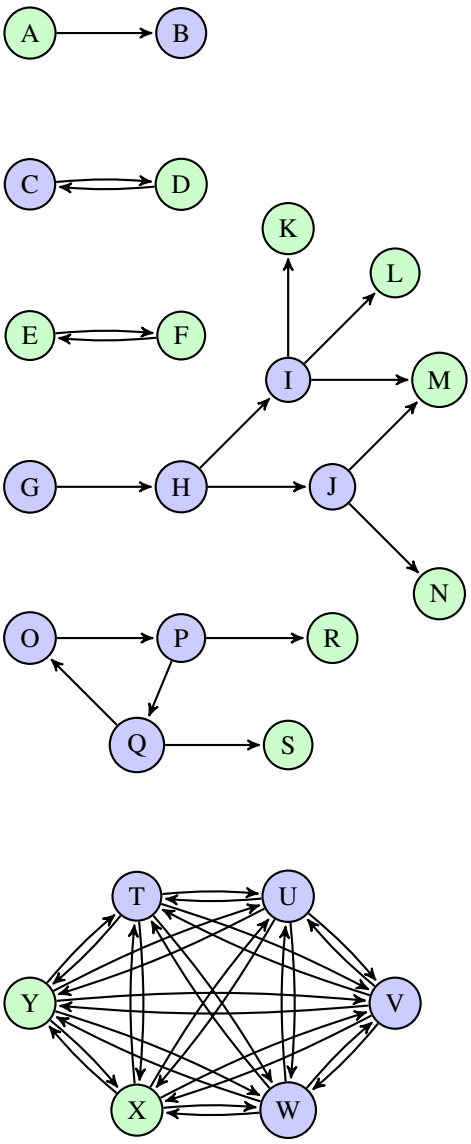

Figure 1. The delegation graph for 25 people we solve as an example in this paper. These people either vote themselves, and are shown as green nodes, or they did not vote, and are shown as blue nodes. Every edge from A to B represents a delegation from person A to person B.

$$
\begin{aligned}
K &= 1 + \frac{1}{3}I \\
L &= 1 + \frac{1}{3}I \\
M &= 1 + \frac{1}{3}I + \frac{1}{2}J \\
N &= 1 + \frac{1}{2}J \\
O &= 1 + \frac{1}{2}Q \\
P &= 1 + O \\
Q &= 1 + \frac{1}{2}P \\
R &= 1 + \frac{1}{2}P \\
S &= 1 + \frac{1}{2}Q \\
T &= 1 + \frac{1}{5}U + \frac{1}{5}V + \frac{1}{5}W
\end{aligned}
$$

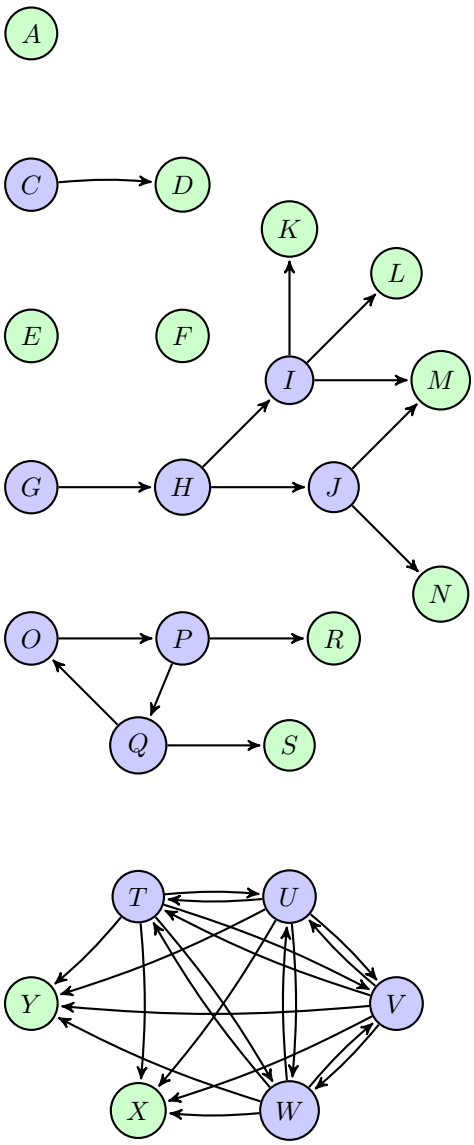

Figure 2. The simplified delegation graph.

$$
\begin{aligned}
U &= 1 + \frac{1}{5}T + \frac{1}{5}V + \frac{1}{5}W \\
V &= 1 + \frac{1}{5}T + \frac{1}{5}U + \frac{1}{5}W \\
W &= 1 + \frac{1}{5}U + \frac{1}{5}V + \frac{1}{5}W \\
X &= 1 + \frac{1}{5}T + \frac{1}{5}U + \frac{1}{5}V + \frac{1}{5}W \\
Y &= 1 + \frac{1}{5}T + \frac{1}{5}U + \frac{1}{5}V + \frac{1}{5}W
\end{aligned}
$$

We can solve system this linear system exactly for all variable A through Y, by converting it to a a matrix system, as shown in equation 3.

This system can easily be solved by inverting the matrix $\mathbf{B}$, if it is not singular, and $\mathbf{B}$ is never singular, since it represents a bijective transformation.

$$\mathbf{S} = \mathbf{B}^{-1} \cdot \mathbf{J}, \tag{1}$$

This solution is shown in equation 4.

Solving the delegation graph: the second method

A second way to look at this, is by constructing the adjacency matrix D of our directed graph. Now, we know that after zero steps in the delegation chain $\mathbf{S} = \mathbf{J}$. After one step in the delegation chain $\mathbf{S} = \mathbf{A}.\mathbf{J}$, after two steps $\mathbf{S} = \mathbf{A}^2.\mathbf{J}$, and so on. Therefore the total number of votes everybody has after an infinite number of delegation steps is

$$\mathbf{S} = \lim_{n \to +\infty} \sum_{i=0}^{n} \mathbf{A}^n \cdot \mathbf{J},$$

and since $\lim_{n \to +\infty} \mathbf{A}^n = \mathbf{0}$,

$$\mathbf{S} = \left(\lim_{n \to +\infty} \sum_{i=0}^{n} \mathbf{A}^n \right) \cdot \mathbf{J}.$$

Because this is a Neumann series

$$\mathbf{S} = (\mathbf{I} - \mathbf{A})^{-1} \cdot \mathbf{J}. \tag{2}$$

Note that $\mathbf{B} = \mathbf{I} - \mathbf{A}$, and hence, the solutions in 1 and 2 are identical.

How to interprete the results

We could just look at the matrix $\mathbf{S}$ to find out the total amount of votes the people who actually voted received from the delegation graph. It might be tempting to interpret the elements of $\mathbf{S}$ of nodes not voting, as the number of votes they would have had, if they would have voted. This is not correct. Take for example the amount of votes of node O. The matrix $\mathbf{S}$ indicates that O would have had 2.333 votes. This is not true, you can easily verify that O actually only would have had 1.75 votes. The reason is that the edge $O \to P$ is removed if O votes, changing the flow of votes altogether.

But there is more information: the matrix $\mathbf{B}^{-1}$ contains more useful information about the origin of each vote. If we take for instance the solution to our specific problem shown in equation 4, we can see that each element of $\mathbf{B}_{ij}$ indicates the contribution of node j to the amount of votes of i, if and only if i actually votes. This allows for detailed feedback to each user as to why he voted exactly this, or to why he voted with a certain amount of votes. It is however not possible to trace the exact route of each vote, since it might be – and often is – infinitely long.

How to implement the algorithm

While inverting a matrix is a very common operation, this is non-trivial for large networks. A matrix inversion is of the order $O(N^3)$ for speed and $O(N^2)$ with N the rank of the matrix, here the number of nodes. Since our matrix is however most likely sparse, since the expected number of delegations per user $E[D] \ll N$. This allows for more specialized approaches developed especially for sparse matrices, such that the the memory use scales with $O(ND)$ and the speed scales approximately with $O(N^2)$ as well (Li, 2009). Since usually $E[D]$ is really small, we have used the algorithm provided in the scipy package without notable problems, even though it is slower. It does have the benefit of only needing a limited amount of memory.

Further extensions

This approach can easily be extended to allow for other ideas in liquid voting systems, such as decaying votes when the trust chain grows (by making the sum of outgoing edges smaller than 1), or where the user can weigh the distribution of his vote for his proxies autonomously rather than presuming the vote is distributed equally.

References

Li, S. (2009). *Fast algorithms for sparse matrix inverse computations* (Unpublished doctoral dissertation).

Author Biographies

Jonas Degrave received a M.Sc. degree in electronics engineering at Ghent University in 2012, where he is currently pursuing a Ph.D. in computer science. His research interests are focused on machine learning and how it can be applied on robotics in order to generate more efficient locomotion.

(3)

$$
\mathbf{J}\begin{Bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{Bmatrix}
=
\mathbf{B}\begin{bmatrix}
1 & & & & & & & & & & & & & & & & & & & & & & & \\
 & 1 & & & & & & & & & & & & & & & & & & & & & & \\
 & & -1 & 1 & & & & & & & & & & & & & & & & & & & & \\
 & & & & 1 & & & & & & & & & & & & & & & & & & & \\
 & & & & & 1 & & & & & & & & & & & & & & & & & & \\
 & & & & & & 1 & & & & & & & & & & & & & & & & & \\
 & & & & & & & -1 & 1 & & & & & & & & & & & & & & & \\
 & & & & & & & & & -\frac{1}{2} & -\frac{1}{2} & & & & & & & & & & & & & \\
 & & & & & & & & & & & 1 & & & & & & & & & & & & \\
 & & & & & & & & & & & & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & & & & & & & & & \\
 & & & & & & & & & & & & & & & -\frac{1}{2} & -\frac{1}{2} & & & & & & & \\
 & & & & & & & & & & & & & & & & & 1 & & & & & & \\
 & & & & & & & & & & & & & & & & & & 1 & & & & & \\
 & & & & & & & & & & & & & & & & & & & 1 & & & & \\
 & & & & & & & & & & & & & & & & & & & & 1 & & & \\
 & & & & & & & & & & & & & & & & & & & & & 1 & & \\
 & & & & & & & & & & & & & & & & & & & & & & 1 & \\
 & & & & & & & & & & & & & & & & & & & & & & & 1
\end{bmatrix}
\cdot
\mathbf{S}\begin{Bmatrix}A\\C\\D\\E\\F\\G\\H\\I\\J\\K\\L\\M\\N\\O\\P\\Q\\R\\S\\T\\U\\V\\W\\X\\Y\end{Bmatrix}
$$

$$
\overbrace{\begin{pmatrix} A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \\ M \\ N \\ O \\ P \\ Q \\ R \\ S \\ T \\ U \\ V \\ W \\ X \\ Y \end{pmatrix}}^{\mathbf{s}}
=
\begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1.5 \\ 1.5 \\ 1.5 \\ 1.5 \\ 2.25 \\ 1.75 \\ 2.333333 \\ 3.333333 \\ 2.666667 \\ 2.666667 \\ 2.333333 \\ 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \\ 3 \\ 3 \end{pmatrix}
=
\overbrace{\begin{bmatrix}
1 & & & & & & & & & & & & & & & & & & & & & & & \\
 & 1 & & & & & & & & & & & & & & & & & & & & & & \\
 & 1 & 1 & & & & & & & & & & & & & & & & & & & & & \\
 & & & 1 & & & & & & & & & & & & & & & & & & & & \\
 & & & & 1 & & & & & & & & & & & & & & & & & & & \\
 & & & & & 1 & & & & & & & & & & & & & & & & & & \\
 & & & & & & 1 & & & & & & & & & & & & & & & & & \\
 & & & & & 0.500 & & 1 & & & & & & & & & & & & & & & & \\
 & & & & & 0.500 & & & 1 & & & & & & & & & & & & & & & \\
 & & & & & 0.167 & 0.333 & & & 1 & & & & & & & & & & & & & & \\
 & & & & & 0.167 & 0.333 & & & & 1 & & & & & & & & & & & & & \\
 & & & & & 0.417 & 0.333 & 0.500 & & & & 1 & & & & & & & & & & & & \\
 & & & & & 0.250 & & 0.500 & & & & & 1 & & & & & & & & & & & \\
 & & & & & & & & & & & & & 1.333 & 0.333 & 0.667 & & & & & & & & \\
 & & & & & & & & & & & & & 1.333 & 1.333 & 0.667 & & & & & & & & \\
 & & & & & & & & & & & & & 0.667 & 0.667 & 1.333 & & & & & & & & \\
 & & & & & & & & & & & & & 0.667 & 0.667 & 0.333 & 1 & & & & & & & \\
 & & & & & & & & & & & & & 0.333 & 0.333 & 0.667 & & 1 & & & & & & \\
 & & & & & & & & & & & & & & & & & & 1.250 & 0.417 & 0.417 & 0.417 & & \\
 & & & & & & & & & & & & & & & & & & 0.417 & 1.250 & 0.417 & 0.417 & & \\
 & & & & & & & & & & & & & & & & & & 0.417 & 0.417 & 1.250 & 0.417 & & \\
 & & & & & & & & & & & & & & & & & & 0.417 & 0.417 & 0.417 & 1.250 & & \\
 & & & & & & & & & & & & & & & & & & 0.500 & 0.500 & 0.500 & 0.500 & 1 & \\
 & & & & & & & & & & & & & & & & & & 0.500 & 0.500 & 0.500 & 0.500 & & 1
\end{bmatrix}}^{\mathbf{B}^{-1}}
\cdot
\overbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}}^{\mathbf{J}}
\tag{4}
$$