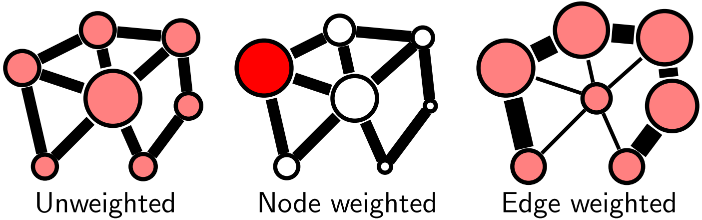
Model Reduction for Edge-Weighted Personalized PageRank

D. Bindel

21 Oct 2015

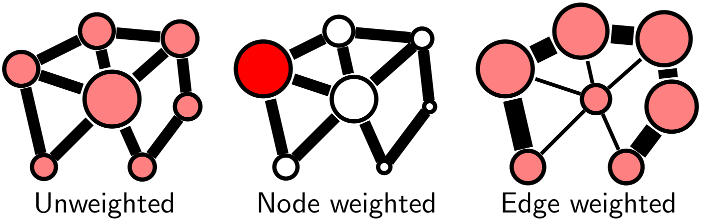
PageRank Problem



Goal: Find "important" vertices in a network

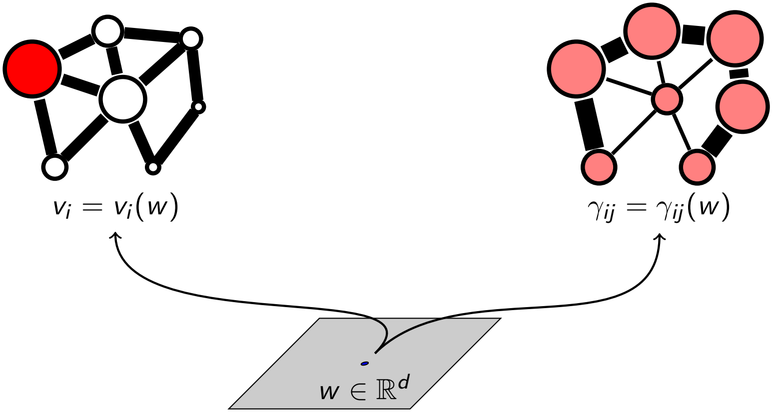
- Basic approach uses only topology
- Weights incorporate prior info about important nodes/edges

PageRank Model



- Random surfer model: $x^{(t+1)} = \alpha P x^{(t)} + (1-\alpha)v$ where $P = AD^{-1}$
- Stationary distribution: $Mx = b$ where $M = (I - \alpha P)$, $b = (1-\alpha)v$

Edge Weight vs Node Weight Personalization



Introduce *personalization parameters* $w \in \mathbb{R}^d$ in two ways:

Node weights: $M\ x(w) = b(w)$

Edge weights: $M(w)\ x(w) = b$

Edge Weight vs Node Weight Personalization

Node weight personalization is well-studied

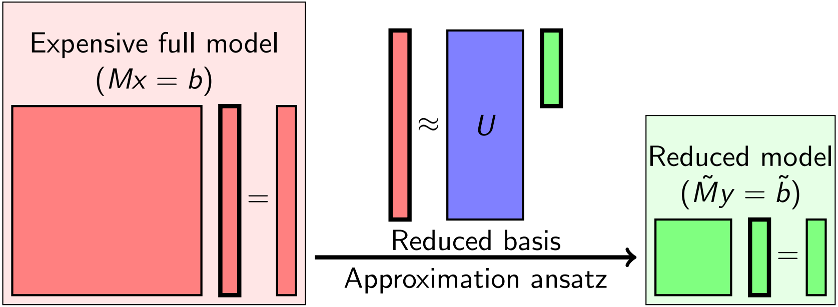
- Topic-sensitive PageRank: fast methods based on linearity
- Localized PageRank: fast methods based on sparsity

Some work on edge weight personalization

- ObjectRank/ScaleRank: personalize weights for different edge types
- But lots of work incorporates edge weights *without* personalization

Our goal: General, fast methods for edge weight personalization

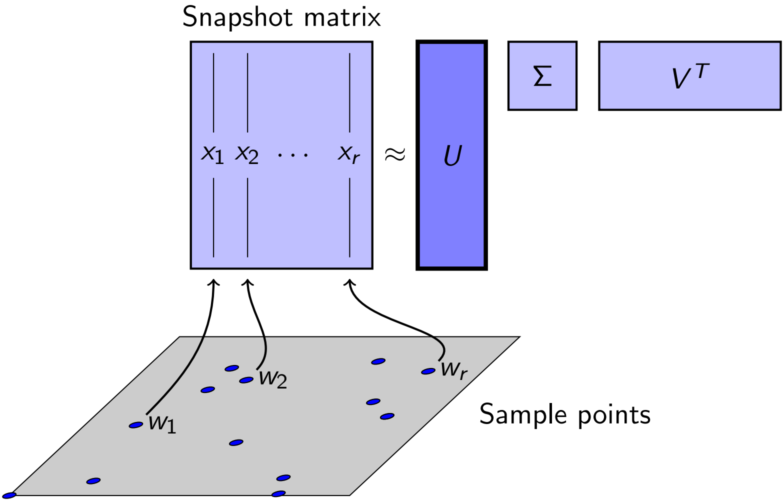
Model Reduction



Model reduction procedure from physical simulation world:

- *Offline*: Construct *reduced basis* $U \in \mathbb{R}^{n \times k}$
- *Offline*: Choose $\geq k$ equations to pick approximation $\hat{x} = Uy$
- *Online*: Solve for $y(w)$ given w and reconstruct $\hat{x}$

Reduced Basis Construction: SVD (aka POD/PCA/KL)



Choosing Good Spaces

What is the best possible approximation $\hat{x} = Uy$?

$$\min_y \|Uy - x(w)\|_2 \leq \sigma_{k+1}\|x\|_2 + e_{\mathrm{interp}}(w)$$

where

$$e_{\mathrm{interp}}(w) = \left\| x(w) - \sum_{j=1}^{r} x(w_j) c_j(w) \right\|_2$$

is error in an interpolant.

- Pay attention where x has large derivatives!
- Also suggests sampling strategies (sparse grids, adaptive methods)

Approximation Ansatz

Want $r = MUy - b \approx 0$. Consider two approximation conditions:

Method	Ansatz	Properties
Bubnov-Galerkin	$U^T r = 0$	Good accuracy empirically Fast for $P(w)$ linear
DEIM	$\min \|r_{\mathcal{I}}\|$	Fast even for nonlinear $P(w)$ Complex cost/accuracy tradeoff

Petrov-Galerkin a bit more accurate than Bubnov-Galerkin – future work.

Bubnov-Galerkin Method

$$U^T \left[M U y - b \right] = 0.$$

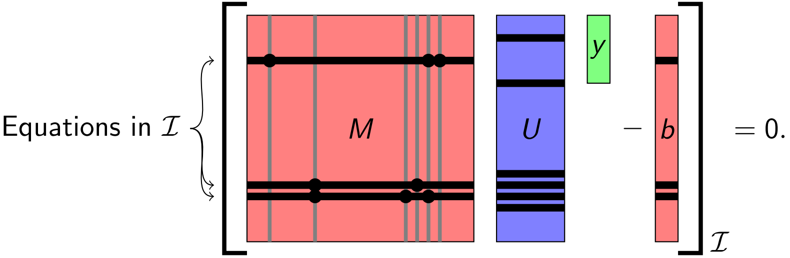
- Linear case: w_i = probability of transition with edge type i

$$M(w) = I - \alpha \left(\sum_i w_i P^{(i)} \right), \quad \tilde{M}(w) = I - \alpha \left(\sum_i w_i \tilde{P}^{(i)} \right)$$

  where we can precompute $\tilde{P}^{(i)} = U^T P^{(i)} U$

- Nonlinear: Cost to form $\tilde{M}(w)$ comparable to cost of PageRank!

Discrete Empirical Interpolation Method (DEIM)



- Ansatz: Minimize $\|r_{\mathcal{I}}\|$ for chosen indices $\mathcal{I}$
- Only need a few rows of M (and associated rows of U)
- Difference from physics applications: high-degree nodes!

Error Behavior

Similar error analysis framework for both Galerkin and DEIM

$$\text{Consistency} + \text{Stability} = \text{Accuracy}$$

- Consistency: Does the subspace contain good approximants?
- Stability: Is the approximation subproblem far from singular?

Characterize stability by a quasi-optimality condition

$$\|x - Uy\| \leq \min_z C\|x - Uz\|$$

Quasi-optimality (Galerkin)

Define $\tilde{M} = W^T M U$ and $\Pi = U \tilde{M}^{-1} W^T M$:

$$x - Uy = (I - \Pi)x \qquad\qquad 0 = (I - \Pi)U$$

So we have the Galerkin error relation

$$x - Uy = (I - \Pi)(x - Uz)$$

for any candidate solution Uz. Take norms and minimize over z:

$$\|e\| \leq (1 + \kappa_G)\|e_{\text{min}}\|$$

where

$$\kappa_G \equiv \|U\| \|\tilde{M}^{-1}\| \|W^T M\|.$$

Can actually bound offline for linear parameterization.

Quasi-optimality (DEIM)

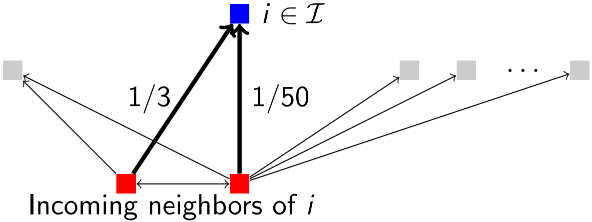
DEIM error relation derived similarly,

$$e = [I - \Pi]\, e_* = \left[I - U(M_{\mathcal{I},:}U)^\dagger M_{\mathcal{I},:} \right] e_*.$$

- Keep $M_{\mathcal{I},:}U$ far from singular $\implies$ keep DEIM stable
  - Also helps to explicitly enforce $\sum_i \hat{x}_i = 1$
- Suggests pivoting schemes to choose a "good" $\mathcal{I}$

Interpolation Costs

Consider subgraph relevant to one interpolation equation:



- *Really* care about weights of edges incident on $\mathcal{I}$
  - Need more edges to normalize (unless $A(w)$ is linear)
- High in/out degree are expensive but informative
- **Key question**: how to choose $\mathcal{I}$ to balance **cost** vs **accuracy**?

Interpolation Accuracy

- Key: keep $M_{\mathcal{I},:}$ far from singular.
- Want to pick $\mathcal{I}$ *once*, so look at rows of

$$Z = \begin{bmatrix} M(w_1)U & M(w_2)U & \dots \end{bmatrix}$$

  for sample parameters $w^{(i)}$.
- Pivoted QR-like greedy row selection with proxy measures for
  - **Cost**: Nonzeros in row (+ assoc columns if normalization required)
  - **Accuracy**: Residual when projecting row onto those previously selected
- Several heuristics for cost/accuracy tradeoff (see paper)

Online Costs

If $\ell = \#$ PR components needed, online costs are:

Form $\tilde{M}$	$O(dk^2)$ for B-G
	More complex for DEIM
Factor $\tilde{M}$	$O(k^3)$
Solve for y	$O(k^2)$
Form Uy	$O(k\ell)$

Online costs **do not** depend on graph size!
(unless you want the whole PR vector)

Example Networks

DBLP (citation network)

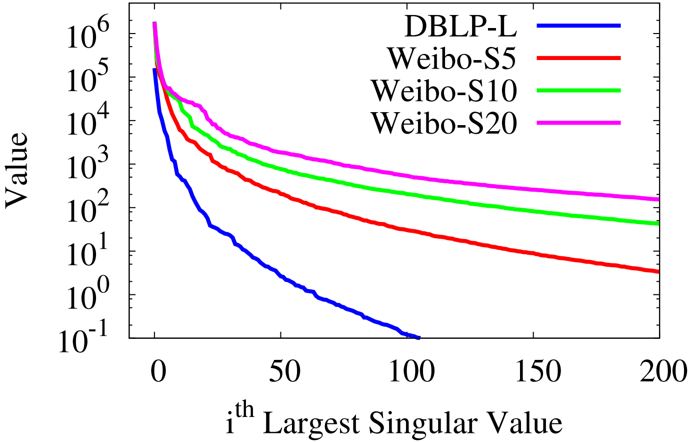
- 3.5M nodes / 18.5M edges
- Seven edge types $\Longrightarrow$ seven parameters
- $P(w)$ linear
- Competition: ScaleRank

Weibo (micro-blogging)

- 1.9M nodes / 50.7M edges
- Weight edges by topical similarity of posts
- Number of parameters = number of topics (5, 10, 20)

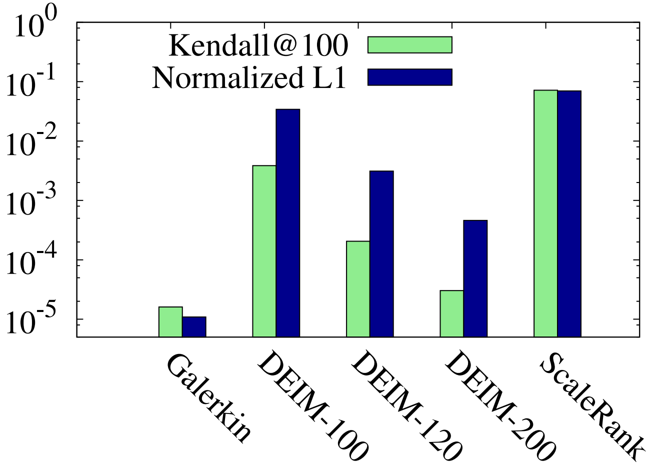
(Studied global and local PageRank – see paper for latter.)

Singular Value Decay

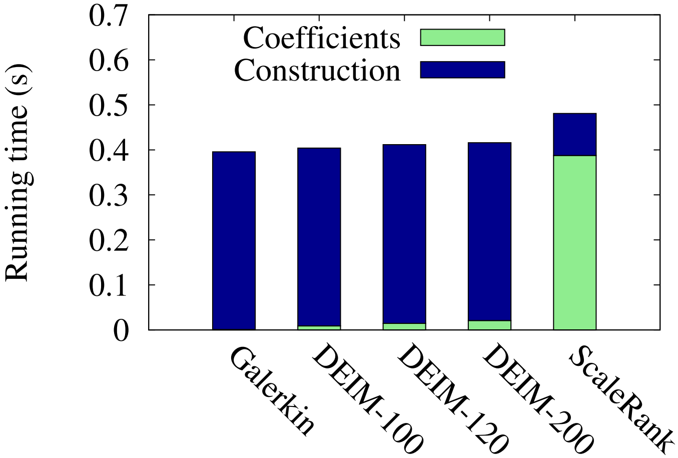


$r = 1000$ samples, $k = 100$

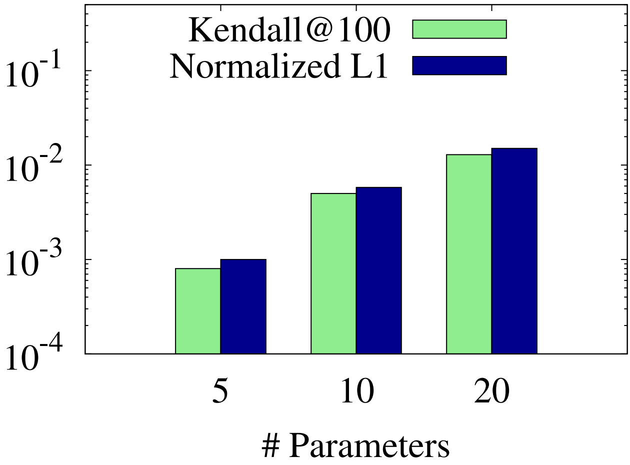
DBLP Accuracy



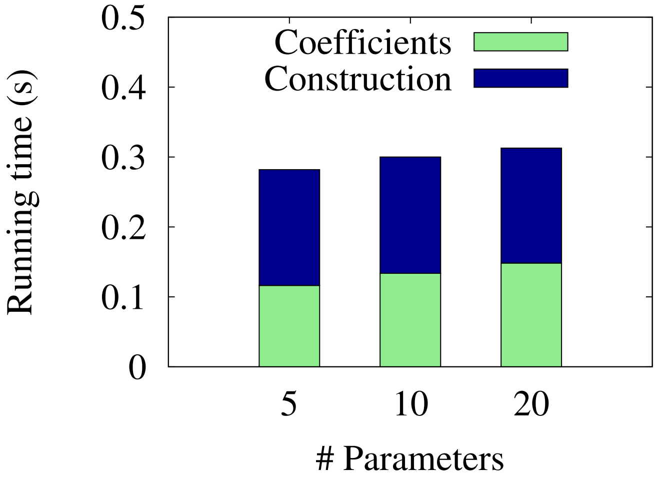
DBLP Running Times (All Nodes)



Weibo Accuracy



Weibo Running Times (All Nodes)



Application: Learning to Rank

Goal: Given $T = \{(i_q, j_q)\}_{q=1}^{|T|}$, find w that mostly ranks i_q over j_1.
(c.f. Backstrom and Leskovec, WSDM 2011)

Standard idea: Gradient descent

$$\frac{\partial x}{\partial w_j} = M(w)^{-1} \left[\alpha \frac{\partial P(w)}{\partial w_j} x(w) \right]$$

Dominant cost: $d + 1$ solves with the PageRank system $M(w)$

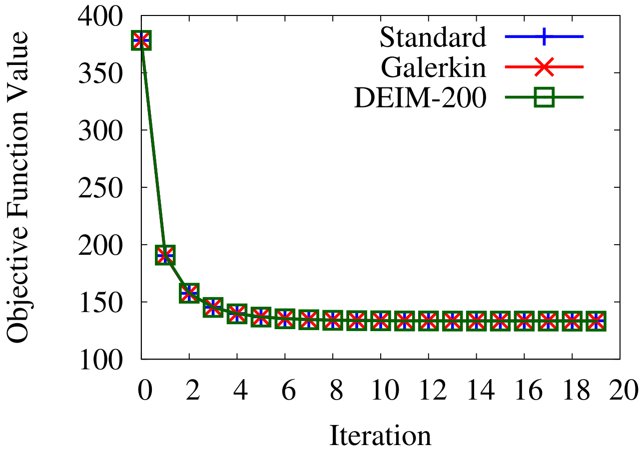
- One PageRank solve to evaluate a loss function
- One PageRank solve per parameter to evaluate gradients

Application: Learning to Rank

Goal: Given $T = \{(i_q, j_q)\}_{q=1}^{|T|}$, find w that mostly ranks i_q over j_1.
(c.f. Backstrom and Leskovec, WSDM 2011)

- Standard: Gradient descent on full problem
  - One PR computation for objective
  - One PR computation for each gradient component
  - Costs $d + 1$ PR computations per step
- With model reduction
  - Rephrase objective in reduced coordinate space
  - Use factorization to solve PR for objective
  - Re-use same factorization for gradient

DBLP Learning Task



(8 papers for training + 7 params)

The Punchline

Test case: DBLP, 3.5M nodes, 18.5M edges, 7 params

Cost per Iteration:

Method	Standard	Bubnov-Galerkin	DEIM-200
Time(sec)	159.3	0.002	0.033

Roads Not Taken

In the paper (but not the talk)

- Selecting interpolation equations for DEIM
- Localized PageRank experiments (Weibo and DBLP)
- Comparison to BCA for localized PageRank

Room for future work! Analysis, applications, systems, ...

Questions?

Edge-Weighted Personalized PageRank:
Breaking a Decade-Old Performance Barrier

Wenlei Xie, David Bindel, Johannes Gehrke, and Al Demers

KDD 2015, paper 117

Sponsors:

- NSF (IIS-0911036 and IIS-1012593)
- iAd Project from the National Research Council of Norway