LR-PPR: Locality-Sensitive, Re-use Promoting, Approximate Personalized PageRank Computation [*]

Jung Hyun Kim
Arizona State University
Tempe, AZ 85287, USA
jkim294@asu.edu

K. Selçuk Candan
Arizona State University
Tempe, AZ 85287, USA
candan@asu.edu

Maria Luisa Sapino
University of Torino
I-10149 Torino, Italy
mlsapino@di.unito.it

ABSTRACT

Personalized PageRank (PPR) based measures of node proximity have been shown to be highly effective in many prediction and recommendation applications. The use of personalized PageRank for large graphs, however, is difficult due to its high computation cost. In this paper, we propose a *Locality-sensitive*, *Re-use promoting*, *approximate personalized PageRank* (LR-PPR) algorithm for efficiently computing the PPR values relying on the *localities* of the given seed nodes on the graph: (a) The LR-PPR algorithm is *locality sensitive* in the sense that it reduces the computational cost of the PPR computation process by focusing on the local neighborhoods of the seed nodes. (b) LR-PPR is *re-use promoting* in that instead of performing a *monolithic* computation for the given seed node set using the entire graph, LR-PPR divides the work into localities of the seeds and *caches* the intermediary results obtained during the computation. These cached results are then *reused* for future queries sharing seed nodes. Experiment results for different data sets and under different scenarios show that LR-PPR algorithm is highly-efficient and accurate.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Miscellaneous

Keywords

Personalized PageRank; Locality-Sensitivity; Reuse-Promotion

1. INTRODUCTION

Node distance/proximity measures are commonly used for quantifying how *nearby* or otherwise *related to* two or more nodes on a graph are. Path-length based definitions [17] are useful when the *relatedness* can be captured solely based on the properties of the nodes and edges on the *shortest* path (based on some definition of path-length). Random-walk based definitions, such as hitting distance [16] and personalized page rank (PPR) score [4, 13, 21] of node relatedness, on the other hand, also take into account the density of the edges: unlike in path-based definitions, random walk-based definitions of relatedness also consider how tightly connected two nodes are and argue that nodes that have many paths between them can be considered more related. Random-walk based techniques encode the structure of the network in the form a transition matrix of a stochastic process from which the node relationships can be inferred. When it exists, the convergence probability of a node n gives the ratio of the time spent at that node in a sufficiently long random walk and, therefore, neatly captures the connectivity of the node n in the graph. Therefore, many web search and recommendation algorithms, such as PageRank [5], rely on random-walks to identify significant nodes in the graph: let us consider a weighted, directed graph $G(V, E)$, where the weight of the edge $e_j \in E$ is denoted as $w_j (\geq 0)$ and where $\left(\sum_{e_j \in outedge(v_i)} w_j\right) = 1.0$. The PageRank score of the node $v_i \in V$ is the stationary distribution of a random walk on G, where at each step with probability $1-\beta$, the random walk moves along an outgoing edge of the current node with a probability proportional to the edge weights and with probability β, the walk jumps to a random node in V. In other words, if we denote all the PageRank scores of the nodes in V with a vector $\vec{\pi}$, then

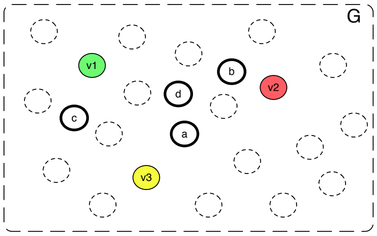


Figure 1: Key questions: Given a graph, G, and a seed set of nodes $S = \{v_1, v_2, v_3\}$ in G, can we rank the remaining nodes in the graph regarding their relationships to the set S? Which of the nodes a through d is the most interesting given the seed set of nodes v_1 through v_3?

$$\vec{\pi} = (1-\beta)\mathbf{T}_G \times \vec{\pi} + \beta\vec{j},$$

where $\mathbf{T}_G$ denotes the transition matrix corresponding to the graph G (and the underlying edge weights) and $\vec{j}$ is a *teleportation* vector where all entries are $\frac{1}{\|V\|}$.

1.1 Proximity and PageRank

An early attempt to contextualize the PageRank scores is the *topic sensitive PageRank* [12] approach which adjusts the PageRank scores of the nodes by assigning the *teleportation* probabilities in vector $\vec{j}$ in a way that reflects the graph nodes' degrees of

[*] This work is supported by NSF Grant 1043583 "MiNC: NSDL Middleware for Network- and Context-aware Recommendations".

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2505651.

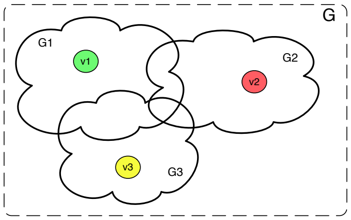
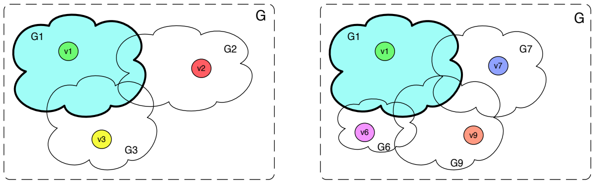


Figure 2: Locality-sensitivity: Computation of PPR should focus on the neighborhoods (localities) of the seeds



(a) PPR query 1

(b) PPR query 2

Figure 3: Re-use promotion: Two PPR queries sharing a seed node (v_1) should also share relevant work

match to the search topic. [6, 7] were among the first works which recognized that random-walks can also be used for measuring the degree of *association*, *relatedness*, or *proximity* of the graph nodes to a given *seed node set*, $S \subseteq V$ (Figure 1). An alternative to this approach is to modify (as in topic sensitive PageRank [12]) the teleportation vector, $\vec{j}$: instead of jumping to a random node in V with probability β, the random walk jumps to one of the nodes in the seed set, S, given by the user. More specifically, if we denote the personalized PageRank scores of the nodes in V with a vector $\vec{\phi}$, then

$$\vec{\phi} = (1-\beta)\mathbf{T}_G \times \vec{\phi} + \beta\vec{s},$$

where $\vec{s}$ is a re-seeding vector, such that if $v_i \in S$, then $\vec{s}[i] = \frac{1}{\|S\|}$ and $\vec{s}[i] = 0$, otherwise. One key advantage of this approach over modifying the transition matrix as in [6] is that the term β can be used to directly control the *degree of seeding (or personalization)* of the PPR score. However, the use of personalized PageRank for large graphs is difficult due to the high cost of solving for the vector $\vec{\phi}$, given β, transition matrix $\mathbf{T}_G$, and the seeding vector $\vec{s}$. One way to obtain $\vec{\phi}$ is to solve the above equation for $\vec{\phi}$ mathematically. Alternatively, PowerIteration methods [14] simulate the dissemination of probability mass by repeatedly applying the transition process to an initial distribution $\vec{\phi}_0$ until a convergence criterion is satisfied. For large data sets, both of these processes are prohibitively expensive. Recent advances on personalized PageRank includes top-k and approximate personalized PageRank algorithms [1, 3, 8, 10, 11, 20, 22] and parallelized implementations on MapReduce or Pregel based batch data processing systems [2, 15]. The FastRWR algorithm presented in [22] for example partitions the graph into subgraphs and indexes partial intermediary solutions. Unfortunately, for large data sets, FastRWR requires large number of partitions to ensure that the intermediary metadata (which requires dense matrix representation) fits into the available memory and this negatively impacts execution time and accuracy.

1.2 Contributions of this Paper

In this paper, we argue that we can improve *both* scalability and accuracy through a *Locality-sensitive*, *Re-use promoting*, *approximate personalized PageRank* (LR-PPR) algorithm: LR-PPR is

- *locality sensitive* in the sense that it reduces the computational cost of the PPR computation process and improve accuracy by focusing on the neighborhoods of the seed nodes (Figure 2); and
- *re-use promoting* in that it enables *caching* and *re-use* of significant portions of the intermediary work for the individual seed nodes in future queries (Figure 3).

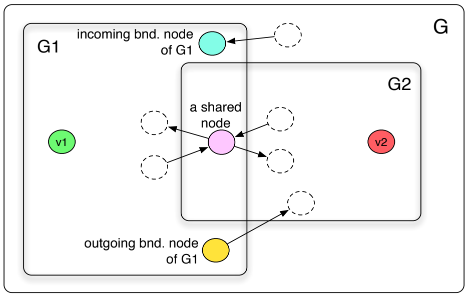


Figure 4: Incoming and outgoing boundary nodes/edges and a node shared between two localities

In the following section, we first formally introduce the problem and then present our solution for *locality-sensitive*, *re-use promoting*, *approximate personalized PageRank* computations. We evaluate LR-PPR for different data sets and under different scenarios in Section 3. We conclude in Section 4.

2. PROPOSED APPROACH

Let $G = (V, E)$ be a directed graph. For the simplicity of the discussion, without any loss of generality, let us assume that G is unweighted[1]. Let us be given a set $S \subseteq V$ of seed nodes (Figure 1) and a personalization parameter, β. Let $\mathcal{G}_S = \{G_h(V_h, E_h) \mid 1 \leq h \leq K\}$ be $K = \|S\|$ subgraphs of G, such that

- for each $v_i \in S$, there exists a corresponding $G_i \in \mathcal{G}_S$ such that $v_i \in V_i$ and
- for all $G_h \in \mathcal{G}_S$, $\|G_h\| \ll \|G\|$.

We first formalize the locality-sensitivity goal (Figure 2):

Desideratum 1: Locality-Sensitivity. Our goal is to compute an approximate PPR vector, $\vec{\phi}_{apx}$, using $\mathcal{G}_S$ instead of G, such that $\vec{\phi}_{apx} \sim \vec{\phi}$, where $\vec{\phi}$ represents the true PPR scores of the nodes in V relative to S: i.e.,

$$\vec{\phi}_{apx} \sim \vec{\phi} = (1-\beta)\mathbf{T}_G \times \vec{\phi} + \beta\vec{s},$$

where $\mathbf{T}_G$ is the transition matrix corresponding to G and $\vec{s}$ is the re-seeding vector corresponding to the seed nodes in S.

We next formalize the re-use promotion goal (Figure 3):
Desideratum 2: Reuse-Promotion. Let S_1 and S_2 be two sets of seed nodes and let v_i be a node such that $v_i \in S_1 \cap S_2$. Let also the approximate PPR vector, $\vec{\phi}_{apx,1}$ corresponding to S_1 have already been computed using $\mathcal{G}_{S_1}$ and let us assume that the approximate PPR vector, $\vec{\phi}_{apx,2}$ corresponding to S_2 is being requested. The part of the work performed when processing $G_i \in \mathcal{G}_{S_1}$ (corresponding to v_i) should not need to be re-performed when processing $G_i \in \mathcal{G}_{S_2}$, when computing $\vec{\phi}_{apx,2}$ using $\mathcal{G}_{S_2}$.

2.1 Combined Locality and its Boundary

Unlike existing approximate PPR algorithms [1, 3, 8, 10, 11, 20, 22], LR-PPR is *location sensitive*. Therefore, given the set, S, of

[1] Extending the proposed algorithms to weighted graphs is trivial.

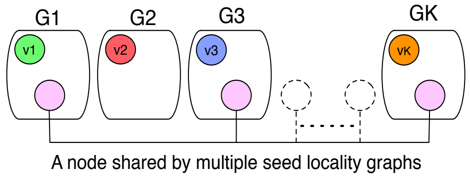


Figure 5: An equivalence set consists of the copies of a node shared across multiple seed locality graphs

seed nodes and the corresponding localities, $\mathcal{G}_S$, the computation focuses on the *combined locality* $G^+(V^+, E^+) \subseteq G$, where

$$V^+ = \bigcup_{1 \le l \le K} V_l \text{ and } E^+ = \bigcup_{1 \le l \le K} E_l.$$

Given a combined locality, G^+, we can also define its *external graph*, $G^-(V^-, E^-)$, as the set of nodes and edges of G that are outside of G^+ and boundary nodes and edges. As shown in Figure 4, we refer to $v_i \in V_l$ as an *outgoing boundary node* of G_l if there is an outgoing edge $e_{i,j} = [v_i \to v_j] \in E$, where $v_j \notin V_l$; the edge e_j is also referred to as an *outgoing boundary edge* of G_l. The set of all outgoing boundary nodes of G_l is denoted as $V_{outbound,l}$ and the set of all outgoing boundary edges of G_l is denoted as $E_{outbound,l}$. Note that $V_{outbound,l} \subseteq V_l$, whereas $E_{outbound,l} \cap E_l = \emptyset$.

We also define *incoming boundary nodes* ($V_{inbound,l}$) and *incoming boundary edges* ($E_{inbound,l}$) similarly to the outgoing boundary nodes and edges of G_l, but considering inbound edges to these subgraphs. More specifically, $E_{inbound,l}$ consists of edges of the form $[v_i \to v_j] \in E$, where $v_j \in V_l$ and $v_i \notin V_l$.

2.2 Localized Transition Matrix

Since LR-PPR focuses on the combined locality, G^+, the next step is to combine the transition matrices of the individual localities into a combined transition matrix. To produce accurate approximations, this *localized transition matrix*, however, should nevertheless take the external graph, G^-, and the boundaries between G^- and G^+, into account.

2.2.1 *Transition Matrices of Individual Localities*

Let $v_{(l,i)}$ $(1 \le l \le K)$ denote a re-indexing of vertices in V_l. If $v_{(l,i)} \in V_l$ and $v_c \in V$ s.t. $v_{(l,i)} = v_c$, we say that $v_{(l,i)}$ is a member of an *equivalence set*, $\mathcal{V}_c$ (Figure 5). Intuitively, the equivalence sets capture the common parts across the localities of the individual seed nodes. Given $G_l(V_l, E_l) \subseteq G$ and an appropriate re-indexing, we define the corresponding local transition matrix, $\mathbf{M}_l$, as a $\|V_l\| \times \|V_l\|$ matrix, where

- $\left(\nexists e_{i,j} = [v_{(l,i)} \to v_{(l,j)}] \in E_l\right) \to \mathbf{M}_l[j,i] = 0$ and
- $\left(\exists e_{i,j} = [v_{(l,i)} \to v_{(l,j)}] \in E_l\right) \to \mathbf{M}_l[j,i] = \frac{1}{out(v_{(l,i)})}$, where $out(v_{(l,i)})$ is the number of outgoing edges of v_i.

2.2.2 *Localization of the Transition Matrix*

Given the local transition matrices, $\mathbf{M}_1$ through $\mathbf{M}_K$, we *localize* the transition matrix of G by approximating it as

$$\mathbf{M}_{apx} = \mathbf{M}_{bd} + \mathbf{M}_0,$$

where $\mathbf{M}_{bd}$ is a block-diagonal matrix of the form

$$\begin{pmatrix} \mathbf{M}_1 & \mathbf{0}_{\|V_1\| \times \|V_2\|} & \dots & \mathbf{0}_{\|V_1\| \times \|V_K\|} & \mathbf{0}_{\|V_1\| \times 1} \\ \mathbf{0}_{\|V_2\| \times \|V_1\|} & \mathbf{M}_2 & \dots & \mathbf{0}_{\|V_2\| \times \|V_K\|} & \mathbf{0}_{\|V_2\| \times 1} \\ \dots & \dots & \dots & \dots & \dots \\ \mathbf{0}_{\|V_K\| \times \|V_1\|} & \mathbf{0}_{\|V_K\| \times \|V_2\|} & \dots & \mathbf{M}_K & \mathbf{0}_{\|V_K\| \times 1} \\ \mathbf{0}_{1 \times \|V_1\|} & \mathbf{0}_{1 \times \|V_2\|} & \dots & \mathbf{0}_{1 \times \|V_K\|} & \mathbf{M}_{K+1} \end{pmatrix},$$

where $\mathbf{M}_{K+1}$ is equal to the 1×1 matrix $\mathbf{0}_{1\times 1}$. Intuitively, $\mathbf{M}_{bd}$ combines the K subgraphs into one transition matrix, without considering common nodes/edges or incoming/outgoing boundary edges and ignoring all outgoing and incoming edges. All the external nodes in G^- are accounted by a single node represented by the 1×1 matrix $\mathbf{M}_{K+1}$.

A key advantage of $\mathbf{M}_{bd}$ is that it is block-diagonal and, hence, there are efficient ways to process it. However, this block-diagonal matrix, $\mathbf{M}_{bd}$, cannot accurately represent the graph G as it ignores potential overlaps among the individual localities and ignores all the nodes and edges outside of G^+. We therefore need a *compensation matrix* to

- make sure that nodes and edges shared between the localities are not double counted during PPR computation and
- take into account the topology of the graph external to both localities G_1 through G_K.

2.2.3 *Compensation Matrix,* $\mathbf{M}_0$

Let t be $(\|V_1\| + \|V_2\| + \ldots + \|V_K\| + 1)$. The compensation matrix, $\mathbf{M}_0$, is a $t \times t$ matrix accounting for the boundary edges of the seed localities as well as the nodes/edges in G^-. $\mathbf{M}_0$ also ensures that the common nodes in V_1 through V_K are not double counted during PPR calculations. $\mathbf{M}_0$ is constructed as follows:

Row/column indexing: Let $v_{l,i}$ be a vertex in V_l. We introduce a row/column indexing function, $ind()$, defined as follows:

$$ind(l,i) = \left(\sum_{1 \le h < l} \|V_h\|\right) + i$$

Intuitively the indexing function, $ind()$, maps the relevant nodes in the graph to their positions in the $\mathbf{M}_0$ matrix.

Compensation for the common nodes: Let $e_{l,i,j}$ be an edge $[v_{(l,i)} \to v_{(l,j)}] \in E_l$ and let $v_{(l,j)}$ be a member of the equivalence set $\mathcal{V}_c$ for some $v_c \in V$. Then, if $\|\mathcal{V}_c\| > 1$

- $\mathbf{M}_0[ind(l,j), ind(l,i)] = -\frac{\|\mathcal{V}_c\|-1}{\|\mathcal{V}_c\|} \times \frac{1}{out(G, v_{l,i})}$ and
- $\forall v_{(h,k)} \in \mathcal{V}_c$ s.t. $v_{(h,k)} \neq v_{(l,j)}$, we have

$$\mathbf{M}_0[ind(h,k), ind(l,i)] = -\frac{1}{\|\mathcal{V}_c\|} \times \frac{1}{out(G, v_{l,i})},$$

where $out(G, v)$ is the outdegree of node v in G. Intuitively, the compensation matrix re-routes a portion of the transitions going towards a shared node in a given locality V_l to the copies in other seed localities. This prevents the transitions to and from the shared node from being mis-counted.

Compensation for outgoing boundary edges: The compensation matrix needs to account also for outgoing boundary edges that are not accounted for by the neighborhood transition matrices $\mathbf{M}_1$ through $\mathbf{M}_K$:

- *Accounting for boundary edges from nodes in* V_l *to nodes in* V_h: $\forall [v_{(l,i)} \to v_{(h,j)}] \in E_{outbound,l}$
  - $\mathbf{M}_0[ind(h,j), ind(l,i)] = \frac{1}{out(v_{(l,i)})}$
- *Accounting for boundary edges from nodes in* V_l *to graph nodes that are in* V^-:

  if $\exists [v_{(l,i)} \to v] \in E_{outbound,l}$ s.t. $v \in V^-$

- $\mathbf{M}_0[t, ind(l,i)] = \frac{bnd(v_{(l,i)})}{out(v_{(l,i)})}$, where $bnd(v_{(l,i)})$ is the number of edges of the form $[v_{(l,i)} \rightarrow v] \in E_{outbound,l}$ where $v \in V^-$

else $\mathbf{M}_0[t, ind(l,i)] = 0$

The compensation matrix records all outgoing edges, whether they cross into another locality or they are into external nodes in G^-. If a node has more than one outgoing edge into the nodes in G^-, all such edges are captured using one single compensation edge which aggregates all the corresponding transition probabilities.

Compensation for incoming boundary edges (from G^-): Similarly to the outgoing boundary edges, the compensation matrix needs also to account for incoming boundary edges that are not accounted for by the neighborhood transition matrices $\mathbf{M}_1$ through $\mathbf{M}_K$. Since incoming edges from other localities have been accounted for in the previous step, here we only need to consider incoming boundary edges (from G^-). Following the formulation in [23], we account for incoming edges where the source is external to G^+ and the destination is a vertex $v_{(l,i)}$ in V_l by inserting an edge from the dummy node to $v_{(l,i)}$ with a weight that considers the outdegrees of all external source nodes; i.e., $\forall v_{(l,i)}$ s.t. $\exists [v_k \rightarrow v_{(l,i)}] \in E_{inbound,l}$ where $v_k \in V^-$ and $v_{(l,i)}$ is in the equivalence set $\mathcal{V}_c$ for a $v_c \in V$, $\mathbf{M}_0[ind(l,i), t]$ is equal to

$$\frac{1}{\|\mathcal{V}_c\|} \frac{\sum_{([v_k \rightarrow v_{(l,i)}] \in E_{inbound,l}) \wedge (v_k \in V^-)} \frac{1}{out(G, v_k)}}{\|V^-\|},$$

where $out(G, v)$ is the outdegree of node v in G.

Compensation for the edges in G^-: We account for edges that are entirely in G^- by creating a self-loop that represents the sum of outdegree flow between all external nodes averaged by the number of external nodes; i.e.,

$$\mathbf{M}_0[t,t] = \frac{\sum_{v \in V^-} \frac{out(G^-, v)}{out(G, v)}}{\|V^-\|},$$

where $out(G^-, v)$ and $out(G, v)$ are the outdegrees of node v in G^- and G, respectively.

Completion: For any matrix position p, q not considered above, no compensation is necessary; i.e., $\mathbf{M}_0[p,q] = 0$.

2.3 L-PPR: Locality Sensitive PPR

Once the block-diagonal local transition matrix, $\mathbf{M}_{bd}$, and the compensation matrix, $\mathbf{M}_0$, are obtained, the next step is to obtain the PPR scores of the nodes in V^+. This can be performed using any fast PPR computation algorithm discussed in Section 1.1.

Note that the overall transition matrix $\mathbf{M}_{apx} = \mathbf{M}_{bd} + \mathbf{M}_0$ is approximate in the sense that all the nodes external to G^+ are clustered into a single node, represented by the last row and column of the matrix. Otherwise, the combined matrix $\mathbf{M}_{apx}$ accurately represents the nodes and edges in the "*merged localities graph*" combining the seed localities, G_1 through G_K. As we see in Section 3, this leads to highly accurate PPR scores with better scalability than existing techniques.

2.4 LR-PPR: Locality Sensitive and Reuse Promoting PPR

Our goal is not only to leverage locality-sensitivity as in L-PPR, but also to boost sub-result re-use. Remember that, as discussed above, the localized transition matrix $\mathbf{M}_{apx}$ is equal to $\mathbf{M}_{bd} + \mathbf{M}_0$ where (by construction) $\mathbf{M}_{bd}$ is a block-diagonal matrix, whereas $\mathbf{M}_0$ (which accounts for shared, boundary, and external nodes) is relatively sparse. We next use these two properties of the decomposition of $\mathbf{M}_{apx}$ to efficiently compute approximate PPR scores of the nodes in V^+. In particular, we rely on the following result due to [22], which itself relies on the Sherman-Morisson lemma [18]:

> Let $\mathbf{C} = \mathbf{A} + \mathsf{U}\mathbf{S}\mathbf{V}$. Let also $(\mathbf{I} - c\mathbf{A})^{-1} = \mathbf{Q}^{-1}$. Then, the equation
>
> $$\vec{r} = (1-c)(\mathbf{I} - c\mathbf{A})^{-1}\vec{e}$$
>
> has the solution
>
> $$\vec{r} = (1-c)(\mathbf{Q}^{-1}\vec{e} + c\mathbf{Q}^{-1}\mathbf{U}\mathbf{\Lambda}\mathbf{V}\mathbf{Q}^{-1}\vec{e}),$$
>
> where
>
> $$\mathbf{\Lambda} = (\mathbf{S}^{-1} - c\mathbf{V}\mathbf{Q}^{-1}\mathbf{U})^{-1}.$$
>
> If $\mathbf{A}$ is a block diagonal matrix consisting of k blocks, $\mathbf{A}_1$ through $\mathbf{A}_k$, then $\mathbf{Q}^{-1}$ is also a block diagonal matrix consisting of k corresponding blocks, $\mathbf{Q}_1^{-1}$ through $\mathbf{Q}_k^{-1}$, where $\mathbf{Q}_i^{-1} = (\mathbf{I} - c\mathbf{A}_i)^{-1}$.

We use the above observation to efficiently obtain PPR scores by setting $c = (1-\beta)$, $\mathbf{C} = \mathbf{M}_{apx}$, $\mathbf{A} = \mathbf{M}_{bd}$, and $\mathsf{U}\mathbf{S}\mathbf{V} = \mathbf{M}_0$. In particular, we divide the PPR computation into two steps: a locality-sensitive and re-usable step involving the computation of the $\mathbf{Q}^{-1}$ term using the local transition matrices and a run-time computation step involving the compensation matrix.

2.4.1 *Locality-sensitive and Re-usable $\mathbf{Q}_{bd}^{-1}$*

Local transition matrices, $\mathbf{M}_1$ through $\mathbf{M}_K$ corresponding to the seeds v_1 through v_K are constant (unless the graph itself evolves over time). Therefore, if $\mathbf{Q}_h^{-1} = (\mathbf{I} - (1-\beta)\mathbf{M}_h)^{-1}$ is computed and cached once, it can be reused for obtaining $\mathbf{Q}_{bd}^{-1}$, which is a block diagonal matrix consisting of $\mathbf{Q}_1^{-1}$ through $\mathbf{Q}_{K+1}^{-1}$ (as before, the last block, $\mathbf{Q}_{K+1}^{-1}$, is simply equal to $1_{1\times1}$):

$$\begin{pmatrix} \mathbf{Q}_1^{-1} & \mathbf{0}_{\|V_1\|\times\|V_2\|} & \cdots & \mathbf{0}_{\|V_1\|\times\|V_K\|} & \mathbf{0}_{\|V_1\|\times1} \\ \mathbf{0}_{\|V_2\|\times\|V_1\|} & \mathbf{Q}_2^{-1} & \cdots & \mathbf{0}_{\|V_2\|\times\|V_K\|} & \mathbf{0}_{\|V_2\|\times1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{0}_{\|V_K\|\times\|V_1\|} & \mathbf{0}_{\|V_K\|\times\|V_2\|} & \cdots & \mathbf{Q}_K^{-1} & \mathbf{0}_{\|V_K\|\times1} \\ \mathbf{0}_{1\times\|V_1\|} & \mathbf{0}_{1\times\|V_2\|} & \cdots & \mathbf{0}_{1\times\|V_K\|} & \mathbf{Q}_{K+1}^{-1} \end{pmatrix},$$

2.4.2 *Computation of the LR-PPR Scores*

In order to be able to use the above formulation for obtaining the PPR scores of the nodes in V^+, in the query time, we need to decompose the compensation matrix, $\mathbf{M}_0$, into $\mathsf{U}_0\mathbf{S}_0\mathbf{V}_0$. While obtaining a precise decomposition in run-time would be prohibitively expensive, since $\mathbf{M}_0$ is sparse and since we are looking for an approximation of the PPR scores, we can obtain a fairly accurate low-rank approximation of $\mathbf{M}_0$ efficiently [22]:

$$\mathbf{M}_0 \simeq \tilde{\mathbf{U}}_0\tilde{\mathbf{S}}_0\tilde{\mathbf{V}}_0.$$

Given this decomposition, the result vector $\vec{\phi}_{apx}$, which contains the (approximate) PPR scores of the nodes in V^+, is computed as

$$\vec{\phi}_{apx} = \beta\left(\mathbf{Q}_{bd}^{-1}\vec{s} + (1-\beta)\mathbf{Q}_{bd}^{-1}\tilde{\mathbf{U}}_0\mathbf{\Lambda}\tilde{\mathbf{V}}_0\mathbf{Q}_{bd}^{-1}\vec{s}\right),$$

where

$$\mathbf{\Lambda} = \left(\tilde{\mathbf{S}}_0^{-1} - (1-\beta)\tilde{\mathbf{V}}_0\mathbf{Q}_{bd}^{-1}\tilde{\mathbf{U}}_0\right)^{-1}.$$

Note that the compensation matrix $\mathbf{M}_0$ is query specific and, thus, the work done for the last step cannot be reused across queries. However, as we experimentally verify in Section 3, the last step is relatively cheap and the earlier(costlier) steps involve re-usable work. Thus, caching and re-use through LR-PPR enables significant savings in execution time. We discuss the overall complexity and the opportunities for re-use next.

2.5 Complexity and Re-use

Analysis of LR-PPR points to the following advantages: First of all, computation is done using only local nodes and edges. Secondly, most of the results of the expensive sub-tasks can be cached and re-used. Moreover, costly matrix inversions are limited to the smaller matrices representing localities and small matrices of size $r \times r$. Various subtasks have complexity proportional to $\|V^+\|^2$, where $\|V^+\| = \sum_{1 \leq l \leq K} \|V_l\|$. While in theory the locality V_l can be arbitrarily large, in practice we select localities with a bounded number of nodes; i.e., $\forall_{1 \leq l \leq K}, \|V_l\| \leq L$ for some $L \ll \|V\|$.

As described above LR-PPR algorithm supports caching and re-use of some of the intermediary work. The process results in local transition matrices, each of which can be cached in $O(\|E_l\|)$ space (where E_l is the number edges in the locality) assuming a sparse representation. The algorithm also involves a matrix inversion, which results in a dense matrix; as a result, caching the inverted matrix takes $O(\|V_l\|^2)$ space (where V_l is the number of vertices in the locality). If the locality is size-constrained, this leads to constant space usage of $O(L^2)$, where L is the maximum number of nodes in the locality. If the inverted matrix of a locality is cached, then the local transition matrix does not need to be maintained further. For cache replacement, any frequency-based or predictive cache-replacement policy can be used.

3. EXPERIMENTAL EVALUATION

In this section, we present results of experiments assessing the efficiency and effectiveness of the *Locality-Sensitive, Re-use Promoting Approximate Personalized PageRank* (LR-PPR) algorithm. Table 1 provides overviews of the three data sets (from http : //snap.stanford.edu/data/) considered in the experiments. We considered graphs with different sizes and edge densities. We also varied numbers of seeds and the distances between the seeds (thereby varying the overlaps among seed localities). We also considered seed neighborhoods (or localities) of different sizes.

Experiments were carried out using a 4-core Intel Core i5-2400, 3.10GHz, machine with 8GB memory and 64-bit Windows 7 Enterprise. Codes were executed using Matlab 7.11.0(2010b). All experiments were run 10 times and averages are reported.

3.1 Alternative Approaches

Global PPR: This is the default approach where the entire graph is used for PPR computation. We compute the PPR scores by solving the equation presented in Section 1.1.

FastRWR: This is an approximation algorithm, referred to as NB_LIN in [22]. The algorithm reduces query execution times by partitioning the graph into subgraphs and preprocessing each partition. The pre-computed files are stored on disk and loaded to the memory during the query stage. To be fair to FastRWR, we selected the number of partitions in a way that minimizes its execution time and memory and maximizes its quality.

L-PPR: This is our locality sensitive algorithm, where instead of using the whole graph, we use the *localized graph* created by combining the locality nodes and edges as described in Section 2.2. Once the localized transition matrix is created, the PPR scores are computed by solving the equation presented in Section 1.1.

LR-PPR: This is the locality sensitive and re-use promoting algorithm proposed described in detail in Section 2.4.

The restart probability, β, is set to 0.15 for all approaches.

3.2 Evaluation Measures

Efficiency: This is the amount of time taken to load the relevant (cached) data from the disk plus the time needed to carry out the operations to obtain the PPR scores.

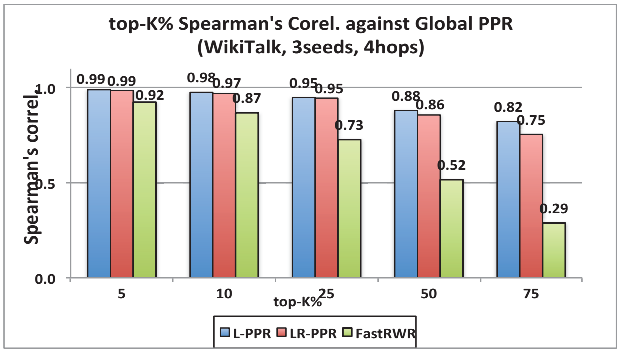


Figure 6: Accuracies of L-PPR, LR-PPR, and FastRWR against the Global PPR for different numbers of target nodes

Accuracy: For different algorithm pairs, we report the Spearman's rank correlation

$$\frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}},$$

which measures the agreement between two rankings (nodes with the same score are assigned the average of their positions in the ranking). Here, x and y are rankings by two algorithms and $\bar{x}$ and $\bar{y}$ are average ranks. To compute the rank coefficient, a portion of the highest ranked nodes in the merged graph according to x are considered. As default, we considered 10% highest ranked nodes; but we also varied the target percentage (5%, 10%, 25%, 50%, 75%) to observe how the accuracy varies with result size.

Memory: We also report the amount of data read from the cache.

3.3 Results and Discussions

Table 2 presents experimental results for FastRWR, L-PPR, and LR-PPR. First of all, all three algorithms are much faster than Global PPR. As expected, in *small data sets* (Epinions and Slashdot) FastRWR works faster than L-PPR and LR-PPR, though in many cases, it requires more memory. In *large data sets*, however, L-PPR and LR-PPR significantly outperform FastRWR in terms of query processing efficiency and run-time memory requirement.

In terms of accuracy, the proposed locality sensitive techniques, L-PPR and LR-PPR, constantly outperform FastRWR. This is because, FastRWR tries to approximate the whole graph, whereas the proposed algorithms focus on the relevant localities. FastRWR requires large number of partitions to ensure that the intermediary metadata (which requires dense matrix representation) fits into memory and this negatively impacts accuracy. Our locality-sensitive algorithms, L-PPR and LR-PPR, avoid this and provide high accuracy with low memory consumption, especially in large graphs, like WikiTalk.

Figure 6 confirms that the accuracies of L-PPR and LR-PPR both stay high as we consider larger numbers of top ranked network nodes for accuracy assessment, whereas the accuracy of FastRWR suffers significantly when we consider larger portions of the merged locality graph.

Figure 7 studies the execution time behavior for L-PPR, LR-PPR, and FastRWR for different number of seed nodes. As the figure shows, the time cost increases for both L-PPR and LR-PPR algorithms as the number of seeds increases. But, the cost of LR-PPR (which leverages re-use) increases much slower than the cost of L-PPR and both remain significantly cheaper than FastRWR.

Table 1: Data sets

Data Set	Overall Graph Characteristics		Locality Graph Characteristics		Seeds	
	# nodes	# edges	# nodes per neighborhood	# edges per neighborhood	# seeds	seed distances (hops)
Epinions	∼76K	∼500K	from ∼200 to ∼2000	from ∼10K to ∼75K	2-3	3-4
SlashDot	∼82K	∼870K	from ∼700 to ∼5000	from ∼10K to ∼75K	2-3	3-4
WikiTalk	∼2.4M	∼5M	from ∼700 to ∼6000	from ∼10K to ∼75K	2-8	3-4

Table 2: Summary of the results for different configurations (in all scenarios, individual seed localities have ∼75K edges)

Data set	Seeds		Merged Network		Execution Time (sec.)				Top-10% Spearman's Correl. (vs. Global PPR)			Memory usage(MB)		
	# seeds	Dist (#hops)	Avg # nodes	Avg # edges	Global PPR	Fast RWR	L-PPR	LR-PPR	Fast RWR	L-PPR	LR-PPR	Fast RWR	L-PPR	LR-PPR
Epinions ∼76K nodes ∼500K edges	2	3	∼2.2K	∼90K	27.81	0.21	0.37	0.14	0.963	0.997	0.990	178.3	2.9	36.3
	2	4	∼3.0K	∼99K	27.58	0.22	0.51	0.20	0.960	0.998	0.990		3.1	55.2
	3	3	∼2.7K	∼108K	27.30	0.21	0.58	0.26	0.967	0.998	0.990		4.6	57.6
	3	4	∼3.5K	∼120K	27.90	0.22	0.76	0.36	0.967	0.997	0.991		4.7	77.7
SlashDot ∼82K nodes ∼870K edges	2	3	∼5.9K	∼117K	21.79	0.35	0.70	0.53	0.955	0.973	0.990	302.1	5.0	228.1
	2	4	∼5.7K	∼125K	21.85	0.35	0.78	0.42	0.943	0.965	0.983		4.9	172.8
	3	3	∼7.1K	∼141K	21.74	0.36	1.12	0.95	0.957	0.971	0.990		7.6	325.9
	3	4	∼7.2K	∼159K	22.93	0.38	1.39	0.83	0.958	0.976	0.986		7.2	256.0
WikiTalk ∼2.4M nodes ∼5M edges	2	3	∼5.7K	∼102K	681.08	16.28	0.75	0.37	0.868	0.958	0.944	1429.0	15.5	114.5
	2	4	∼5.8K	∼100K	693.44	16.22	0.73	0.37	0.870	0.930	0.909		16.2	120.7
	3	3	∼6.3K	∼101K	701.34	16.32	0.75	0.37	0.877	0.937	0.902		24.0	211.6
	3	4	∼6.7K	∼103K	706.26	16.34	0.78	0.36	0.869	0.976	0.967		28.7	197.5

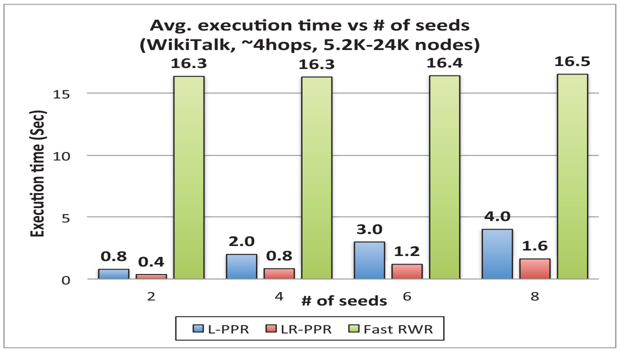


Figure 7: Execution times of L-PPR, LR-PPR, and FastRWR for different numbers of seed nodes

4. CONCLUSIONS

In this paper, we presented a *Locality-sensitive, Re-use promoting, approximate Personalized PageRank (LR-PPR)* algorithm for efficiently computing the PPR values relying on the localities of the seed nodes on the graph. Instead of performing a *monolithic* computation for the given seed node set using the entire graph, LR-PPR divides the work into localities of the seeds and caches the intermediary results obtained during the computation. These cached results can then be reused for future queries sharing seed nodes. Experiments showed that the proposed LR-PPR approach provides significant gains in execution time relative to existing approximate PPR computation techniques, where the PPR scores are computed from scratch using the whole network. LR-PPR also outperforms L-PPR, where the PPR scores are computed in a locality-sensitive manner, but without significant re-use.

5. REFERENCES

[1] K. Avrachenkov, N. Litvak, D. Nemirovsky, E. Smirnova, and M. Sokol. Quick Detection of Top-k Personalized PageRank Lists. WAW'11, 2011.

[2] B.Bahmani, K.Chakrabarti, and D. Xin. Fast personalized PageRank on MapReduce. In SIGMOD'11. 973-984. 2011.

[3] B.Bahmani, A.Chowdhury, and A.Goel. Fast incremental and personalized PageRank. PVLDB. 4, 3, 173-184, 2010.

[4] A. Balmin, V. Hristidis, and Y.Papakonstantinou. ObjectRank: Authority-based keyword search in databases. VLDB, 2004.

[5] S. Brin and L. Page. "The anatomy of a large-scale hypertextual Web search engine". Computer Networks and ISDN Systems 30: 107-117, 1998.

[6] K. S. Candan and W.-S. Li. Using random walks for mining web document associations. In *PAKDD*, pp. 294-305, 2000.

[7] K. S. Candan and W.-S. Li. Reasoning for Web document associations and its applications in site map construction. Data Knowl. Eng. 43(2), 2002.

[8] K. Csalogany, D.Fogaras, B. Racz, and T. Sarlos. Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments Internet Math. 2,3, 333-358, 2005.

[9] F. Fouss, A. Pirotte, J. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. TKDE, 2007.

[10] Y. Fujiwara, M. Nakatsuji, M. Onizuka, and M. Kitsuregawa. Fast and exact top-k search for random walk with restart. PVLDB. 5, 5, 442-453. 2012.

[11] M. Gupta, A. Pathak, and S. Chakrabarti. Fast algorithms for Top-k Personalized PageRank Queries. In WWW'08. 1225-1226. 2008.

[12] T.H. Haveliwala. Topic-sensitive PageRank. WWW'02. 517-526. 2002.

[13] G. Jeh and J. Widom. Scaling personalized web search. Stanford University Technical Report. 2002.

[14] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, and G.H. Golub. Extrapolation methods for accelerating PageRank computations. In WWW'03 261-270. 2003.

[15] G. Malewicz, *et al.* Pregel: a system for large-scale graph processing. SIGMOD'10, 2010.

[16] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time, CIKM'08, 2008.

[17] C. Palmer, P. Gibbons, and C. Faloutsos. Anf: a fast and scalable tool for data mining in massive graphs. KDD'02, 2002.

[18] W. Piegorsch and G. E. Casella. Inverting a sum of matrices. In SIAM Review, 1990.

[19] P. Sarkar, A.W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. ICML'08, 2008.

[20] H. H. Song, *et al.* Scalable proximity estimation and link prediction in online social networks. In *Internet Measurement Conference*, pp. 322–335. 2009.

[21] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. *KDD*, pp. 747–756, 2007.

[22] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast Random Walk with Restart and Its Applications. In ICDM'06. 613-622. 2006.

[23] Y. Wu and L. Raschid, ApproxRank: Estimating Rank for a Subgraph, ICDE'09, 54-65, 2009.