Sidebar: Functional PageRank (PR)

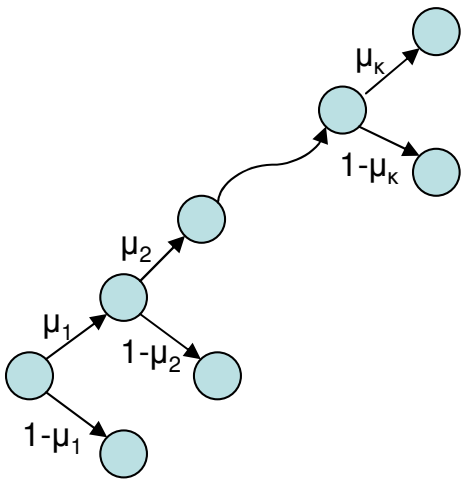
Computing PageRank (PR)

- PageRank as a *random surfer process*: Start surfing from a random node and keep following links with probability μ restarting with probability $1-\mu$; the node for restarting will be selected based on a personalization vector v. The ranking value x_i of a node i is the probability of visiting this node during surfing.
- PR can also be cast in power series representation as $x = (1-\mu)\sum_{j=0}^{k} \mu^j S^j v$; S encodes column-stochastic adjacencies.

Functional rankings

- A general method to assign ranking values to graph nodes as $x = \sum_{j=0}^{k} \zeta_j S^j v$. PR is a functional ranking, $\zeta_j = (1-\mu)\mu^j$.
- Terms attenuated by outdegrees in S *and* damping coefficients ζ_j.

Functional Rankings Through Multidamping [Kollias, Gallopoulos, AG, TKDE'13]



Computing μ_j in multidamping

Simulate a functional ranking by random surfers following emanating links with probability μ_j at step j given by :

$$\mu_j = 1 - \frac{1}{1+\frac{\rho_{k-j+1}}{1-\mu_{j-1}}}, j = 1, ..., k,$$

where $\mu_0 = 0$ and $\rho_{k-j+1} = \frac{\zeta_{k-j+1}}{\zeta_{k-j}}$

Examples

LinearRank (LR) $x^{\mathrm{LR}} = \sum_{j=0}^{k} \frac{2(k+1-j)}{(k+1)(k+2)} S^j v : \mu_j = \frac{j}{j+2}, j = 1, ..., k.$

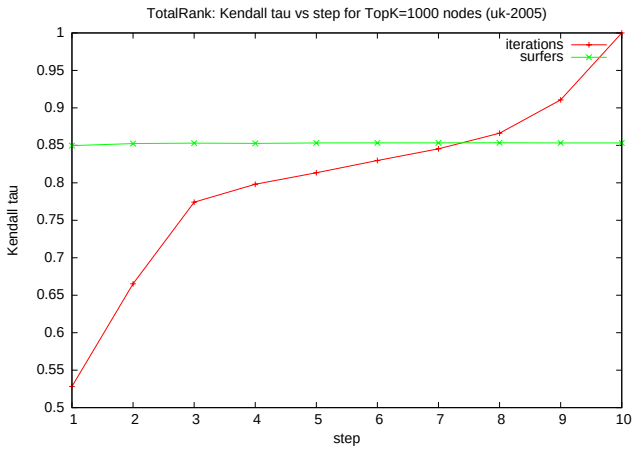
TotalRank (TR) $x^{\mathrm{TR}} = \sum_{j=0}^{\infty} \frac{1}{(j+1)(j+2)} S^j v : \mu_j = \frac{k-j+1}{k-j+2}, j = 1, ..., k.$

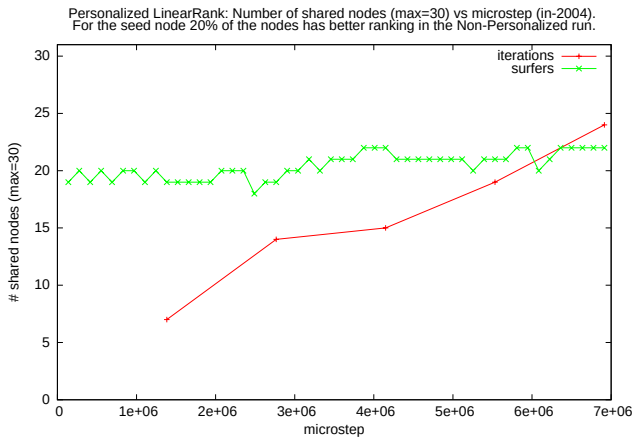
Multidamping and Computational Cost

Advantages of multidamping

- Interpretability and Design!
- Reduced computational cost in *approximating* functional rankings using the Monte Carlo approach. A random surfer terminates with probability $1 - \mu_j$ at step j.
- Inherently parallel and synchronization free computation.

Multidamping Performance

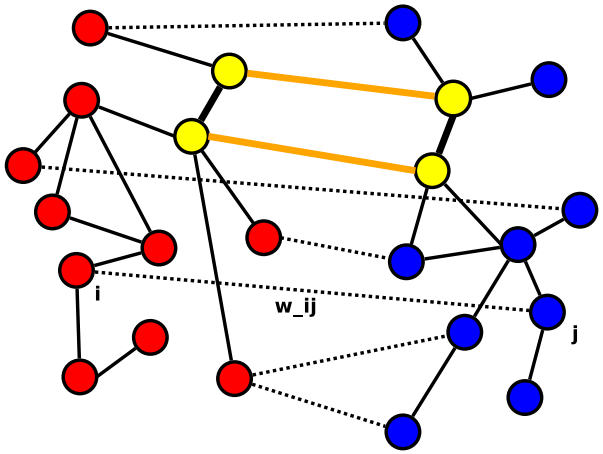




Approximate ranking: Run n surfers to completion for graph size n. How well does the computed ranking capture the "reference" ordering for top-k nodes, compared to standard iterations of equivalent computational cost/number of operations? *[Left]*

Approximate personalized ranking: Run $< n$ surfers to completion (each called a microstep, x-axis), but only from a selected node (personalized). How well can we capture the "reference" top-k nodes, i.e., how many of them are shared (y-axis), compared to the iterative approach of equivalent computational cost? *[Right]*

Sidebar: Graph Alignment



- **Node similarity:** Two nodes are similar if they are linked by other similar node pairs. By pairing similar nodes, the two graphs become *aligned*.

- Let $\tilde{A}$ and $\tilde{B}$ be the normalized adjacency matrices of the graphs (normalized by columns), H_{ij} be the independently known similarity scores (preferences matrix) of nodes $i \in V_B$ and $j \in V_A$, and μ be the fractional contribution of topological similarity.

- To compute X, IsoRank iterates:

$$X \leftarrow \mu \tilde{B} X \tilde{A}^T + (1 - \mu) H$$

Network Similarity Decomposition (NSD) [Kollias, Mohammadi, AG, TKDE'12]

Network Similarity Decomposition (NSD)

- In n steps of we reach
  $X^{(n)} = (1-\mu)\sum_{k=0}^{n-1} \mu^k \tilde{B}^k H(\tilde{A}^T)^k + \mu^n \tilde{B}^n H(\tilde{A}^T)^n$
- Assume that $H = uv^T$ (1 component). Two phases for X:
  1. $u^{(k)} = \tilde{B}^k u$ and $v^{(k)} = \tilde{A}^k v$ *(preprocess/compute iterates)*
  2. $X^{(n)} = (1-\mu)\sum_{k=0}^{n-1} \mu^k u^{(k)} {v^{(k)}}^T + \mu^n u^{(n)} {v^{(n)}}^T$ *(construct X)*

  This idea extends to s components, $H \sim \sum_{i=1}^{s} w_i z_i^T$.
- NSD computes matrix-vector iterates and builds X as a sum of outer products; these are much cheaper than triple matrix products.

We can then apply Primal-Dual or Greedy Matching (1/2 approximation) to extract the actual node pairs.

NSD: Performance [Kollias, Madan, Mohammadi, AG, BMC RN'12]

Species	Nodes	Edges
celeg (worm)	2805	4572
dmela (fly)	7518	25830
ecoli (bacterium)	1821	6849
hpylo (bacterium)	706	1414
hsapi (human)	9633	36386
mmusc (mouse)	290	254
scere (yeast)	5499	31898

Species pair	NSD (secs)	PDM (secs)	GM (secs)	IsoRank (secs)
celeg-dmela	**3.15**	152.12	7.29	783.48
celeg-hsapi	**3.28**	163.05	9.54	1209.28
celeg-scere	**1.97**	127.70	4.16	949.58
dmela-ecoli	**1.86**	86.80	4.78	807.93
dmela-hsapi	**8.61**	590.16	28.10	7840.00
dmela-scere	**4.79**	182.91	12.97	4905.00
ecoli-hsapi	**2.41**	79.23	4.76	2029.56
ecoli-scere	**1.49**	69.88	2.60	1264.24
hsapi-scere	**6.09**	181.17	15.56	6714.00

- We compute similarity matrices X for various pairs of species using Protein-Protein Interaction (PPI) networks. $\mu = 0.80$, uniform initial conditions (outer product of suitably normalized 1's for each pair), 20 iterations, one component.
- We then extract node matches using PDM and GM.
- *Three orders of magnitude speedup* from NSD-based approaches compared to IsoRank.

NSD: Parallelization [KKG JPDC'13, Submitted, KMSAG ParCo'13 Submitted]

Parallelization: NSD has been ported to parallel and distributed platforms.

- We have aligned up to million-node graph instances using over 3K cores.
- We process graph pairs of over a billion nodes and twenty billion edges each (!), on MapReduce-based distributed platforms.