Model Reduction for Edge-Weighted Personalized PageRank

D. Bindel

11 Apr 2016
The Computational Science & Engineering Picture

- MEMS
- Smart grids
- Networks
- Linear algebra
- Approximation theory
- Symmetry + structure
- HPC / cloud
- Simulators
- Solvers
Collaborators

Wenlei Xie LinkedIn
David Bindel Cornell
Johannes Gehrke Microsoft
Al Demers Cornell
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:** \( x^{(t+1)} = \alpha Px^{(t)} + (1 - \alpha)v \) where \( P = AD^{-1} \)
- **Stationary distribution:** \( Mx = b \) where \( M = (I - \alpha P), b = (1 - \alpha)v \)
Introduce *personalization parameters* $w \in \mathbb{R}^d$ in two ways:

- **Node weights:** $M \times(w) = b(w)$
- **Edge weights:** $M(w) \times(w) = b$
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
Different ways to personalize $\implies$ different algorithm options

1. **Linear**: Take an edge of type $i$ with probability $\alpha w_i$

   \[ P(w) = \sum_{i=1}^{d} w_i P^{(i)} \]

2. **Scaled linear**: Take an edge with probability $\propto$ (linear) edge weight

   \[ P(w) = A(w) D(w)^{-1}, \quad A(w) = \sum_{i=1}^{d} w_i A^{(i)}, \quad D(w) = \sum_{i=1}^{d} w_i D^{(i)}, \]

3. **Fully nonlinear**: Both $A$ and $P$ depend nonlinearly on $w$
Model Reduction

Expensive full model \((Mx = b)\)

\[
\begin{bmatrix}
M \\
\end{bmatrix}
\begin{bmatrix}
x \\
\end{bmatrix}
\approx
\begin{bmatrix}
\tilde{M} \\
\end{bmatrix}
\begin{bmatrix}
\tilde{x} \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
u \\
\end{bmatrix}
\]

Reduced basis

Reduced model \((\tilde{M}y = \tilde{b})\)

\[
\begin{bmatrix}
\end{bmatrix}
\]

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)

\[
\begin{bmatrix}
\mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_r
\end{bmatrix} \approx \mathbf{U} \mathbf{V}^T
\]

Snapshot matrix

Sample points

\( \mathbf{w}_1 \), \( \mathbf{w}_2 \), \ldots, \( \mathbf{w}_r \)
Choosing Good Spaces

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

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

where

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

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:

<table>
<thead>
<tr>
<th>Method</th>
<th>Ansatz</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubnov-Galerkin</td>
<td>$U^Tr = 0$</td>
<td>Good accuracy empirically</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fast for $P(w)$ linear</td>
</tr>
<tr>
<td>DEIM (collocation)</td>
<td>$\min</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Complex cost/accuracy tradeoff</td>
</tr>
</tbody>
</table>

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

\[ \begin{bmatrix} U^T & M & U \end{bmatrix} - b = 0. \]

- 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!
Equations in $\mathcal{I}$

\[
\begin{bmatrix}
M & U \\
\end{bmatrix} \begin{bmatrix}
\mathcal{I} \\
y \\
- b \\
\end{bmatrix} = 0.
\]

- Ansatz: Minimize $\|r_\mathcal{I}\|$ for chosen indices $\mathcal{I}$
- Only need a few rows of $M$ (and associated rows of $U$)
  - If given $A(w)$, also need column sums for normalization.
- 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

\[
\| x - Uy \| \leq \min_z C \| x - Uz \|
\]
Define a solution projector:

\[ \Pi x = \text{approximate solution when true solution is } x \]

Note that \( \Pi U = U \).

The error projector \( I - \Pi \) maps a true solution to error

\[ e = x - \Pi x = (I - \Pi)x \]

Note that \( (I - \Pi)U = 0 \).

If \( e_{\text{min}} = x - Uz \) is the smallest norm error in the space, then

\[ e = (I - \Pi)x - (I - \Pi)Uz = (I - \Pi)e_{\text{min}} \]

Therefore, a bound on \( \|I - \Pi\| \leq 1 + \|\Pi\| \) establishes quasi-optimality.
Quasi-Optimality: Galerkin and DEIM

Galerkin: \[ \Pi = U \tilde{M}^{-1} W^T M \quad \tilde{M} \equiv W^T M U \]

DEIM: \[ \Pi = U \tilde{M}^\dagger M_{\mathcal{I},:} \quad \tilde{M} \equiv M_{\mathcal{I},:} U \]

- Key to stability: \( \tilde{M} \) far from singular
- Suggests pivoting schemes for “good” \( \mathcal{I} \) in DEIM
  - Also helps to explicitly enforce \( \sum_i \hat{x}_i = 1 \)
- Can bound \( \|\Pi\| \) offline for Galerkin + linear parameterization.
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 $A(w)$ linear)
- Cost to include $i \in \mathcal{I}$: $|\{j, k : a_{ij} \neq 0 \text{ and } a_{kj} \neq 0\}|$
- High in/out degree are expensive but informative

Incoming neighbors of $i$
Interpolation Cost and Accuracy

- **Key question**: how to choose \( \mathcal{I} \) to balance \textbf{cost} vs \textbf{accuracy}?
- Want to pick \( \mathcal{I} \) once, so look at rows of
  \[
  Z = \begin{bmatrix}
  M(w_1)U & M(w_2)U & \ldots 
  \end{bmatrix}
  \]
  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)
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 $\rightarrow$ 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

\[ \text{Value} \]

- DBLP-L
- Weibo-S5
- Weibo-S10
- Weibo-S20

\[ i^{th} \text{ Largest Singular Value} \]

\[ r = 1000 \text{ samples, } k = 100 \]
DBLP Accuracy

The diagram shows a comparison of different methods for approximating the DBLP accuracy. The methods include Galerkin, DEIM-100, DEIM-120, DEIM-200, and ScaleRank. The accuracy is measured using Kendall@100 and the normalized L1 norm. The graph compares these methods across different scales, with the y-axis showing the normalized L1 norm on a logarithmic scale from $10^{-5}$ to $10^0$.
DBLP Running Times (All Nodes)

![Bar Chart](image)

- **Coefficients**
- **Construction**

- **Galerkin**
- **DEIM-100**
- **DEIM-120**
- **DEIM-200**
- **ScaleRank**
Weibo Accuracy

Kendall@100
Normalized L1

# Parameters

5 10 20
Weibo Running Times (All Nodes)

D. Bindel
Purdue
11 Apr 2016
Application: Learning to Rank

Goal: Given $T = \{(i_q, j_q)\}_{q=1}^{\lvert T \rvert}$, 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
Objective Function Value vs. Iteration

- Standard
- Galerkin
- DEIM-200

(8 papers for training + 7 params)
The Punchline

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

Cost per Iteration:

<table>
<thead>
<tr>
<th>Method</th>
<th>Standard</th>
<th>Bubnov-Galerkin</th>
<th>DEIM-200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec)</td>
<td>159.3</td>
<td>0.002</td>
<td>0.033</td>
</tr>
</tbody>
</table>
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
Welcome to SIAM ALA in Hong Kong!

Hong Kong Baptist University, 4-8 May 2018