Graph Search and Learning

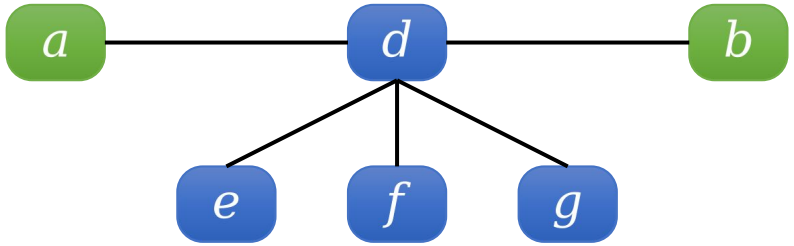
CS6101

Soumen Chakrabarti

Map

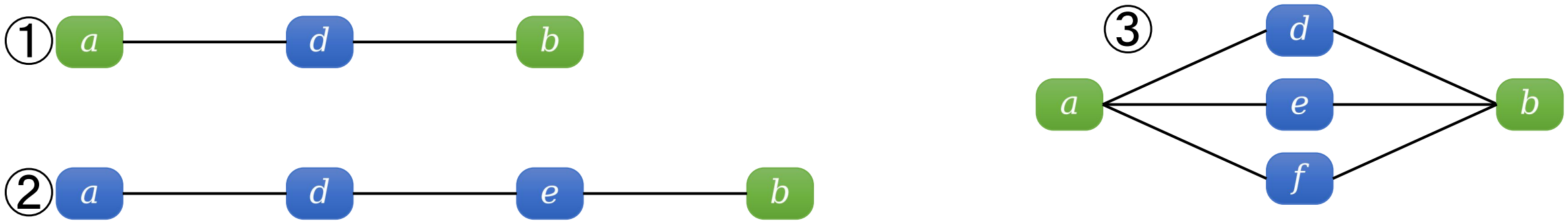
- Graphs add vital signal to local (text) features
- Useful in (at least) two task categories
  - Node property prediction
    - Page topic, spamminess, prestige, centrality, relevance
  - Edge or node-pair (property) prediction
    - Node pair relatedness
    - Edge existence, recommendation, polarity
- Starting point: How related are two given nodes?
  - Subjective, but can lay out a few axioms

Relatedness: why not shortest path?



- Shortest a—b path has two hops in both cases
- In the second case, d is like someone who delivers pizza to both a and b, but many others too
- Relatedness between a and b should be larger without distraction from other nodes

Relatedness: why not max flow?



- All edges have unit edge capacity
- Max flow from a to b is 1 in ① and 3 in ③
- Most would agree a and b are more strongly related in graph ③ than graph ①
- But max flow fails to detect that relatedness is more for ① than ② (shorter path)

Relatedness: The three principles

- Two nodes are more strongly related if
  - There is a short path from one to the other
  - There are many paths from one to the other
  - Paths do not have nodes on them with large degree ("low distraction paths")
- How to combine these three considerations into a principled relatedness measure?
- That can work with rich node and text features (e.g., text) as well?
- Random walk in graphs ↔ resistive electrical networks equivalence will be useful
  - Will set up through series of warmup examples

Warmup: linear equations related to graphs

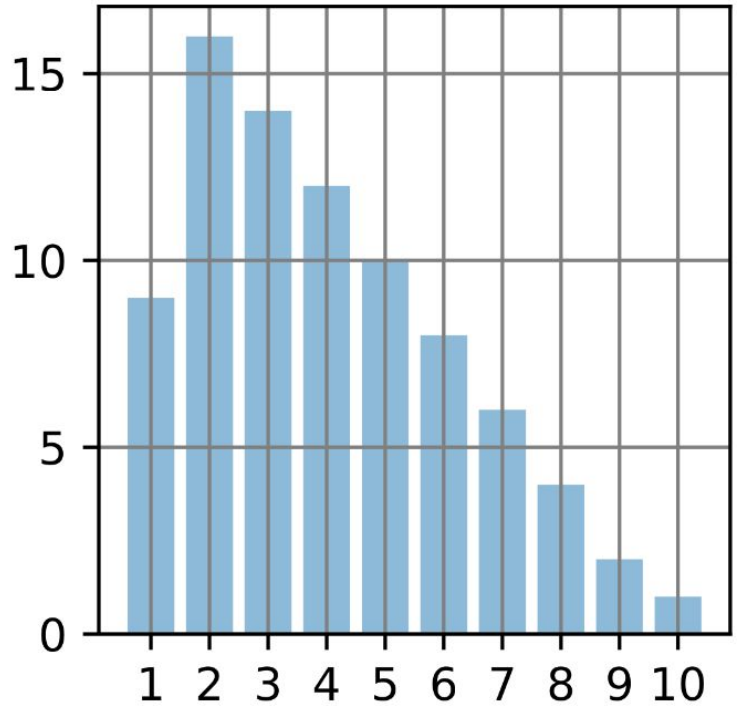
- Undirected chain graph with nodes $1, 2, \cdots, n$
- Walk begins at node 1 and ends when node n is visited for the first time
- At each step, walker goes from a node to a neighbor uniformly at random
- Define $v(i)$ as the expected number of visits to node i
  - By definition, $v(n) = 1$
- Half the time surfer is at $n-1$, they then visit n
  - $\frac{1}{2}v(n-1) = v(n) = 1 \Rightarrow v(n-1) = 2$
- Half the time surfer is at 2, they move to 1 next
  - $v(1) = 1 + \frac{1}{2}v(2)$ (1 for initial visit, not transition from 2)

Chain graph, continued

- If surfer is at 2 now, must have been at 1 or 3 in previous time step $\Rightarrow \nu(2) = \nu(1) + \frac{1}{2}\nu(3)$
- For nodes $3 \leq i \leq n-3$, $\nu(i) = \frac{\nu(i-1)+\nu(i+1)}{2}$
- General solution is

$$\nu(i) = \begin{cases} n-1 & i = 1 \\ 2(n-i) & 2 \leq i \leq n-1 \\ 1 & i = n \end{cases}$$

- Before reaching n for the first time, will visit 1 expected $n-1$ times!
- Expected length of walk is $\Theta(n^2)$



Markov walks in undirected graphs

- From chain to general graph
- Nodes are states, edges are transitions
- Every time step, from a node, step to one neighbor with uniform probability (unweighted edges)
- Symmetric node-node adjacency matrix A
  - Assume single connected component
  - Positive sum for every row and column
- Convention: row = to node i, column = from node j
  - Compute column sums of A: these are node degrees $d[j]$
  - Have to go *some* node i from node j
  - Define $C[i, j] = A[i, j]/d[j] \Rightarrow$ columns of C sum to 1 ("column-stochastic")

State (multinomial) probability vector

- C is the single-step transition probability matrix (not symmetric in general)
  - $C[i, j] = C[j \to i]$ probability of walking from j to i
- $p^{(t)}[i]$ is the probability of being in node i at time step t
  - Have to be at some node: $\sum_i p^{(t)}[i] = 1$
- To be in node i at time t, must have been at some neighbor j in previous step $t - 1$
  - $p^{(t)}[i] = \sum_j p^{(t-1)}[j]\, C[i, j]$
- Collect into matrix-vector product $p^{(t)} = C\, p^{(t-1)}$
- If a steady-state exists, $p^{(\infty)} = C\, p^{(\infty)}$ is an eigenvector
  - Sometimes written as $\pi = C\, \pi$

Steady state and return time

- For connected undirected graph, $p^{(\infty)}$ exists
- Let M be the number of edges
- Check that $\pi[i] = d[i]/2M$ is a steady state
- I.e., visit rate is proportional to degree
  - Not generally true for directed graphs
- Return time $R(i)$ = expected number of steps between two consecutive visits to i
- Claim: $R(i) = 1/\pi[i]$
  - Intuition: walk becomes a multinomial sampler π
  - Gap between getting two consecutive draws of one ‘face’ (node) — geometrically distributed
- Undirected chain graph
  - $\pi[0] = \pi[n] = 1/2n$; $\pi[1] = \cdots = \pi[n-1] = 2/2n = 1/n$
  - $R(0) = R(n) = 2n$; $R(1) = \cdots = R(n-1) = n$

Hitting time from node i to node k on chain

- Starting at node i, the expected number of steps to reach node j for the first time is the hitting time $H(i, j)$
- In the undirected chain graph, what is $H(k-1, k)$?
- Walk ends when we reach k for the first time
- Therefore, nodes to the right of k effectively do not exist
- $R(k) = 2k$
- To return from k to k, we need to step to $k-1$ and then hit k from $k-1 \Rightarrow R(k) = 1 + H(k-1, k)$
- Solves to $H(k-1, k) = 2k - 1$
  - Although, w.p. $1/2$, we walk straight from $k-1$ to k

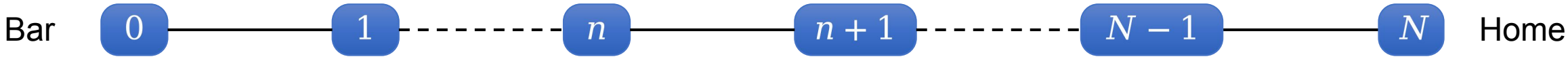
Hitting time in chain graph

- How about $H(i,k)$ with $i < k \leq n-1$?
  - To hit k, must reach $k-1$ earlier
  - Leads to $H(i,k) = H(i,k-1) + H(k-1,k)$
    $= H(i,k-1) + 2k-1$
  - Unrolling recurrence, $H(i,k) = k^2 - i^2$
  - Confirm special case: $H(0,k) = k^2$
- Infinite random walk on integer number line
  - Start at 0, toss coin, head$\rightarrow +1$, tail$\rightarrow -1$
  - Coordinate is current head minus tail imbalance
  - To achieve imbalance k, have to toss k^2 times
- Commute time between i, j is $H(i,j) + H(j,i)$

Cover time

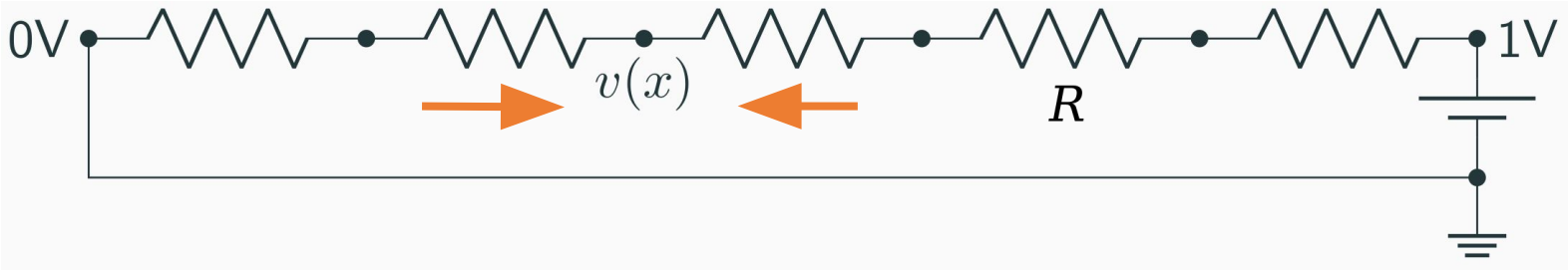
- Cover time $c(i)$ starting at node i is the expected number of steps until all nodes are visited at least once
  - In chain graph with nodes $\{0, \cdots, k\}$, $c(0) = k^2$
- What is the cover time of any node in a clique?
  - The more we cover, the less likely the next step covers a new node
- Kite graph
- Dumb-bell graph

Terminating (absorbing) walk



- Starter example: chain walk
  - If we reach bar or home, stay there for ever
  - Otherwise, step left w.p. 1/2 and right w.p. 1/2
- Start from node n: $0 < n < N$
- $p(n)$ is the probability of reaching home first
  - $p(0) = 0,\ \ p(N) = 1$
  - For $0 < n < N$, $p(n) = \frac{1}{2}p(n-1) + \frac{1}{2}p(n+1)$

Resistive network



- 1V battery connected to R-Ohm resistors in series
- Node 0 is at 0V, node N is at 1V
- $v(x)$ is the voltage at node x
- By Kirchoff's current law, for $0 < x < N$,

$$\frac{v(x-1) - v(x)}{R} + \frac{v(x+1) - v(x)}{R} = 0$$

- (No charge accumulates at node x)
- Solves to $v(x) = \frac{1}{2}v(x-1) + \frac{1}{2}v(x+1)$, exactly the same form as $p(n)$

Escape probability

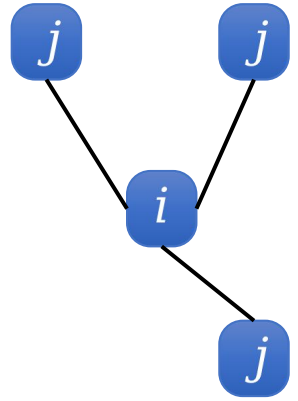
- Designated sink node s, target node t, start node j
- $E(s,t;j)$ the probability of reaching t before ever visiting s
- Base cases $E(s,t;s)=0,\ E(s,t;t)=1$
- For $j \neq s,t$ we move to a neighbor i of j and recurse from i
  - With unweighted edges, we get $E(s,t;j)=\frac{1}{d(j)}\sum_{i:\,(i,j)\in E} E(s,t;i)$
  - With edge transition probabilities $C[*,j]$, we get $E(s,t;j)=\sum_{i:\,(i,j)\in E} C[i,j]\ E(s,t;i)$
- Recall $C[i,j]=\frac{A[i,j]}{\sum_l A[l,j]}$, this lets us write

$$E(s,t;j)=\frac{\sum_{i:\,(i,j)\in E} A[i,j]\ E(s,t;i)}{\sum_{i:\,(i,j)\in E} A[i,j]}$$

- E.P. at node j is edge-weighted average of E.P. of neighbors i of j

Harmonic functions on graphs

- In general, for node j with neighbors $\{i\}$, Kirchoff's current law says (for non-battery, non-ground nodes)



$$\sum_{i:(i,j)\in E} \frac{v(i) - v(j)}{R_{ij}} = 0$$

- For easier notation, use conductance $g_{ij} = 1/R_{ij}$ and rewrite as

$$\sum_{j:(i,j)\in E} g_{ij}(v(i) - v(j)) = 0$$

- Collecting terms, we get

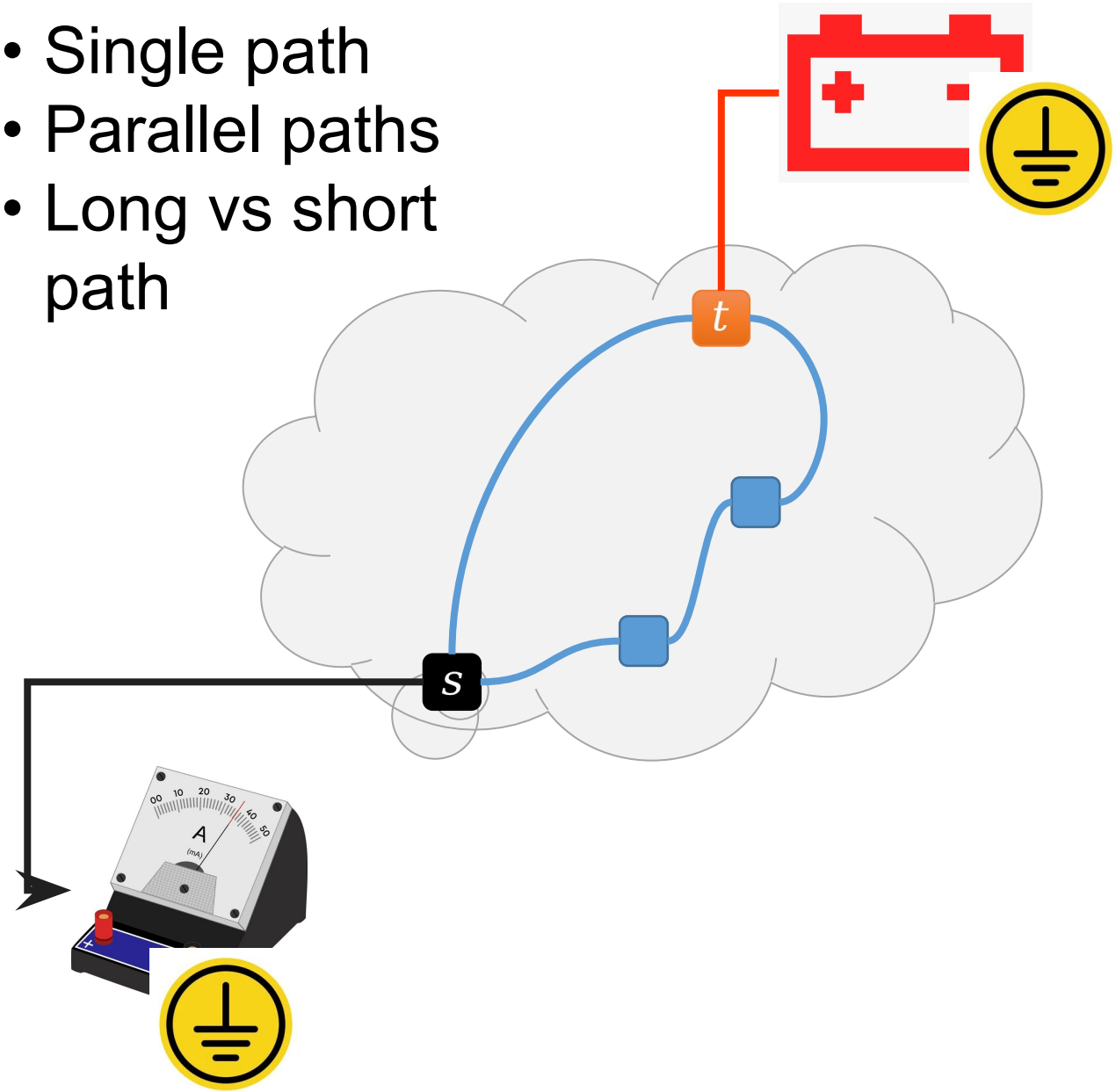
$$v(j) = \frac{\sum_{i:(i,j)\in E} g_{ij} v(i)}{\sum_{i:(i,j)\in E} g_{ij}}$$

Voltage at a node is the conductance-weighted average of voltages at neighboring nodes

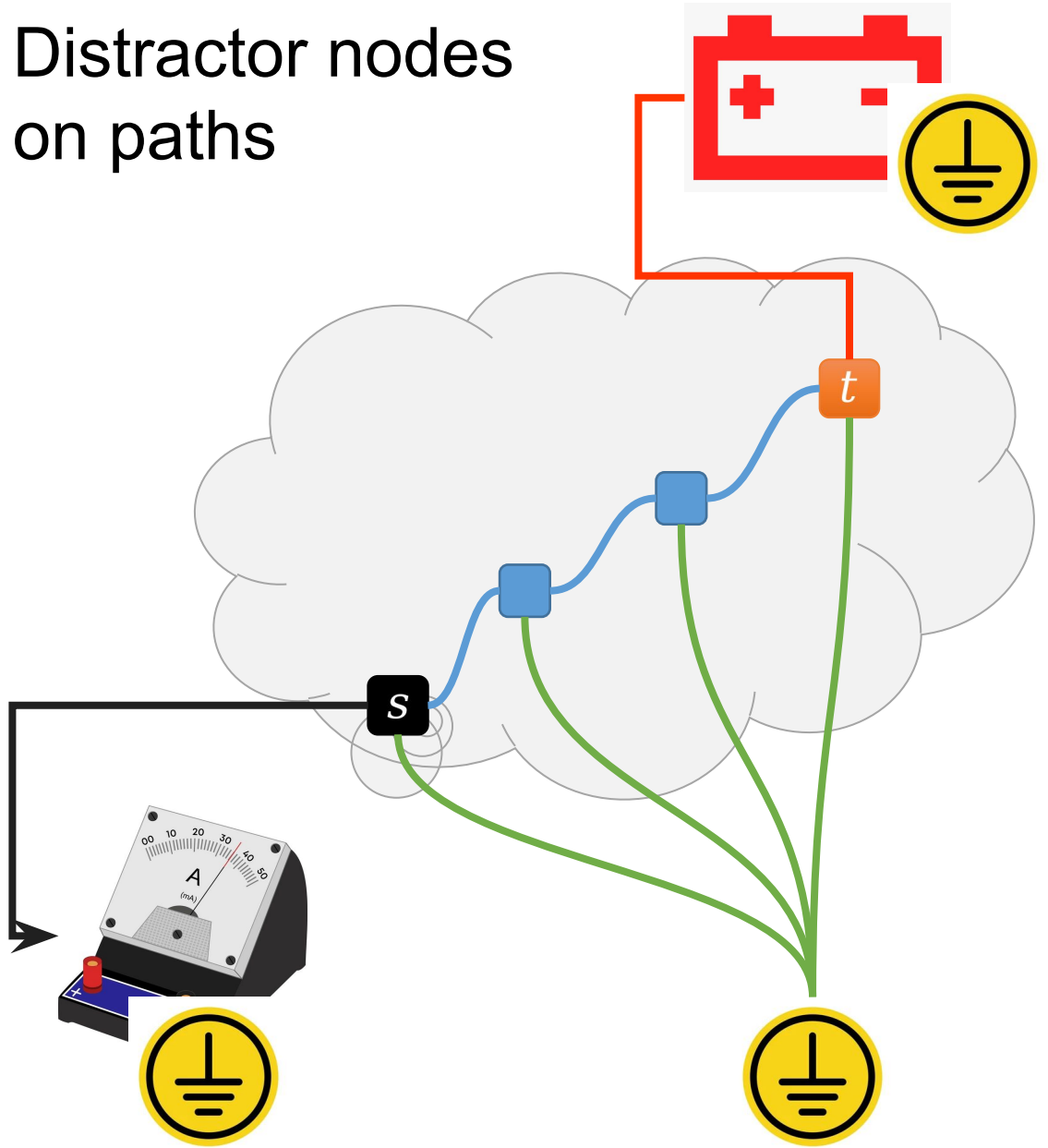
- Except for boundary conditions $v(s) = 0$, $v(t) = 1$

Effective conductance

- Single path
- Parallel paths
- Long vs short path



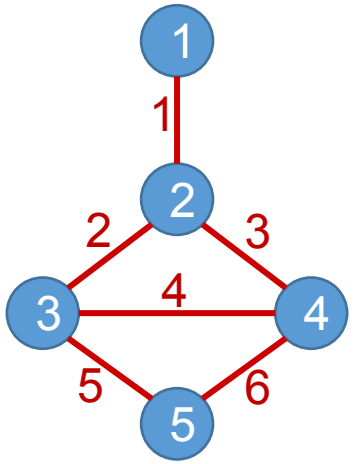
- Distractor nodes on paths



Pseudo-relevance feedback

- Node i = doc, relevance $r(i)$ wrt a query
  - May be produced by a neural scoring network
- Edge = doc-doc similarity $a(i, j) \geq 0$
  - May be output by another network
- For now, assume $r(i)$ and $a(i, j)$ are fixed
- Final score for ranking purpose is $f(i)$ ("decision variables") with two-part objective
  - $f(i)$ should approximate $r(i)$: $\sum_i (f(i) - r(i))^2$
  - $f(i)$ should be smooth across edges: $\sum_{i,j} a(i, j)(f(i) - f(j))^2$

Node-edge adjacency matrix B



$A =$

	1	2	3	4	5
1					
2					
3					
4					
5					

$B =$

	1	2	3	4	5	6
1						
2						
3						
4						
5						

Orient each edge arbitrarily, one -1 and one $+1$ in each column

$B\ B^{\mathsf{T}} =$

	1	2	3	4	5	6
1						
2						
3						
4						
5						

	1	2	3	4	5
1					
2					
3					
4					
5					
6					

$=$

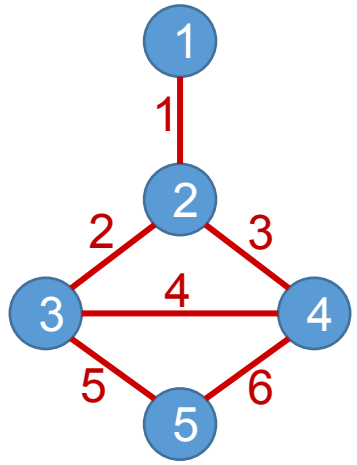
	1	2	3	4	5
1					
2					
3					
4					
5					

Off-diagonals are $-A$

Diagonals hold node degrees

Why is $BB^\top$ interesting?

- Smoothness loss is $(f_1 - f_2)^2 + (f_2 - f_3)^2 + (f_2 - f_4)^2 + \cdots$
- One term for each edge
- Let $f^\top = [f_1\ f_2\ f_3\ f_4\ f_5]$
- What is $f^\top B$?



1	2	3	4	5

	1	2	3	4	5	6
1						
2						
3						
4						
5						

=

1	2	3	4	5	6

- Smoothness loss is therefore $f^\top B = f^\top BB^\top f$
- $BB^\top$ is positive semidefinite
- Therefore this is a convex quadratic term

PageRank

- Query-independent score of 'authority' of a page (node) in the Web graph of hyperlinks (directed edges)
- Seeley, 1949: "… we are involved in an "infinite regress": [an actor's status] is a function of the status of those who choose him; and their [status] is a function of those who choose them, and so ad infinitum.
- Edge direction is critical: $i \rightarrow j$ increases authority of j but not i
- What if node-node adjacency matrix A is allowed to be asymmetric?
  - Normalize columns as before to get C, and find fixpoint $p^{(\infty)} = Cp^{(\infty)}$ as before
  - What can go wrong?

Directed walks (e.g., on Web graph)

- Complications: dead ends and cycles
- Markov walk on such graphs may lead to problematic (non-unique, degenerate, or non-convergent) $p^{(\infty)}$
- Two sufficient conditions avoid these problems
  - Single strongly connected component (directed path from any node to another) $\rightarrow$ $p^{(\infty)}$ is well-defined and unique
  - Aperiodic (cycles of almost all lengths exist) $\rightarrow$ $p^{(\infty)}$ converges starting from any $p^{(0)}$

Teleport

- Hyperparameter (walk probability) $\alpha \in (0,1)$
- Random walker currently at node j
  - If j has out-neighbors, total probability of walking to neighbors is α
  - If j is a dead end, probability of walking out is 0
- With the remaining probability ($1-\alpha$ or 1), teleport (jump) to any node in the graph
  - Choose target uniformly at random, or
  - Sample a multinomial 'personalization' teleport distribution r
- "Random walk with restart" (RWR)
- Effectively superpose a complete directed graph on top of the input graph, satisfying both requirements

PageRank recurrence

- For notational simplicity assume no dead-end nodes
- If random walker is now at node i, then
  - Either walking in from a neighbor j, along edge (j, i), with probability $\alpha C[i, j]$
  - Or jumped from some arbitrary node k, with probability $(1 - \alpha) r[i]$
- If steady state probability of visiting node i is $p[i]$,

$$p[i] = \sum_{j:(j,i)\in E} \alpha C[i, j] p[j] + \sum_{j \in V} (1 - \alpha) p[j] \, r[i]$$

- Or, in compact matrix form,

$$p = \alpha C p + (1 - \alpha) r$$

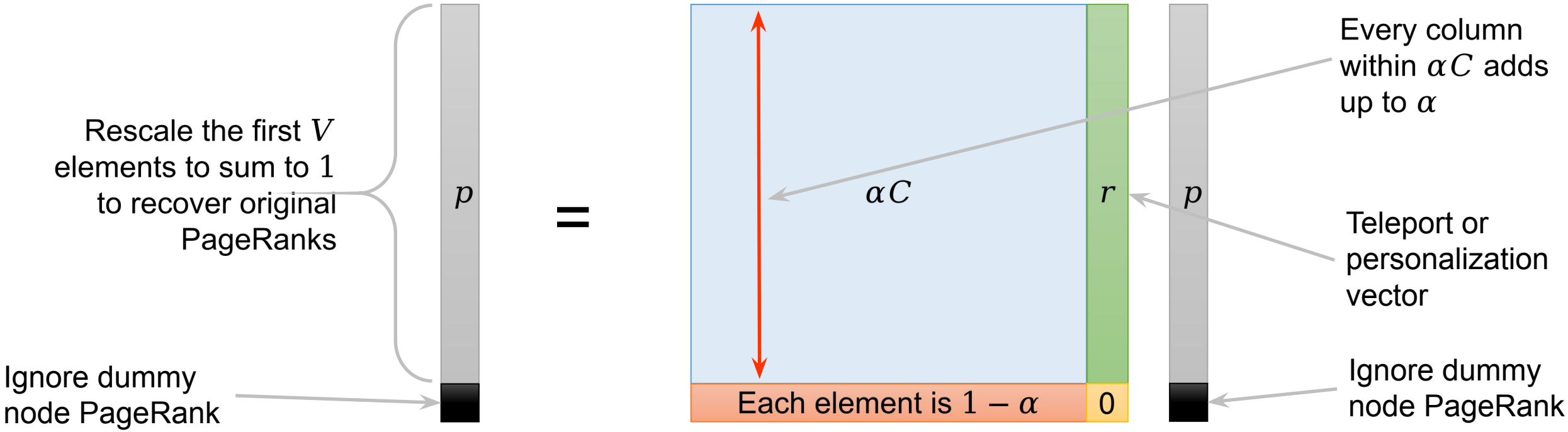
- Solve for p efficiently, with $|V| \approx 10^{10}, |E| \approx 10^{12}$
- Use $p[i]$ as a static (query independent) feature in learning to rank
- How to tune α? Why are some values suitable?

Solving the recurrence

- PageRank equation $p = \alpha C p + (1 - \alpha) r$
  - $\mathbb{I} p - \alpha C p = (\mathbb{I} - \alpha C) p = (1 - \alpha) r$ or $p = (1 - \alpha)(\mathbb{I} - \alpha C)^{-1} r$
  - Here $(\mathbb{I} - \alpha C)^{-1} \in \mathbb{R}^{V \times V}, r \in \mathbb{R}^{V}$
  - Does matrix inverse of $(\mathbb{I} - \alpha C)$ always exist?
  - Even if it exists, it is dense and impractical to store or manipulate
- A more practical form is $p = (1 - \alpha) \sum_{n \geq 0} \alpha^n C^n r$
  - Converges because C is column-stochastic and $\alpha < 1$

A homogeneous form

- Instead of all-to-all teleport, use a dummy node
  - All-to-dummy then dummy-to-all 'walk' edges
- Leads to a padded $\widehat{C} \in \mathbb{R}^{(V+1)\times(V+1)}$ and corresponding padded $\widehat{p} \in \mathbb{R}^{(V+1)}$ in the homogeneous form $\widehat{p} = \widehat{C}\widehat{p}$
- Rescale non-dummy nodes in $\widehat{p}$ to add up to 1 to recover p



Power iteration

- To solve $\widehat{p} = \widehat{C}\widehat{p}$ via power iteration
- Initialize $\widehat{p}^{(0)}$ 'suitably'
  - Can be initially random
  - Later, warm start from previous runs
  - (Hope is that Web graph is changing relatively slowly)
- Repeat for $t = 1,2, \cdots$
  - $\widehat{p}^{(t)} \leftarrow \widehat{C}\widehat{p}^{(t-1)}$
- Until $\|\widehat{p}^{(t)} - \widehat{p}^{(t-1)}\|$ is "small enough"
- Postprocess "$\widehat{p}^{(\infty)}$" to recover PageRanks

RAM-friendly, IO-friendly matrix-vector product

- Luckily, $\widehat{C}$ is sparse with $O(V + E)$ nonzeros
- Store as a list on disk, with records $\langle i, j, \hat{c}_{i,j} \rangle$
- Need to hold $\widehat{p}^{(t)}$, $\widehat{p}^{(t-1)}$ in RAM
  - Otherwise need repeated sorts, more expensive
- Repeat
  - Initialize $\widehat{p}^{(t)}$ to all zeros
  - Scan edge list; for each $\langle i, j, \hat{c}_{i,j} \rangle$
    - $\widehat{p}^{(t)}[i]$ += $\hat{c}_{i,j}$ $\widehat{p}^{(t-1)}[j]$
  - Check for convergence and quit loop if done
  - Swap (pointers to) $\widehat{p}^{(t)}$, $\widehat{p}^{(t-1)}$

Power iterations to find dominant eigenvector

- Simple notation $p \leftarrow Cp$
- If C is column-stochastic, renormalizing p is not required in principle
- May still do it in practice, for numerical stability
- C's eigenvalues satisfy $1 = \lambda_1 > |\lambda_2| > |\lambda_3| > \cdots$
- Corresponding eigenvectors $u_1, u_2, u_3, \cdots$ form orthonormal basis
- Express $p^{(0)} = a_1 u_1 + a_2 u_2 + a_3 u_3 + \cdots$ where $a_1, a_2, a_3, \cdots$ are coordinates in the new basis

$$p^{(1)} = Cp^{(0)} = a_1 Cu_1 + a_2 Cu_2 + a_3 Cu_3 + \cdots$$
$$= a_1 u_1 + a_2 \lambda_2 u_2 + a_3 \lambda_3 u_3 + \cdots$$

- After k iterations, $p^{(1)} = a_1 u_1 + a_2 \lambda_2^k u_2 + a_3 \lambda_3^k u_3 + \cdots$
- Observe that only $a_1 u_1$ does not shrink, all other terms do
- Can we shrink lower eigenvectors faster?

Acceleration

- Suppose, after some number of iterations, we got lucky and all but top two eigenvectors are gone:

$$p^{(k-2)} = u_1 + a_2 u_2$$

- In the next two iterations,

$$p^{(k-1)} = C p^{(k-2)} = u_1 + a_2 \lambda_2 u_2$$
$$p^{(k)} = C p^{(k-1)} = u_1 + a_2 \lambda_2^2 u_2$$

- Using these, we can solve for u_1
  - $u_1[i] = p^{(k-2)}[i] - a_2 u_2[i]$

$$= p^{(k-2)}[i] - \frac{\left(p^{(k-1)}[i] - p^{(k-2)}[i]\right)^2}{p^{(k)}[i] - 2p^{(k-1)}[i] + p^{(k-2)}[i]}$$

- Periodically, force $p^{(k)}$ to this value

TotalRank: More principled choice of α

- Here we fix teleport r for simplicity
- Instead write $p(\alpha)$ as a function of α
- Since we don't know α, marginalize
- $\int_0^1 p(\alpha)\,d\alpha = \int_0^1 \left[\sum_{k\geq 0} (1-\alpha)\,\alpha^k C^k\right] r\,d\alpha$

$$= \sum_{k\geq 0} C^k r \int_0^1 (1-\alpha)\alpha^k d\alpha$$

$$= \left[\sum_{k\geq 0} \frac{C^k}{(k+1)(k+2)}\right] r$$

- How to compute this infinite sum?

Recap

- Random walk in graphs ↔ electrical networks
- Node-to-node relatedness
  - Hitting, commute, cover times, escape probability
- Node popularity or prestige
  - Steady state visit rates
- PageRank
  - Need for teleport (irreducible, aperiodic)
  - Dominant eigenvector, (accelerated) power method
  - Next: (query-time) personalization

Personalized PageRank

- General recurrence $p_r = \alpha C p_r + (1 - \alpha) r$
- Subscript r in p_r because PageRank will be regarded as a function of r
- Teleport vector r can be designed from application needs
  - $r[i]$ proportional to query relevance of doc i
  - $r[i] > 0$ if user has visited page i (often)
- Recall we can write $p_r = (1 - \alpha)(\mathbb{I} - \alpha C)^{-1} r$
- The highlighted part is a matrix M
- Thus, p_r is a linear function of r: $p_r = Mr$

Impulse teleport vector

- For a fixed node i, a special value $r = \delta_i$ is defined as

$$\delta_i[j] = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

  - "Impulse teleport" to only one node i
- Corresponding p_{δ_i} is called the personalized PageRank vector $\mathrm{PPV}(i) \in \mathbb{R}^V$
- $\mathrm{PPV}(i)$ is a kind of soft "reachability map" from node i to all V nodes
- Imagine we have precomputed and stored $\mathrm{PPV}(i)$ for all nodes i
- From these, can we recover p_r for any r?

Linearity and superposition

- Thanks to linearity
  - $M(ur) = p_{ur} = up_r = u(Mr)$ with $u \in \mathbb{R}$
  - (Although ur may not be a valid multinomial teleport distribution)
  - Likewise, $p_{r+r'} = p_r + p_{r'}$ (ditto for $r + r'$)
- Given arbitrary teleport r, we can decompose it among the basis $\{\delta_i : i \in V\}$ as $r = \sum_i u_i \delta_i$
- Accordingly, $p_r = \sum_i u_i \mathrm{PPV}(i)$
- "We have precomputed and stored $\mathrm{PPV}(i)$ for all nodes i" — can be prohibitively expensive

Pushdown and "backward propagation"

- Recall $p_r = p_{\alpha C r} + (1-\alpha)r$
- Aside: $\sum_{v:(u,v)\in E} C[v,u]\delta_v = C\delta_u$ (verify this)
- Using the pushdown property,

$$\sum_{v:(u,v)\in E} \alpha C[v,u] p_{\delta_v} = p_{\sum_{(u,v)\in E} \alpha C[v,u]\delta_v}$$

- Therefore we can write

$$\alpha \sum_{v:(u,v)\in E} C[v,u]\, p_{\delta_v} = p_{\alpha C \delta_u} = \alpha C p_{\delta_u}$$

- leading to

$$p_{\delta_u} = \alpha \sum_{v:(u,v)\in E} C[v,u]\, p_{\delta_v} + (1-\alpha)\delta_u$$

- Note, vector p_{δ_u} is being expressed in terms of vectors p_{δ_v} of out-neighbors v where $(u,v) \in E$

Synchronous and asynchronous PageRank

- $q \leftarrow r, p \leftarrow 0$
- while $|q|$ is large do
  - $p \leftarrow p + (1-\alpha)q$
  - $q \leftarrow \alpha C q$

- $q \leftarrow r, p \leftarrow 0$
- while some $q(u)$ is large
  - $\hat{q} \leftarrow q(u)$; $q(u) \leftarrow 0$
  - $p(u)$ += $(1-\alpha)\hat{q}$
  - for out-neighbors v
    - $q(v)$ += $\alpha C[v,u]\hat{q}$

Iter		

Similar loop invariants can be set up in both forms

What is $\mathrm{PPV}_v = p_{r=\delta_v} \in \Delta^{|V|-1}$?

- If unit heat is injected into node v, how much heat goes lands up at all nodes?
- By linearity, if h "heat units" are injected into v, node w will get $p_{\delta_v}[w]$ "heat units"
- Can short-cut if p_{δ_v} is available

Forward propagation

- For input teleport distribution vector r, PageRank is a solution to $p = \alpha C p + (1 - \alpha) r$
- p is a (linear) function or r, and we write it as p_r
- If $\{u\}$ link to v and we know all $p_r[u]$, we can compute $p_r[v]$ via "forward propagation" as

$$\sum_{u:(u,v)\in E} \alpha C[v,u] p_r[u] + \sum_u r[u](1-\alpha)$$
$$= \sum_{u:(u,v)\in E} \alpha C[v,u] p_r[u] + (1-\alpha)$$

- Note, scalar quantity $p_r[v]$ is expressed in terms of scalar quantities $p_r[u]$ of in-neighbors u of node v

SimRank motivation

- Recall $p_{\delta_i}[j]$ is the PageRank of node j when teleport is to only node i
  - How often a RWR restarting at i visits j
  - Incorporates all three relatedness principles
  - But not generally symmetric: $p_{\delta_i}[j] \neq p_{\delta_j}[i]$
- Some applications need a symmetric measure of node-node similarity based on their graph neighborhoods
  - Here we focus on edges from other nodes into i, j
  - Call them the in-neighbors $N(i)$, $N(j)$
  - (Can incorporate also out-links, local features, etc.)

SimRank formulation

- A node is maximally similar to itself: $s(i, i) = 1$
- To compute $s(i, j)$ with $i \neq j$
  - Let $i' \in N(i)$, $j' \in N(j)$
  - Recursively compute $s(i', j')$
  - $s(i', j')$ 'suitably' contributes to $s(i, j)$

$$s(i, j) = \frac{1}{|N(i)||N(j)|} \sum_{i' \in N(i), j' \in N(j)} \alpha \; s(i', j')$$

- $\alpha \in (0,1)$ is a dampener much like in PageRank
- Is $s(i, j)$ well-defined and convergent?
- Possible to compute efficiently for all i, j pairs?

Hyperlink induced topic search (HITS)

- Query dependent graph collection stage
  - Pages with high text match scores
  - Their in- and out-neighbors (one hop)
- Each node u has two kinds of prestige score
  - Authority score $a[u]$, similar to PageRank, groundbreaking papers
  - Hub score $h[u]$, "reflected glory", good survey paper
- Mutual recurrence
  - Good hubs endorse good authorities:
    $h[u] = \sum_{v:(u,v)\in E} a(v)$ or $\boldsymbol{h} = \boldsymbol{E}\,\boldsymbol{a}$
  - Good authorities endorsed by good hubs:
    $a[v] = \sum_{u:(u,v)\in E} h(u)$ or $\boldsymbol{a} = \boldsymbol{E}^{\top}\boldsymbol{h}$

HITS calculations

- Equivalently, $\boldsymbol{h} = \boldsymbol{E}\boldsymbol{a} = \boldsymbol{E}\boldsymbol{E}^{\mathsf{T}}\boldsymbol{h}$ and $\boldsymbol{a} = \boldsymbol{E}^{\mathsf{T}}\boldsymbol{h} = \boldsymbol{E}^{\mathsf{T}}\boldsymbol{E}\boldsymbol{a}$
- Unlike PageRank, $\boldsymbol{E}\boldsymbol{E}^{\mathsf{T}}$ and $\boldsymbol{E}^{\mathsf{T}}\boldsymbol{E}$ are not stochastic; norm control needed
- Initialize $\boldsymbol{h} \leftarrow \mathbf{1},\ \boldsymbol{a} \leftarrow \mathbf{1}$
- Some number of iterations:
  - $\boldsymbol{h}_{\approx} \leftarrow \boldsymbol{E}\boldsymbol{a}$; $\boldsymbol{h} \leftarrow \boldsymbol{h}_{\approx}/\|\boldsymbol{h}_{\approx}\|_1$ // hub update
  - $\boldsymbol{a}_{\approx} \leftarrow \boldsymbol{E}^{\mathsf{T}}\boldsymbol{h}$; $\boldsymbol{a} \leftarrow \boldsymbol{a}_{\approx}/\|\boldsymbol{a}_{\approx}\|_1$ // authority update

Stability

Learning to rank in graphs