Network Science Summer School

Universiteit Utrecht

Day program

09:00–10:00:

Introductions

10:00–12:00:

Introduction to network science

Practical + discussion

12:00-13:00

Lunch

13:00–16:30:

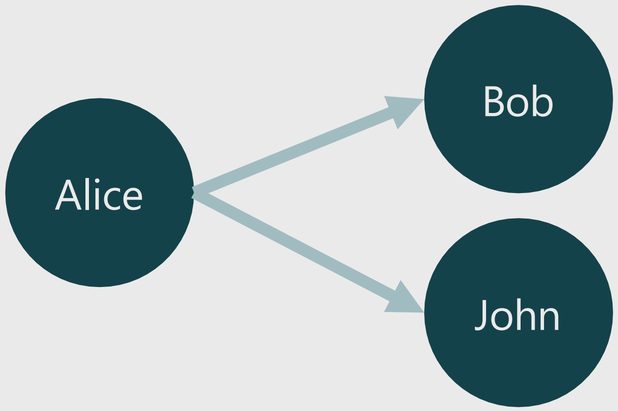
Network representation

Centrality

Intro to linear algebra

Why? Multiplying sparse matrices is fast (relatively)

Network representation



Adjacency list: (edgelist)
- Adv: It is dense: Only keeping edges
- Disadvantage: Hard to work with

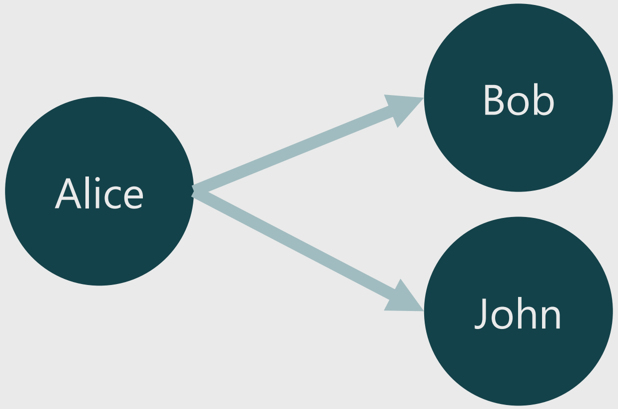
Source	Target	Weigth
Alice	Bob	1
Alice	John	1

Adjacency matrix:
- Adv: Linear algebra is easy
- Disadvantage: It is sparse (mostly zeros).1E6 nodes → 1 trillion options

Target → ↓ Source	Alice	Bob	John
Alice	0	1	1
Bob	0	0	0
John	0	0	0

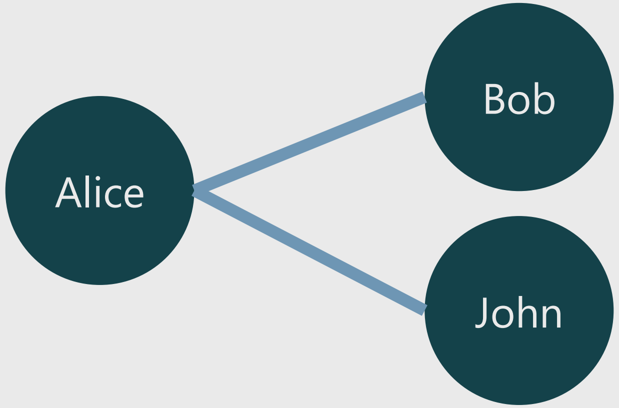
In computers → Sparse matrices: Best of both worlds

Directed networks



Target → ↓ Source	Alice	Bob	John
Alice	0	1	1
Bob	0	0	0
John	0	0	0

Undirected networks



Target → ↓ Source	Alice	Bob	John
Alice	0	1	1
Bob	1	0	0
John	1	0	0

Some terms

A =

Target → ↓ Source	Alice	Bob	John
Alice	0	1	1
Bob	0	0	0
John	0	0	0

Diagonal

Trace = Sum of elements in the diagonal

Identity matrix (I) =

	1	0	0
	0	1	0
	0	0	1

I @ A = A

Transpose (A^T, A') =

(python) A.T

Target → ↓ Source	Alice	Bob	John
Alice	0	0	0
Bob	1	0	0
John	1	0	0

Symmetric matrix: A = A.T (e.g. undirected network)

Python exercise notebook 2, ex.1

Python:

- Convert between formats
- Plot matrix

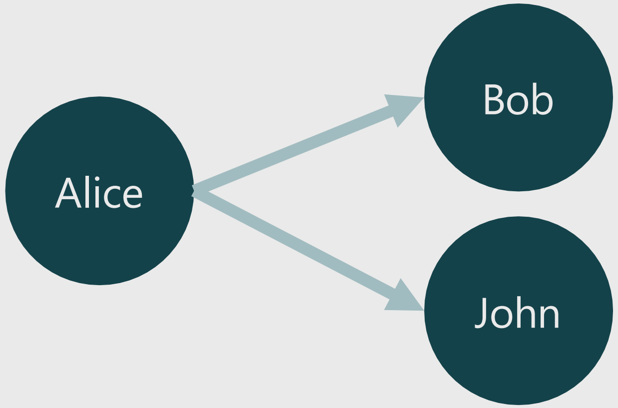
Transposing = changing the direction

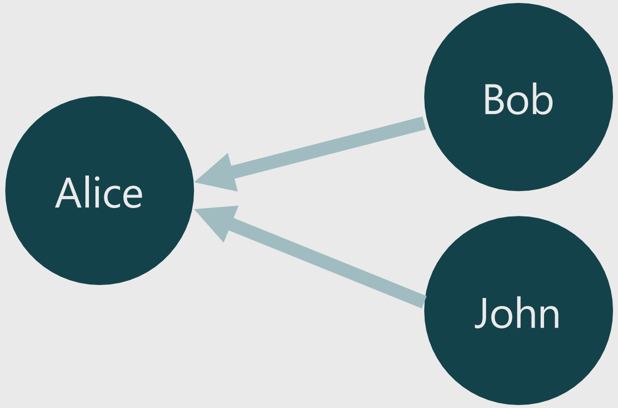
A =

Target → ↓ Source	Alice	Bob	John
Alice	0	1	1
Bob	0	0	0
John	0	0	0

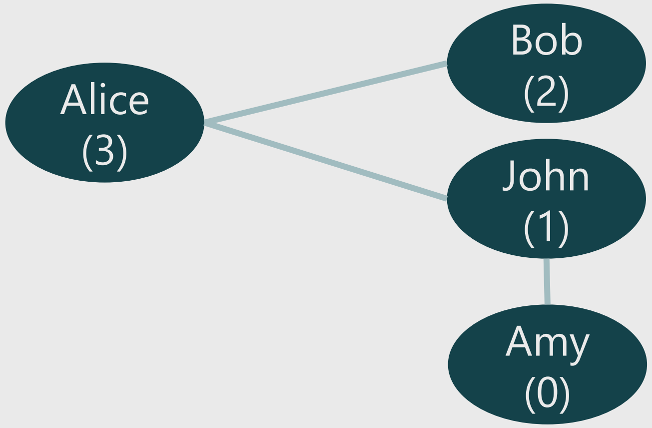
A.T =

Target → ↓ Source	Alice	Bob	John
Alice	0	0	0
Bob	1	0	0
John	1	0	0





Matrix multiplication: sum



Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	1	0	0	0
John	1	0	0	1
Amy	0	0	1	0

@

Node	kids
Alice	3
Bob	2
John	1
Amy	0

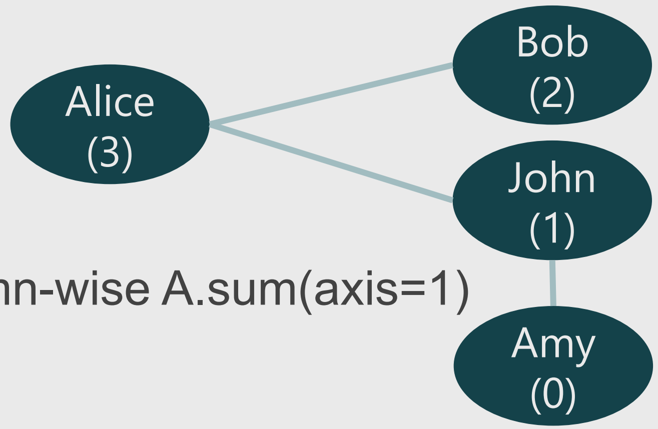
=

Node	kids
Alice	0*3 + 1*2 + 1*1 + 0*0 = 3
Bob	1*3 + 0*2 + 0*1 + 0*0 = 3
John	1*3 + 0*2 + 0*1 + 1*0 = 3
Amy	0*3 + 0*2 + 1*1 + 0*0 = 1

A @ M = SM

(N x N) @ (N x 1) = (N x 1)

Matrix multiplication: average



Divide by the degree. We get it by summing the adjacency elements column-wise A.sum(axis=1)

A @ M / A.sum(1)

(N x N) @ (N x 1) / (N x 1) = (N x 1) / (N x 1) = (N x 1)

Target → ↓ Origin	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	1	0	0	0
John	1	0	0	1
Amy	0	0	1	0

@

Node	Kids
Alice	3
Bob	2
John	1
Amy	0

divided by

Target → ↓ Source	Sum
Alice	2
Bob	1
John	2
Amy	1

=

Node	Kids
Alice	3
Bob	3
John	3
Amy	1

divided by

Target → ↓ Source	Sum
Alice	2
Bob	1
John	2
Amy	1

=

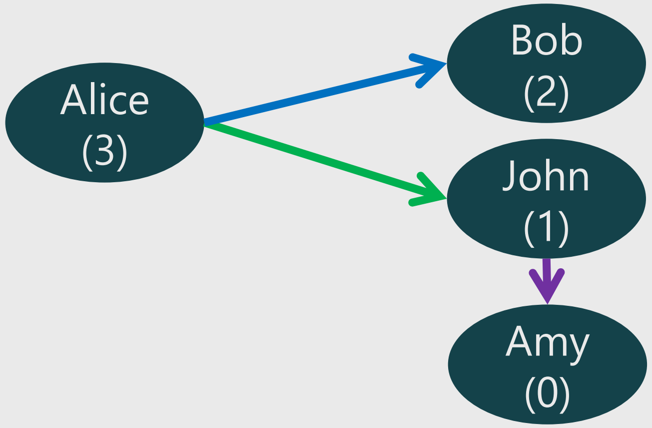
Node	Kids
Alice	1.5
Bob	3
John	1.5
Amy	1

Python exercise notebook 2, ex.2

Calculate the average number of children of your friends using matrix multiplication

Matrix multiplication: paths

Interpretation A: Presence of path between node i and j



A^2 =

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

@

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

=

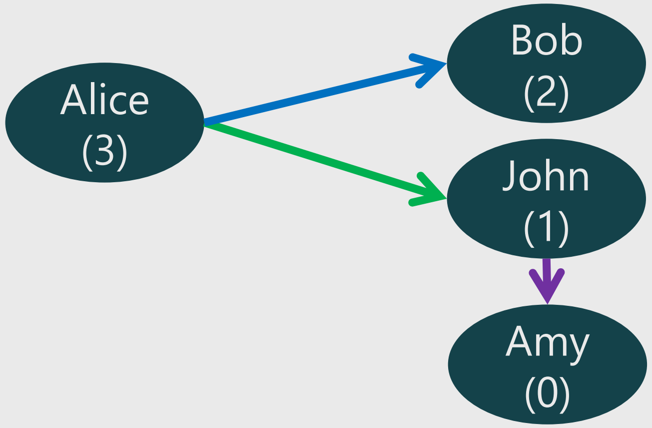
Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0	0	1
Bob	0	0	0	0
John	0	0	0	0
Amy	0	0	0	0

From you → to your neighbors

From your neighbors → to their neighbors

You → the neig. of your neig.

Matrix multiplication: paths



Interpretation A: Presence of path between node i and j
Interpretation A^2: Number of path between node i and j in two steps
Interpretation A^3: Number of path between node i and j in three steps
…

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

@

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

=

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0	0	1
Bob	0	0	0	0
John	0	0	0	0
Amy	0	0	0	0

Alice → Alice (0) * Alice → Amy (0)
+ Alice → Bob (1) * Bob → Amy (0)
+ Alice → John (1) * John → Amy (1)
+ Alice → Alice (0) * Alice → Amy (1)

Python exercise notebook 2, ex.3a

Matrix multiplication: number of people reached in <3 steps

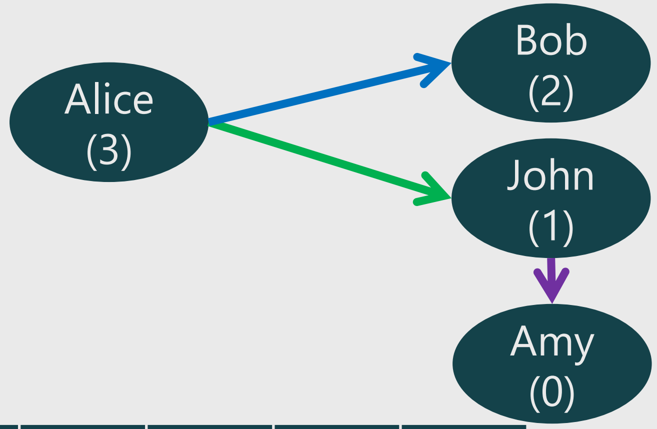
Number of paths in two or three steps from node i to node j: ***N = A + A^2 + A^3***
We need to remove duplicate paths: ***N = N>0***
We need to remove paths from us to ourselves ***N.setdiag(0)***

Matrix multiplication: number of triangles

Number of paths in two or three steps from node i to node j in three steps: ***A^3***
We are interested in the diagonal

Undirected network? Divide the triangles by two (two directions)
Counting the total number of triangles? Divide the trace by 3

Matrix multiplication: number of triangles



A^2

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0	0	1
Bob	0	0	0	0
John	0	0	0	0
Amy	0	0	0	0

@

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

=

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0	0	0
Bob	0	0	0	0
John	0	0	0	0
Amy	0	0	0	0

Alice → Alice in two steps * Alice → Alice (0)
Alice → Bob in two steps * Bob -> Alice (0)
Alice → John in two steps * John -> Alice (0)
Alice → Amy in two steps * Amy -> Alice (0)

Diagonal of A^3

Alice → X_1 * X_1 → X_1 * X_1 → Alice +
Alice → X_1 * X_1 → X_2 * X_2 → Alice +
...

Python exercise notebook 2, ex.3b

(already done, just check solutions)

Centrality measures

Nice explanations:

https://aksakalli.github.io/2017/07/17/network-centrality-measures-and-their-visualization.html

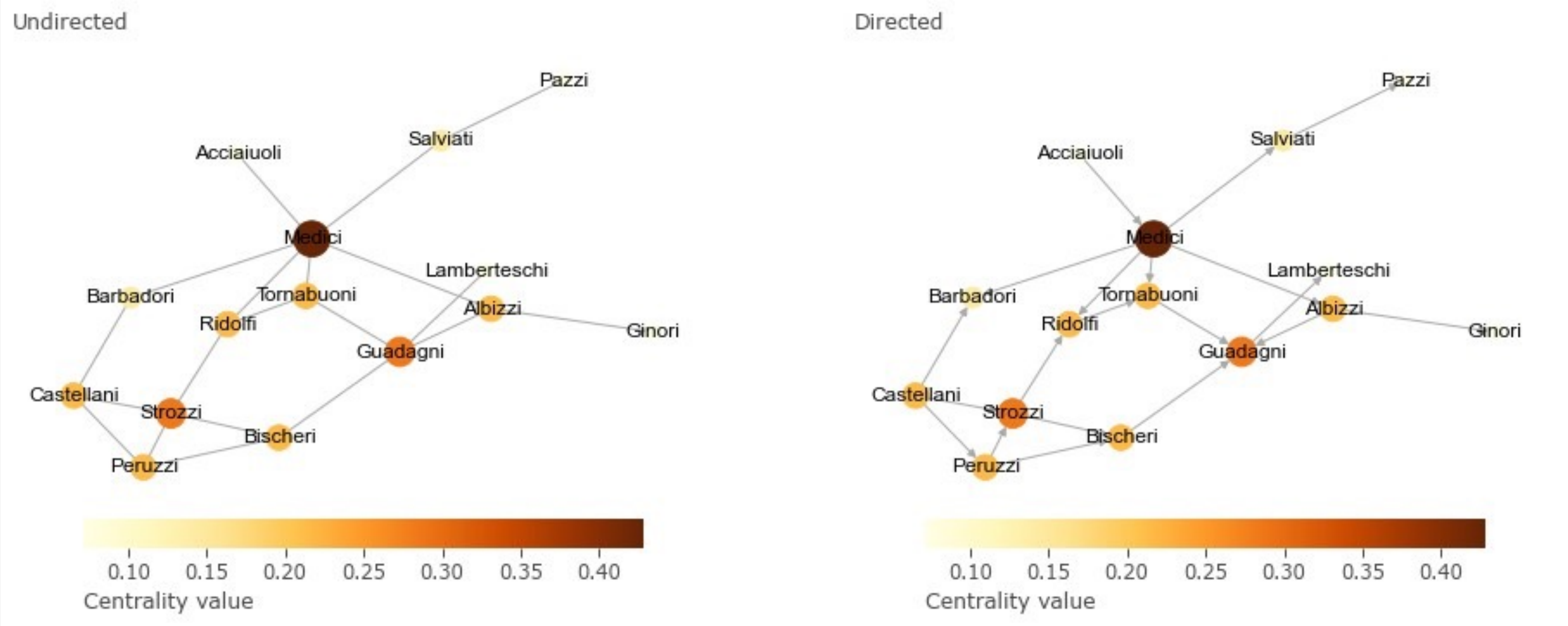
Networks: an introduction (Newman)

Degree centrality = $^{d_i}/_{N\text{-}1}$

d_i = degree of node i

$N - 1$ = number of nodes - 1 (max. potential number of partners without self-edges/multi-edges)

Measures the **local** influence of the node



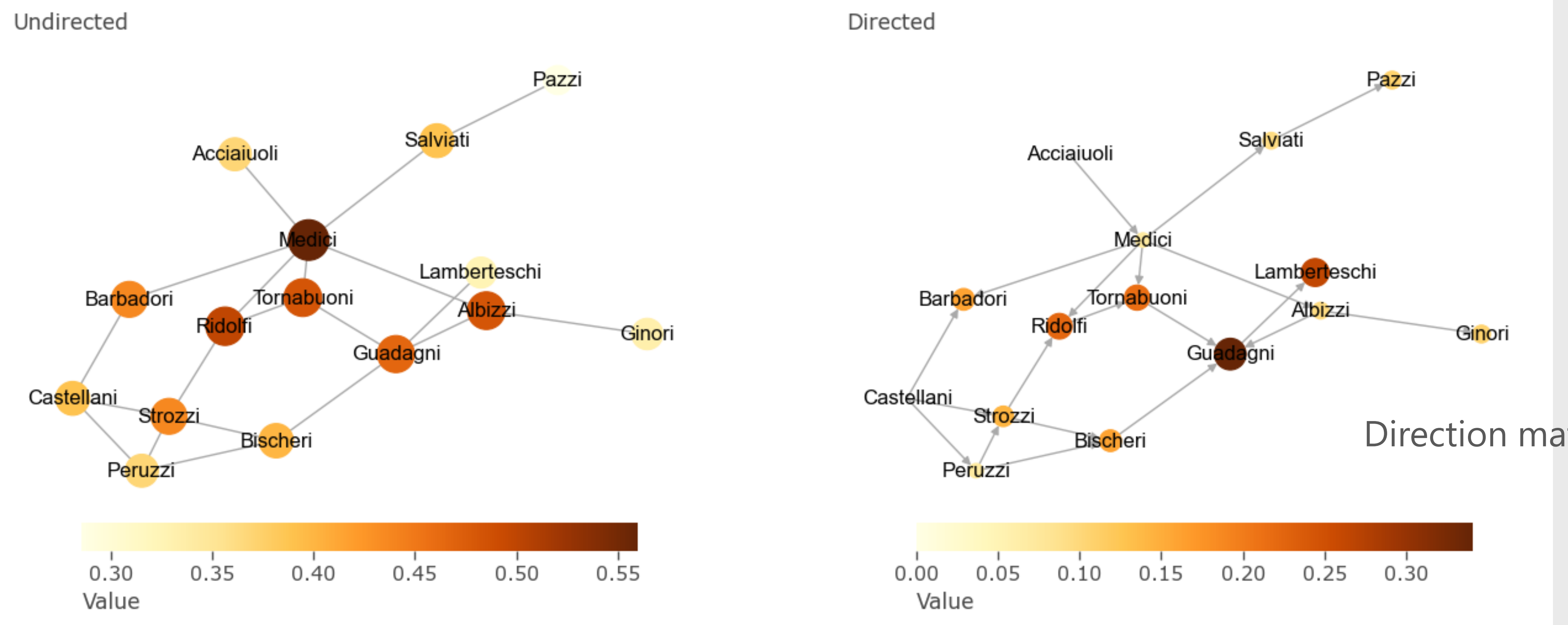
Closeness centrality = $^{1}/_{l_i}$

l_i = average distance of node i to all other nodes := $l_i = \frac{1}{N}\sum_j d_{ij}$

d_{ij} = shortest distance from node i to node j

Only useful in fully connected networks

Measures the **most central** node in the network (closest to get to all other nodes)



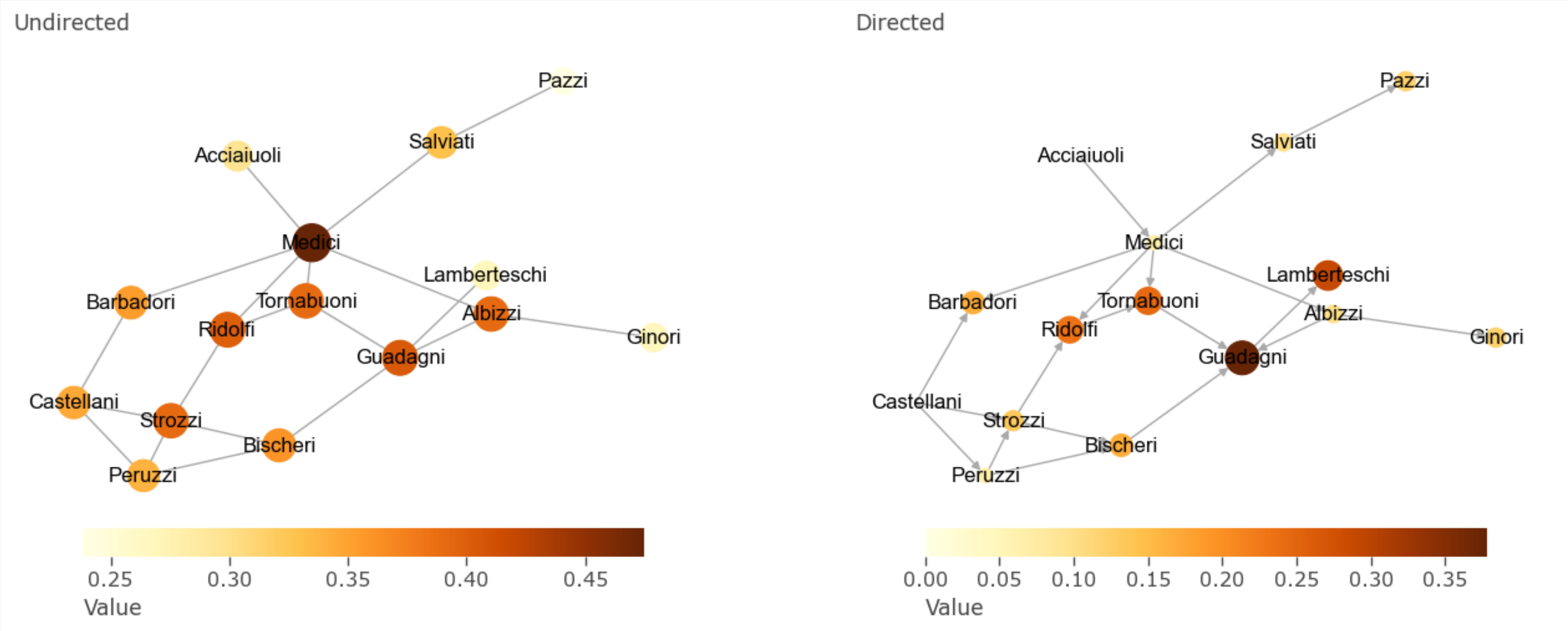
Direction matters!

Harmonic closeness centrality = $^1/_{N\text{-}1} \sum_{ij|i\neq j} {^1/_{d_{ij}}}$

d_{ij} = shortest distance from node i to node j

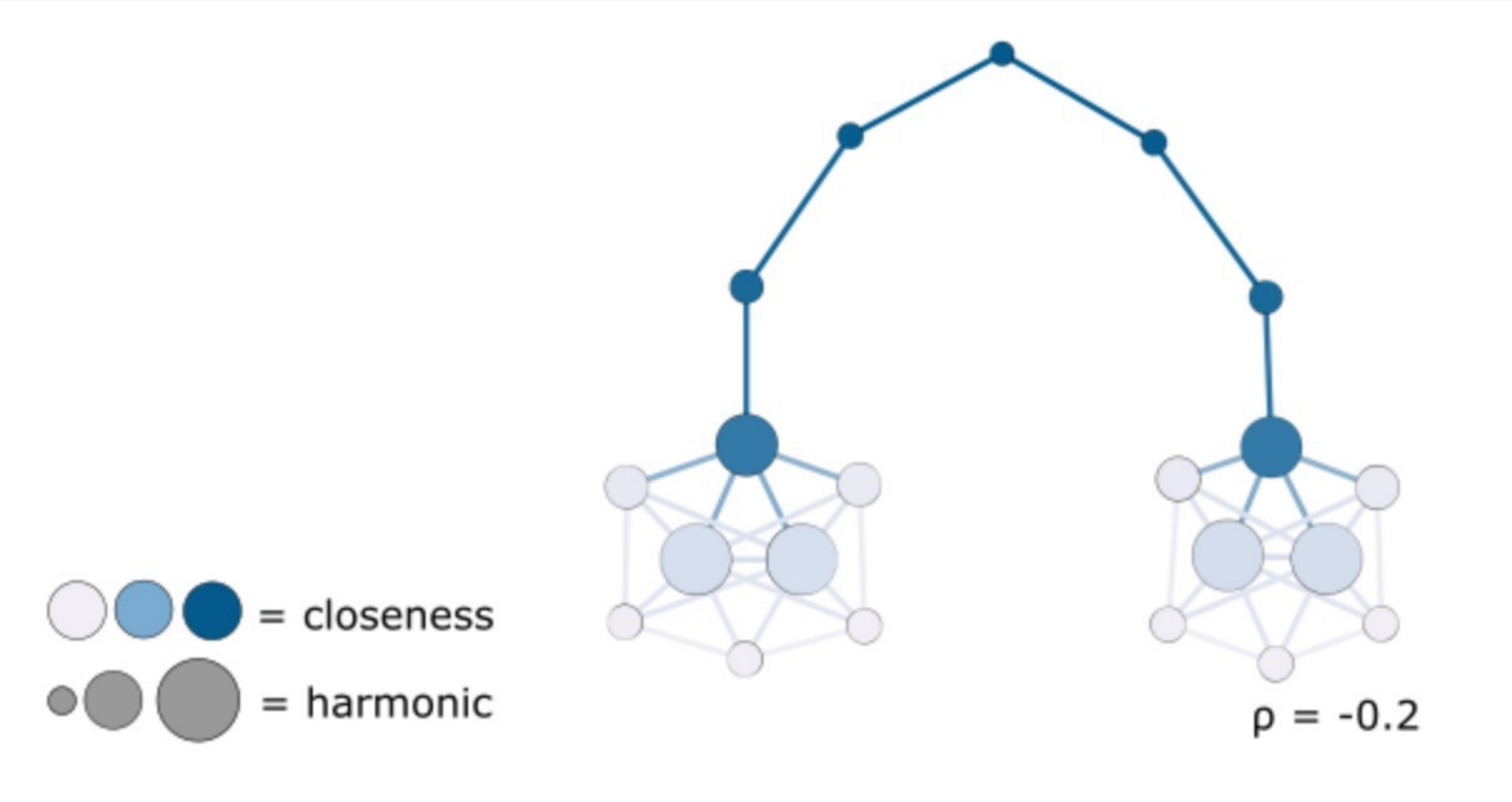
Useful also in disconnected networks. Gives more weight to closer nodes.

Measures the **most central** node in the network (harmonic average)



Bavelas, 1950

Closeness vs harmonic



https://symbio6.nl/en/blog/analysis/harmonic-centrality

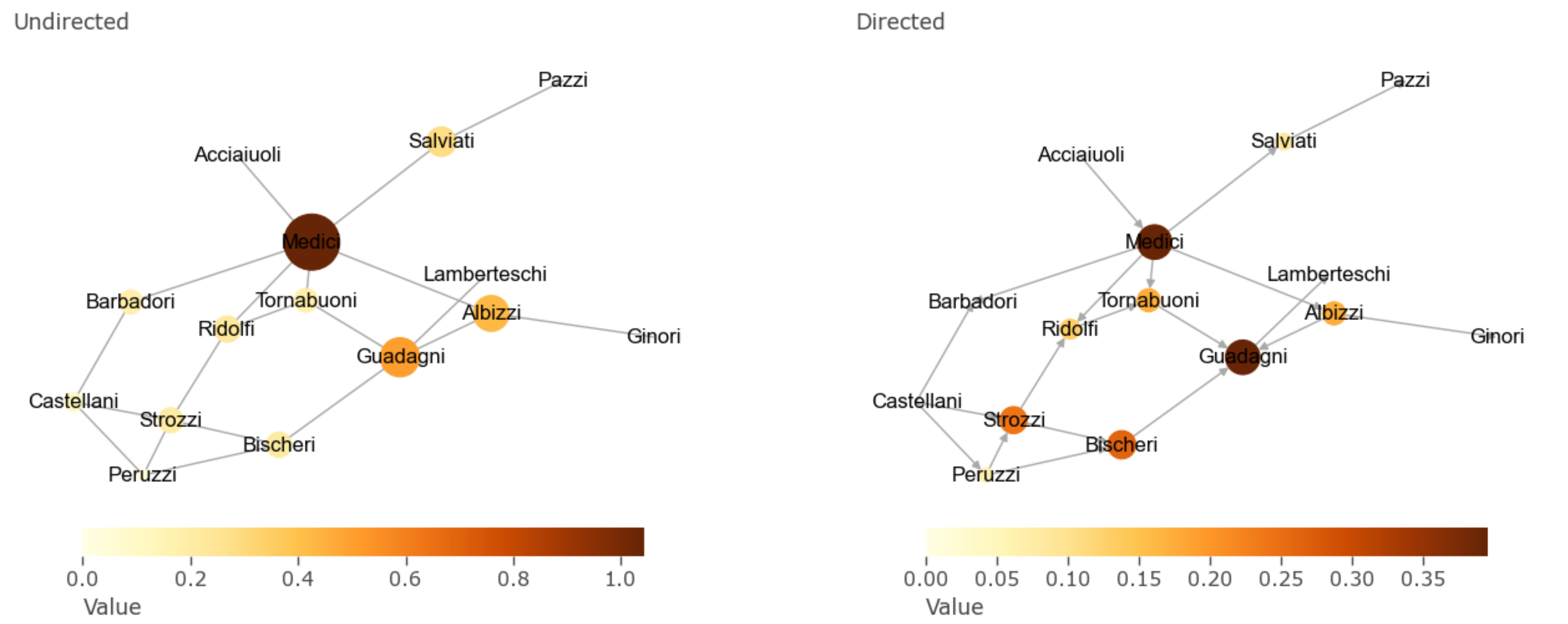
Betweeness centrality = $1/_{n^2} \sum_{st} n_{st}^{i}$

n_{st}^{i} = 1/g if node i lies on the g shortest paths between nodes s and t

Assumptions:

- every pair of nodes in the network exchanges messages at the same average rate
- messages always take the shortest available path though the network

Measures **brokerage** in the network ➔ disruption of these nodes = disruption of communication



Freeman (1977), and Anthonisse (1971, unpublished)

Eigenvector centrality = $\lambda^{-1} \sum_j A_{ij} e_j$

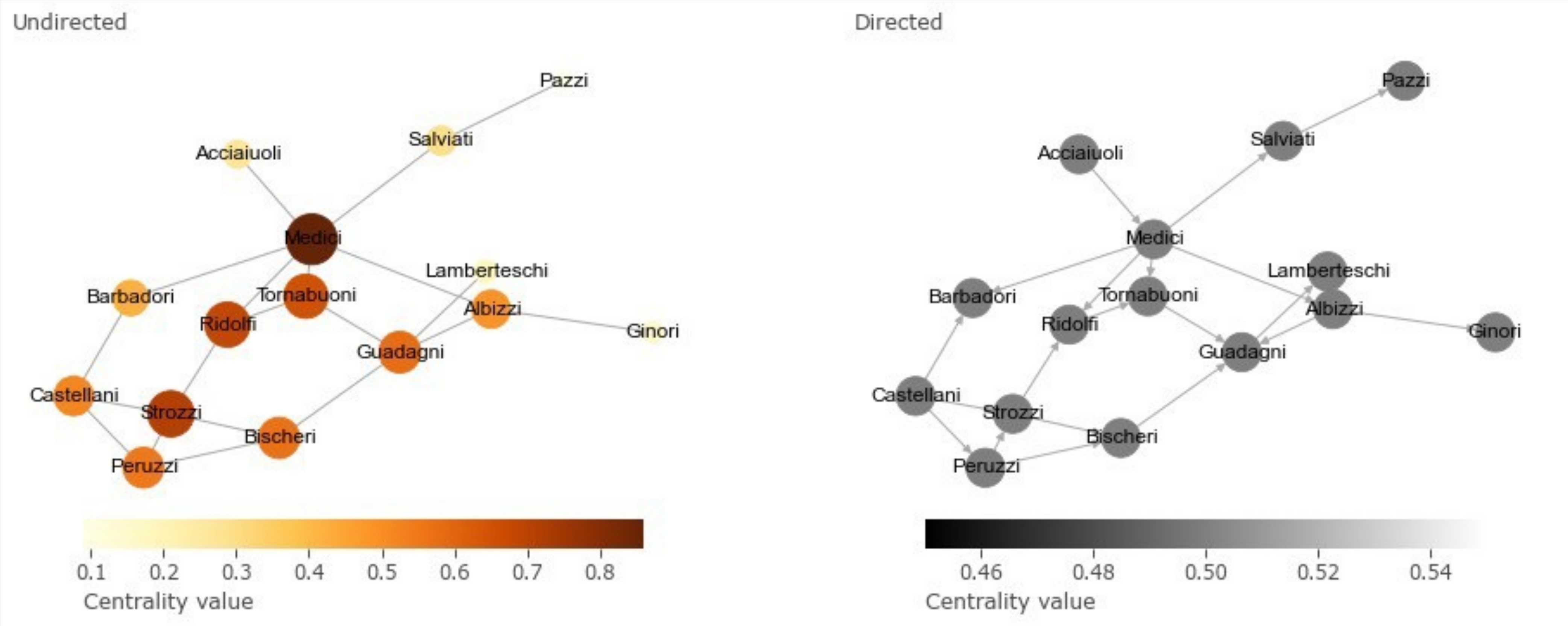
Takes into account how central your neighbors are.

e_j = eigenvector centrality of node j

λ = largest eigenvalue

Measures total **influence** in the network (assuming all nodes are the same)

Only for undirected, fully-connected networks!



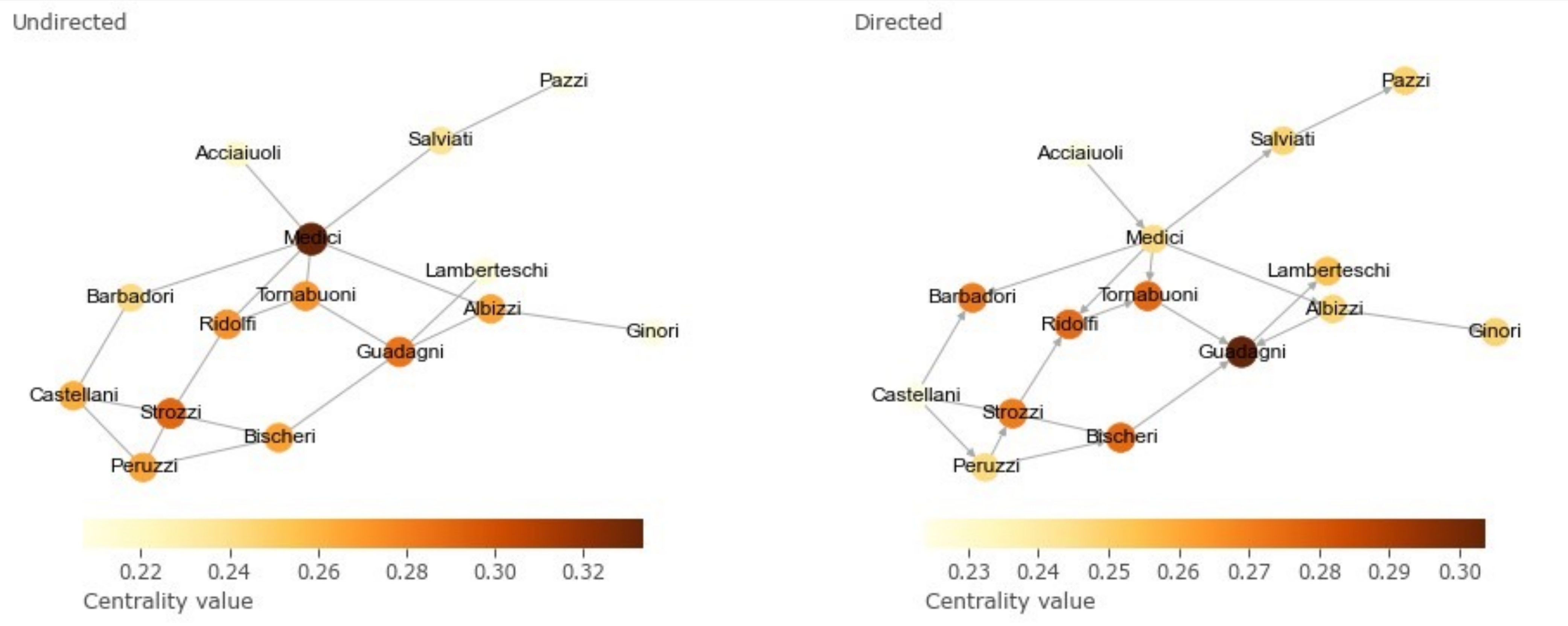
Bonacich, 1987

Katz centrality = $\alpha \sum_j A_{ij} k_j + \beta$

$\boldsymbol{k_j}$ = Katz centrality of node *j*

Takes into account how central your neighbors are, **each node has a minimum value of $\boldsymbol{\beta}$,** and the balance between the constant and the eigenvector part is controlled by $\boldsymbol{\alpha}$

Measures total **influence** in the network (assuming all nodes are the same)



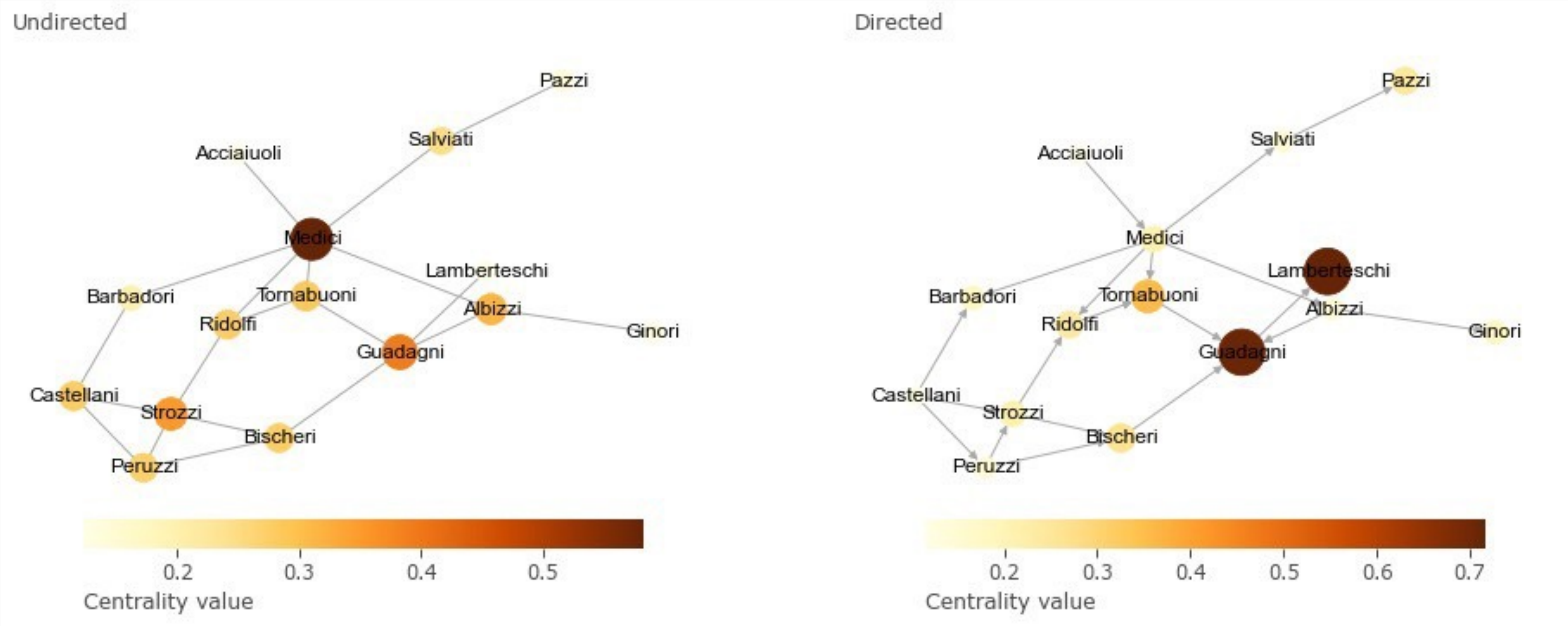
Katz, 1953

Pagerank centrality = $\alpha \sum_j A_{ij} {}^{p_j}/_{d_j} + \beta$

d_j = Degree of node *j*

p_j = Pagerank centrality of node j

Takes into account how central your neighbors are. Each node has a minimum value of β**, the pagerank of your neighbours is normalized by their out-degree,** and the balance between the constant and the eigenvector part is controlled by α



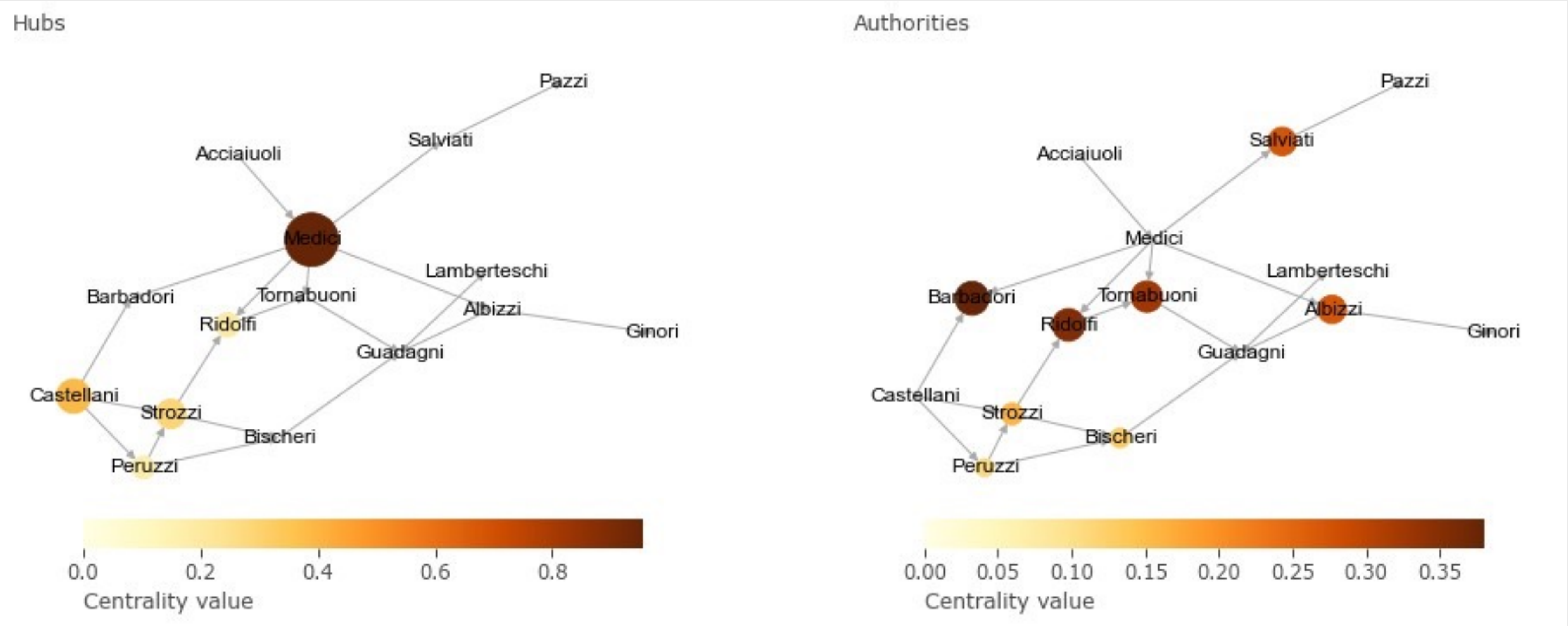
Bonacich, 1987

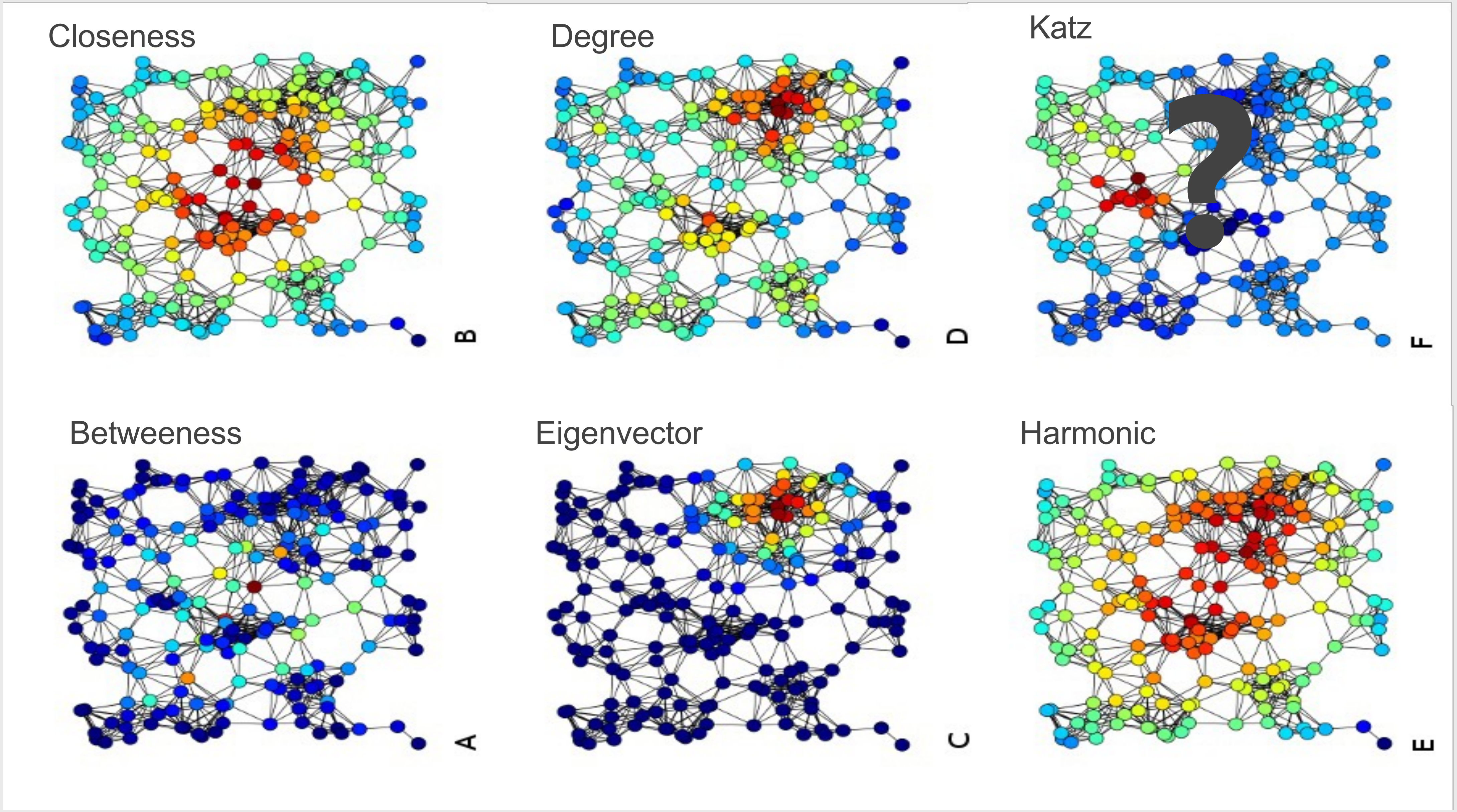
Hubs and authorities (HITS)

A node may be important if it points to others with high centrality, e.g., a review article pointing to prestigious articles

Authorities are nodes that contain useful information on a topic of interest and **hubs** are nodes that tell us where the best authorities are to be found (Newman). Two centralities: authority (a) and hub (h) centrality. Only for directed networks!

$h_\mathrm{i} = \alpha \sum_\mathrm{j} A_\mathrm{ij} a_\mathrm{j}$ and $a_\mathrm{i} = \alpha \sum_\mathrm{j} A_\mathrm{ij} h_\mathrm{j}$

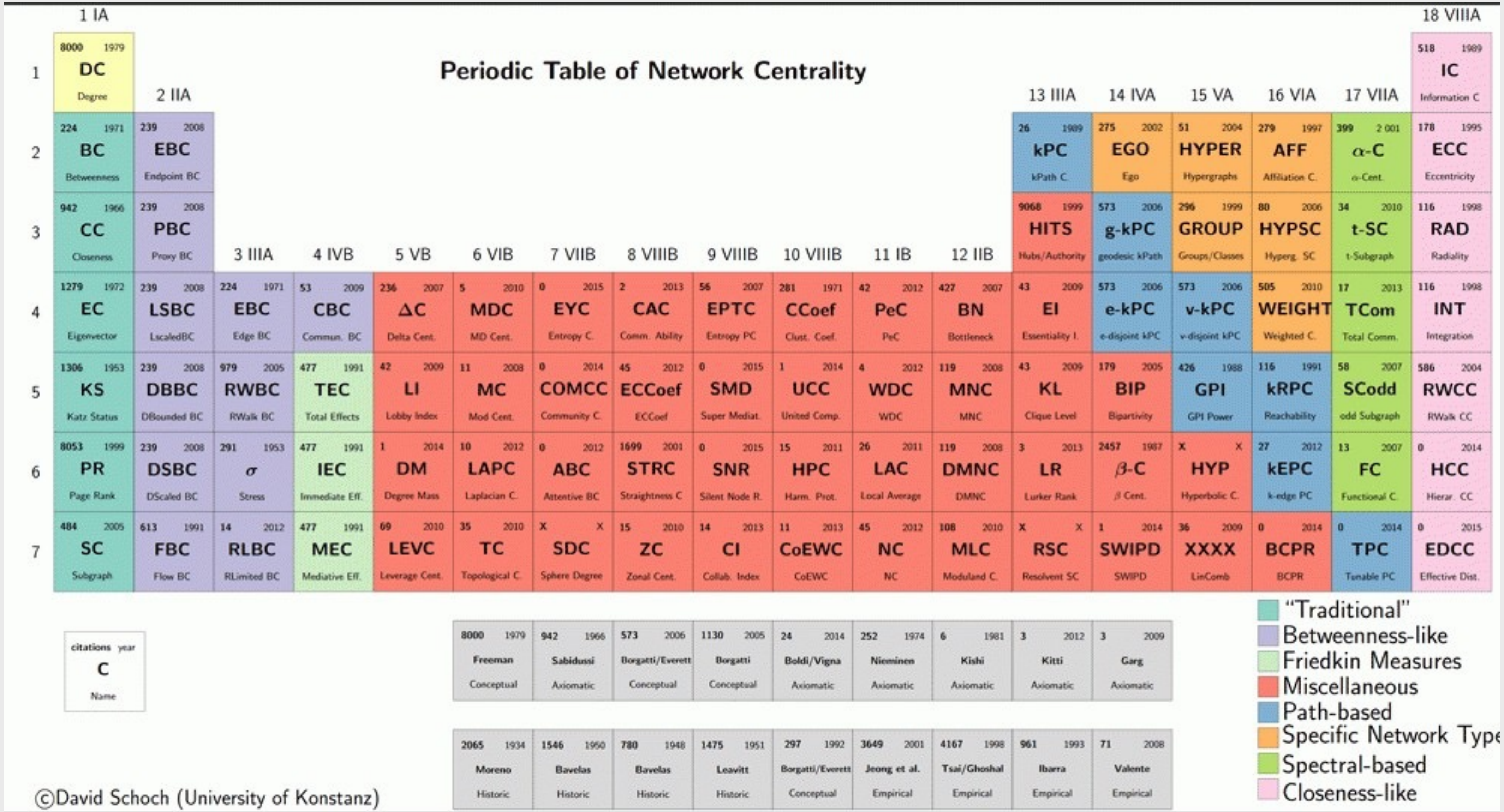




Source: Wikipedia (@Tapiocozzo) – katz centrality looks strange

Use a centrality measure that fits your theory, not the one that gives you the best results

Consider what is the real objective (e.g. is it to enable low-income individuals to increase their social capital?) *(https://petterhol.me/2019/01/11/the-importance-of-being-earnest-about-node-importance/)*



Chains

Sometimes data is represented as chains

- Life trajectory: Aranda → León → Vermont → Amsterdam
- Ownership chain: (right figure)

They allow you to do other analysis:

- Importance of the node based on how often it is found in between
- Importance of the node based on how many people jump to you

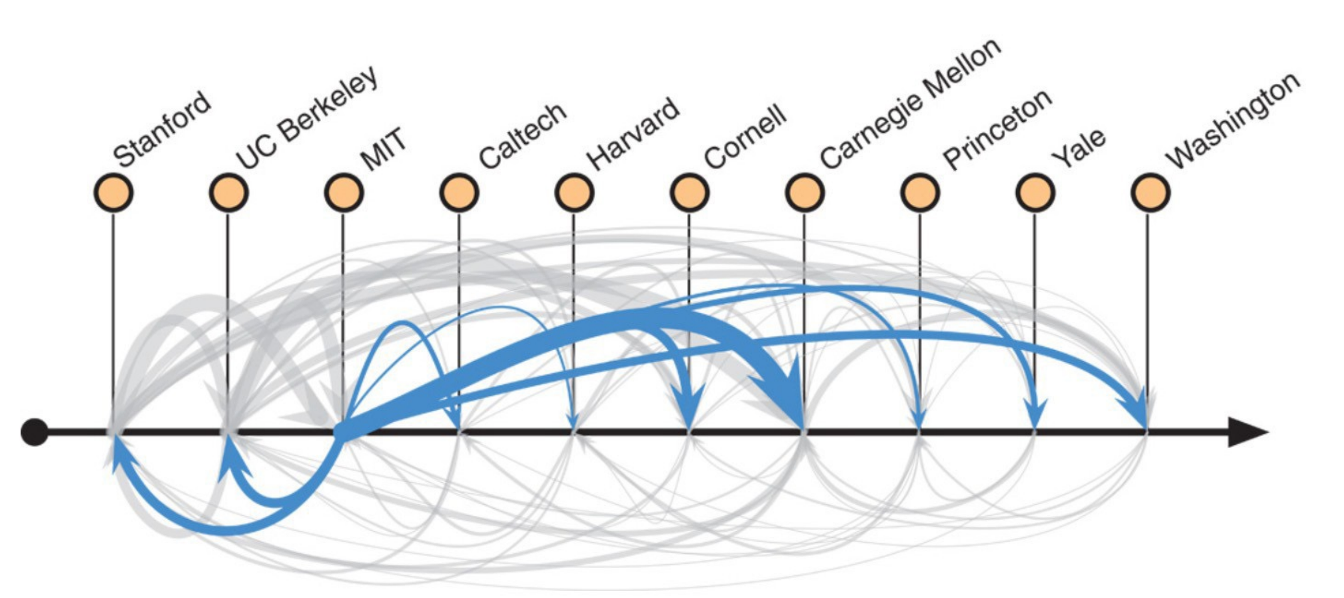
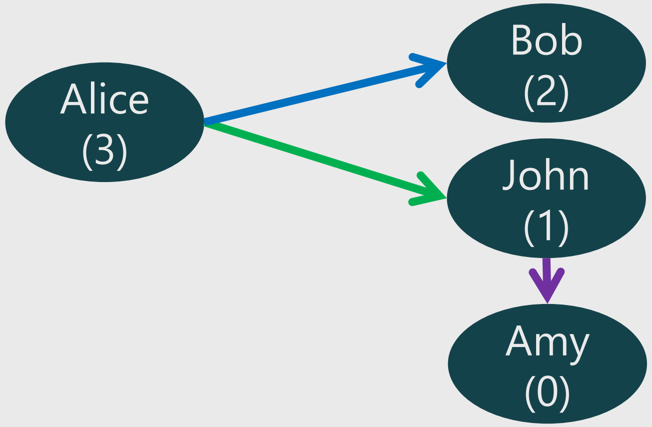


Fig. 1 Prestige hierarchies in faculty hiring networks. *Clauset, Arbesman and Larremore (2015)*

Practical 2:
Exercise 4 and 5

Linear algebra and centrality measures

Degree



Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

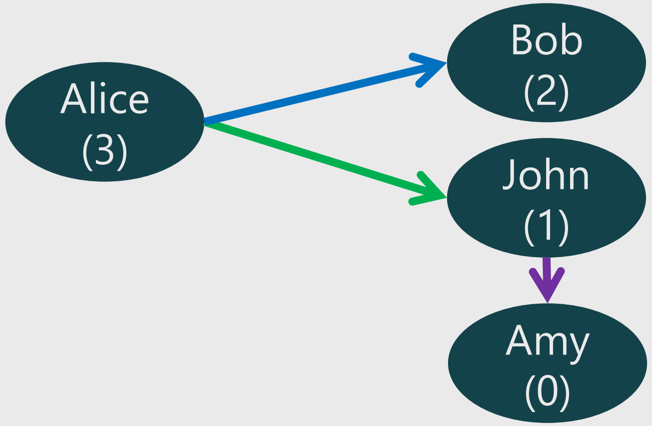
@

Alice	1
Bob	1
John	1
Amy	1

=

	Out-Degree
Alice	2
Bob	0
John	1
Amy	0

Matrix multiplication: paths



Interpretation A: Presence of path between node i and j

Interpretation A^2: Number of path between node i and j in two steps

Interpretation A^3: Number of path between node i and j in three steps

…

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

@

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	1	1	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

=

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0	0	1
Bob	0	0	0	0
John	0	0	0	0
Amy	0	0	0	0

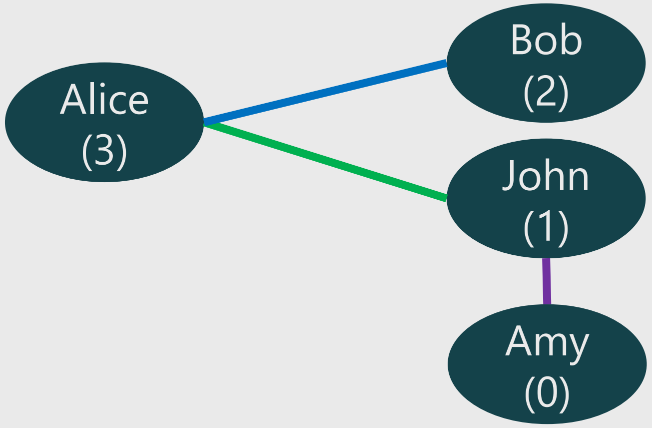
Alice ➔ Alice (0) * Alice ➔ Amy (0)

+ Alice ➔ Bob (1) * Bob ➔ Amy (0)

+ Alice ➔ John (1) * John ➔ Amy (1)

+ Alice ➔ Alice (0) * Alice ➔ Amy (1)

Another view on matrix multiplications: Random walks on undirected networks



Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0.5	0.5	0
Bob	1	0	0	0
John	0.5	0	0	0.5
Amy	0	0	1	0

@

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0.5	0.5	0
Bob	1	0	0	0
John	0.5	0	0	0.5
Amy	0	0	1	0

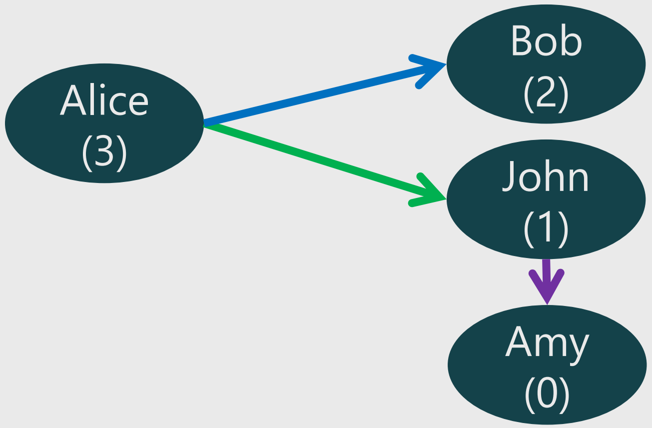
=

A random walker starting in Alice will go 50% of the times to Bob, 50% of the times to John.

- From Bob it goes 100% of the times back to Alice
- From John it goes 50% of the times to John, 50% back to Alice

If we let the random walker walk forever ➔ The fraction of time spend at each node converges to the **degree centrality** of the node

Another view on matrix multiplications: Random walks on directed networks



Transition matrix (row-normalized A)

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0.5	0.5	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

@

Target → ↓ Source	Alice	Bob	John	Amy
Alice	0	0.5	0.5	0
Bob	0	0	0	0
John	0	0	0	1
Amy	0	0	0	0

=

A random walker starting in Alice will go 50% of the times to Bob, 50% of the times to John.

- From Bob it gets trapped
- From John it goes 100% of the times to Amy and gets trapped

If we let random walkers walk forever ➔ They gets trapped in the extremes!

Solution: PageRank (the beta parameter can be understood as a teletransportation probablity)

Practical 3:
Working with networks using Gephi

- Follow this tutorial (slides 1–23 only!): https://gephi.org/users/quick-start/
- In community detection use the "stochastic blockmodel" instead of modularity maximization (or try both)
- You can choose to use our own data (https://tinyurl.com/network-game) or the Twitter data.

Python exercise
notebook 2, ex.7