Efficient and Accurate Label Propagation on Dynamic Graphs and Label Sets

Michele Covell and Shumeet Baluja
Google Research
Google Inc., Mountain View CA, USA
covell@google.com shumeet@google.com

Abstract—**Many web-based application areas must infer label distributions starting from a small set of sparse, noisy labels. Previous work has shown that graph-based propagation can be very effective at finding the best label distribution across nodes, starting from partial information and a weighted-connection graph. In their work on video recommendations, Baluja et al. showed high-quality results using *Adsorption*, a normalized propagation process. An important step in the original formulation of Adsorption was re-normalization of the label vectors associated with each node, between every propagation step. That interleaved normalization forced computation of all label distributions, in synchrony, in order to allow the normalization to be correctly determined. Interleaved normalization also prevented use of standard linear-algebra methods, like stabilized bi-conjugate gradient descent (*BiCGStab*) and Gaussian elimination. We show how to replace the interleaved normalization with a single pre-normalization, done once before the main propagation process starts, allowing use of selective label computation *(label slicing)* as well as large-matrix-solution methods. As a result, much larger graphs and label sets can be handled than in the original formulation and more accurate solutions can be found in fewer propagation steps. We further extend that work to handle graphs that change and expand over time. We report results from using pre-normalized Adsorption in topic labeling for web domains, using label slicing and BiCGStab. We also report results from using incremental updates on changing co-author network data. Finally, we discuss two options for handling mixed-sign (positive and negative) graphs and labels.**

Keywords - graph propagation, large-scale labeling, incremental connection-graph changes, stabilized bi-conjugate gradient descent, Gaussian elimination, topic discovery, web domains.

I. INTRODUCTION

Many different approaches have recently been proposed to label propagation across weighted graphs of nodes [1]-[7]. Applications include searching for, recommending, and advertising against image, audio, and video content. These labeling problems must handle millions of interconnected entities (users, domains, content segments) and thousands of competing labels (interests, tags, recommendations, topics). These applications share the characteristics of having a limited amount of label data, often of uneven quality, associated with a large graph of weighted connections between many nodes, some unlabeled and some partially labeled.

We build on the work done by Zhu and Ghahramani [3], Baluja et al. [2] and Covell and Baluja [1]. The Baluja paper [2] described *Adsorption*, a graph-based approach to estimating label distributions, which was applied to providing YouTube video recommendations. The resulting top-pick recommendation was more accurate than the next-best alternative algorithm for all users who had watched 3 or more previous videos, with accuracy improvements of up to 100% for the most frequent watchers. In Adsorption [1],[2], each node (e.g., each video for which we are building a recommendation list) has a limited capacity for labels (e.g., the proposed recommendations for that video). Baluja et al. [2] enforce this constraint by interleaving a normalization step at each node, in between every propagation step. Without this normalization, the solution is not guaranteed to converge.

The interleaved normalization step is needed for convergence but prevents label slicing: under the original formulation [2], we cannot find the estimated distribution of a subset of labels without solving for the full set of labels first. Furthermore, the interleaved normalization prevents the use of most standard linear-algebra techniques, such as Gaussian elimination of nodes that are not of direct interest (though they still are needed for their effect on the remainder of the graph). Additionally, methods for rapid convergence to the final solution, such as stabilized bi-conjugate gradient descent (BiCGStab), cannot be used in the original formulation.

We start the paper with a recap of the original Adsorption application and mathematical description [2], in Section II. This paper then reviews and expands on the work presented by Covell and Baluja [1] for pre-normalizing the Adsorption graph and label weights, such that there is no need for interleaved normalization (Section III). With this, we can use BiCGStab and Gaussian elimination. Our graph size contains more than 10 million nodes and 4 billion interconnections (i.e., more than 10 million rows and more than 4 billion non-zero entries in the corresponding matrix), which is more than we can reasonably handle in straightforward implementations of these techniques. Instead, we use implementations of BiCGStab and Gaussian elimination in the MapReduce framework. We describe these implementations briefly, in Sections IV and V. In Section VI, we present our results on topic labeling of web domains, using a graph based on shared keywords between pages across the domains.

In Section VII, we extend the pre-normalized framework [1] to handle fast updates for graphs with newly added nodes and changing connection weights between existing nodes. In Section VIII, we demonstrate this incremental-update approach on a co-author network, as seen originally seen in 2003 and then updated in 2005. Finally, in Section IX, we discuss two alternative approaches to handling negative associations.

II. Adsorption (with interleaved normalization)

The original formulation of Adsorption [2] can be described as an iteration using two systems of equations:

$$\underline{\underline{\tilde{X}}}_{n+1} = \sigma \underline{\underline{X}}_n + \beta \underline{\underline{W}}\,\underline{\underline{X}}_n + \left[\begin{array}{cc} \gamma \underline{\underline{L}} & \delta \underline{1} \end{array}\right] \tag{1}$$

$$\left\{\underline{\underline{X}}_{n+1}\right\}_{i*} = \left\{\underline{\underline{\tilde{X}}}_{n+1}\right\}_{i*} \Big/ \left\|\left\{\underline{\underline{\tilde{X}}}_{n+1}\right\}_{i*}\right\|_1 \tag{2}$$

where double underlining indicates a matrix of values, a single underline is a vector, not-underlined values are scalars, and the tilde indicates a not-normalized set of values. The matrix $\underline{\underline{W}}$ holds the connection weights with row i giving the incoming connections into the i'th node. This matrix often is symmetric, to start with, but this property is not required and will be given up later to allow for pre-normalization. The matrix $\underline{\underline{L}}$ holds the weights of the *injection label* information. These are often noisy or incomplete label sets based on some prior information, with the graph propagation as a way to improve and expand these label sets. In $\underline{\underline{L}}$, each label is associated with a column and the weights for the injection labels for the i'th node of the graph are in the i'th row of the matrix. In addition to the true labels, in $\underline{\underline{L}}$, Baluja et al. [2] add an *abandonment label*, represented in Equation (1) by the appended column $\delta \underline{1}$. The scalar δ can be thought of in many different ways: as the loss in certainty about any of the labels that are propagated for one hop in the graph; as the number of random walks through the graph that end with "abandonment", giving no final label set; as the regularization margin in the system of equations. The other scalars (σ, β, and γ) allow graph-wide balancing of the previous (same-node) labels, of the propagated neighbors' labels, and of the injection labels. Finally, the matrix $\underline{\underline{X}}_n$ is the label distribution estimate, with the i'th row containing the estimated labels for the i'th node, including as the last column the abandonment label. In this context, the node's abandonment weight provides a measure, at that node, of the label uncertainty.

Equation (1) creates a new *un-normalized* estimate of the steady-state label distribution across all the nodes using a weighted combination of the previous normalized estimate for the distribution ($\underline{\underline{X}}_n$), of a graph-weighted propagated version of that same distribution ($\underline{\underline{W}}\,\underline{\underline{X}}_n$), of injection labels ($\underline{\underline{L}}$), and of the abandonment label (δ). Equation (2) provides a normalized estimate of the label distribution, by dividing each row of the estimate from Equation (1) by the L_1 norm of the full label set, including the abandonment label.

Iterating over Equations (1) and (2) together is guaranteed to converge to a stable steady-state solution, as long as δ is greater than 0. Baluja et al. [2] used this algorithm to successfully provide video recommendations that, using a top-pick-accuracy measure, outperformed alternative approaches. Our goal is to provide a formulation for the same Adsorption algorithm that does not require per-propagation-step normalization, allowing us to use label slicing and standard linear-algebra tools.

III. Pre-normalized Adsorption

We achieve our goal of pre-normalized Adsorption by first assuming that all associations in our graph and in our label injection are non-negative. Specifically: $sign\left(\left\{\underline{\underline{X}}_n\right\}_{ij}\right) \geq 0$, $sign\left(\left\{\underline{\underline{W}}\right\}_{ij}\right) \geq 0$, and $sign\left(\left\{\underline{\underline{L}}\right\}_{ij}\right) \geq 0$.

This non-negative assumption works well with the partial-information applications that are the most common ones in large-graph labeling formulations: for example, in video recommendation, we can say that two videos are often watched together, within a single viewing session, but it is much more difficult to say that two videos are negatively associated (that watching one means you are significantly less likely to watch the other), since we seldom have enough training data to make such an assertion with any confidence.

For those applications where we do have confidence in negative label-to-node associations (negative values in $\underline{\underline{L}}$), we can handle these by introducing a negated label column and using positive associations with the negated label where we would have otherwise used negative associations with the positive label. Handling negative node-to-node connections (negative values in $\underline{\underline{W}}$) is also possible. We go over all of these cases in more detail in Section IX.

Assuming we have non-negative values in our component matrices, we can consider the denominator of Equation (2) in more detail:

$$\left\|\left\{\underline{\underline{\tilde{X}}}_{n+1}\right\}_{i*}\right\|_1 = \left\|\left\{\left(\sigma \underline{\underline{I}} + \beta \underline{\underline{W}}\right)\underline{\underline{X}}_n + \left[\begin{array}{cc} \gamma \underline{\underline{L}} & \delta \underline{\underline{1}} \end{array}\right]\right\}_{i*}\right\|_1 \tag{3}$$

$$= \sum_j \left(\sum_k \left\{\sigma \underline{\underline{I}} + \beta \underline{\underline{W}}\right\}_{ik} \left\{\underline{\underline{X}}_n\right\}_{kj} \right) + \sum_j \left\{\left[\begin{array}{cc} \gamma \underline{\underline{L}} & \delta \underline{\underline{1}} \end{array}\right]\right\}_{ij} \tag{4}$$

$$= \sum_k \left\{\sigma \underline{\underline{I}} + \beta \underline{\underline{W}}\right\}_{ik} \sum_j \left\{\underline{\underline{X}}_n\right\}_{kj} + \gamma \sum_j \left\{\underline{\underline{L}}\right\}_{ij} + \delta \tag{5}$$

$$= \sum_k \left\{\sigma \underline{\underline{I}} + \beta \underline{\underline{W}}\right\}_{ik} + \gamma \left\|\left\{\underline{\underline{L}}\right\}_{i*}\right\|_1 + \delta \tag{6}$$

$$= \sigma + \beta \left\|\left\{\underline{\underline{W}}\right\}_{i*}\right\|_1 + \gamma \left\|\left\{\underline{\underline{L}}\right\}_{i*}\right\|_1 + \delta \tag{7}$$

Equation (3) simply provides the expansion of the L_1 row norm using the propagation Equation (1). Equation (4) makes use of the non-negativity conditions that we are requiring, in order to remove the absolute values implied by the L_1 norm and expands the norm summation, as well as the summation implicit in the $\underline{\underline{W}}\,\underline{\underline{X}}_n$ matrix multiply. Equation (5) swaps the order of summation, allowing us to make use of the unit L_1 row norm for $\underline{\underline{X}}_n$ in Equation (6). Simplifying the summations and noting the use of the row-norm definitions for $\underline{\underline{L}}$ and $\underline{\underline{W}}$ finally results in Equation (7).

The useful property of Equation (7) is that $\left\| \left\{ \tilde{\underline{\underline{X}}}_{n+1} \right\}_{i*} \right\|_1$ depends only the initial combination weights and the row norms of $\underline{\underline{L}}$ and $\underline{\underline{W}}$. We can use this property to pre-normalize by first defining

$$\lambda_i = \sigma + \beta \left\| \left\{ \underline{\underline{W}} \right\}_{i*} \right\|_1 + \gamma \left\| \left\{ \underline{\underline{L}} \right\}_{i*} \right\|_1 + \delta \tag{8}$$

$$\hat{\sigma}_i = \sigma / \lambda_i \qquad \hat{\underline{\underline{\sigma}}} = diag(\hat{\sigma}_i) \tag{9}$$

$$\hat{\beta}_i = \beta \left\| \left\{ \underline{\underline{W}} \right\}_{i*} \right\|_1 / \lambda_i \qquad \hat{\underline{\underline{\beta}}} = diag(\hat{\beta}_i) \tag{10}$$

$$\hat{\gamma}_i = \gamma \left\| \left\{ \underline{\underline{L}} \right\}_{i*} \right\|_1 / \lambda_i \qquad \hat{\underline{\underline{\gamma}}} = diag(\hat{\gamma}_i) \tag{11}$$

$$\hat{\delta}_i = \delta / \lambda_i \qquad \hat{\underline{\delta}} = vec(\hat{\delta}_i) \tag{12}$$

and then using these new quantities in a pre-normalized Adsorption algorithm.

$$\underline{\underline{X}}_{n+1} = \hat{\underline{\underline{\sigma}}} \underline{\underline{X}}_n + \hat{\underline{\underline{\beta}}} \underline{\underline{W}} \underline{\underline{X}}_n + \left[\begin{array}{cc} \hat{\underline{\underline{\gamma}}} \underline{\underline{L}} & \hat{\underline{\delta}} \end{array} \right] \tag{13}$$

Note that direct use of Equation (13) is exactly the power-iteration approach to finding the solution (used in [2]) and will give the same solutions at every iteration as the combination of Equations (1) and (2): the pre-normalization has the exact same effect, even though it is only done once, as the interleaved normalizations. Equation (13), therefore, also is guaranteed to converge to a stable solution, just as the original Adsorption algorithm is guaranteed. The advantage is that we do not need to normalize at each step and, as a result, we can compute an incomplete set of labels, while still deriving the benefits of the full label set to limit belief within the set of labels that are interested in. This slicing directly reduces the computational costs by the same percentage as the percentage of dropped labels. Furthermore, with the use of Equation (13) as the system of equations for which we want a solution, we can use standard linear-algebra tools, like BiCGStab (for faster convergence) and Gaussian elimination (for shrinking our graph matrix). We discuss these algorithms and their large-graph implementations next.

IV. Map-Reduce formulation of Stabilized Bi-Conjugate Gradient Descent (BiCGStab)

In [2], Baluja et al. implicitly use power iteration to solve their system of constraints. For symmetric systems of constraints, gradient-descent methods can find solutions in fewer iterations, for any given level of accuracy (as measured by the average residual error). However, due to the pre-normalization of Adsorption, we no longer have a symmetric matrix, and must move to bi-conjugate gradient approaches. Since the most direct generalization (biconjugate gradient descent) is not numerically stable, we focus on stabilized biconjugate gradient descent [8], which has been shown to converge more uniformly than power iteration, without the numerical issues of (not-stabilized) bi-conjugate gradient descent. We ran several simulations using power iteration and BiCGStab, based on random graph matrices with the same level of regularization as we expect to see through the abandonment variable in our true graphs. In these tests, when the graph matrix and the beginning label estimates were non-sparse, on average, BiCGStab converged to the correct solution 12 times faster than the power-iteration method (e.g., BiCGStab would converge in two iterations, requiring only 5 graph-matrix multiplies, while power iteration would require 60 iterations, needing 60 graph-matrix multiplies to converge to the same level of accuracy).

When the graph matrix and the beginning label estimates were sparse, there were similar differences in the rate of convergence, away from the "wavefront boundary". We use the term *wavefront* to emphasize that (for both power iteration and BiCGStab), updates are done in such a way that non-zero values propagate through the graph according to the neighborhood connections. When the labels are sparsely injected, non-zero values move in a "wave", outward from non-zero areas into areas that were zero (due to sparseness). Both power iteration and BiCGStab rely on the graph matrix to determine the label-estimate update, so both have their non-zero wavefronts progress in the same way.

Due to the size of the graph over which we will be operating, we implemented BiCGStab using three MapReduce [9] stages per iteration. Using the notation from the Wikipedia article on BiCGStab [10], we have a distinct set of vectors for each of the labels on which we want to estimate the final distribution. We arrive at the BiCGStab components $\underline{\underline{A}}$ and $\underline{b}$ (at least conceptually) by separating $\hat{\underline{\underline{\gamma}}}\underline{\underline{L}}$ into columns corresponding to $\underline{b}$, by separating $\underline{\underline{X}}_n$ into columns corresponding to $\underline{x}_n$ and by using

$$\underline{\underline{A}} = \underline{\underline{I}} - \hat{\underline{\underline{\sigma}}} - \hat{\underline{\underline{\beta}}}\underline{\underline{W}} \tag{14}$$

We select an initial *shadow direction* $\hat{\underline{r}}_0$ for each column aligned with its first-pass residual vector, $\underline{r}_0$. Note that computing the first-pass residual vector takes one MapReduce to compute $\underline{r}_0 = b - \underline{\underline{A}}\underline{x}_0$. (For our applications, $\underline{b}$ itself is often a good initial estimate for $\underline{x}$.) It is this separate estimation of each column (where each column corresponds to a single label) that makes label slicing so simple and powerful in combination with BiCGStab.

Unlike [10], we mark all our auxiliary variables with the iteration on which they were computed, since this makes our Reduce processing more uniform and reliable: therefore, we use α_n, $\underline{s}_n$ and $\underline{t}_n$ here (instead of their un-versioned form from [10]). To allow the remaining framework to operate smoothly, starting from the initialization (the 0'th pass), we also use the settings for our auxiliary variables that are suggested in [10], namely: $\rho_0 = \alpha = \omega_0 = 1$ and $\underline{v}_0 = \underline{p}_0 = \underline{0}$

For all iterations after this initialization, there are 3 MapReduce stages: (*A*) updating the search direction and its projection through $\underline{\underline{A}}$; (*B*) updating the shadow direction and its projection through $\underline{\underline{A}}$; and (*C*) combining the computed components to give a new state estimate and residual.

For all three MapReduce stages, the reduce processing is the same: from the set of inputs computed in the Map stage, as well as the inputs passed directly through to the Reducer from previous stages or iterations, keep and combine the results for each variable (auxiliary variables, residual, and state estimate) that is marked with the highest iteration number observed for that variable, and throw away earlier versions.

A. Updating the search direction and its projection

1) *Map (shared) context:*

   a. From initial selection: $\underline{\hat{r}}_0$

   b. From previous iteration:

   $$\rho_{n-1},\ \alpha_{n-1},\ \omega_{i-1},\ \underline{r}_{n-1},\ \underline{v}_{n-1},\ \underline{p}_{n-1}$$

   c. From pre-map computation:

   $$\rho_n = \left\langle \underline{\hat{r}}_0, \underline{r}_{n-1} \right\rangle$$

   $$\underline{p}_n = \underline{r}_{n-1} + \left(\tfrac{\rho_n}{\rho_{n-1}}\right)\left(\tfrac{\alpha_{n-1}}{\omega_{n-1}}\right)\left(\underline{p}_{n-1} - \omega_{n-1}\eta_{n-1}\right)$$

2) *Map computation:*

   For each row in $\underline{\underline{A}}$*, compute* $\left\{\underline{\eta}_n\right\}_i = \left\{\underline{\underline{A}}\right\}_{i*} \underline{p}_n$

B. Updating the shadow direction and its projection

1) *Map (shared) context:*

   a. From initial selection: $\underline{\hat{r}}_0$

   b. From previous iteration: $\underline{r}_{n-1}$

   c. From previous stage of current iteration:

   $$\rho_n,\ \underline{\eta}_n$$

   d. From pre-map computation:

   $$\alpha_n = \rho_n \Big/ \left\langle \underline{\hat{r}}_0, \underline{\eta}_n \right\rangle$$

   $$\underline{s}_n = \underline{r}_{n-1} - \alpha_n \eta_n$$

2) *Map computation:*

   For each row in $\underline{\underline{A}}$*, compute* $\left\{\underline{t}_n\right\}_i = \left\{\underline{\underline{A}}\right\}_{i*} \underline{s}_n$

C. Combining components for residual and state estimates

1) *Map (shared) context:*

   a. From previous iteration: $\underline{x}_{n-1}$

   b. From previous stages of current iteration:

   $$\alpha_n,\ \underline{s}_n,\ \underline{t}_n,\ \underline{p}_n$$

2) *Map computation: For each label, compute*

   $$\omega_n = \left\langle \underline{s}_n, \underline{t}_n \right\rangle \Big/ \left\langle \underline{t}_n, \underline{t}_n \right\rangle$$

   $$\underline{x}_n = \underline{x}_{n-1} + \alpha_n \underline{p}_n + \omega_n \underline{s}_n$$

   $$\underline{r}_n = \underline{s}_n - \omega_n t_n$$

V. MAPREDUCE FORMULATION OF GAUSSIAN ELIMINATION

Label slicing allows us to compute our distributions on the subset of labels that are of most interest, while still benefiting from the constraints effectively imposed by the full label set. In a similar way, Gaussian elimination allows us to compute our distribution on a subset of nodes (domains), while still benefiting from the indirect interconnections that are formed through the nodes that we do not want to explicitly include in our calculation. The computational savings provided by Gaussian elimination is linear with the percentage reduction in the number of graph connections. In addition, Gaussian elimination can speed up convergence, by effectively increasing the wavefront-propagation speed through those parts of the graph that were originally connected via the eliminated nodes.

Gaussian elimination is much simpler to implement in the MapReduce framework than BiCGStab, requiring only a single stage and capable of handling elimination of multiple nodes per run. The Reduce processing in the MapReduce is a straight pass-through of the outputs from the map stage.

To make the description more concise, define

$$\underline{\underline{A}}_{keep} = \left\{\underline{\underline{A}}\right\}_{i*} \quad \underline{\underline{L_\gamma}}_{keep} = \left\{\underline{\hat{\gamma}}\underline{\underline{L}}\right\}_{i*} \quad i \in \left\{ \begin{array}{c} nodes \\ to\ be\ kept \end{array} \right\}$$

$$\underline{\underline{A}}_{remove} = \left\{\underline{\underline{A}}\right\}_{j*} \quad \underline{\underline{L_\gamma}}_{remove} = \left\{\underline{\hat{\gamma}}\underline{\underline{L}}\right\}_{j*} \quad j \in \left\{ \begin{array}{c} nodes\ to\ be \\ eliminated \end{array} \right\}$$

Using this notation, the map processing is

1) *Map (shared) context:*

   From stored representation:

   $$\underline{\underline{A}}_{remove},\ \underline{\underline{L_\gamma}}_{remove}$$

2) *Map computation: For each row, i, in* $\underline{\underline{A}}_{keep}$ *and* $\underline{\underline{L_\gamma}}_{keep}$

   a) Initialize

   $$\underline{\underline{\tilde{A}}}_{keep} = \underline{\underline{A}}_{keep},\quad \underline{\underline{\tilde{A}}}_{remove} = \underline{\underline{A}}_{remove}$$

   $$\underline{\underline{\tilde{L}_\gamma}}_{keep} = \underline{\underline{L_\gamma}}_{keep},\quad \underline{\underline{\tilde{L}_\gamma}}_{remove} = \underline{\underline{L_\gamma}}_{remove}$$

   b) Compute the pivot strength, π_{ij}, for each $j \in \left\{nodes\ to\ be\ eliminated\right\}$:

   $$\pi_{ij} = \left\{\underline{\underline{\tilde{A}}}_{keep}\right\}_{ij} \Big/ \left\{\underline{\underline{\tilde{A}}}_{remove}\right\}_{jj}$$

and select the elimination node, $\tilde{j}$, with the smallest amplitude $\left|\pi_{ij}\right|$

   c) Eliminate all non-zero entries in the $\tilde{j}$'th column in $\left\{\underline{\underline{\tilde{A}}}_{keep}\right\}_{i*}$ and $\underline{\underline{\tilde{A}}}_{remove}$, with matched operations on $\left\{\underline{\underline{\tilde{L}_\gamma}}_{keep}\right\}_{i*}$ and $\underline{\underline{\tilde{L}_\gamma}}_{remove}$:

$$\left\{\underline{\underline{\tilde{A}}}_{keep}\right\}_{ik} \leftarrow \left\{\underline{\underline{\tilde{A}}}_{keep}\right\}_{ik} - \pi_{i\tilde{j}} \left\{\underline{\underline{\tilde{A}}}_{remove}\right\}_{\tilde{j}k}$$

$$\left\{\underline{\underline{\tilde{L}_\gamma}}_{keep}\right\}_{ik} \leftarrow \left\{\underline{\underline{\tilde{L}_\gamma}}_{keep}\right\}_{ik} - \pi_{i\tilde{j}} \left\{\underline{\underline{\tilde{L}_\gamma}}_{remove}\right\}_{\tilde{j}k}$$

$$\left\{\underline{\underline{\tilde{A}}}_{remove}\right\}_{nk} \leftarrow \left\{\underline{\underline{\tilde{A}}}_{remove}\right\}_{nk} - \tilde{\pi}_{n\tilde{j}} \left\{\underline{\underline{\tilde{A}}}_{remove}\right\}_{\tilde{j}k} \quad \forall n \neq \tilde{j}$$

$$\left\{\underline{\underline{\tilde{L}_\gamma}}_{remove}\right\}_{nk} \leftarrow \left\{\underline{\underline{\tilde{L}_\gamma}}_{remove}\right\}_{nk} - \tilde{\pi}_{n\tilde{j}} \left\{\underline{\underline{\tilde{L}_\gamma}}_{remove}\right\}_{\tilde{j}k} \quad \forall n \neq \tilde{j}$$

$$\text{with } \tilde{\pi}_{nj} = \left\{\underline{\underline{\tilde{A}}}_{remove}\right\}_{nj} \Big/ \left\{\underline{\underline{\tilde{A}}}_{remove}\right\}_{jj}$$

   d) Remove row $\tilde{j}$ from $\underline{\underline{\tilde{A}}}_{remove}$, $\underline{\underline{\tilde{L}_\gamma}}_{remove}$

   e) Repeat (b), (c), and (d), until there are no more rows (nodes) to be removed.

   f) Output $\left\{\underline{\underline{\tilde{A}}}_{keep}\right\}_{i*}$ and $\left\{\underline{\underline{\tilde{L}_\gamma}}_{keep}\right\}_{i*}$

VI. Large-scale Domain-Level Topic Labeling

Baluja et al. [2] already showed the usefulness of the Adsorption approach in video recommendations. The pre-normalized Adsorption algorithm [1] provides identical results at a fraction of the computational cost using the new formulation with label slicing, Gaussian elimination, and BiCGStab. The final computational cost is reduced by the product of the savings of all three approaches (label slicing, BiCGStab and Gaussian elimination).

In our previous paper [1], we explored using pre-normalized Adsorption for topic labeling on web domains, for search and advertising. Many pages URLs, and even whole domains, are poorly classified by standard topic-analysis approaches, due to having little in the way of machine-understandable content to classify. A standard example of this problem are domains that primarily host images or video – while the page URL can be examined for clues to the topic, as well as the linked-to URLs, the results are impoverished and noisy. If we can improve the topic labeling, we could more accurately index these pages for search and for content-matched advertisement.

Specifically, we created a graph with domains as nodes and a measure of shared searches for cross-domain pairs of URLs as the weighted connections between nodes. Our measure looked at, for each search term, the click rates for each URL served in the results and set the strength of the URL-URL-term triple to the lower of the click rates between the paired URLs. The connection weight between pairs of URLs is the sum over all triples that terminate at those two URLs. To aggregate from URL-pair connections, up to domain-pair connections, we sum across those URL-pair connections where the first of the pair of URLs is from the first domain and the second is from the second domain. Similarly, our injection labeling is based on combining topic analysis of the URLs within the domain, dropping those topics that were based on keywords that showed too much within-domain variance in their strength. We aggregate the link and topic-label strength up to the domain level to improve coverage and reliability of our graph connections. Even with this aggregation of URLs to domain-level nodes and filtering of keyword labels to within-domain-stable sets, our initial data provides a graph of about 13 million domains (nodes), with about 4 billion node-to-node connections based on analysis of more than 253 million search terms. Our topic analysis provides more than 4,500 general topics, using traditional text-based classification.

- Clothing
  - Women's, Men's, Children's
  - Athletic, Casual, Formal, Outerwear, Sleepwear
  - Shoes, Boots
- Accessories
  - Jewelry, Watches, Purses
- Toys
  - Building Toys, Dolls, Stuffed Animals, Ride-on Toys
- Gifts
  - Flowers, Cards, Party Items, Holiday Items
- Discounts
  - Coupons, Loyalty Cards

Figure 1. Examples from selected 71 commercial topics.

From this set of 4,500 topics, we focused on 71 commercial topics (see Figure 1 for examples). The computational savings (over the original Adsorption approach) for the label slicing alone was a factor of 63 times. We do not include this savings in the remainder of this discussion, since it is available to both power iteration and BiCGStab, as long as we are using the pre-normalized Adsorption formulation. That said, it is the most significant source of computational savings, compared to the original work [2].

We ran this set of 71 labels through two iterations of BiCGStab (5 graph-matrix multiplies) and through 70 iterations of the power method, both starting from the same initial estimate. Figure 2 shows the size of the per-node residual for BiCGStab on these labels (using an L_1 norm). As with our small-scale simulations, at the end of our second iteration, the not-insignificant residuals occurred at the 3% of the nodes that were at the "wavefront boundary" of one or more of the topic labels. This level of convergence, with just 5 matrix multiplies, is not seen in the power-iteration solution until the 62th iteration (an additional savings of nearly 12.5 times).

Since the goal of our label propagation is to increase the richness and extent of the topic labeling on poorly labeled (or unlabeled) domains without over-extending into domains that are not related to our commercial subset, it is helpful to look at the statistics summarized in Figures 3 through 5.

Figure 3 gives a measure of the richness of our labels on commercial domains and how that richness increases as a function of iteration. The plot shows the percentages of domains by how many commercial-topic labels are seen on that domain. If a domain is commercial, the more commercial labels that are associated with the domain, the richer the topic description. As shown by the plots, our

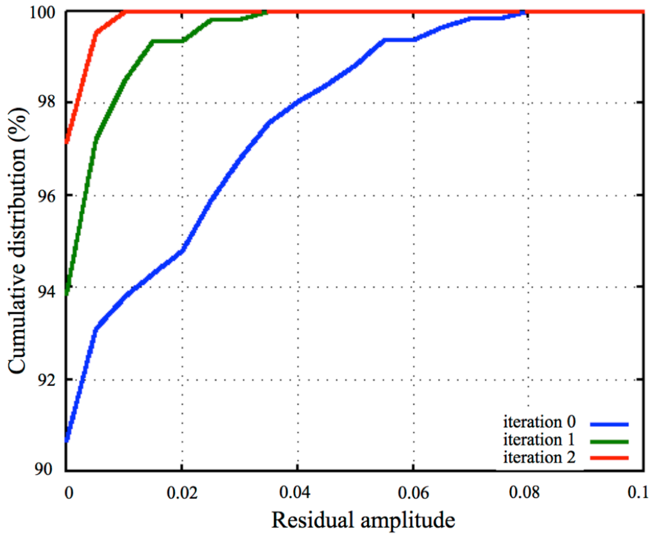

Figure 2. Cumulative residual distribution (by iteration). [1]

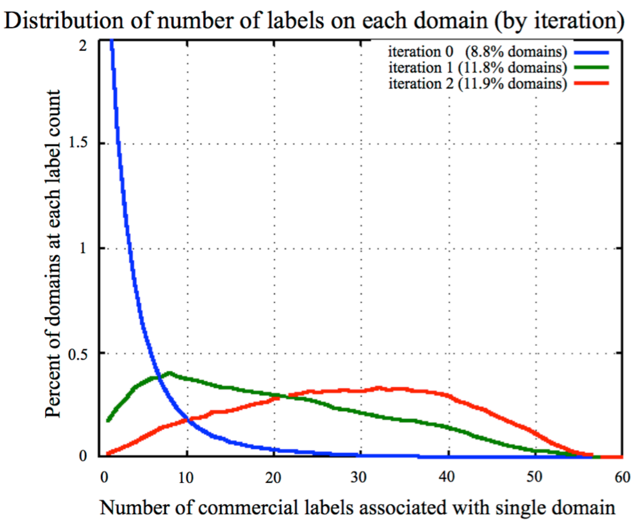

Figure 3. Node-level coherence of commercial labels. [1]

injection labels (those given by topic analysis) within each domain provides sparse topic labels, with the largest percentage of commercial domains having only one label. Since our 71 commercial topics are actually a hierarchical set, this sparseness is unlikely to be correct for most domains. By the end of the second iteration, the mode of that distribution has moved to around 30 topic labels per commercial domain.

Also, the legend in Figure 3 gives us the information needed to check that we are not just expanding the support of our commercial-topic labels indiscriminately across the full domain graph. The first iteration extends the support of the commercial labels by a third, from just under 9% of all domains to just under 12%, suggesting the addition of a subset of the unlabeled domains within the graph. After the first iteration, the support of the commercial-label set is effectively unchanged. This can be traced back to the effect of pre-normalizing on the full set of topic labels. Even though the non-commercial topics are not being explicitly computed in our iterations, they still have an effect, keeping the commercial labels from spreading onto distant (in the graph-connection sense) domains, as they otherwise would as the commercial wavefront progressed. This highlights both one of the main advantages of the original Adsorption as well as the most compelling advantage of the pre-normalized Adsorption. With the original Adsorption, each node has a limited capacity for supporting labels, thereby limiting propagation – but enforcing that limited capacity forced computation of all label distributions, not just the labels of interest. With pre-normalized Adsorption, there is still the per-node limited capacity for supporting labels, but we achieve that capacity limit by pre-normalizing, freeing us to compute only at that subset of labels that we are interested in, without having those labels spread unchecked.

Up to now, our analysis of our results has focused on the richness and extent of our commercial labels but not on the likely quality of the mix of labels that we are introducing onto commercial nodes. Since our topics are structured into a hierarchical framework, intuitively what we would like is to have each commercial site labeled mostly by closely related subsets of the available topics. We can use *dendrite distances* between the labels to capture this sense of closeness among the sets of labels associated with each domain node. As with standard dendrite measures, for each pair of labels on a domain, we count the number of hierarchical topic links that we have to go across in order to travel from one topic label to the other. We lengthen that distance by one for each generation that *both* labels have to travel back through, in order to penalize siblings more than grandparent-grandchild relations. As an example, if we need to calculate the distance between women's jewelry and men's clothing and we have the two tree branches "Jewelry → Women's Accessories → Apparel" and "Men's Clothing → Apparel", our dendrite distance measure would be 4: two (for "Women's Jewelry" to "Apparel") plus one (for "Men's Clothing" to "Apparel") plus one (for the one generation removal from direct descendent connection).

As a way to evaluate our label distributions on domains with 2 to 6 labels, we computed all pairwise dendrite distances within each domain and averaged them (again, on a per-domain basis). Due to the use of the topic hierarchy in our dendrite-distance measure, smaller distances amongst the labels on a single domain correspond to more believable topic mixes. Figure 4 shows our results, as function of iteration. When the initial topic labeling provides more than one label, it includes many dissimilar labels, with the mode of the dendrite average distance being up between 6 and 7. Our propagation reduces that average distance, filling in parent and children nodes, to give a mode that is just above one. While parents could always be filled in by knowing the hierarchical structure of our topic labels, the propagation graph is doing this without that knowledge – it is finding these associations purely through propagation of neighbor

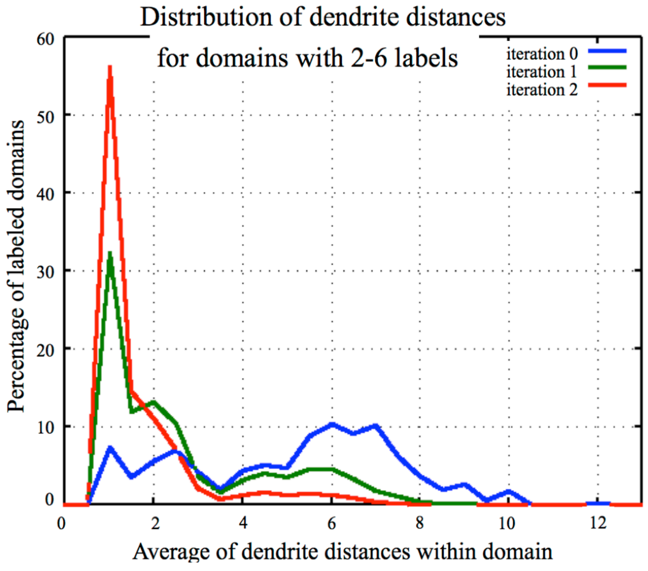

Figure 4. Dendrite topic-label distance on domains with 2-6 labels (by iteration). [1]

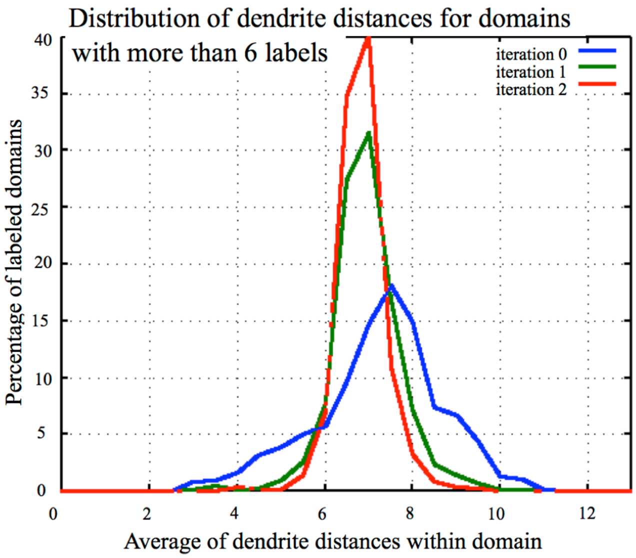

Figure 5. Dendrite topic-label distance on domains with more than 6 labels (by iteration). [1]

labels. (Furthermore, we could not use the tree-structure meta-information to fill in the correct children labels – if we blindly used the tree structure, we would get numerous nearby but irrelevant labels.) For this set of nodes, we are enriching the topic description without introducing unrelated labels. This measure of quality is a stringent one, since at no point do we use the dendrite structure to limit our propagation.

Figure 5 shows a similar measure, for domains with more than 6 labels, again averaging the dendrite distances within each node. We did this separation between Figure 4, for domains with 2-6 commercial labels, and Figure 5, for domains with more than 6 commercial labels, since the dendrite distances across larger sets of labels, taken from the same hierarchy will have a larger minimum-average distance than will smaller sets of labels. For small sets, you can often find 2-6 labels, with all parent-child or sibling relationships with one another but, for large sets of labels, this is not possible and first and second cousin relationships become a major part of even the most compact set of labels. Same as with Figure 4, Figure 5 shows that the average dendrite distance decreases with each iteration, even on nodes with more than 6 labels. Since closely related sets of topic labels are more likely to be a full and accurate description of the domain topic, our topic labeling seems to be improved by our graph propagation work.

All of the measurements conducted on the propagation of web labels on this large set of domains indicate an improvement in search indexing and content-matched advertising. In the future, we will expand these experiments in two directions. First, we will run live trials, with full user-facing experiments, to determine the quality improvement in the user experience. Second, we will increase our graph size and specificity by including individual URLs, for those sites that have enough textual information to support that level of analysis.

VII. Efficient Updating on Dynamic Graphs

In nearly every application of graph-based label propagation, the graph changes over time: in video recommendation, new videos are added and old videos fade in popularity; in social networks, new users join, new friendships are made, and old friendships are ended; and, in topic labeling, the connection strengths between domains change as their content is updated. These changes occur gradually and most of the label distributions within the newly changed graph are only mildly perturbed from those labels that were computed for the original matrix, making it more efficient to do incremental updates than to restart the labeling process from scratch. The largest changes are associated with newly added nodes and labels and with the nodes that connect to either those sets. We focus on the changes to the graph and the labels to create an efficient update process.

Since we are now considering a change in the graph and label distribution, which will necessitate breaking the matrices into pieces, we first define a more compact notation for our pre-normalized adsorption state equation. Instead of using Equation (13), we will use

$$\underline{\underline{X}} = \underline{\underline{\hat{W}}}\,\underline{\underline{X}} + \underline{\underline{\hat{L}}} \qquad (15)$$

where $\underline{\underline{\hat{W}}} = \underline{\underline{\hat{\sigma}}} + \underline{\underline{\hat{\beta}}}\,\underline{\underline{W}}$ and $\underline{\underline{\hat{L}}} = \left[\begin{array}{cc} \underline{\underline{\hat{\gamma}}}\,\underline{\underline{L}} & \underline{\underline{\hat{\delta}}} \end{array}\right]$. Equation (15) is identical to Equation (13), with the exception of the symbols that we use to describe it. This notation hides the iteration subscript that we previously associated with $\underline{\underline{X}}$, so that we will be able to use the subscript location for identifying sub-matrices.

To refer to the two related but distinct sets of graph and label weights, we will use a superscript of "-" or "+" to distinguish the pre-change and post-change versions of the graph, respectively. So:

$$\underline{\underline{X}}^- \approx \underline{\underline{\hat{W}}}^- \underline{\underline{X}}^- + \underline{\underline{\hat{L}}}^- \qquad (16)$$

is the pre-change version of the graph state equations, with $\underline{\underline{X}}^-$ as the inferred label distributions that we have already computed for the pre-change graph, and

$$\underline{\underline{X}}^+ = \underline{\underline{\hat{W}}}^+ \underline{\underline{X}}^+ + \underline{\underline{\hat{L}}}^+ \qquad (17)$$

is the post-change version of the graph state equations, with $\underline{\underline{X}}^+$ as the inferred label distributions that we need to compute for the post-change graph. We also define difference matrices:

$$\Delta\underline{\underline{Y}} = \underline{\underline{Y}}^+ - \underline{\underline{Y}}^- \qquad (18)$$

where $\underline{\underline{Y}}$ can be any of $\underline{\underline{X}}$, $\underline{\underline{\hat{W}}}$, or $\underline{\underline{\hat{L}}}$.

In order to allow us to use matrix operations across these two graph descriptions, we assume that we have added all-zero rows and columns as needed to the pre-change graph, to allow for the newly added nodes and labels that we need for the post-change description and that we have added all-zero rows and columns as needed to the post-change graph, to allow for the newly removed nodes and labels that were present in the pre-change description. We rearrange the rows and columns, so that we group these to all-zero rows and columns and use the notation

$$\underline{\underline{Y}} = \begin{bmatrix} \underline{\underline{Y}}_{||} & \underline{\underline{Y}}_{|-} & \underline{\underline{Y}}_{|+} \\ \underline{\underline{Y}}_{-|} & \underline{\underline{Y}}_{--} & \underline{\underline{0}} \\ \underline{\underline{Y}}_{+|} & \underline{\underline{0}} & \underline{\underline{Y}}_{++} \end{bmatrix} \tag{19}$$

where $\underline{\underline{Y}}$ can be any of $\underline{\underline{X}}$, $\underline{\underline{\hat{W}}}$, or $\underline{\underline{\hat{L}}}$ and $\underline{\underline{Y}}^-$ refers to the pre-change versions of these matrices and $\underline{\underline{Y}}^+$ refers to the post-change versions. The subscripts "|", "-", and "+" distinguish between nodes and labels according to their need to be part of the pre- and post-change graphs. The first of the subscript pair refers to the row characteristics and the second refers to the column characteristics. The "-" subscript is for rows or columns that are only needed for the pre-change matrices (i.e., they are identically zero for the post-change matrices). The "+" subscript is for rows or columns that are only needed for the post-change matrices (i.e., they are identically zero for the pre-change matrices). The "|" subscript is for rows or columns that are needed for both pre- and post-change matrices. Notice that we can assert that the sub-matrices that would have represented interactions between "-" and "+" rows and columns are known to be identically zero, since these two sets of nodes and labels do not occur (non-trivially) in the same matrices. Also, using this grouping, we know that $\underline{\underline{Y}}^-_{|+} = \underline{\underline{Y}}^-_{++} = \underline{\underline{Y}}^-_{+|} \equiv \underline{\underline{0}}$ and $\underline{\underline{Y}}^+_{|-} = \underline{\underline{Y}}^+_{--} = \underline{\underline{Y}}^+_{-|} \equiv \underline{\underline{0}}$. Finally, we will use the same matrix partitioning for the difference matrices, $\Delta\underline{\underline{Y}}$, as we have described above for the pre- and post-change matrices.

Our goal is to find a good estimate of $\underline{\underline{X}}^+$ from $\underline{\underline{X}}^-$, with as few computations as possible. Starting from the post-change equation and recasting it in terms of the pre-change matrices and the difference matrices:

$$\underline{\underline{X}}^+ = \underline{\underline{\hat{W}}}^+ \underline{\underline{X}}^+ + \underline{\underline{\hat{L}}}^+ \tag{20}$$

$$\underline{\underline{X}}^- + \Delta\underline{\underline{X}} = \underline{\underline{\hat{W}}}^+ \left(\underline{\underline{X}}^- + \Delta\underline{\underline{X}}\right) + \underline{\underline{\hat{L}}}^- + \Delta\underline{\underline{\hat{L}}} \tag{21}$$

$$\underline{\underline{X}}^- + \Delta\underline{\underline{X}} = \left(\underline{\underline{\hat{W}}}^- + \Delta\underline{\underline{\hat{W}}}\right)\underline{\underline{X}}^- + \underline{\underline{\hat{W}}}^+ \Delta\underline{\underline{X}} + \underline{\underline{\hat{L}}}^- + \Delta\underline{\underline{\hat{L}}} \tag{22}$$

Rearranging Equation (22):

$$\Delta\underline{\underline{X}} = \underline{\underline{\hat{W}}}^+ \Delta\underline{\underline{X}} + \Delta\underline{\underline{\hat{L}}} + \Delta\underline{\underline{\hat{W}}}\,\underline{\underline{X}}^- + \left(\underline{\underline{\hat{W}}}\,\underline{\underline{X}}^- + \underline{\underline{\hat{L}}}^- - \underline{\underline{X}}^-\right) \tag{23}$$

Since we have a good estimate of $\underline{\underline{X}}^-$ from the pre-change description, we use $\underline{\underline{X}}^- \approx \underline{\underline{\hat{W}}}^- \underline{\underline{X}}^- + \underline{\underline{\hat{L}}}^-$ to remove the final term from Equation (23), giving

$$\Delta\underline{\underline{X}} = \underline{\underline{\hat{W}}}^+ \Delta\underline{\underline{X}} + \left(\Delta\underline{\underline{\hat{L}}} + \Delta\underline{\underline{\hat{W}}}\,\underline{\underline{X}}^-\right) \tag{24}$$

Finally, we define

$$\Delta\underline{\underline{X}}^0 = \Delta\underline{\underline{\hat{L}}} + \Delta\underline{\underline{\hat{W}}}\,\underline{\underline{X}}^- \tag{25}$$

to get:

$$\Delta\underline{\underline{X}} = \underline{\underline{\hat{W}}}^+ \Delta\underline{\underline{X}} + \Delta\underline{\underline{X}}^0 \tag{26}$$

There are several things of note about Equations (25) and (26). From Equation (25), $\Delta\underline{\underline{X}}^0$ is exactly the estimate for $\Delta\underline{\underline{X}}$, if our previous estimate was $\underline{\underline{0}}$. It is also used in later iterations, as a persistent input, so explicitly saving it reduces the computation needed on later iterations. Finally, $\Delta\underline{\underline{X}}^0$ is much sparser than $\underline{\underline{\hat{L}}}$, which will be useful in our discussion of Equation (26).

Equation (26) is an update equation, similar to Equation (15). The reason that Equation (26) is preferred over Equation (15) is (as just noted) $\Delta\underline{\underline{X}}^0$ is much sparser than $\underline{\underline{\hat{L}}}$ and that $\Delta\underline{\underline{X}}$ is much sparser than $\underline{\underline{\hat{X}}}$, even after several iterations. This sparseness reduces the amount of computation needed per iteration.

We can further improve the efficiency and compactness of our computation by not computing values for the nodes that are not needed for the post-change description. We can now use our matrix partitioning to remove these extra entries. We can also use our knowledge that $\underline{\underline{Y}}^-_{|+} = \underline{\underline{Y}}^-_{++} = \underline{\underline{Y}}^-_{+|} \equiv \underline{\underline{0}}$ and $\underline{\underline{Y}}^+_{|-} = \underline{\underline{Y}}^+_{--} = \underline{\underline{Y}}^+_{-|} \equiv \underline{\underline{0}}$ to simplify the formula. When we use those zero identities along with the sub-matrix notation in Equation (26), we get:

$$\begin{bmatrix} \Delta\underline{\underline{X}}_{||} & -\underline{\underline{X}}^-_{|-} & \underline{\underline{X}}^+_{|+} \\ -\underline{\underline{X}}^-_{-|} & -\underline{\underline{X}}^-_{--} & \underline{\underline{0}} \\ \underline{\underline{X}}^+_{|+} & \underline{\underline{0}} & \underline{\underline{X}}^+_{++} \end{bmatrix} = \begin{bmatrix} \Delta\underline{\underline{\hat{W}}}_{||} & -\underline{\underline{\hat{W}}}^-_{|-} & \underline{\underline{\hat{W}}}^+_{|+} \\ -\underline{\underline{\hat{W}}}^-_{-|} & -\underline{\underline{\hat{W}}}^-_{--} & \underline{\underline{0}} \\ \underline{\underline{\hat{W}}}^+_{|+} & \underline{\underline{0}} & \underline{\underline{\hat{W}}}^+_{++} \end{bmatrix} \begin{bmatrix} \Delta\underline{\underline{X}}_{||} & -\underline{\underline{X}}^-_{|-} & \underline{\underline{X}}^+_{|+} \\ -\underline{\underline{X}}^-_{-|} & -\underline{\underline{X}}^-_{--} & \underline{\underline{0}} \\ \underline{\underline{X}}^+_{|+} & \underline{\underline{0}} & \underline{\underline{X}}^+_{++} \end{bmatrix}$$
$$+ \begin{bmatrix} \Delta\underline{\underline{\hat{L}}}_{||} & -\underline{\underline{\hat{L}}}^-_{|-} & \underline{\underline{\hat{L}}}^+_{|+} \\ -\underline{\underline{\hat{L}}}^-_{-|} & -\underline{\underline{\hat{L}}}^-_{--} & \underline{\underline{0}} \\ \underline{\underline{\hat{L}}}^+_{|+} & \underline{\underline{0}} & \underline{\underline{\hat{L}}}^+_{++} \end{bmatrix} + \begin{bmatrix} \Delta\underline{\underline{\hat{W}}}_{||} & -\underline{\underline{\hat{W}}}^-_{|-} & \underline{\underline{\hat{W}}}^+_{|+} \\ -\underline{\underline{\hat{W}}}^-_{-|} & -\underline{\underline{\hat{W}}}^-_{--} & \underline{\underline{0}} \\ \underline{\underline{\hat{W}}}^+_{|+} & \underline{\underline{0}} & \underline{\underline{\hat{W}}}^+_{++} \end{bmatrix} \begin{bmatrix} \underline{\underline{X}}^-_{||} & \underline{\underline{X}}^-_{|-} & \underline{\underline{0}} \\ \underline{\underline{X}}^-_{-|} & \underline{\underline{X}}^-_{--} & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{0}} & \underline{\underline{0}} \end{bmatrix} \tag{27}$$

We can reduce Equation (27) down to the four submatrix update equations that constrain label distributions in the post-change network ($\Delta\underline{\underline{X}}_{\vert\vert}$, $\underline{\underline{X}}^{+}_{\vert+}$, $\underline{\underline{X}}^{+}_{+\vert}$, and $\underline{\underline{X}}^{+}_{++}$). Focusing on those four update equations:

$$\Delta\underline{\underline{X}}_{\vert\vert} = \underline{\underline{\hat{W}}}^{+}_{\vert\vert}\Delta\underline{\underline{X}}_{\vert\vert} + \underline{\underline{\hat{W}}}^{+}_{\vert+}\underline{\underline{X}}^{+}_{+\vert} + \Delta\underline{\underline{\hat{L}}}_{\vert\vert} + \Delta\underline{\underline{\hat{W}}}_{\vert\vert}\underline{\underline{X}}^{-}_{\vert\vert} - \underline{\underline{\hat{W}}}^{-}_{\vert-}\underline{\underline{X}}^{-}_{-\vert} \tag{28}$$

$$\underline{\underline{X}}^{+}_{\vert+} = \underline{\underline{\hat{W}}}^{+}_{\vert\vert}\underline{\underline{X}}^{+}_{\vert+} + \underline{\underline{\hat{W}}}^{+}_{\vert+}\underline{\underline{X}}^{+}_{++} + \underline{\underline{\hat{L}}}^{+}_{\vert+} \tag{29}$$

$$\underline{\underline{X}}^{+}_{+\vert} = \underline{\underline{\hat{W}}}^{+}_{+\vert}\Delta\underline{\underline{X}}_{\vert\vert} + \underline{\underline{\hat{W}}}^{+}_{++}\underline{\underline{X}}^{+}_{+\vert} + \underline{\underline{\hat{L}}}^{+}_{+\vert} + \underline{\underline{\hat{W}}}^{-}_{+\vert}\underline{\underline{X}}^{-}_{\vert\vert} \tag{30}$$

$$\underline{\underline{X}}^{+}_{++} = \underline{\underline{\hat{W}}}^{+}_{+\vert}\underline{\underline{X}}^{+}_{\vert+} + \underline{\underline{\hat{W}}}^{+}_{++}\underline{\underline{X}}^{+}_{++} + \underline{\underline{\hat{L}}}^{+}_{++} \tag{31}$$

Reformatting Equations (28) through (31) back into a single partitioned matrix gives:

$$\begin{bmatrix} \Delta\underline{\underline{X}}_{\vert\vert} & \underline{\underline{X}}^{+}_{\vert+} \\ \underline{\underline{X}}^{+}_{+\vert} & \underline{\underline{X}}^{+}_{++} \end{bmatrix} = \begin{bmatrix} \underline{\underline{\hat{W}}}^{+}_{\vert\vert} & \underline{\underline{\hat{W}}}^{+}_{\vert+} \\ \underline{\underline{\hat{W}}}^{+}_{+\vert} & \underline{\underline{\hat{W}}}^{+}_{++} \end{bmatrix} \begin{bmatrix} \Delta\underline{\underline{X}}_{\vert\vert} & \underline{\underline{X}}^{+}_{\vert+} \\ \underline{\underline{X}}^{+}_{+\vert} & \underline{\underline{X}}^{+}_{++} \end{bmatrix} + \begin{bmatrix} \Delta\underline{\underline{X}}^{0}_{\vert\vert} & \underline{\underline{\hat{L}}}^{+}_{\vert+} \\ \Delta\underline{\underline{X}}^{0}_{+\vert} & \underline{\underline{\hat{L}}}^{+}_{++} \end{bmatrix} \tag{32}$$

where

$$\Delta\underline{\underline{X}}^{0}_{\vert\vert} = \Delta\underline{\underline{\hat{L}}}_{\vert\vert} + \Delta\underline{\underline{\hat{W}}}_{\vert\vert}\underline{\underline{X}}^{-}_{\vert\vert} - \underline{\underline{\hat{W}}}^{-}_{\vert-}\underline{\underline{X}}^{-}_{-\vert} \tag{33}$$

$$\Delta\underline{\underline{X}}^{0}_{+\vert} = \underline{\underline{\hat{L}}}^{+}_{+\vert} + \underline{\underline{\hat{W}}}^{-}_{+\vert}\underline{\underline{X}}^{-}_{\vert\vert} \tag{34}$$

Using Equations (32) through (34) allows us to find an update to the inferred label matrices, starting from the inferred labels for the pre-change graph, even when there are nodes and labels that have been newly added or completely deleted. This approach saves computation on each iteration, since the inferred-label change matrix will be non-zero on a much smaller number of nodes and labels than the full inferred-label matrices are. Further savings can be had by computing and caching the initial update matrices described in Equations (33) and (34) for use in later iterations. We stop iterating on the inferred-label change matrix when the per-entry residuals are similar in size to residuals that we ignored in using $\underline{\underline{X}}^{-} \approx \underline{\underline{\hat{W}}}^{-}\underline{\underline{X}}^{-} + \underline{\underline{\hat{L}}}^{-}$ or when the inferred-label change matrix is no more sparse than the full post-change inferred-label matrix. At that point, the post-change inferred label matrix should be reconstructed, using the values of $\underline{\underline{X}}^{+}_{\vert+}$, $\underline{\underline{X}}^{+}_{++}$, and $\underline{\underline{X}}^{+}_{+\vert}$ as given by Equation (32) and using $\underline{\underline{X}}^{+}_{\vert\vert} = \underline{\underline{X}}^{-}_{\vert\vert} + \Delta\underline{\underline{X}}_{\vert\vert}$ for the last non-trivial submatrix of $\underline{\underline{X}}^{+}$.

The convergence of Equation (32) is guaranteed only indirectly. Equation (32) is formed as the difference of two state equations (one for the pre-change graph and one for the post-change graph). Both of those two state equations, having eigenvalues that are strictly inside the unit circle, are guaranteed to converge. The difference between them will therefore converge.

VIII. Incremental Updating of Co-Author Network Inferences

To demonstrate the use of the incremental update of label distributions on changing graphs, we used condensed-matter collaboration data, posted at [11] from work done by Newman [12]. This co-author network data was first collected from physics pre-print publications for 1995 through 1999 but was twice updated, first to contain co-author connections from 1995 to 2003 and later to extend that time frame to 2005. The 1995-2003 network contains 31,163 authors (nodes) while the 1995-2005 network has 40,421 authors. Based on exact matching of names, all except 57 of the authors from the 2003 network could be uniquely matched to the authors in the 2005 network. These 57 authors had ambiguous matches (e.g., there were 3 "PARK, S" author nodes in both the 2003 and 2005 networks). This original pair of networks had 30% new authors added (and 0.2% dropped, due to ambiguity) between 2003 and 2005, in addition to having changes in the connection weights between the authors that were in both networks. To create the co-author graph, Newman [12] scanned the Los Alamos e-Print Archive on condensed-matter physics for the years in question. For each paper in that database that had *n* authors, for *n* > 1, he added (or strengthened) a connection between each pair of co-authors by a weight of 1/(*n* – 1). In this way, the connections made from each co-author to other researchers is increased by one, for each paper that is a collaborative effort. Newman's research [12] describes the core characteristics of this network: the mean (collaborative) papers-per-author in this field is 3.87 (with a standard deviation of 5); the mean number of authors per paper is 2.66 (with a standard deviation of 1); and the mean number of collaborators for each author is 5.86 (with a standard deviation of 9).

This type of co-author graph can be used to recommend new collaborations to each author in the network, based on propagating the names of potential collaborators. To this end, we use parts of these co-author weighted graphs as our un-normalization node-to-node matrix, $\underline{\underline{W}}$ as used in Equation (1). We created an un-normalized label matrix, $\underline{\underline{L}}$ as used in Equation (1), from the author names, using as an (un-normalized) injection weight the number of papers on which that author collaborated. Using the number of papers as this label weight will result in the prolific authors' names being recommended as potential collaborators more widely and strongly than less prolific authors. To complete the un-normalized constraint equation, we somewhat arbitrarily set $\beta = \gamma = 1$ and $\sigma = \delta = 2$ in Equation (1).

The addition of 30% new authors between 1995-2003 and 1995-2005 is large enough that the incremental update approach would provide little, if any, computational

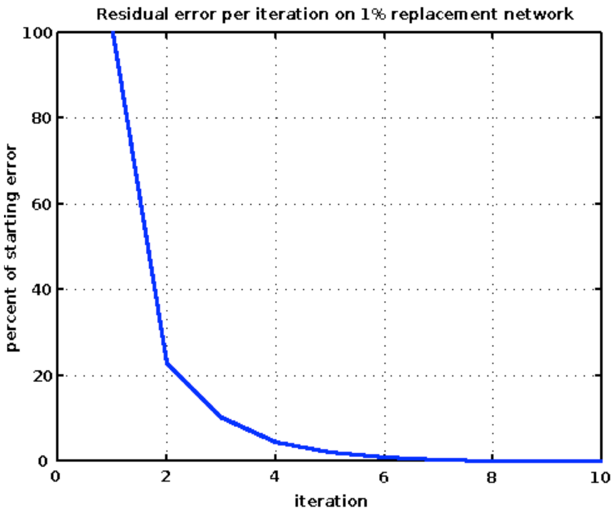

Figure 6. Residual error on 1% replacement network, starting from the label distribution from the pre-change graph, as a function of iteration (using the power method of solution)

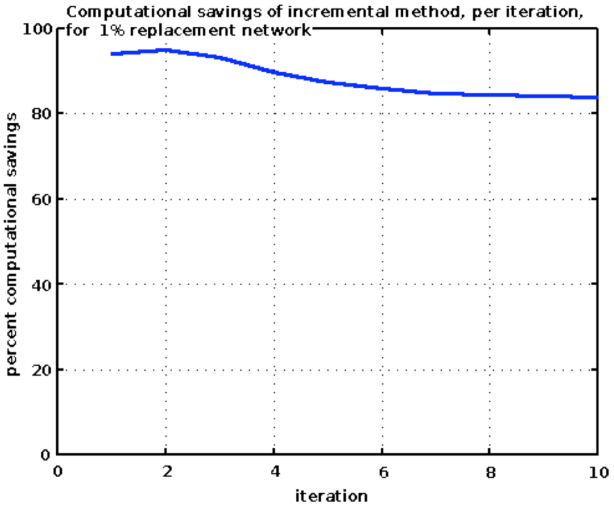

Figure 7. Computational savings on 1% replacement network using incremental updating compared to using the same starting estimate with the full post-change network.

savings: with this much change, $\Delta\underline{\underline{X}}^0$ is almost as dense as $\underline{\underline{\hat{L}}}$ and, due to the authorship fan-out becomes nearly as dense as the full label matrix, $\underline{\underline{\hat{X}}}$, by the second iteration. To concretely demonstrate the potential for large computational savings in incrementally changing networks, we employed a subset of the 1995-2003 and 1995-2005 data. We first reduced the size of both of the 2003 and the 2005 networks down to the same 27,519 authors, ones with unambiguous matches who occurred with a reasonable weight and connectivity in both data sets. From this shared set, we picked equal numbers of distinct nodes to drop from each of the reduced-2003 and reduced-2005 networks, so that both reduced networks retained equal numbers of nodes (authors) by picking the least well-connected nodes and dropping them from one of the two networks. For our “1% replacement experiment”, we did this with 1% of the nodes, so that both the pre- and the post-change graphs had 27,244 nodes with 275 of the nodes that are in the reduced-2003 graph being dropped and replaced with a distinct set of 275 nodes for the reduced-2005 graph. The connection weights between the 99% of the nodes (26,969 nodes) that appeared in both of these reduced graphs changed in whatever way that was indicated by the original 2003 or 2005 data from [11]. In a similar manner, we created our “17% replacement experiment”, removing two distinct sets of 3,997 nodes from the reduced-2003 and -2005 networks to create two graphs with 23,522 nodes each, 17% of which are present in only one of the two graphs. As before, the connection weights for the nodes that were shared between the graphs was allowed to change, according to the original 2003 or 2005 data from [11].

Once these two pairs of un-normalized networks (pre- and post-change networks for the 1% and 17% replacement experiments) were formed, we separately normalized the matrices for each of the four networks, as described by Equations (8) to (13). These normalizations are based on the entries that are actually in each network: there is no leakage from pre-change networks into the normalization of the post-change network (nor vice-versa) and there is no leakage from anything done in the 1% replacement networks to the 17% replacement networks (nor vice-versa).

The node-to-node connection occupancies were about 0.04% on both the pre- and post-change graphs ($\underline{\underline{\hat{W}}}^-$ and $\underline{\underline{\hat{W}}}^+$) for both the 1% and 17% replacement experiments. This is nearly twice the collaboration rate reported by Newman [12], in part due to the longer time period covered (8 and 10 years, in contrast with 5 years) and in part due to the selection bias for how we created the reduced 2003 and 2005 networks. In contrast to the occupancy of $\underline{\underline{\hat{W}}}^-$ and $\underline{\underline{\hat{W}}}^+$, the occupancy of $\Delta\underline{\underline{\hat{W}}}$ was about 50% of that level (so, 0.02% occupancy) for the 17% replacement experiment and about 5% of that level (so, 0.002% occupancy) for the 1% replacement. These occupancy levels are higher than expected by a factor of 3-5 times, due to the changes in the weights between $\underline{\underline{\hat{W}}}_{\|}^-$ and $\underline{\underline{\hat{W}}}_{\|}^+$. These weights change, even though neither of the connected nodes are added or removed, since the connection (un-normalized) weights are taken from the 2003 and the 2005 co-author data, respectively, as well as indirect effects from changing normalization on rows that do connect with new or removed nodes.

In both experiments, we start from the pre-change label distributions, $\underline{\underline{X}}^-$. We computed these label distributions using power-method iterations for 20 iterations. This brought the per-entry residual errors on the label

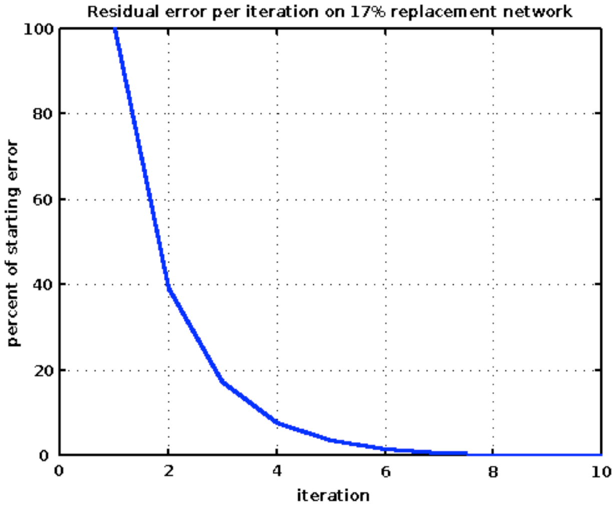

Figure 8. Residual error on 17% replacement network, starting from the label distribution from the pre-change graph, as a function of iteration (using the power method of solution)

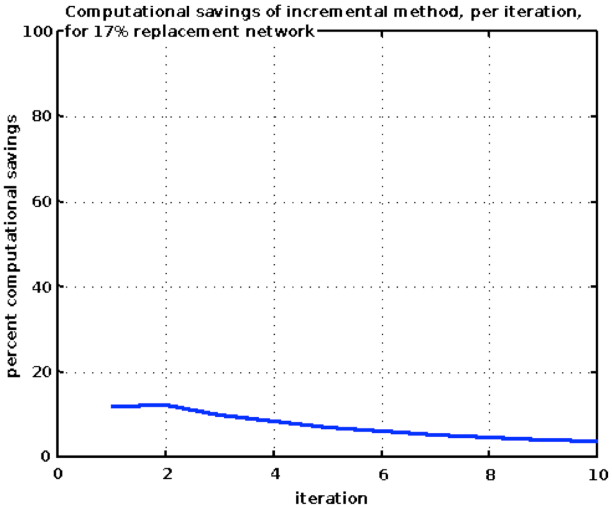

Figure 9. Computational savings on 17% replacement network using incremental updating compared to using the same starting estimate with the full post-change network.

distributions (as measured by $\left|\underline{\underline{X}}_{n+1} - \underline{\underline{X}}_{n}\right|_{ij}$) down to below 10^{-8} for all entries. The result had non-trivial entries on about 70% of the pre-change label distribution matrix, for both 1% and 17% replacement experiments.

Figures 6 and 7 show our results for our 1% replacement experiment. Figures 8 and 9 show our results for our 17% replacement experiment. In both Figures 6 and 8, we show the reduction in the residual label error, starting from the label distribution computed for the pre-change graph (the reduced-2003 graph), when applied to the post-change graph. For these graphs, we use the power-iteration method (instead of bi-conjugate gradient descent). Using power iteration and starting from a well-converged solution for the pre-change graph, the residual errors are exactly the same, whether we use the incremental or non-incremental update methods. We can also see that, for these graph pairs, the residual error is rapidly reduced, falling to well under 1% of the initial error within 8 iterations. The rate at which that reduction happens is slower for the 17% replacement experiment than for the 1% replacement but only by a factor of less than two, in terms of the remaining residual (relative to the starting residual).

Figures 7 and 9 show the computational savings, derived from using the incremental update equations. For the incremental-update method, we use Equation (32), starting with $\Delta\underline{\underline{X}} = \underline{\underline{0}}$ and counting the number of multiplies needed to determine $\Delta\underline{\underline{X}}^{0}$ as part of that first iteration. For the full-update method, we use Equation (20), using $\underline{\underline{X}}^{-}$ as the starting estimate for $\underline{\underline{X}}^{+}$. The computational savings of using the incremental-update approach is very large for the 1% replacement experiment, with as much as 95% of the multiplies needed for Equation (20) avoided by Equation (32), with the same error rates. Even after 10 iterations, the savings is above 84% of the multiplies that would be needed for Equation (20). The savings of the incremental approach is much more modest when 17% of the nodes have been replaced (as well as other edges changing weight). Under those conditions, the savings are 12-13%, for early iterations but falls to only 4% savings (per iteration) by the 10'th iteration. The difference between 1% and 17% replacement, in terms of the levels of savings from incremental updating, can be traced back to the non-zero occupancy rates in $\Delta\underline{\underline{X}}^{0}$. For the 1% replacement experiment, the occupancy of $\Delta\underline{\underline{X}}^{0}$ was 2%; for the 17% replacement experiment, it was 25%. Both are less dense than the 70% occupancy of the pre-change label distribution ($\underline{\underline{X}}^{-}$) but the initial cost is higher than this difference in sparseness would suggest. Part of that reduction in savings is due to the computation of $\Delta\underline{\underline{X}}^{0}$ itself. Furthermore, for the 17% replacement experiment, the initial difference in sparseness is greatly reduced by even the third iteration, due to the co-authorship fan out in $\hat{\underline{\underline{W}}}^{+}$.

The specific savings and convergence rates will depend on the specific network configurations and changes that are being used. However, these co-author network examples are somewhat representative of the types of networks that exist for many problems, in that there are only sparse interconnections (having a fan-out level of only 0.04% of what is possible) and having multiple cliques. Fortunately, in large-scale real-world usage scenarios (e.g., YouTube and social networks like G+) updates are computed daily or even more frequently. As a result, the amount of change that will be encountered in the network between update cycles, compared to the overall size of the network, is small. The real-world computational savings of this procedure will be enormous in practice, allowing responsive targeting to occur using these propagation approaches.

IX. Negative Associations Across Nodes And Between Nodes And Labels

To this point, our derivation has relied on having only non-negative values in our node-to-node connections and in our injection label weights. By restricting our matrices to non-negative values and by pre-normalizing as described in Equations (8) through (12), we are guaranteed to maintain the same unit L$_1$ norm for all iterations, for all rows of $\underline{\underline{X}}_n$.

In most large graph-based problems, there is no difficulty with restricting the description in this way. In the earlier mentioned example of video-to-video recommendation, we can say that two videos are often watched together, within a single viewing session, but it is much more difficult to say that two videos are negatively associated (that watching one means you are significantly less likely to watch the other); rarely is there enough training data to confidently ascertain disassociations.

Nonetheless, there are several cases in which negative information can be useful. First consider the use of social networks in movie and music recommendations and reviews. Often, users express dislike for particular songs or movies. In this case, though the node-to-node connections (user-to-user) remain positive, the label (the movie/song/artist) may now have a negative connection weight to some nodes. We need to be able to handle and propagate these negative reviews in the same way as positive reviews, as both can shape public opinion. The simplest way to do this is to separately represent positive-bias and negative-bias labels, even when they refer to the same underlying label: in our example we would double the number of labels that we used, with one for "likes" and one for "dislikes" for each movie/song. Representing this using the notation for Equation (15) we would have:

$$\left[\begin{array}{cc} \underline{\underline{X}}^p & \underline{\underline{X}}^n \end{array}\right] = \underline{\underline{\hat{W}}} \left[\begin{array}{cc} \underline{\underline{X}}^p & \underline{\underline{X}}^n \end{array}\right] + \left[\begin{array}{cc} \underline{\underline{\hat{L}}}^p & \underline{\underline{\hat{L}}}^n \end{array}\right] \qquad (35)$$

where $\underline{\underline{\hat{L}}}^p = \max\left(\underline{\underline{\hat{L}}}, \underline{\underline{0}}\right)$ are positive injection labels (declared "likes") and $\underline{\underline{\hat{L}}}^n = \max\left(-\underline{\underline{\hat{L}}}, \underline{\underline{0}}\right)$ are negative injection labels (declared "dislikes"). For Equation (35), we are still requiring that all entries in $\underline{\underline{\hat{W}}}$ be non-negative. The result is that the inferred label weights are split, with positive-association weights in $\underline{\underline{X}}^p$ and negative-association weights in $\underline{\underline{X}}^n$.

As long as $\underline{\underline{X}}^p$ and $\underline{\underline{X}}^n$ are represented separately, the L$_1$ row norm of $\left[\begin{array}{cc} \underline{\underline{X}}^p & \underline{\underline{X}}^n \end{array}\right]$ will remain one, throughout our iterative estimation, without renormalizing. This approach will have the effect that the amount of 'attention' paid to controversial movies/songs will be higher, in terms of the portion of the available L$_1$ unit length, than it would be if we did not split positive and negative labels into separate entries. For some applications, this separation of opposite extremes may be the right thing to do. For example, it may be that separately listing strong positive and negative recommendations for a controversial movie, and thereby having that movie feature prominently in terms amount of attention it is given, is better than having the recommendations for that movie "wash out" to neutral, by not exposing the controversy in opinions.

An alternative example would be to use a social-network of voters in conjunction with political-opinion labels to help guide a politician's stance on legislation. The social network could capture regional differences within the politician's constituency. Since the politician needs to be seen as serving all represented regions equally, understating the key issues of a region that has a divided stand on some controversial issue does not serve the politician well. The politician is probably better served by emphasizing the issues with political consensus, for those regions, rather than effectively ignoring all the opinions of the region by understating what they do agree upon. As such, this situation may be better handled by the re-combining and re-normalizing approach we outline later in this section.

In the previous examples, we explored the use of negative values associated with a node's labels. Now, we consider the case where there are negative connection weights between nodes. Consider the case of financial-fund analysis. If we are trying to find closely related (as well as nearly opposite) investment opportunities, we could create a graph with one node for each fund under study and with the node-to-node connection being set by the statistically significant market-adjusted price-change correlations. The labels would then be the fund symbols themselves, optionally with long and short positions represented in $\underline{\underline{X}}^p$ and $\underline{\underline{X}}^n$, respectively. In this framework, there are many combinations of funds or instruments that would show negative connection weights between them. A fairly simple example would be the connection from either a purchase-to-open put contract or a sell-to-open call contract to the underlying security. Both option contracts are clearly distinct in their valuation from the underlying security (with the time-value changes having the largest exogenous impact) but both have price changes that are strongly (negatively) correlated with the security's price changes. A less direct example of negative connections between funds would by an ultra-short fund on a market sector and any of the largest firms in that sector (for example, QID and AAPL).

To handle negative node-to-node connections, we use a similar slicing-and-doubling approach as we used for negative label weights. Specifically, we can say

$$\begin{bmatrix} \underline{\underline{X}}^p \\ \underline{\underline{X}}^n \end{bmatrix} = \begin{bmatrix} \underline{\underline{\hat{W}}}^p & \underline{\underline{\hat{W}}}^n \\ \underline{\underline{\hat{W}}}^n & \underline{\underline{\hat{W}}}^p \end{bmatrix} \begin{bmatrix} \underline{\underline{X}}^p \\ \underline{\underline{X}}^n \end{bmatrix} + \begin{bmatrix} \underline{\underline{\hat{L}}}^p \\ \underline{\underline{\hat{L}}}^n \end{bmatrix} \quad (36)$$

where $\underline{\underline{\hat{W}}}^p = \max\left(\underline{\underline{\hat{W}}}, \underline{\underline{0}}\right)$ and $\underline{\underline{\hat{W}}}^n = \max\left(-\underline{\underline{\hat{W}}}, \underline{\underline{0}}\right)$. Note that Equation (36) has two "partial" rows per node, one (with positive associations) in $\underline{\underline{X}}^p$ and the other (with negative associations) in $\underline{\underline{X}}^n$. It is the concatenation of these two parts, $\begin{bmatrix} \underline{\underline{X}}^p & \underline{\underline{X}}^n \end{bmatrix}$, that will maintain a unit L1 row norm, while the energy split between $\underline{\underline{X}}^p$ and $\underline{\underline{X}}^n$ in each row can vary widely, from one iteration to another. Also note that Equation (35) is a re-arranged and simplified version of Equation (36), with $\underline{\underline{\hat{W}}}^n = \underline{\underline{0}}$.

In the movie/music recommendation example, we suggested that keeping track of diverse opinions about the same movie/song made sense (since controversial movies/music are fundamentally different than ones that do not evoke strong opinions either way). In contrast, in this financial example, keeping positive and negative label entries separately (corresponding to holding long and short positions of the same security) may not be the correct approach. Similarly, in the politician's example, it may be better to refocus the rhetoric on non-controversial issues, instead of tracking the degree of controversy.

Therefore, we investigate the effects of recombining and reweighting positive and negative inferred labels, $\underline{\underline{X}}^p$ and $\underline{\underline{X}}^n$. When we recombine $\underline{\underline{X}}^p$ and $\underline{\underline{X}}^n$, by taking $\underline{\underline{X}} = \underline{\underline{X}}^p - \underline{\underline{X}}^n$, we will end up with a row norm less than one, in all rows where one or more non-zero entries of $\underline{\underline{X}}^p$ and $\underline{\underline{X}}^n$ overlap. The total *reduction* in the i^{th} row norm will be

$$L_i^{loss} = 2\sum_j \min\left(\underline{\underline{X}}_{ij}^p, \underline{\underline{X}}_{ij}^n\right) \quad (37)$$

To avoid under-emphasizing the combined information from rows with some conflicting labels, we then need to renormalize, to bring the row norm back up to one. We can do this selectively on only those rows where L_i^{loss} exceeds some pre-defined threshold. When that happens, the label vector for that i^{th} row should be replaced with

$$\underline{\underline{X}}_{ij}^p = \max\left(0, \frac{\underline{\underline{X}}_{ij}^p - \underline{\underline{X}}_{ij}^n}{1 - L_i^{loss}}\right) \quad (38)$$

and

$$\underline{\underline{X}}_{ij}^n = \max\left(0, \frac{\underline{\underline{X}}_{ij}^n - \underline{\underline{X}}_{ij}^p}{1 - L_i^{loss}}\right) \quad (39)$$

This need for re-normalization, unfortunately, means that we can no longer use label slicing, since we need to be able to track the row norm for this normalization process. It also complicates the use of stabilized bi-conjugate gradient descent, since that approach introduces interdependences between update iterations that are not easily adjusted for changing scale. When we use this re-normalizing method, we need to be aware of the increases in the computational costs that result. However, this method does allow us to handle situations that respond best to "fair and equal" representations of each node within the network, throughout the computation, even in the presence of conflicting labeling. The cost/benefit trade-off must be carefully considered in the context of each application.

X. Conclusions

This paper improves the computational efficiency of Adsorption, a graph-based labeling approach that has already been shown to be highly effective. We do so by replacing propagation-interleaved normalization with pre-normalization, without changing the results provided by Adsorption. Specifically, if the power-method approach to finding a solution is used, as it was with Adsorption, the answers at every iteration will be exactly the same using either the original or the pre-normalized Adsorption. The advantage of the pre-normalized Adsorption is computational efficiency in determining the label distribution. With the pre-normalized version, we can use label slicing, to compute only those labels that are of direct interest, without losing the beneficial belief-limiting characteristics of the full label set. Label slicing reduces the computational cost linearly with the percentage of dropped labels. Similarly, we can use Gaussian elimination, to compute the labels only on those nodes that are of direct interest, without losing the effects of the connections that occur indirectly through currently not-of-interest nodes. Finally, we can speed up convergence to the steady-state solution by a factor of 12 (in numbers of graph matrix multiples), by using stabilized biconjugate gradient descent, instead of power iteration. We tested the pre-normalized Adsorption in a new, large-scale application area, topic labeling on web domains, with promising results.

Additionally, we explored extensions to address two real-world scenarios, in which network propagation will play an important role: (1) networks with both positive and negative connections as well as positive and negative associations with labels and (2) gradually changing networks in which nodes are added and removed (as well as having weight changes between existing nodes). Examples of changing networks include searching for, recommending, and advertising against image, audio, and video content. These labeling problems must handle millions of interconnected entities (users, domains, content segments) and thousands of competing labels (interests, tags, recommendations, topics). By using a label update matrix (instead of the full label distribution matrix) in our update equations, we were able to converge to the correct label distribution on the changed network in the same number of iterations as we would need for the full network but with only one tenth of the computation (for a network that changed by 1% node replacement). This savings drops rapidly as the percentage replacement increased but was still significant, even at 17% node replacement. We demonstrated the incremental update using co-author networks. We note that, in real-world cases in which rapid and continual updated of large (10^7 – 10^9 node) networks is required, the methods proposed in this paper will make propagation methods feasible.

References

[1] M. Covell and S. Baluja, "Efficient and Accurate Label Propagation on Large Graphs and Label Sets," *Proceedings of the International Conferences on Advances in Multimedia*, IARIA, April 2013, pp. 12-18.

[2] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly, "Video suggestion and discovery for YouTube: taking random walks through the view graph," *Proceedings of the International Conference on World Wide Web*, ACM, April 2008, pp. 895-904.

[3] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," CMU technical report, CMU-CALD-02-107, 2002.

[4] P.P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira, "Weakly-supervised acquisition of labeled class instances using graph random walks," *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* Association of Computational Linguistics, October 2008, pp. 582-590.

[5] Y. Jing and S. Baluja, "Visual Rank: applying Page Rank to large-scale image search," *Transactions on Pattern Analysis and Machine Intelligence,* IEEE, vol. 30, November 2008, pp. 1877-1890.

[6] J. Liu, W. Lai, X S. Hua, Y. Huang, and S. Li, "Video search re-ranking via multi-graph propagation," *Proceedings of the International Conference on Multimedia,* ACM, September 2007, pp. 208-217.

[7] M. Speriosu, N. Sudan, S. Upadhyay, and J. Baldridge, "Twitter polarity classification with label propagation over lexical links and the follower graph," *Proceedings of the Workshop on Unsupervised Learning in Natural Language Processing,* Association of Computational Linguistics, July 2011, pp. 53-63.

[8] H. A. Van der Vorst, "Bi-CGSTAB: A Fast and Smoothly Converging Variant of BiCG for the Solution of Nonsymmetric Linear Systems," *Journal on Scientive and Statistical Computing,* SIAM, vol. 13, March 1992, pp. 631-644.

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Proceeds of the Symposium on Operating Systems Design and Implementation*, USENIX, December 2004, pp. 137-150.

[10] Wikipedia, "Biconjugate Gradient Stablized Method," http://en.wikipedia.org/wiki/Biconjugate_gradient_stabilized_method [retrieved February, 2013].

[11] M. E. J. Newman, "Condensed Matter Collaborations 2003" and "Condensed Matter Collaborations 2005," http://www-personal.umich.edu/~mejn/netdata [retrieved September, 2013].

[12] M. E. J. Newman, "Structure of Scientific Collaboration Networks," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 98, January 2001, pp. 404-409.