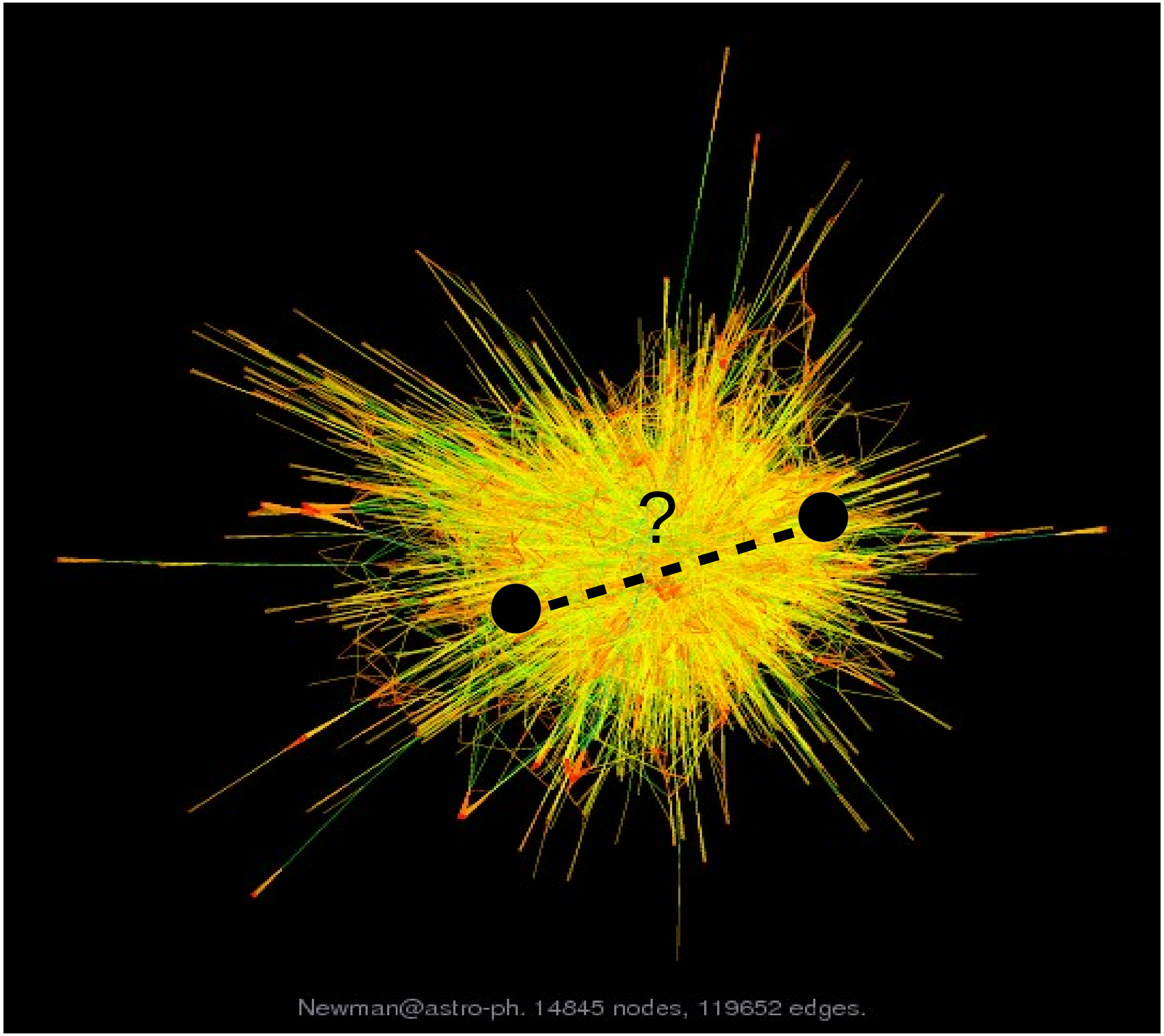
Social matching and link prediction

Link prediction

- Given a snapshot of a social network
- **Question**: infer which new interactions among its members are likely to occur in the near future [Liben-Nowell and Kleinberg, 2004, 2007]

Link prediction in a nutshell



Why link prediction

- Social matching
  - See further in this presentation
- Inferring missing links
- Testing models of evolving networks
  - See discussion about in Nowell and Kleinberg's paper

Link prediction – formalizing the problem

- **Given**
  - Social network G = (V, E)
  - Each edge e = (u, v) represents an interaction between two nodes u and v and comes with a timestamp t(e)
    - Parallel edges with possibly different timestamps possible
  - $G[t_1, t_2]$: graph consisting of all edges with timestamps between t_1 and t_2
- **Problem**: given $G[t_1, t_2]$ return a list of edges likely to belong to $G[t_3, t_4]$, with $t_3 > t_2$
  - Some caveats necessary

Performance

- We have a training graph $G[t_1, t_2]$ and accordingly a set E_{old} of edges
  - This is the training set
- We also have a test graph $G[t_3, t_4]$ and accordingly a set E_{true} of edges appeared in (t_3, t_4)
  - This is the test set
- Algorithm returns a list E_{new} of predicted edges for interval (t_3, t_4)
- We can use precision and recall

$$P = \frac{|E_{new} \cap E_{true}|}{|E_{new}|} \qquad R = \frac{|E_{new} \cap E_{true}|}{|E_{true}|}$$

Link prediction algorithms

- The general link predictor outputs a ranked list L_p of predicted edges taken from A x A, where $G[t_1, t_2] = (A, E_{old})$
- The i-th item of the list is the i-th most plausible new link (according to the algorithm)
  - So, list ranked according to decreasing algorithm's confidence
- How this is done in practice
  - Algorithm assigns a weight score(x, y) to each $(x, y) \in A \times A$
  - This is essentially a similarity score

Link prediction and user profiling

- **Collaborative filtering**
  - Given a collection of user-item pairs (u, i) summarizing past purchase history
  - *Problem*: predict which users are going to buy which items in the near future

Link prediction and user profiling

- **Collaborative filtering**
  - Given a collection of user-item pairs (u, i) summarizing past purchase history
  - *Problem*: predict which users are going to buy which items in the near future
- **Link prediction**
  - Given a collection of past user-user interaction (edges) (x, y)
  - *Problem*: predict which user interaction are going to occur in the near future

Link prediction and user profiling/2

- **Collaborative filtering (no ratings)**
  - Each user u is a list **u** = $\{a_1, a_2, \ldots\}$ of items she "purchased" in the past (whatever this means)

Link prediction and user profiling/2

- **Collaborative filtering (no ratings)**
  - Each user u is a list $\mathbf{u} = \{a_1, a_2, \dots\}$ of items she “purchased” in the past (whatever this means)
- **Link prediction**
  - Every node u is a list $\mathbf{u} = \{v_1, v_2, \dots\}$ of nodes she interacted with in the past
    - Note that the same node might appear more than once
- *Caveats*
  - Many approaches/techniques carry over
  - Some differences at the application level
    - Privacy issues can be more important
    - Trust and reputation

Link predictors [Kleinberg, Liben-Nowell 2007]

- **Neighbourhood based**
  - Common neighbours
  - Jaccard's coefficient
  - Adamic/Adar coefficient
  - Preferential attachment
- **Path-based**
  - Katz measure
  - Hitting time
  - Personalized Pagerank
- **Other techniques**
  - Low-rank matrix approximations
  - Clustering
  - ...

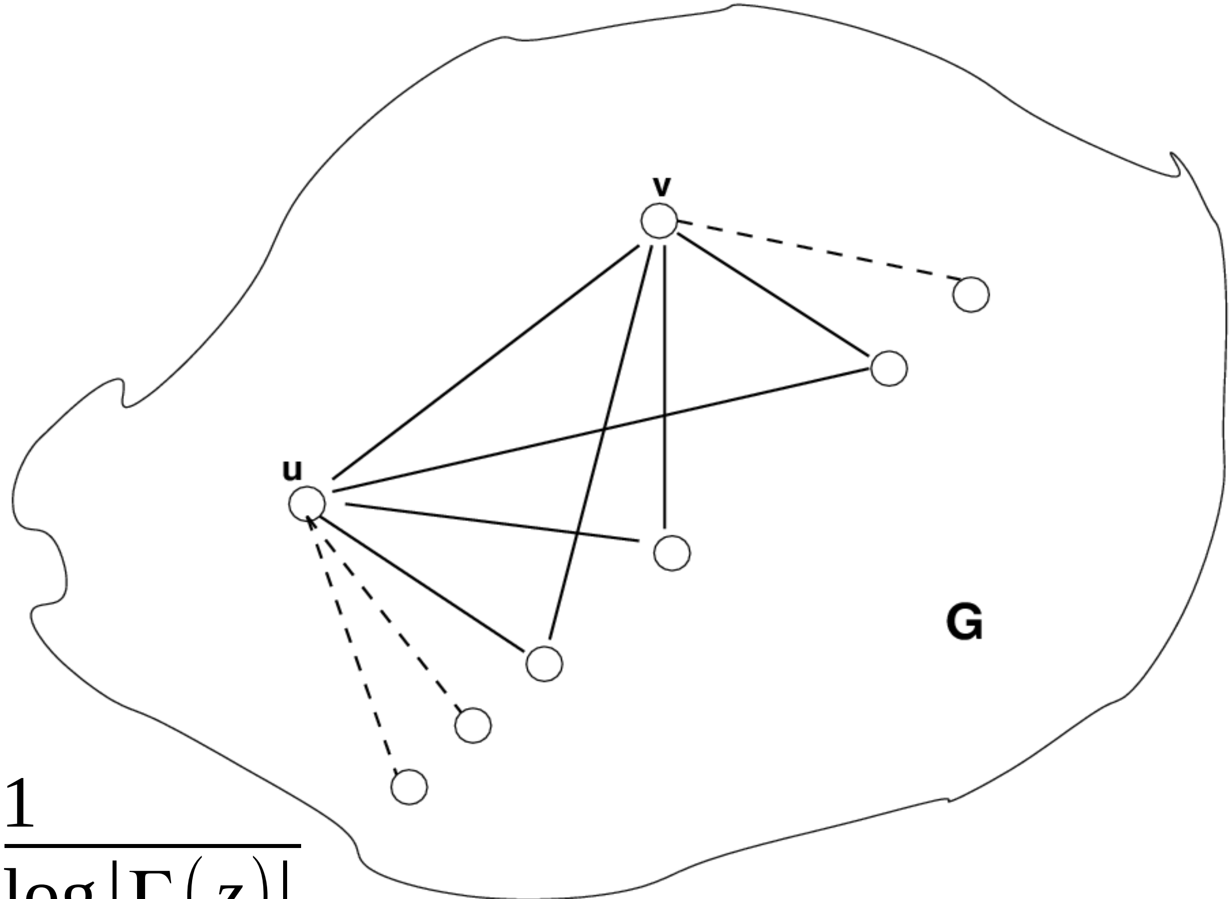
Neighbourhood based

$$cn(u,v) = |\Gamma(u) \cap \Gamma(v)|$$

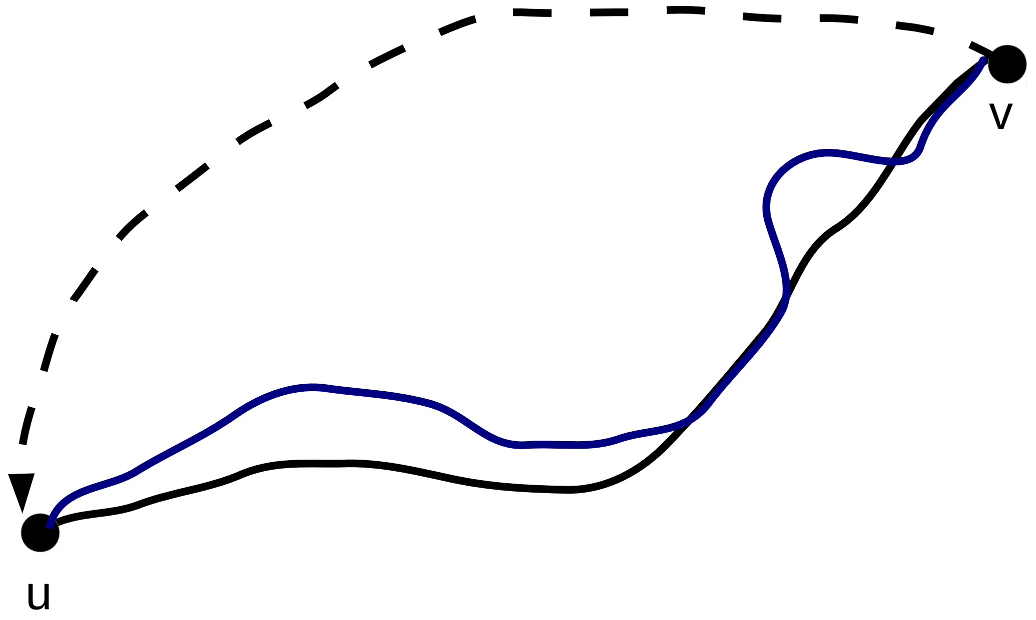
$$J(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

$$pa(u,v) = |\Gamma(u)| \cdot |\Gamma(v)|$$

$$AA(u,v) = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log|\Gamma(z)|}$$

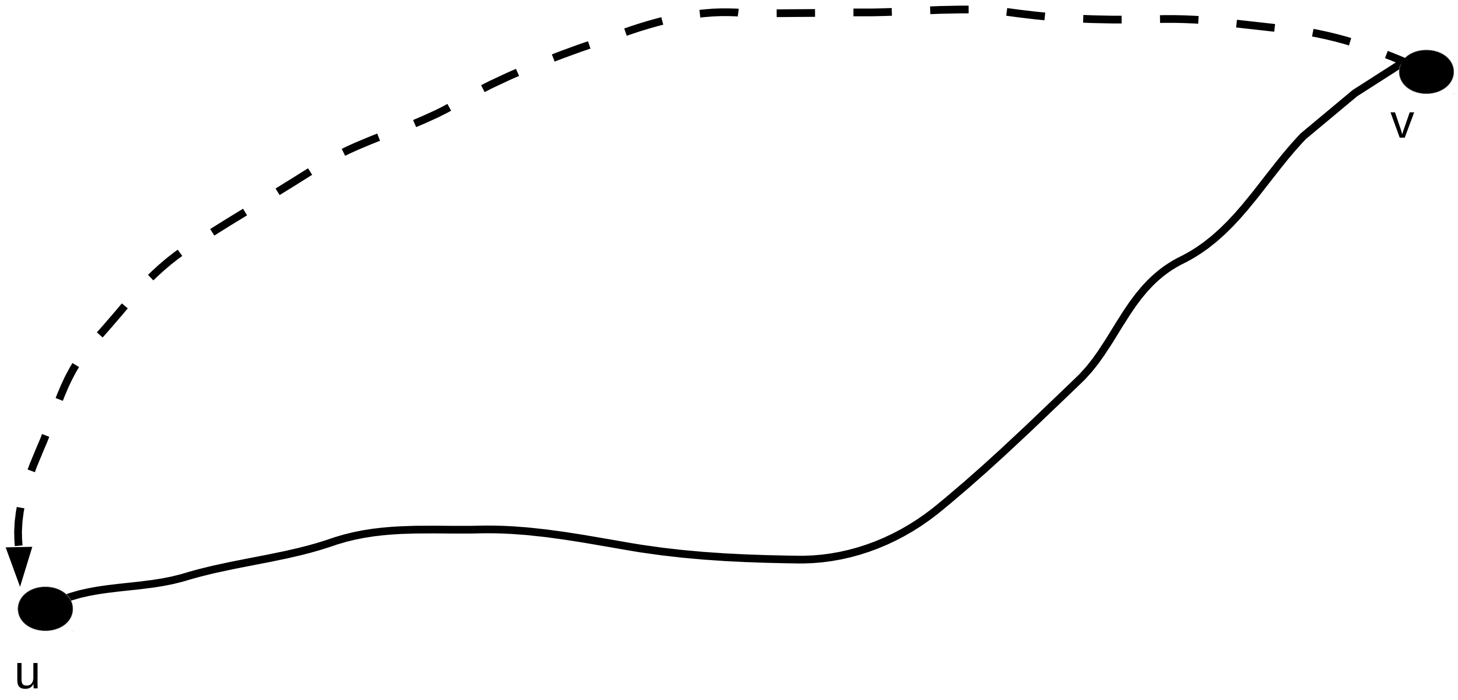


Path based – Katz coefficient



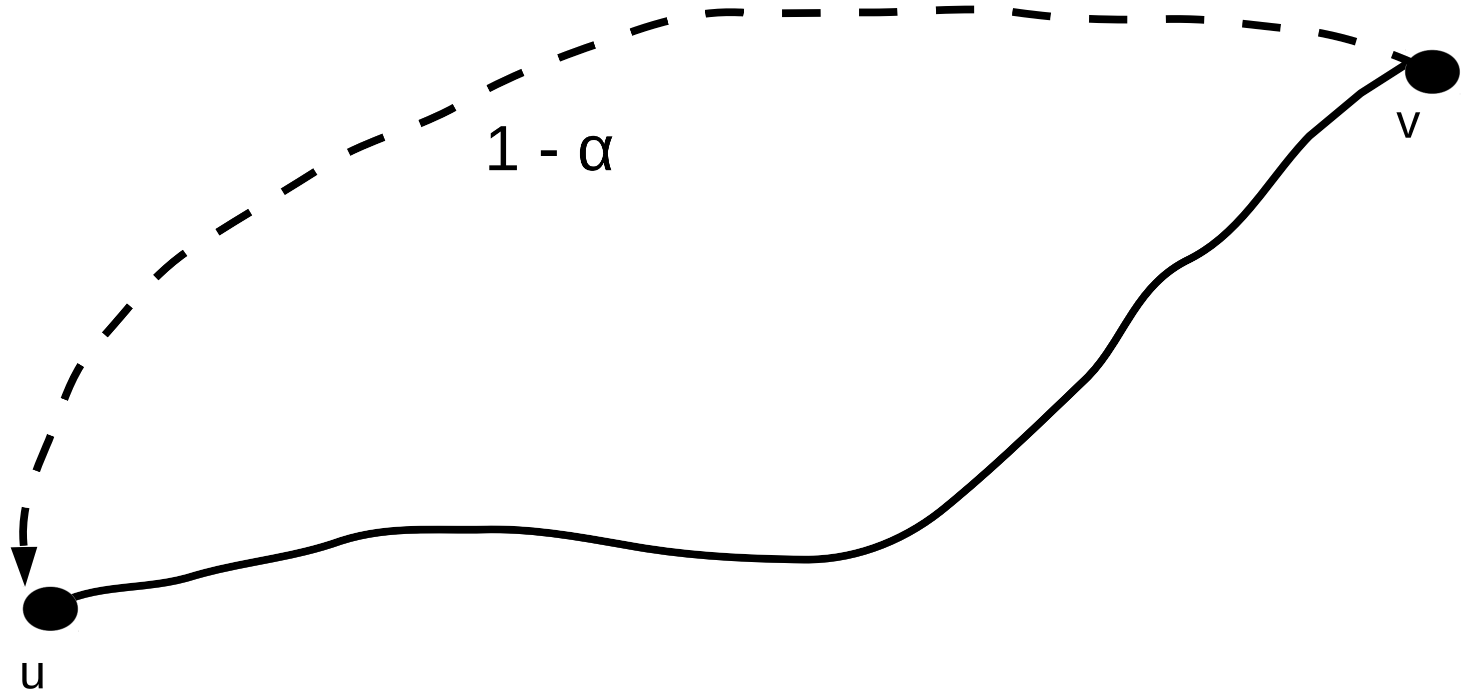
$$K_{\beta}(u,v)=\sum_{i=1}^{\infty}\beta^{i}\left|paths^{(l)}(u,v)\right|$$

Path based – Commute/Hitting time



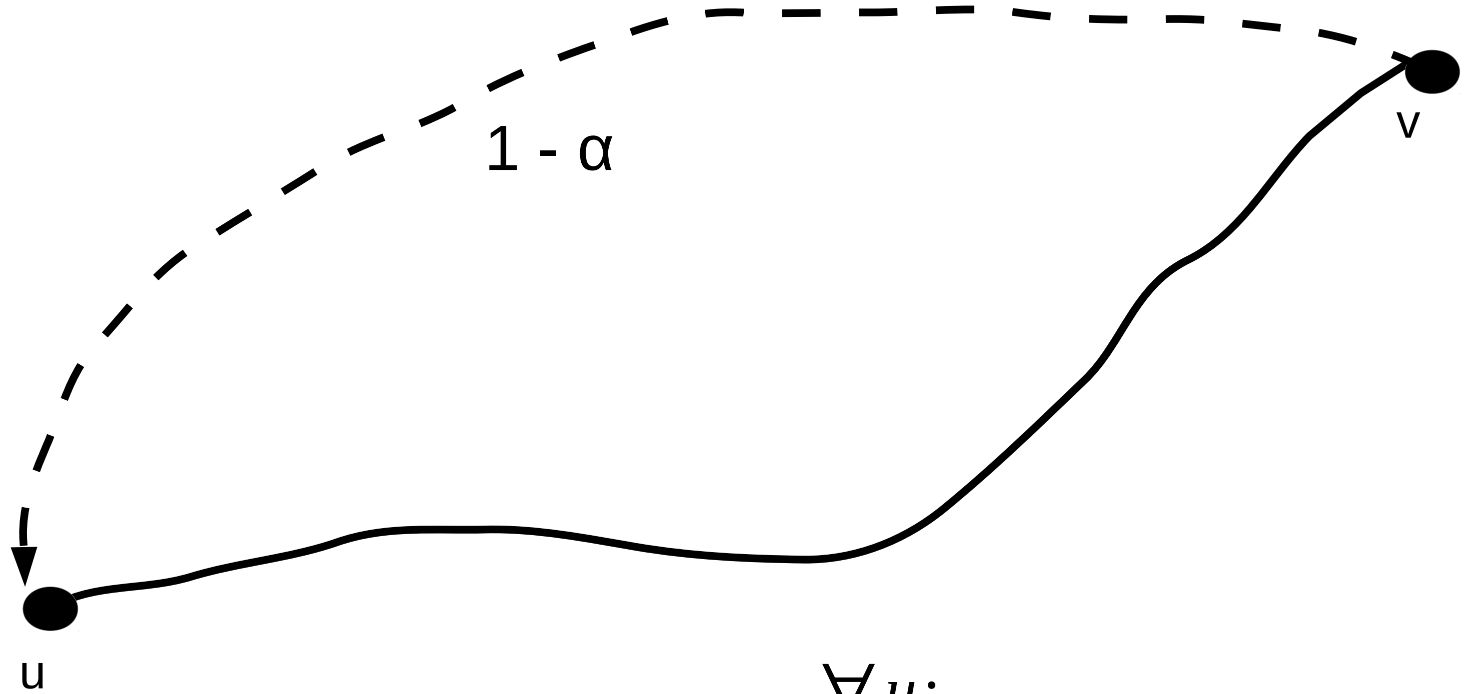
- Perform a random walk on the graph
- H(u, v) = expected number of steps to go from u to v
- C(u, v) = expected number of steps to go from u to v and back
- Can be efficiently computed for all pairs (still expensive!)

Personalized PR



- **To compute score$_u$(v):**
- Start a RW at v. At every node v:
- Jump u.a.r. with probability α. Jump to u with probability 1 – α
- score$_u$(v) is stationary probability of v
- This is PR with initial personalization vector $\mathbf{e}_u$

Personalized PR/2



- $\mathbf{e}_v$ is v's canonical vector
- **P** is the transition matrix
- Easy to prove that this corresponds to an ergodic MC hence:
- *For every u, here is a unique stationary distribution* $\boldsymbol{\pi}_u$

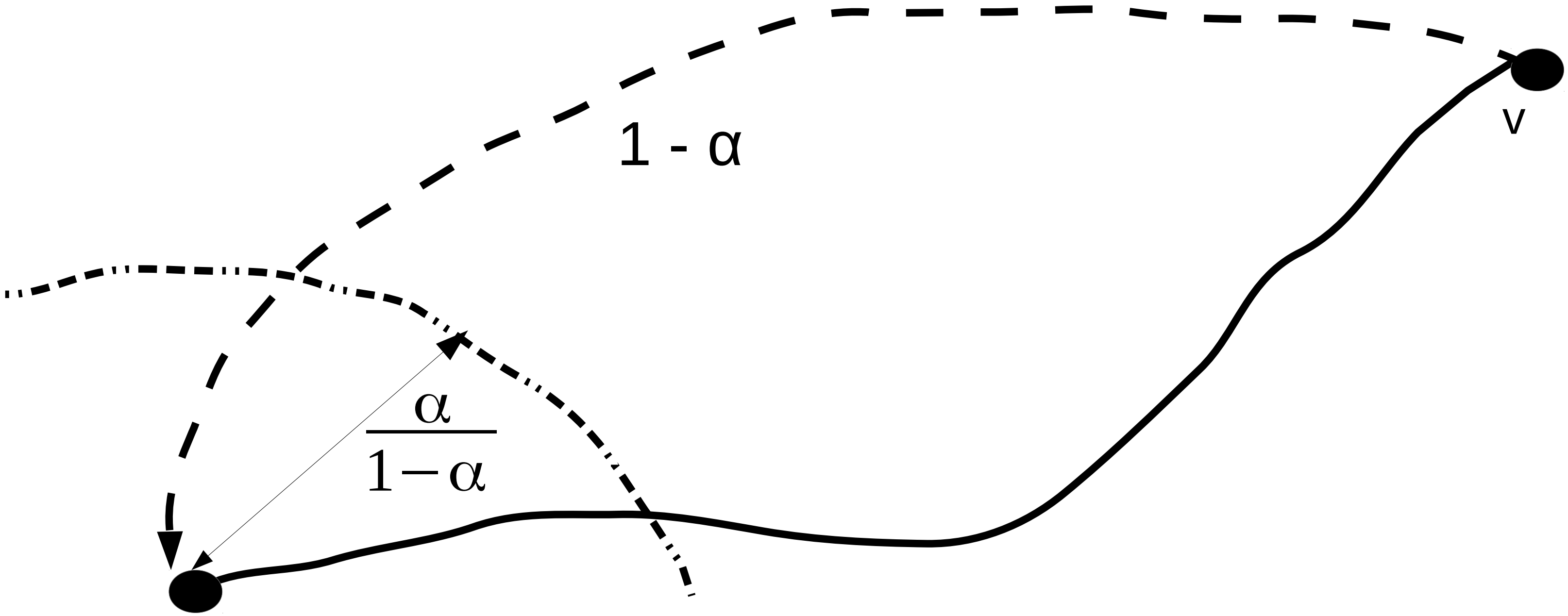
$$\forall u:$$

$$\pi_{u,u}(t+1)=\frac{1-\alpha}{n}+\alpha\sum_{v\in N(u)}\frac{\pi_{u,v}(t)}{deg(v)}$$

$$\pi_{u,w}(t+1)=\alpha\sum_{v\in N(u)}\frac{\pi_{u,v}(t)}{deg(v)}\; when\; w\neq u.$$

$$\boldsymbol{\pi}_{\boldsymbol{u}}^T(t+1)=\frac{1-\alpha}{n}\boldsymbol{e}_u+\alpha\,\boldsymbol{\pi}_{\boldsymbol{u}}^T(t+1)\boldsymbol{P}$$

Personalized PR/3



- For every u, we have scores $score_u(v) = \pi_{u.v}$ for every v, including u itself
- Computational cost: we need to compute a Personalized Pagerank (PPR) vector for every node in the network
- Notice that, in expectation, the personalized RW starting at u returns to u after $\alpha/(1 - \alpha)$ steps → Prove!
- Intuitively, this means that most of the time the RW is visiting a neighbourhood of u at most $O(\alpha/(1 - \alpha))$ hops away from it
- As a consequence, nodes with higher $\pi_{u.v}$'s are “closer” to u

SimRank [Jeh, Widom 2002]

- SimRank is the fixed point of the following recursive definition:
- simrank(x, x) = 1. In general:

$$simrank(x,y) = \gamma \cdot \frac{\sum_{u \in \Gamma(x)} \sum_{v \in \Gamma(y)} simrank(u,v)}{|\Gamma(u)| \cdot |\Gamma(v)|}$$

- Where γ is a parameter in [0, 1]

Other approaches

- Use noise reduction techniques as low-rank matrix approximations
  - Then, view matrix A obtained this way as a (weighted) adjacency matrix and apply predictors to it
- Clustering
  - Use a clustering procedure to identify (and remove) more “tenuous” edges in the network

Performance

- In absolute terms, performance is not very high
- This follows since new links in a social network often form for reasons exogenous to the network itself
- Still, experimental analysis in [Liben-Nowell, Kleinberg 2007] shows improvements over random predictions by orders of magnitude

Bibliography

- [Liben-Nowell and Kleinberg, 2007] D. Liben-Nowell and Jon Kleinberg. "The link-prediction problem for social networks." Journal of the American society for information science and technology 58.7 (2007): 1019-1031.
  - Preliminary version on proceedings of CIKM 2003
  - http://dm.kaist.ac.kr/kse625/resources/Liben-Nowell_2007.pdf
- [Terveen, McDonald, 2005] L. Terveen, D. W. McDonald. "Social matching: A framework and research agenda." ACM transactions on computer-human interaction 12.3 (2005): 401-434.
  - Available at http://grouplens.org/system/files/Terveen-SocialMatching.pdf