Efficient Node Proximity and Node Significance Computations in Graphs

by

Jung Hyun Kim

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2017 by the
Graduate Supervisory Committee:

Kasim Selçuk Candan, Chair
Hasan Davulcu
Hanghang Tong
Maria Luisa Sapino

ARIZONA STATE UNIVERSITY

August 2017

ABSTRACT

Node proximity measures are commonly used for quantifying how nearby or otherwise related to two or more nodes in a graph are. Node significance measures are mainly used to find how much nodes are important in a graph. The measures of node proximity/significance have been highly effective in many predictions and applications. Despite their effectiveness, however, there are various shortcomings. One such shortcoming is a scalability problem due to their high computation costs on large size graphs and another problem on the measures is low accuracy when the significance of node and its degree in the graph are not related. The other problem is that their effectiveness is less when information for a graph is uncertain. For an uncertain graph, they require exponential computation costs to calculate ranking scores with considering all possible worlds.

In this thesis, I first introduce Locality-sensitive, Re-use promoting, approximate Personalized PageRank (LR-PPR) which is an approximate personalized PageRank calculating node rankings for the locality information for seeds without calculating the entire graph and reusing the precomputed locality information for different locality combinations. For the identification of locality information, I present Impact Neighborhood Indexing (INI) to find impact neighborhoods with nodes' fingerprints propagation on the network. For the accuracy challenge, I introduce Degree Decoupled PageRank (D2PR) technique to improve the effectiveness of PageRank based knowledge discovery, especially considering the significance of neighbors and degree of a given node. To tackle the uncertain challenge, I introduce Uncertain Personalized PageRank (UPPR) to approximately compute personalized PageRank values on uncertainties of edge existence and Interval Personalized PageRank with Integration (IPPR-I) and Interval Personalized PageRank with Mean (IPPR-M) to compute ranking scores for the case when uncertainty exists on edge weights as interval values.

ACKNOWLEDGMENTS

I would like to thank to my advisor, Dr. Kasim Selçuk Candan for his support and encouragement. He continually and convincingly conveyed a spirit of adventure in regard to the research and taught me how to challenge and solve research problems in various approaches. Without his guidance and advice, this dissertation would not have been accomplished. I would also like to express my appreciation to Dr. Hasan Davulcu, Dr. Hanghang Tong, and Dr. Maria Luisa Sapino for being a part of my committee and sharing the insights of my thesis.

I would like to thank Parth Nagarkar, Xilun Chen, Xinsheng Li, Shengyu Huang, Sicong Liu, Yash Garg, Mao-Lin Li, Hans Behrens, and Silvestro Poccia at the EMIT lab members. It was my pleasure to work together in the researches and projects.

I also thank to Youngchoon Park at Johnson Controls Inc. and Weidong Li at American Express. I was fortunate to work in there as an intern specially under their guidance. It was great opportunities to improve and develop my skills with industrial views.

I am eternally express my love to my parents, Keewon Kim and Joowan Kim. During my PhD study, they gave me endless encouragement and emotional support.

Last but not least, I am very thankful to my beautiful wife, Soojin Jung. She encouraged and lifted me when I have difficulties in my research stood my side always, and supported me every time. I would also like to thank to my lovely kids who are Claire Chaelin Kim and Jayden Sejoon Kim. They are my precious pleasures and gave me happiness during my PhD study.

TABLE OF CONTENTS

Page

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER

1 INTRODUCTION 1

1.1 Shortcomings of Existing Techniques 5

1.2 Research Contributions 10

1.2.1 Locality-sensitive, Re-use promoting, approximate Personalized PageRank 10

1.2.2 Impact Neighborhood Indexing (INI) in Diffusion Graph 11

1.2.3 Relationship between Node Degrees and Node Significances 12

1.2.4 Personalized PageRank in Uncertain Graphs with Mutually Exclusive Edges 13

1.2.5 Personalized PageRank in Uncertain Interval Valued Graphs 14

1.3 Dissertation Overview 14

2 RELATED WORKS 16

2.1 Context-Sensitive PageRank 16

2.2 Improvements to the PageRank Function 17

2.2.1 Approximate Personalized PageRank 17

2.2.2 Partial Matrix Computation for Personalized PageRank 18

2.3 PageRank Optimization 19

2.4 Influential Node Identification 20

2.4.1 Information Flows within Networks and Influential Node Identification 20

2.4.2 Network Distance: Definitions and Indexing 21

CHAPTER Page

2.5 Graphs with Uncertainty 22

3 LOCALITY-SENSITIVE, RE-USE PROMOTING, APPROXIMATE PERSONALIZED PAGERANK COMPUTATION 24

3.1 Introduction 24

3.2 Proposed Approach 25

3.2.1 Combined Locality and its Boundary 26

3.2.2 Localized Transition Matrix 27

3.2.3 L-PPR: Locality Sensitive PPR 34

3.2.4 LR-PPR: Locality Sensitive and Reuse Promoting PPR 34

3.3 Complexity and Re-use 42

3.4 Optimizations 45

3.4.1 Locality Selection 46

3.4.2 Caching 46

3.4.3 Parallelization Opportunities 47

3.5 Experimental Evaluation 47

3.5.1 Alternative Approaches 49

3.5.2 Evaluation Measures 51

3.5.3 Results and Discussions 52

4 IMPACT NEIGHBORHOOD INDEXING IN DIFFUSION GRAPHS 68

4.1 Introduction 68

4.1.1 Contributions and Structure of the Chapter 68

4.2 Key Concepts 70

4.2.1 Propagation 70

4.2.2 Reinforcement 71

CHAPTER Page

4.2.3 Zero-Erasure Neighborhoods (ZENs) 72

4.2.4 Impact Neighborhoods 74

4.3 Impact Neighborhood Indexing (INI): Overview and Challenges 75

4.3.1 Outline of the Basic INI Process 75

4.4 Reducing Costs 77

4.5 Reducing False Positives 79

4.5.1 Signature Length and False Positives 80

4.5.2 Grid-Signatures 80

4.6 Index Reuse 84

4.6.1 Bit Masking 84

4.6.2 $\mathcal{E}$-Erasure Neighborhoods 85

4.7 Implementation Discussions 86

4.8 Experiments 88

4.8.1 Verification of INI's General Properties 90

4.8.2 Efficiency of INI on Real Graphs 92

5 PAGERANK REVISITED: RELATIONSHIP BETWEEN NODE DEGREES AND NODE SIGNIFICANCES 95

5.1 Introduction 95

5.1.1 Tight Coupling of PageRank Scores of Nodes and their Degrees 95

5.1.2 PageRank Revisited: De-coupling Node Significance from Node Degrees 97

5.2 Degree De-Coupled PageRank 98

5.2.1 Desideratum 98

5.2.2 Degree De-coupling Transition Matrix 99

CHAPTER Page

5.3 Case Studies 104

5.3.1 Setup 104

5.3.2 Measures 106

5.3.3 Impact of De-Coupling in Different Applications (Unweighted Graphs) 107

5.3.4 Relationship between α and $\mathbf{p}$ 111

5.3.5 Relationship between β and $\mathbf{p}$ in Weighted Graphs 114

6 PERSONALIZED PAGERANK IN UNCERTAIN GRAPHS WITH MUTUALLY EXCLUSIVE EDGES 118

6.1 Introduction 118

6.2 Problem Formulation 120

6.2.1 Possible Worlds of an Uncertain Edge 122

6.2.2 Possible Worlds of a Graph 123

6.2.3 PPR under Uncertainty 123

6.3 Naive Approaches 124

6.3.1 Exhaustive Approaches 124

6.3.2 Collapsing-based Approaches 125

6.3.3 Flattening-based Approaches 127

6.4 UPPR: Proposed Approach 129

6.4.1 Special Case: Two Possible Worlds 129

6.4.2 General Case: $>$ 2 Possible Worlds 132

6.4.3 Accuracy of UPPR 138

6.4.4 Efficient Computation of UPPR Scores 139

CHAPTER Page

6.4.5 Hybrid Computation in the Presence of Large Numbers of Uncertain Edges . . . 140

6.5 Experiments . . . 141

6.5.1 Datasets and Setup . . . 141

6.5.2 Alternative Approaches . . . 142

6.6 Results and Discussions . . . 142

7 PERSONALIZED PAGERANK IN UNCERTAIN GRAPHS WITH UNCERTAIN EDGE WEIGHTS . . . 148

7.1 Introduction . . . 148

7.2 Problem Formulation . . . 150

7.2.1 Interval Edges and Interval Graphs . . . 150

7.2.2 Naive Approach: PPR Computation with Sampling . . . 151

7.2.3 Personalized PageRank in an Interval Graph . . . 152

7.3 Interval Personalized PageRank with Integration (IPPR-I) . . . 153

7.3.1 Interval Transition Matrix for Interval Graphs . . . 153

7.3.2 Interval Personalized PageRank with Integration (IPPR-I) . . . 157

7.4 Interval Personalized PageRank with Mean (IPPR-M) . . . 157

7.5 Mixing Factor and Localized Graph . . . 158

7.6 Experimental Evaluation . . . 160

7.6.1 Datasets and setup . . . 161

7.6.2 Measure . . . 162

7.6.3 Results and Discussions . . . 162

8 CONCLUSIONS . . . 165

CHAPTER Page

8.1 Locality-Sensitive, Re-use Promoting, Approximate Personalized PageRank . . . 165
8.2 Impact Neighborhood Indexing in Diffusion Graphs . . . 166
8.3 Degree Decoupled PageRank . . . 166
8.4 Uncertain Personalized PageRank . . . 167
8.5 Interval Personalized PageRank . . . 167
9 FUTURE WORKS . . . 169
9.1 SVD Decomposition of an Interval Valued Matrix . . . 169
9.2 Matrix Inverse on an Interval Valued Matrix . . . 171
9.2.1 Interval Matrix Arithmetic . . . 171
9.2.2 Matrix Inverse in a Diagonal Interval Valued Matrix . . . 172
REFERENCES . . . 174

LIST OF TABLES

Table Page

3.1 Data Sets 48

3.2 FastRWR Performance for Different Data Sets and Configurations 50

3.3 Summary of Execution Time Results for Different Configurations on 10K Seed Localities 52

3.4 Summary of Accuracy Results for Different Configurations on 10K Seed Localities 53

3.5 Summary of Memory Usage Results for Different Configurations on 10K Seed Localities 53

3.6 Summary of Execution Time Results for Different Configurations on ~75K Seed Localities 54

3.7 Summary of Accuracy Results for Different Configurations on ~75K Seed Localities 54

3.8 Summary of Memory Usage Results for Different Configurations on ~75K Seed Localities 55

4.1 Index Creation and Querying times, and Memory Usage for INI 89

5.1 Spearman's Rank Correlation Between the Node Degree Ranks and The Node Ranks' Based on PageRank Scores for Various Data Graphs . 96

5.2 Ranks of Graph Nodes of Different Degrees on a Sample Graph for Different De-coupling Weights, p 101

5.3 Data Sets and Data Graphs 103

6.1 Data Sets 142

6.2 Uncertainty Scenarios 143

6.3 UPPR Vs. MC Method on the Facebook Graph 146

7.1 Data and Parameters 161

LIST OF FIGURES

Figure Page

1.1 Ambiguity in Wikipedia and Its Potential impact on the Proximity/Cluster Analysis 8

1.2 Locality-sensitivity: Computation of PPR Should Focus on The neighborhoods (Localities) of the Seeds 10

1.3 Re-use Promotion: Two PPR Queries Sharing a Seed Node (v_1) should Also Share Relevant Work 10

3.1 Incoming and Outgoing Boundary Nodes/Edges and a Node Shared between Two Localities 27

3.2 An Equivalence Set Consists of the Copies of a Node Shared across Multiple Seed Locality Graphs 28

3.3 The Matrix, $\mathbf{M}_{Bd}$ and the Corresponding Graph 29

3.4 Accounting for Shared Nodes in the Compensation Matrix, $\mathbf{M}_0$ 31

3.5 Accounting for the Edges That Are Outgoing from a Locality 32

3.6 Accounting for the Edges That Are Originating from The nodes That Are Outside of the Localities of G_1 and G_2 33

3.7 (a) The Probability from Node v_i to Node v_j in the original Graph $\mathbf{G}$ and (b) Locality Graph G^+ with an external Node G^- 38

3.8 Accuracies of L-PPR, LR-PPR, FastRWR, and GMRES-PPR Against the Global PPR for Different Numbers of Target Nodes 57

3.9 Performances of L-PPR, LR-PPR, FastRWR, and GMRES-ppr on the Size of the Combined Localities Network (Epinion Data Set) 58

3.10 Execution Times of the Algorithms L-PPR, LR-PPR, FastRWR, and GMRES-PPR for Different Data Sets of Varying Sizes 59

Figure Page

3.11 Performances of L-PPR, LR-PPR, and FastRWR on the Size of the Combined Localities Network (WikiTalk Data Set) 60

3.12 Distribution of the Execution Times for L-PPR and LR-PPR for the Epinions Data Set 61

3.13 Performance of LR-PPR as a Function of the Size of the Combined Localities Network (Epinion Data Set, 3 Seeds, ~4 Hops) 62

3.14 The Impact of the Ratio of the Boundary Edges on the Execution Time for L-PPR and LR-PPR (Epinions, 3 Seeds, with Distance ~ 4 Hops) . . 63

3.15 Impact of the Boundary Edges for the SlashDot Data Set (3 Seeds) 63

3.16 Offline Parallelization Execution Time of LR-PPR Generating Locality Graphs and Calculating $\mathbf{Q}_h^{-1}$ for Four Seeds on Different Number of Cores 65

3.17 Performances of L-PPR, LR-PPR, and FastRWR on the Size of the Combined Localities Network (SlashDot Data Set) 66

3.18 Performances of L-PPR, LR-PPR, and FastRWR on the Size of the Combined Localities Network (LiveJournal Data Set) 67

4.1 The Fingerprint of Node n_i Propagates in the Graph, Subject to Bit Erasures 69

4.2 Node n_i Receives the Fingerprints of the Nodes Within Its zero-erasure Neighborhood Intact 71

4.3 Propagation with Reinforcement: Darker Shaded Nodes Have higher Probability of Receiving the Message Intact 72

4.4 Combining Signatures for Reduced Cost 77

4.5 Partial Signatures n_i Receives from the Nodes Outside of Its Zero-erasure Neighborhood Makes up the Exo-neighborhood Noise on n_i 79

Figure Page

4.6 Grid-signatures 81

4.7 Creation of Grid-signatures 83

4.8 Simple Graph Topologies: (a) in a Linear Graphand (b) in a Lattice Graph 88

4.9 Using Smaller Number of Signature Bits Than b (Computed in Section 4.5) Results in False Positives 91

4.10 (a) A Fixed Erasure May Lead to Drops in Accuracy, Whereas (b) Adaptive Erasure Helps Improve Accuracy 92

4.11 Distribution of Matching Nodes for a Target Radius of 4 on a Linear Graph (1000 Runs). 93

4.12 Index Re-use with Bit-masking 93

4.13 Index Re-use Through ε-erasure ($\varepsilon = 1$ and $\varepsilon = 3$): predicted and observed number of matches are well aligned 94

5.1 Transition Probabilities from Node $v_i = A$ to All Its Neighbors v_j in Different p Values 101

5.2 Application Group A: $p > 0$ is Optimal (i.e., Node Degrees Need to Be Penalized) 108

5.3 Application Group B: $p = 0$ Is Optimal 109

5.4 Application Group C: $p < 0$ Is Optimal (i.e., Node Degrees Need to Be Boosted) 110

5.5 Correlations Between Node Degrees and Application specific Significances for Different Data Graphs 111

5.6 Relationship Between p and α, for Application Group A, Where $p > 0$ Is Optimal (i.e., Degrees Need to Be Penalized) 112

Figure Page

5.7 Relationship Between p and α, For Application Group B, Where $p = 0$ Is Optimal 113

5.8 Relationship Between p and α, for Application Group C, Where $p < 0$ Is Optimal (i.e., Node Degrees Need to Be Boosted) 114

5.9 Relationship Between p and β, for Application Group A, Where $p > 0$ Is Optimal (i.e., Node Degrees Need to Be Penalized) 115

5.10 Relationship Between p and β, for Application Group B, Where $p = 0$ Is Optimal 116

5.11 Relationship Between p and β, for Application Group C, Where $p < 0$ Is Optimal (i.e., Node Degrees Need to Be Boosted) 117

6.1 A Graph with Certain and Uncertain Edges 120

6.2 Alternative (Naive) Approaches for Computing PPR values on an Uncertain Graph 125

6.3 Flattening of the Uncertain Graph in Figure 6.1 Into an (Approximate) Certain Graph 127

6.4 An Example of Multiple Uncertain/Certain Edges with Same Target Nodes 134

6.5 An Example of an Uncertain Edge with and without ϵ 137

6.6 Results on the Facebook Data Set, for Different Amount of Uncertainty with Different Edge Semantics 144

6.7 Results in Different Graphs of Different Sizes 145

7.1 Interval Edge Weights When a User Have Different Degree of Interests . 149

7.2 Examples of Interval-valued/Sampled/Scalar-valued Graphs 152

7.3 An Example of One Interval Edge 155

Figure Page

7.4 An Example of Rankings of IPPR-R and IPPR-M 159

7.5 Rank Correlation Results on Different Range of Intervals 162

7.6 Rank Correlation Results on Different Outgoing Degrees of Nodes 163

7.7 Rank Correlation Results on Random Weights, Random Out-degrees, Random % of Interval Edges 164

7.8 Results of Execution on IPPR-M and IPPR-I 164

Chapter 1

INTRODUCTION

There has been a growing interest in the research area of node proximity computation in a graph. Given data such as history, interest, and connections, people have tried to find relevant nodes for nodes in different graph-structured application domains such as social networks, biology, web search, and recommendation systems[66]. Node distance/proximity measures are commonly used for quantifying how nearby or otherwise related to two or more nodes on a graph are. In many graph applications, how a given pair of nodes on a graph relates to each other is determined by the underlying graph topology. Given a graph, measures of node proximity are available as estimates of node similarity and can be defined in two different ways. The first definition is a Path-length based definition. This is 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). The straightforward approach is to use a recursive breath-first search (BFS) to find the path between nodes but this approach is very costly. To overcome the time complexity for the use of online query, the shortest path problems for the distance between nodes can be solved with two steps that are preprocessing and answering queries [4, 110, 115]. Preprocessing algorithm computes certain information such as an index of data or a data structure between every pair of nodes for the preparation of the second step in the offline. On the second step, the distance can be answered very efficiently in almost constant time. The main problems on this shortest path query are the size of indexing and the execution time of the answer on the online query. When the graph size is very large, it is inefficient for the space cost.

On the other hand, random-walk based definitions, such as hitting distance [24, 82] and PageRank score [17, 70], of node relatedness, also take into account the density of 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. Many web search and recommendation algorithms rely on random-walks to identify significant nodes in the graph. Since enumerating all paths among the graph nodes would require time exponential in the size of the graph, random-walk based techniques encode the structure of the network in the form of a transition matrix of a stochastic process from which the node significance can be inferred.

A stochastic process is said to be Markovian if the conditional probability distribution of future states, given the present, depends only on the present. A Markov chain is a discrete-time stochastic process which is conditionally independent of the past states. A basic random walk on a graph, $G(V, E)$, on the other hand, is a Markov chain whose state at any time is described by a vertex of G and the transition probability is distributed equally among all outgoing edges. These random-walk based models are used heavily in many application domains, including data mining, bioinformatics, and queuing theory. Since the next state of a Markovian chain only depends on current state and given the current state, is conditionally independent of the past states, for a process with finite number of states, the transition probability distribution can be represented as a matrix. The (i, j)'th element of this matrix, T_{ij}, describes the probability that, given that the current state is i, the process will be in state j in the next time unit; i.e., $T_{ij} = P(S_{now+1} = j | S_{now} = i)$. Given this 1-step transition probability, the n-step transition probabilities can be computed as the n'th extrapolation of the transition matrix. If the transition matrix T is irreducible (each state is accessible from all other states) and aperiodic (for any state s_i, the greatest

common divisor of $\{n \geq 1 | T_{ii}^n > 0\}$ is equal to 1), then in the long run the Markov chain has a unique stationary distribution independent of the initial distribution.

Stationary distribution of a random walk is a convergence vector (probability distribution), where no subsequent steps change the probability distribution. It can be shown that if an undirected random walk graph is strongly connected and non-bipartite, then it can be modeled as a Markov Chain with a stationary distribution (the fundamental theorem of Markov Chains). Of course, not all transitional matrices have these properties. Furthermore, it is not always that users are interested in the steady state behaviors of the system, but whether a condition is true in any time in the (bounded) future. In other words, given an initial probability distribution vector π, users may aim finding if $\Theta(T^k\pi)$ for a k, here Θ denotes some (linear) condition. function. For example, the graph node n_j is said to be with Δ hitting distance of the node n_i, of a random walk on the corresponding random-walk graph starting from n_i is expected to visit n_j in at most Δ steps [24].

Hitting time distance[69] is the expected number of steps to reach a target node via random walk from a node. It is similar to Path-length based distance, but it takes into account the density of the edges in the graph: given a node n, nodes within hitting distance δ are those nodes a random walk starting from n is expected to visit in at most δ steps.

PageRank [17] is one of the most widely-used random-walk based methods for measuring node significance. The basic idea of PageRank is that a node is important if it is pointed to by other important nodes – it takes into account the connectivity of nodes in the graph by defining the score of the node v_i as the amount of time spent on v_i in a sufficiently long random walk on the graph. The PageRank algorithm associates a single importance score to each node: 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 α, the random walk moves along an outgoing edge of the current node with a probability proportional to the edge weights and
- with probability $1 - \alpha$, the walk jumps to a random node in V.

More specifically, given a graph $G(V, E)$ where V is a set of nodes in G and E is a set of edges in G, the PageRank scores are represented as $\vec{r}$, where

$$\vec{r} = \alpha \mathbf{T} \vec{r} + (1 - \alpha) \vec{e}$$

where $\mathbf{T}$ is a transition matrix corresponding to the graph G, $\vec{e}$ is a teleportation vector (such that $\vec{e}[i] = \frac{1}{\|V\|}$), and α is the residual probability (or equivalently, $(1 - \alpha)$ is the so-called teleportation probability). Unless the graph is weighted, the transition matrix, $\mathbf{T}$, is constructed such that for a node v with k (outgoing) neighbors, the transition probability from v to each of its (outgoing) neighbors will be $1/k$. If the graph is weighted, then the transition probabilities are adjusted in a way to account for the relative weights of the (outgoing) edges. The basic definition of PageRank associates a convergence score to each node in the graph irrespective of content and context of search.

An alternative to this approach is to modify the teleportation vector, $\vec{j}$: instead of jumping to a random node in V with probability $1 - \alpha$, 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 (PPR) scores of the nodes in V with a vector $\vec{\phi}$, then

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

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. Intuitively, since at each step the random-walk has a non-zero probability of jumping back to the seed nodes from its current node (independently of where the current node is in the graph), the nodes closer to the nodes in S will have larger stationary scores than they would have if the random walk jumped randomly in the entire graph. One key advantage of this approach over modifying the transition matrix as in [19] is that the term $1 - \alpha$ can be used to directly control the degree of seeding (or personalization) of the scores.

1.1 Shortcomings of Existing Techniques

For the measure of node proximity, personalized PageRank (PPR) is an effective measure but there are several shortcomings. A particular shortcoming is that the use of personalized PageRank for large graphs is difficult due to the high cost of solving for the vector $\vec{\phi}$, given $1 - \alpha$, transition matrix $\mathbf{T}$, and the seeding vector $\vec{s}$. One way to obtain $\vec{\phi}$ is to rewrite the stationary state equation of personalized PageRank as

$$\phi = (1 - \alpha)(I - \alpha\mathbf{T})^{-1}\vec{s},$$

and solve the above equation for $\vec{\phi}$ mathematically. Alternatively, PowerIteration methods [56] explicitly 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 as follows:

$$\vec{\phi}_n = \alpha\mathbf{T}\vec{\phi}_{n-1} + (1 - \alpha)\vec{s}.$$

Unfortunately, for large data sets, both of these processes are prohibitively expensive. For the mathematical way, it requires a preprocessing step to pre-compute the $(I - \alpha\mathbf{T})^{-1}$ which takes long time on the inverse matrix computation. Though $\mathbf{T}$ is a sparse

matrix, $(I - \alpha\mathbf{T})^{-1}$ is a dense matrix, so it needs a large amount of memory cost to save the precomputed inverse matrix. Additionally, the preprocessing step takes long time to compute the inverse matrix. For PowerIteration methods, it requires repeated matrix-vector multiplications until the difference between $\vec{\phi}_n$ and $\vec{\phi}_{n-1}$ converges to a criterion but it can be unrealistic in the real world.

Another shortcoming of PageRank (or personalized PageRank) computation is that the ranking scores are tightly coupled with the degrees of the graph nodes. Let us consider an undirected graph $G(V, E)$. There are two distinct factors that contribute to the PageRank of a given node, $v \in V$:

- Factor 1: Significance of Neighbors: The more significant the neighbors of a node are, the higher its likelihood to be also significant.
- Factor 2: Number of Neighbors (Degree of the Node) : Even if the neighbors are not all significant, a large number of neighbors would imply that the node, v, is well-connected and, thus, likely to be structurally important.

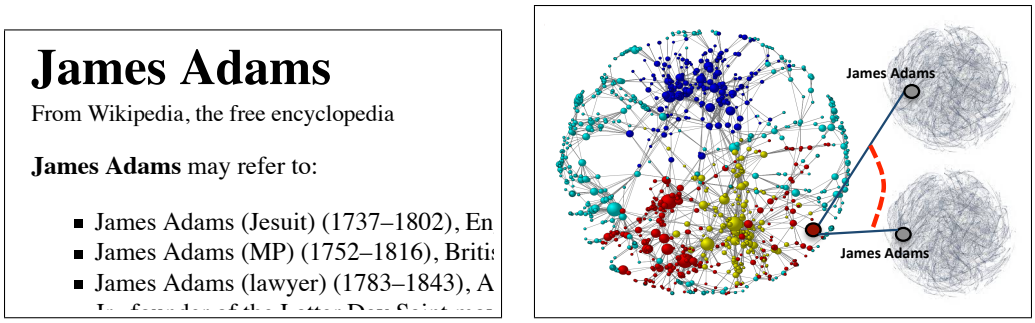
The first factor is how significant the nodes that are neighboring v are through edges incoming to v, and the second factor is the number of edges incident onto these nodes. Intuitively, the more significant the neighbors of a node are, the higher its likelihood to be also significant. Secondly, even if the neighbors are not all significant, a larger number of neighbor implies that the node, v, is well-connected and, thus, likely to be structurally important. Therefore, in theory, these two factors should complement each other. In practice, however, the PageRank formulation described above implies that there is a very tight coupling between the degrees of the nodes in the graph and their PageRank scores.

The problem is that it is possible that the node degree and the node significance are in fact inversely related and that the tight- coupling between node degrees and

PageRank scores might be counter-productive in generating accurate recommendations. Additionally, the mismatch between PageRank score and node significance is not limited to the cases where node degrees are inversely related to the node significance. There are other scenarios where PageRank may, in fact, fail to sufficiently account for the contribution of the node degrees to their significances. In certain applications, the significance of the node may be negatively or not correlated with the node degree, whereas in others PageRank may not be sufficient in accounting for degree contributions. Naturally, in such applications, the naive application of PageRank in generating recommendations may return poor results.

The last shortcoming of personalized PageRank measure is that despite their effectiveness when the underlying graph is certain, these measures become inapplicable and difficult to apply in the presence of graph uncertainties, as they are not designed for graphs that include uncertain information. Consequently, they can be used only for proximity computations when all node and edge information in the given graph are certain/complete. Unfortunately, in many real world web and social-network based applications, it may not be possible to obtain a perfect and complete structure of the underlying knowledge graph for various reasons: genuine lack of information, noise in data collection, or privacy issues, where one is provided with a reduced, clustered, or intentionally noisy and obfuscated version of the graph to hide information[61].

Most existing works on graph uncertainty consider existence uncertainty, where a given edge exists probabilistically and the existence probabilities of the individual edges are assumed to be independent from each other [14, 28, 61, 75, 118, 116, 93, 121]. In practice, however, this assumption does not always hold: we may be aware of the existence of an edge, but we may not know between which pairs of nodes the edge exists. For example, we may be able to deduce that one of the several friends of an individual in a social network may be his/her father, but we may not know which



(a) Ambiguity in data (b) Uncertain edge

Figure 1.1: Ambiguity in Wikipedia and Its Potential impact on the Proximity/Cluster Analysis

friend. As another example, we may know that a name referred to in a web document is one of the many named entities in a knowledge base, but we may not know which one is the correct entity (Figure 1.1(a)). Obtaining node rankings in such a graph is difficult because addition or removal of one single edge can have a drastic effect on proximity [30, 32]: e.g., addition of just one edge may be sufficient to link two otherwise distant node clusters, thereby significantly altering the proximities of a large number of pairs of nodes in the graph (Figure 1.1(b)).

The another type of uncertainty in a graph is an interval value of weights on edges. When the weights of edges are uncertain with interval values, the adjacency matrix and transition matrix of a graph become interval matrices. For the interval matrix, personalized PageRank (PPR) scores can be computed PPR equations $\vec{\phi} = \alpha\mathbf{T}\vec{\phi} + (1-\alpha)\vec{s}$ with interval arithmetics [49]. The problem of this approach is that it requires a lot of time on the interval matrix computations because the interval matrix computation requires a combination of minimum values and maximum values of matrices. For example, for the interval multiplications, let a and c be scalar values. The multiplication is $a \times c$ which takes one computation. When the values have interval with $[a, b]$ and $[c, d]$, the multiplication is

$$[a, b] \times [c, d] = [min(a \times c, a \times d, b \times c, b \times d), max(a \times c, a \times d, b \times c, b \times d),]$$

which takes four times computation. If we compute the interval matrix multiplication of two interval matrices, it takes much longer time than scalar valued matrix computations. There are some tools such as *IntLab* [94] which is optimized to compute interval matrix arithmetics but it still requires long execution compared to scalar valued matrix computations when the size of a matrix is large or the density of a matrix is high.

The another way to compute PPR scores in interval matrices is a sampling technique such as Monte Carlo method[7, 37]. For each interval edge, we randomly samples some values within an interval range. This sampling makes the interval values into discrete values. Given discrete values on edge weights, PPR scores can be computed for all the possible worlds of combination of edge weights. After PPR computations, the average of PPR values becomes the PPR scores of an interval graph. The problem of this approach is that the number of possible worlds may be exponential depending on the number of samples and the number of edges.

To overcome this problem, Ishii and Tempe[51, 52] proposed how to compute PageRank scores in an interval weighted graph. They used the center and the radius of values in the interval matrix and compute PageRank scores with finding the smallest interval vector which are close to the expected PageRank vector by linear programming. Though it finds PageRank scores minimizing the interval range, it does not guarantee the accuracy of PageRank scores since they focus on how to minimize the range of intervals with approximation.

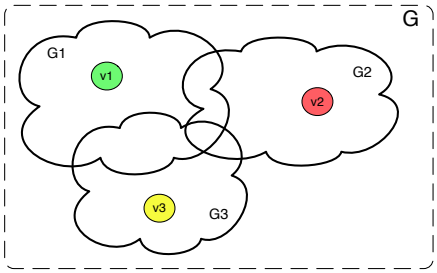
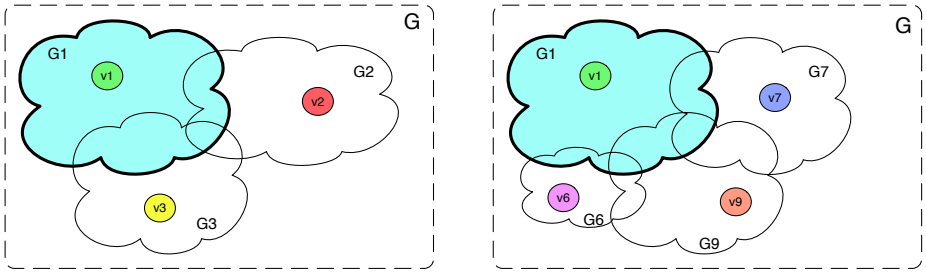


Figure 1.2: Locality-sensitivity: Computation of PPR Should Focus on The neighborhoods (Localities) of the Seeds



(a) PPR query 1 (b) PPR query 2

Figure 1.3: Re-use Promotion: Two PPR Queries Sharing a Seed Node (v_1) should Also Share Relevant Work

1.2 Research Contributions

1.2.1 Locality-sensitive, Re-use promoting, approximate Personalized PageRank

In this thesis, I propose a Locality-sensitive, Re-use promoting, approximate personalized PageRank (LR-PPR) algorithm[63, 64] for efficiently computing the PPR values relying on the localities of the seed nodes on the graph to improve both scalability and accuracy: The LR-PPR algorithm is

- locality sensitive in the sense that it reduces the computational cost of the PPR computation process and improves accuracy by focusing on the neighborhoods of the seed nodes (Figure 1.2); and
- re-use promoting in that, instead of performing a monolithic PPR computation for a given seed node set S (where the intermediary results cannot be re-used

for a different, but similar, seed set S'), LR-PPR divides the work into localities of the seeds and enables caching and re-use of significant portions of the intermediary work for the individual seed nodes in future queries (Figure 1.3): In other words, LR-PPR is able to leverage temporal locality in the users queries:

- This temporal locality may be due to a slow evolution of a given users interest: for example when a user watches a new online movie, this will change the recommendation context only slightly as the users recent movie history (say the last 10 movies watched by the user) will be mostly preserved in the seed set.
- This temporal locality may also be due to popular seeds shared by a lot of users: for example a new hit movie (or say the top 10 movies of the week) may be shared in the seed set of a large portion of the users. LR-PPR leverages such temporal localities to reduce redundant work.

1.2.2 Impact Neighborhood Indexing (INI) in Diffusion Graph

A locality graph consists of a set of graph nodes that are nearby or otherwise related to a seed node. In many applications, relatedness of a pair of graph nodes depends on how information (or influence) flows from one node to the other in the underlying topology, and various algorithms have been proposed to identify influencers or to maximize overall influence in such networks [26, 59, 108]. These algorithms rely on various cascade, contagion, and diffusion models (such as the order-independent cascade and threshold based models) to capture the dynamics of flow within network. In the order-independent cascade model, for example, the likelihood of a node, n, becoming influenced at a given point depends on (a) whether n's neighbors are already influenced, (b) whether they attempt to influence n, and (c) the degree of

influence they have on n. In threshold-based model, however, the influence is assumed to accumulate over time and n becomes influenced when the accumulation passes a threshold. Note that, while their details differ, these various propagation models have two properties in common: (a) decay with distance, meaning that as one moves further away from the source of information/influence, the less likely s/he is to be impacted and (b) reinforcement, meaning that multiple paths over which information/influence is received can reinforce the core message, increasing its impact. Relying on this observation, I introduce the concept of impact neighborhoods, which capture both topological and propagative characteristics of graphs, including decay and reinforcement: We say that a node n is likely to impact another node m in a given network (with decay and reinforcement), if information originating at n reaches m. We define the impact neighborhood of a given node n as the set of nodes that are impacted by n. Impact Neighborhood Indexing in Diffusion Graphs (INI) algorithm computes zero-erasure neighborhoods (ZENs) and impact neighborhoods (for a given impact radius, r). INI propagates fingerprints in the network subject to bit-erasures, modeling decay. During query time, impact neighborhoods are identified by querying the network nodes for the query node's fingerprint.

1.2.3 Relationship between Node Degrees and Node Significances

As I discussed above, one key shortcoming of the conventional PageRank scores is that they are often tightly coupled with the degrees of the graph nodes and in many applications the relationship between the significance of the node and its degree in the underlying network may not be as implied by PageRank-based measure. Degree De-coupled PageRank (D2PR) algorithm[65] improves the effectiveness of PageRank based knowledge discovery and recommendation systems. These techniques suitably

penalize or (if needed) boost[1] the transition strength based on the degree of a given node to adapt the node significances based on the network and application characteristics.

To de-couple the PageRank score from node degrees, it requires to modify the transition matrix. To control the transition matrix, I define a parameter, p which adjusts the transition based on the number of degree of neighbors. When p is negative, it boost the transition probabilities of neighbors which has high degree than other neighbors. When p is positive, it penalize the transition probabilities of neighbors which has high degree. When p is zero, the transition matrix is same as the standard PageRank probabilities. With the de-coupled transition matrix, we can compute de-coupled PageRank scores with the conventional PageRank equation. D2PR algorithm shows how to build degree de-coupled transition matrices for different graphs such as undirected unweighted, directed unweighted, and weighted graphs.

1.2.4 Personalized PageRank in Uncertain Graphs with Mutually Exclusive Edges

In addition to the above challenges on PPR measure, there is another challenge which is a node proximity measure in uncertainty. As it was mentioned in Section 1.1, there is an assumption that all node and edge information in a graph should be complete, so when uncertain data exits, it fails to measure the accurate node proximity. To tackle with this challenge, I propose an efficient Uncertain Personalized PageRank (UPPR) algorithm [67] to approximately compute personalized PageRank values on an uncertain graph with edge uncertainties. The proposed UPPR measure can be applied in uncertain graphs when the existence of edges is uncertain. For the uncertainty of edge existence, there are two semantics which are probabilistic edge existence

[1]In this context, de-coupled does not necessarily imply de-correlated. In fact, D2PR can boost correlation between node degree and PageRank if that is required by the application.

under 1) mutual Exclusion and 2) multiple edge selections. The number of possible worlds in an uncertain graph can be exponential both on different semantics, so it is hard to compute PPR scores with naive approach. UPPR approach avoids enumeration of all possible worlds, yet it is able to achieve comparable accuracy by carefully encoding edge uncertainties in a data structure that leads to fast approximations.

1.2.5 Personalized PageRank in Uncertain Interval Valued Graphs

To overcome the problem that uncertainty exists on edge weights as interval values with range instead of scalar values in an uncertain graph, I propose an effective Interval Personalized PageRank with Mean (IPPR-M) and Interval Personalized PageRank with Integration (IPPR-I) algorithms which compute personalized PageRank scores when the edge weights are interval values. Instead of sampling the interval values of edge weights and then computing PPR scores for the possible worlds approaches, IPPR-I which is an optimal solution, gives the integral formulas to compute the weights of scalar edge weights and interval edge weights and then compute PPR scores. IPPR-M computes approximate PPR scores effectively with the mean of intervals.

1.3 Dissertation Overview

The rest of the dissertation is structured as follow:

- In Chapter 2, I review background and related works in the literature.
- In Chapter 3, I present 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.
- In Chapter 4, I present an impact neighborhood indexing (INI) algorithm that

creates data structures to help quickly identify impact neighborhood of any given node.

- In Chapter 5, I present degree decoupled PageRank (D2PR) techniques to improve the effectiveness of PageRank based knowledge discovery and recommendation systems.

- In Chapter 6, I present an efficient Uncertain Personalized PageRank (UPPR) algorithm to approximately compute personalized PageRank values on an uncertain graph with edge uncertainties.

- In Chapter 7, I present Interval Personalized PageRank with Integration (IPPR-I) and Interval Personalized PageRank with Mean (IPPR-M) algorithms that compute personalized PageRank scores efficiently when the edge weights are uncertain with interval edge weights in graphs.

- In Chapter 8, I conclude the dissertation.

- In Chapter 9, I discuss the future works.

Chapter 2

RELATED WORKS

2.1 Context-Sensitive PageRank

An early attempt to contextualize the PageRank scores is the topic sensitive PageRank [48] 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 match to the search topic. [19, 20] 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, $S \subseteq V$: [19] constructs a transaction matrix, $\mathbf{T}_S$, where edges leading away from the seed nodes are weighted less than edges leading towards the seed nodes. Consequently, the convergence probabilities of the nodes capture both (a) the separations between the seeds and the graph nodes and (b) the connectivity of the nodes in the graph relative to nodes in S.

An alternative approach for contextualizing PageRank scores is to use the PPR techniques [11, 23] discussed in the introduction. One key advantage of this teleportation vector modification based approach over modifying the transition matrix, as in [19], is that the term α can be used to directly control the degree of seeding (or personalization) of the PPR score. In fact, these personalized random-walk and PageRank based measures of node significance have been shown to be highly effective in many prediction and recommendation applications. [24, 82] rely on a random walk hitting time based approach, where the hitting time is defined as the expected number of steps a random walk from the source vertex to the destination vertex will

take.

2.2 Improvements to the PageRank Function

2.2.1 Approximate Personalized PageRank

Naive personalized PageRank requires a lot of time to compute node proximity when the size of graphs is too large, so there have been some efforts to separate the matrix into sparse matrices and dense matrices and then, use different kind of matrix decomposition techniques to compute approximate PPR scores efficiently. The FastRWR algorithm, presented in [106], partitions the graph into subgraphs and indexes partial intermediary solutions. Given a seed node set S then relevant intermediary solutions are combined to quickly solve for approximate PPR scores. Fujiwara et al. [40] permuted the adjacency matrix for sparse matrix, computed the QR decomposition of the matrix in preprocessing, and finally get the node proximity for a single node. Similarly, Shin et al. [97] proposed a method to compute PPR scores by reordering and partitioning the matrix and using block elimination with LU decomposition. The algorithm optimized the computation time with sparsity patterns of the matrices and make the matrix as sparse as possible with reordering. [79] proposed GMRES based algorithm to compute the PPR scores with least number of iterations by exploiting graph structure. They compute core-tree-decomposition, partition into blocks, and compute LU decomposition of the graph in pre-processing step. On the query step, they compute the PPR scores of blocks with pre-conditioned GMRES.

Another way to compute PPR score efficiently is to use sampling technique such as Monte Carlo method. [7, 37] introduced a Monte Carlo End-Point algorithm to compute approximated values of personalized PageRank scores that achieves full personalization by pre-computation of simulated random walks. [9] showed a Person-

alized PageRank algorithm for the incremental graphs with Monte Carlo methods. They pre-compute and store a small number of random walks starting from each node and then fetch the all walk segments for a seed node in query time. In [77], they proposed a distributed Personalized PageRank computation with Monte Calro Full Path algorithm. They pre-process fingerprint that is the approximate PPR vectors for each node and then, in query time, the ranking scores are returned by a linear combination of related fingerprints.

2.2.2 Partial Matrix Computation for Personalized PageRank

Since, in practice, personalized PageRank use only small parts of graph to compute node proximity, [54, 42, 64] used partial information of the graph for PPR computations. Jeh and Widom[54] proposed a procedure using partial vectors and a hubs skeleton. Instead of using and computing the entire web matrix, they constructed a hubs skeleton and hub vectors as partial vectors with identifying interrelationships between vectors and compute PPR scores using these vectors. It reduced the complexity of matrix computation using partial information instead of the entire graph information. Gleich and Polito[42] also used partial information of the graph to compute PPR scores. Given a graph, they divide the nodes into two sets which are active nodes and inactive nodes. Only the set of active nodes is expanded including more nodes that have higher probabilities to have high PPR scores. After the expansion, they computed an approximate PPR scores only using the set of active nodes and their outgoing edges. In [64], they reduced the cost of PPR computation focusing on the neighborhoods of the seed nodes. Instead of the entire graph, they consider the localized graphs which are a subset of nodes and edges that are close to the seeds and an external node that is a set other nodes which are outside of localized graphs and compute approximate PPR scores with small amount of matrix computations

effectively.

2.3 PageRank Optimization

Due to the obvious relationship between ranking and monetary rewards (e.g. through selling of advertisements on web search applications), there has been considerable effort in engineering (or manipulating) graphs in a way to maximize ranking scores of particular nodes. This is commonly referred to as PageRank optimization. One way to achieve this goal is carefully adding or removing certain links: If, for example, one or more colluding web masters can add or remove edges, PageRank scores of target web pages or domain can be increased in the set of domains by cooperating reinforcement learning [6, 88]. [81] established several bounds indicating to what extent the rank of the pages of a website can be changed and the authors derived an optimal referencing strategy to boost PageRank scores. PageRank can be maximized and optimized when Websters can select some edges from a set of edges that are under their control [30, 32].

A related, but opposite, problem is to protect the PageRank scores against negative links (which may indicate, for example, negative influence or distrust in a social network), artificial manipulation, and spam pages. Finding and accounting for negative links [101] and locating and eliminating noisy links can help improve PageRank scores[107]. [12], for example, focused on identifying spam pages and link farms and showed that better PageRank scores can be obtained after filtering spam pages and links. [36] proposed a link spam detection and PageRank demotion algorithm called MaxRank whose ranking of a page takes into account the frequency of visit to the page by a random surfer, but minimizing an average cost per time unit. In [86], to recover link spamming susceptibility, authors showed a refined PageRank with the intuition of exploiting web's decomposability and its hierarchical nature. Eiron et

al. [34] proposed to improve the web page ranking by modifying the PageRank algorithm in a way that penalizes pages based on having fallen out of maintenance. [117] proposed an improvement on the PageRank function to take into account out-degrees of the nodes that are nearby. In particular, the authors proposed to compute a new transition matrix that takes into account out-degrees of those nodes that are within k steps – intuitively, the higher the k-hop out-degree from a given node, the better the node is in terms of being able to serve as a hub from which the user can reach to more information.

2.4 Influential Node Identification

2.4.1 Information Flows within Networks and Influential Node Identification

During the past few years, there has been growing interest in the analysis of information flow within networks. Adar et al. [1], for example, consider information propagation across blog entries. Choudry et al. [31] investigate the impact of data sampling strategies on the analysis of social networks for information diffusion. Other related work in influence diffusion models include [5, 10]. Our work is orthogonal, but nevertheless related, to work that aims locating influencers of a given network. Kempe et al. were one of the first teams who have investigated the problem of optimizing the network for maximum influence spread [58, 59]. Watts and Dodds also studied the conditions under which nodes in a network become influential [108]. [26] proposed a heuristic algorithm, based on local influence regions, to identify nodes in a social network that maximize the spread of influence. Shakarian et al. [96] focused on learning diffusion models and studying the impact of one node on the others in the networks. The authors focus on reasoning with previously learned diffusion models, expressed via generalized annotated programs. More specifically, [96] deals with social network

optimization problems in which -given a goal to achieve (e.g. minimizing the spread of a disease) and given limited resources (e.g., medications)- "key nodes" at which the resources should be allocated are identified. [73] focuses on the related problem of optimal sensor placement to observe information cascades within the network, including disease outbreaks in a population information flow within the blogosphere. In the social network, the information propagation and influential node identification has been main issues on predicting the links [111], understanding the phenomenon [43], and discovering how the internal and external influence of the information can reach a node [83].

2.4.2 Network Distance: Definitions and Indexing

The simplest way to define the distance between a pair of nodes in a network is in terms of the smallest number of edges or hops that separate the nodes. [90, 14], for example, propose algorithms for (approximately) identifying the number of nodes reachable within a given number of hops from a source node. [27, 110, 115] propose algorithms for indexing shortest paths to speed up the hop-distance computation. In [19], Candan and Li presented a random walks based proximity measure for mining associations between nodes in a graph. [11, 23] and others used personalized page rank values to measure proximity in graphs. Work in this direction also includes [104] which uses a connection subgraph constructed by giving each node a goodness score with respect to the query nodes by using random walks with restarts. Similarly, [105] proposes several algorithms based on random walks. [99] also presents methods that approximate a family of proximity measures relying on random-walk techniques. [24] proposed a random walk hitting time based definition of network distance, where the hitting time is defined as the expected number of steps a random walk from the source vertex to the destination vertex will take. [82] also relies on a hitting time based

definition of network distances for query suggestion. [38, 95] proposed algorithms for efficiently computing hitting times; in particular [95] proposes approximate algorithms to tackle the exact hitting time problem.

2.5 Graphs with Uncertainty

Uncertain graphs are commonly used in many applications. For example, in biological networks for the protein interaction, where proteins are represented as nodes and the interactions between them as edges, uncertainty may be introduced when the existence of certain interactions are often only statistically probable [60, 75]. In communication networks, possibility of link failure needs to be accounted for in finding stable and reliable paths for packet delivery with minimum cost: this involves taking into account several forms of uncertainty, including existence uncertainty, ambiguity, and confusion on edges [28].

In web-based applications, such as social networks, uncertainties may exist due to inherent lack of prior knowledge regarding the existence of friendship or influence flow among the users in the underlying network [61] and it may be critical to take into account such forms of uncertainty in predicting which nodes are likely to be connected to which other nodes [76]. Other graph analysis operations that are affected from graph uncertainty include shortest paths, reachability analysis, and subgraph searching. A common challenge in all of these is that, in the presence of uncertainty, the complexity of (already expensive) graph operations becomes more expensive. [22] presented an interval labeled edge model and discussed efficient computation of minimum paths and trees on such uncertain graphs without having to enumerate all possible worlds. [93] and [116] also focused on shortest paths, but on graphs where edges have probabilistic interpretations for existence in uncertain graphs. Given edges that are accompanied with the probability of existence, [55, 60, 75, 118] propose ways to com-

pute reliability and reachability efficiently through Monte-Carlo sampling. [116, 121] proposed pruning techniques to reduce the complexity of subgraph searching and subgraph pattern mining in uncertain graphs by avoiding enumeration of all possible worlds of the uncertain graph.

Several works considered the problem of ranking on graphs with different forms of uncertainties. [51] considered PageRank when web graphs contain erroneous link information and proposed an approximate solution using interval matrices – the proposed approach captures the PageRank scores of the nodes affected by fragile links in terms of lower and upper bounds of PageRank values. A different node-centric uncertain graph model and node ranking approach are presented in [87]: in particular, [87] collapses the uncertain parts of a graph into a cloud graph, where the end of every undetected link is connected to this cloud graph and computes PageRank scores on this transformed graph. [33] considered uncertain graphs, where edges are annotated with existence probabilities and extended the SimRank measure [53] under probabilistic interpretations of edge existence and transition matrices.

Chapter 3

LOCALITY-SENSITIVE, RE-USE PROMOTING, APPROXIMATE PERSONALIZED PAGERANK COMPUTATION

3.1 Introduction

In many graph applications, how a given pair of nodes on a graph are related to each other is determined by the underlying graph topology. Node distance/proximity measures are commonly used for quantifying how nearby or otherwise related to two or more nodes on a graph are. Random-walk based definitions, such as personalized PageRank (PPR) score [11, 23, 54, 104, 105], of node relatedness take into account the density of the edges: intuitively, a node can be said to be more related to another node if there are short paths between them and two nodes are tightly connected and argue that nodes that have many paths between them can be considered more related. Naturally, any distance measure which would require all paths among two nodes to be enumerated would require time exponential in the size of the graph and, thus, would be intractable. 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 HITS [69] and PageRank [17], rely on random-walks to identify significant nodes in the graph.

Unfortunately, for large data sets, random-walk processes are prohibitively expensive. Recent advances on personalized PageRank includes top-k and approximate personalized PageRank algorithms [6, 9, 37, 23, 39, 46, 106, 99] and parallelized implementations on MapReduce or Pregel based batch data processing systems [9, 80].

The FastRWR algorithm, for example, separates a graph into two subgraphs based on inter-edges and intra-edges of clusters and save partial intermediary solutions. For intra-edge subgraphs which are relatively dense, it precomputes the inverse of them and for inter-edge subgraphs, it applies low rank approximation with matrix decomposition. Given a seed node set S then relevant intermediary solutions are combined to quickly solve for approximate PPR scores. Naturally, there is a trade-off between the number of partitions created for the input graph G and the accuracy: the higher the number of partitions, the faster the run-time execution (and smaller the memory requirement), but higher the drop in accuracy. Unfortunately, as we see in Section 3.5.1, 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. To tackle this problem, I propose an efficient Locality-sensitive, Re-use promoting, approximate personalized PageRank (LR-PPR) algorithm which is approximate but efficient in the execution time and accurate close to naive PPR computation.

In the following chapter, I first formally introduce the problem and then present our solution for locality-sensitive, re-use promoting, approximate personalized PageRank computations. In Section 3.4, we discuss optimization and parallelization opportunities. We evaluate LR-PPR for different data sets and under different scenarios in Section 3.5.

3.2 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. Let us be given a set $S \subseteq V$ of seed nodes 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[1] 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:

Desideratum 1: Locality-Sensitivity. Our goal is to compute an approximate PPR vector, $\vec{\phi}_{apx}$, <u>using $\mathcal{G}_S$ instead of G</u>, 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:

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) <u>should not need to be re-performed</u> when processing $G_i \in \mathcal{G}_{S_2}$, when computing $\vec{\phi}_{apx,2}$ using $\mathcal{G}_{S_2}$.

3.2.1 Combined Locality and its Boundary

Unlike existing approximate PPR algorithms [8, 9, 37, 23, 46, 106, 99], LR-PPR is location sensitive. Therefore, given the set, S, of seed nodes and the corresponding localities, $\mathcal{G}_S$, the computation focuses on the combined locality $G^+(V^+, E^+) \subseteq G$,

[1] We discuss alternative ways to select these in Section 3.4.

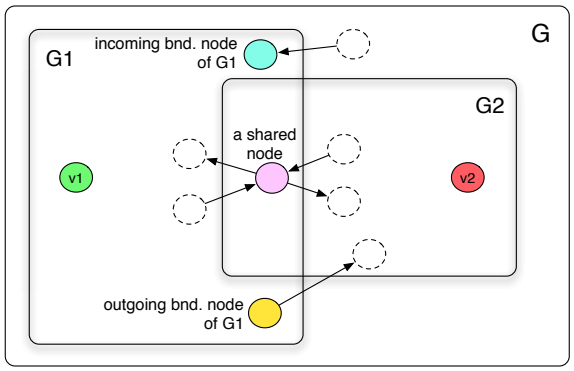


Figure 3.1: Incoming and Outgoing Boundary Nodes/Edges and a Node Shared between Two Localities

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 3.1, 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 $E_{outbound,l} \cap E_l = \varnothing$.

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 $v_i \notin V_l$.

3.2.2 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

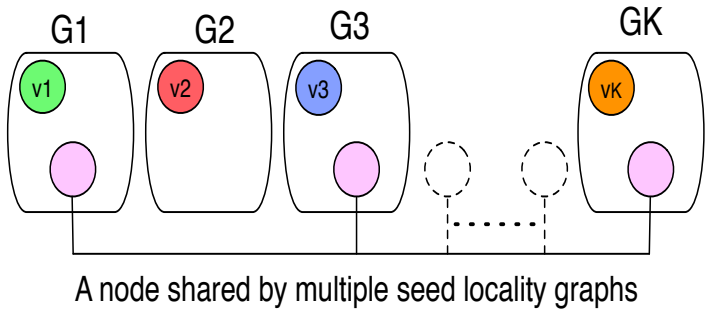


Figure 3.2: An Equivalence Set Consists of the Copies of a Node Shared across Multiple Seed Locality Graphs

nevertheless take the external graph, G^-, and the boundaries between G^- and G^+, into account.

Transition Matrices of Individual Localities

Let $v_{(l,i)}$ $(1 \leq l \leq K)$ 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 3.2). 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

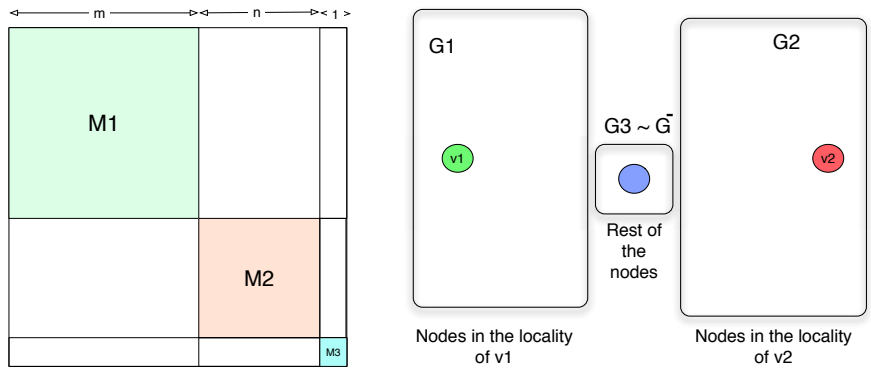
- $\left(\nexists e_{i,j} = [v_{(l,i)} \rightarrow v_{(l,j)}] \in E_l\right) \rightarrow \mathbf{M}_l[j,i] = 0$ and
- $\left(\exists e_{i,j} = [v_{(l,i)} \rightarrow v_{(l,j)}] \in E_l\right) \rightarrow \mathbf{M}_l[j,i] = \frac{1}{out(v_{(l,i)})}$, where $out(v_{(l,i)})$ is the number of outgoing edges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

The $m \times m$ matrix $\mathbf{M}_2$ is also defined similarly considering edges in E_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

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



(a) $\mathbf{M}_{bd}$ matrix (b) corresponding graph

Figure 3.3: The Matrix, $\mathbf{M}_{Bd}$ and the Corresponding Graph

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 (Figure 3.3). All the external nodes in G^- are accounted by a single node represented by the 1×1 matrix $\mathbf{M}_{K+1}$.

As we see later in Section 3.3, 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.

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 $\|\mathcal{V}_c\| > 1$

- $\mathbf{M}_0[ind(l,j), ind(l,i)] = -(\frac{1}{out(G_l, v_{l,i})} - \frac{\|\mathcal{V}_c\|-1}{\|\mathcal{V}_c\|} \times \frac{1}{out(G, v_{l,i})})$ 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 and $out(G_l,v)$ is the outdegree of node v in the subgraph G_l. 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

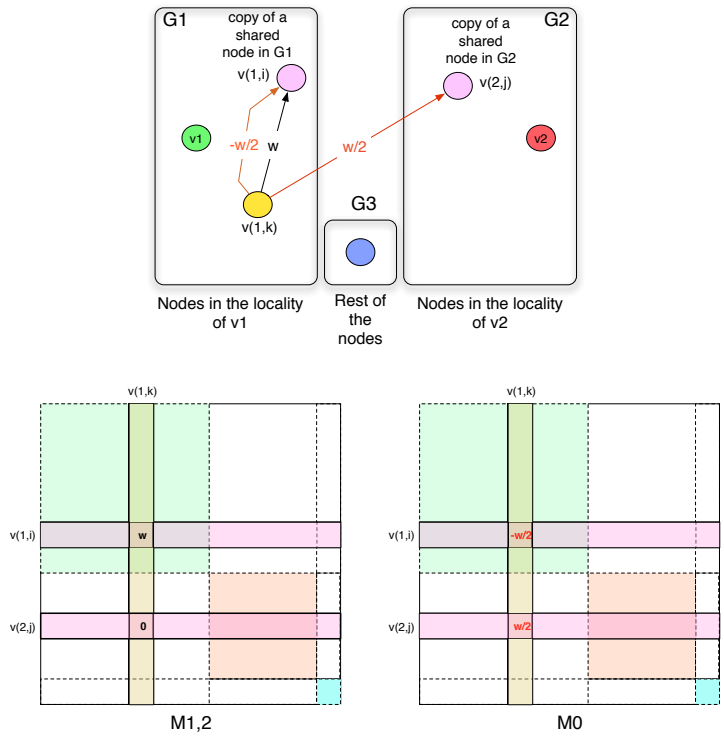


Figure 3.4: Accounting for Shared Nodes in the Compensation Matrix, $\mathbf{M}_0$

in other seed localities (Figure 3.4). 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: $\forall[v_{(l,i)} \rightarrow v_{(h,j)}] \in E_{outbound,l}$
  - $\mathbf{M}_0[ind(h,j), ind(l,i)] = \frac{1}{out(v_{(l,i)})}$ and
  - $\mathbf{M}_0[ind(l,p), ind(l,i)] = -(\frac{1}{out(G_k, v_{l,i})} - \frac{1}{out(G, v_{(l,i)})})$, where $\exists e_{i,p} = [v_{(l,i)} \rightarrow v_{(l,p)}] \in E_l$ and $v_{l,p}$ is not a member of the equivalence set $\mathcal{V}_c$ for any $v_c \in V$
- Accounting for boundary edges from nodes in V_l to graph nodes that are in V^-:

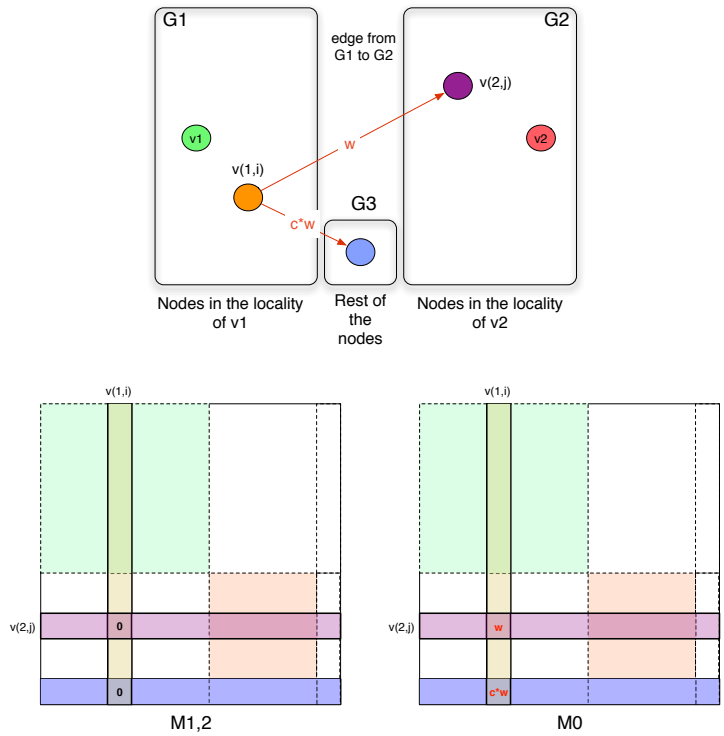


Figure 3.5: Accounting for the Edges That Are Outgoing from a Locality

if $\exists[v_{(l,i)} \rightarrow v] \in E_{outbound,l}$ 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 process of compensating for outgoing boundary edges for a sample case when $K = 2$ is visualized in Figure 3.5. 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

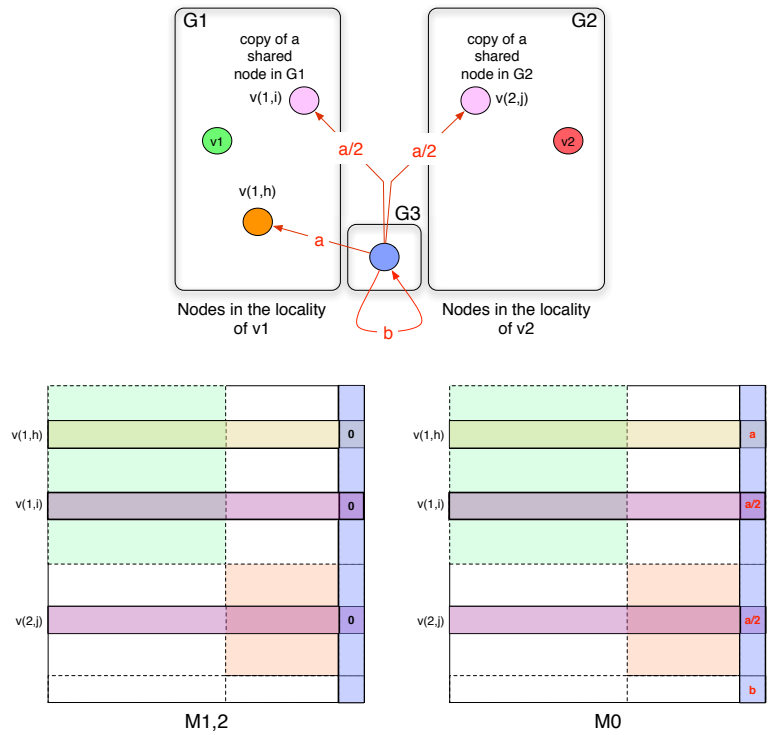


Figure 3.6: Accounting for the Edges That Are Originating from The nodes That Are Outside of the Localities of G_1 and G_2

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 [114], 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. The process of compensating for incoming edges originating from outside of the locality graphs of the seeds is visualized in Figure 3.6.

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. The process of compensating for edges that are outside of the seed localities is also visualized in Figure 3.6.

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

3.2.3 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.

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 3.5, this leads to highly accurate PPR scores with better scalability than existing techniques.

3.2.4 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. Let us restate the problem: Given the block-diagonal local transition matrix, $\mathbf{M}_{bd}$, and the compensation matrix, $\mathbf{M}_0$ (that together make up

the overall transition matrix, $\mathbf{M}_{axp}$) computed as described above, and a re-seeding (or restart) probability, β, we seek to find $\vec{\phi}_{apx}$, where

$$\vec{\phi}_{apx} = \beta(\mathbf{I} - (1-\beta)\mathbf{M}_{axp})^{-1}\vec{s},$$

where $\vec{s}$ is the re-seeding vector for seeds. 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 [106], which itself relies on the Sherman-Morisson lemma [92]:

> Let $\mathbf{C} = \mathbf{A} + \mathsf{USV}$. 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 $\mathsf{USV} = \mathbf{M}_0$. 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.

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 $1_{1\times 1}$):

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

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 [106]:

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

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 3.5, 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.

Error Analysis of LR-PPR

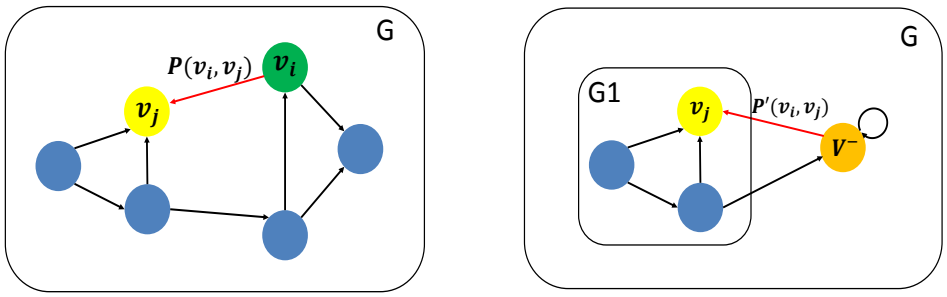
In this section, we describe and analyze the two kind of possible errors for LR-PPR.

Error with the external graph G^-: The fist error can be generated by the compensation part on the incoming boundary edge from the external graph G^-. When we combined all outside nodes from locality graphs into a graph G^- with a node $\mathbf{V}^-$, the compensation for incoming boundary edges from $\mathbf{V}^-$ to $\mathbf{V}^+$ could include some errors. As Figure 3.7(a), in the original transition matrix, the probability $\mathbf{P}(\mathbf{v_i}, \mathbf{v_j})$ from v_i to from v_j is $\frac{1}{outdeg(v_i)}$. With no error, in Figure 3.7(b), we expect that $\mathbf{P'}(\mathbf{v_i}, \mathbf{v_j})$, the probability from v_i which is included in $\mathbf{V}^-$ to v_j would be same as $\mathbf{P}(\mathbf{v_i}, \mathbf{v_j})$. For the ideal case, to get the same probability as $\mathbf{P}(\mathbf{v_i}, \mathbf{v_j})$, the probability $\mathbf{P'}(\mathbf{v_i}, \mathbf{v_j})$ can be computed as

$$\mathbf{P'_{ideal}}(\mathbf{v_i}, \mathbf{v_j}) = \frac{P(v_i)}{\sum_{v \in V^-} P(v)} \times \frac{1}{outdeg(v_i)},$$

where $P(v_i)$ is the probability that the current node on the random walk is v_i among all nodes that are included in $\mathbf{V}^-$. The problem of $\mathbf{P_{ideal}}(\mathbf{V}^-, \mathbf{v_j})$ is that we do not know the probability $P(v_i)$. Instead of using $P(v_i)$, in LR-PPR, we assume that $P(v_i)$ follows the normal distribution and then, $\mathbf{P}(\mathbf{v_i}, \mathbf{v_j})$ is computed as

$$\mathbf{P'}(\mathbf{v_i}, \mathbf{v_j}) = \frac{1}{|V|} \times \frac{1}{outdeg(v_i)}.$$



(a) original graph (b) locality graph with an external node

Figure 3.7: (a) The Probability from Node v_i to Node v_j in the original Graph **G** and (b) Locality Graph G^+ with an external Node G^-

Therefore, the error ϵ between PPR and LR-PPR is defined as $\epsilon = \mathbf{T} - \mathbf{M_a px}$ where T is a transition matrix of G and $M_a px$ is a transition matrix of LR-PPR. Note that ϵ is only related to the incoming boundary edges.

This error is described and shown in [114] but their analysis is not tight enough. They show the $\mathbf{L_1}$ distance between the ideal PageRank and approximated PageRank scores, but the problem is that $\mathbf{L_1}$ distance only shows the average of each score distance though there exists a critical error on a node's score compared other nodes' scores.

We provide more precise and tight bound on the errors The linear equation of Power iteration for PageRank can be written

$$A_0 v_{0i} = \lambda_{0i} v_{0i},$$

where A_0 is a transition matrix of a graph, λ_{0i} is the eigenvalues, and v_{0i} is the eigenvectors for $i = 1, ..., N$. Now suppose that we have a changed matrix and find the eigenvalues and eigenvectors of

$$A v_i = \lambda_i v_i,$$

where $A = A_0 + \delta A$ with the perturbations δA is much smaller than A. The new eigenvalues and eigenvectors can be defined as $\lambda_i = \lambda_{0i} + \delta\lambda_i$ and $v_i = v_{0i} + \delta v_i$. At first,

we assume that we have scaled the eigenvectors such that

$$v_{0j}^{\top} v_{0i} = \delta_{ij},$$

where δ_{ij} is the Kronecker delta. For $Av_i = \lambda_i v_i$, after substituting and expanding, we can get

$$A_0 v_{0i} + \delta A v_{0i} + A_0 \delta v_i + \delta A \delta v_i = \lambda_{0i} v_{0i} + \lambda_{0i} \delta v_i + \delta \lambda_i v_{0i} + \delta \lambda_i \delta v_i.$$

After canceling from $A_0 v_{0i} = \lambda_{0i} v_{0i}$, it becomes

$$\delta A v_{0i} + A_0 \delta v_i + \delta A \delta v_i = \lambda_{0i} \delta v_i + \delta \lambda_i v_{0i} + \delta \lambda_i \delta v_i.$$

For $\delta \lambda_i \delta v_i$, we remove the higher-order terms, and it is simplified as

$$\delta A v_{0i} + A_0 \delta v_i + \delta A \delta v_i = \lambda_{0i} \delta v_i + \delta \lambda_i v_{0i}.$$

As the basis for the perturbed eigenvectors, we construct $\delta v_i = \sum_{j=1}^{N} \omega_{ij} v_{0j}$ where ω_{ij} are small constants that are to be determined and substitute it to

$$\delta A v_{0i} + A_0 \sum_{j=1}^{N} \omega_{ij} v_{0j} + \delta A \delta v_i = \lambda_{0i} \sum_{j=1}^{N} \omega_{ij} v_{0j} + \delta \lambda_i v_{0i}.$$

We can remove the summations by left multiplying by $v_{0i}^{\top}$ using $v_{0j}^{\top} v_{0i} = \delta_{ij}$,

$$v_{0i}^{\top} \delta A v_{0i} + v_{0i}^{\top} A_0 \omega_{ii} v_{0i} + v_{0i}^{\top} \delta A \delta v_i = v_{0i}^{\top} \lambda_{0i} \omega_{ii} v_{0i} + v_{0i}^{\top} \delta \lambda_i v_{0i}.$$

We know that $v_{0i}^{\top} A_0 v_{0i} = v_{0i}^{\top} \lambda_{0i} v_{0i}$, so it leaves

$$v_{0i}^{\top} \delta A v_{0i} + v_{0i}^{\top} \delta A \delta v_i = v_{0i}^{\top} \delta \lambda_i v_{0i}.$$

Rearranging by $\delta \lambda_i$ with $v_{0i}^{\top} v_{0i} = 1$, we can get

$$\delta \lambda_i = v_{0i}^{\top} (\delta A v_{0i} + \delta A \delta v_i).$$

We also can calculate ω_{ia} by left-multiplying $\delta A v_{0i} + A_0 \sum_{j=1}^{N} \omega_{ij} v_{0j} + \delta A \delta v_i = \lambda_{0i} \sum_{j=1}^{N} \omega_{ij} v_{0j} + \delta\lambda_i v_{0i}$ by v_{0a}.

$$\omega_{ia} = \frac{v_{0a}^{\top} \delta A (v_{0i} + \delta v_i)}{\lambda_{0i} - \lambda_{0a}}, \quad i \neq a$$

By replace a to j, we can get

$$\omega_{ij} = \frac{v_{0j}^{\top} \delta A (v_{0i} + \delta v_i)}{\lambda_{0i} - \lambda_{0j}}, \quad i \neq j$$

Because $v_i = v_{0i} + \delta v_i$,

$$\omega_{ij} = \frac{v_{0j}^{\top} \delta A v_i}{\lambda_{0i} - \lambda_{0j}}$$

and after applying to $\delta v_i = \sum_{j=1}^{N} \omega_{ij} v_{0j}$, we can get

$$\delta v_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{v_{0j}^{\top} \delta A v_i}{\lambda_{0i} - \lambda_{0j}} v_{0j}.$$

Because v_i is ordered such that $\lambda_1 = 1$, v_1 can be written

$$v_1 = \sum_{j=2}^{N} \frac{v_{0j}^{\top} \delta A v_1 v_{0j}}{1 - \lambda_{0j}}.$$

Note that δA is related to the error edges incoming from the external node and located on the rightmost in the transition matrix $(N \times N)$ and v_i is a eigenvector $(N \times 1)$. This means that the only last index value of v_i which is an external node probability is multiplied to the rightmost values, so $\delta A v_i$ become a scalar.

Therefore, We define a function of the incoming boundary edges δv_1 as

$$\delta v_1 = \sum_{j=2}^{N} \left(\frac{prob(V^-)}{1 - \lambda_{0j}}\right)\left(v_{0j}^{\top} \epsilon_{incoming} v_{0j}\right),$$

where $prob(V^-)$ is a transition probability among nodes which both of them are in the external node and $\epsilon_{incoming}$ is the incoming boundary edge probability errors. On this function, there are some considerable error dependencies.

- If the error of $prob(V^-)$ is small and is not significant, the overall error becomes small.

- If the incoming boundary edge probability is accurate with small $\epsilon_{incoming}$ (e.g. $P'_{ideal}(v_i, v_j) - P'(v_i, v_j)$), the overall error becomes small.

- The overall error depends on the separation of eigenvectors. If all eigenvectors which are λ_{02}, ..., and λ_{0N} are small, the overall error becomes small.

Error with low rank approximation: The error can be generated by the approximation using low rank approximation. In [106], the error bound of the approximation is described as

$$\|\vec{r} - \hat{\vec{r}}\| = \beta \sum_{i=t+1}^{\|V\|} \frac{1}{(1 - (1 - \beta)\lambda_i)},$$

where $\vec{r}$ and $\hat{\vec{r}}$ be the ranking vectors, t is the rank of the low-rank approximation, and λ_i is the ith largest eigenvalues of the transition matrix, T_G. Note that this error bound equation also depends on the separation of eigenvectors. Though they showed the error bound of the low-rank approximation for eigenvalue decomposition, it is hard to find the error boundary for the general number of partition. The error bound for the approximation with the graph partition can be defined with some extreme cases of the number of partition. In LR-PPR, M_{bd} part is accurate and the only error part is related to the low-rank approximation on the compensation matrix M_0 with graph partition.

For the upper bound error, when each node has a separate partition and the low-rank approximation is applied, the error rate on LR-PPR is high because the compensation matrix M_0 becomes an original matrix and all transition values are approximated.

For the lower bound error, when most of edges are in the locality graphs, the compensation matrix M_0 becomes very sparse. and Intuitively, the less number of locality graphs goes to low-rank approximation and the error become less.

3.3 Complexity and Re-use

We can divide the work underlying the LR-PPR algorithm into five sub-tasks, each processed using only local nodes and edges:

Sub-task 1. The preparatory step in which the localities of the seeds are identified. The computational cost of this depends on the definition of locality. But, in general, the cost of this is linear in the size of the network G^+; i.e., $O(\|G^+\|)$, where $\|G^+\| \ll \|G\|$. Note that the work in this sub-task is entirely re-usable.

Next, the combined local transition matrix, $\mathbf{M}_{bd}$, and the compensation matrix, $\mathbf{M}_0$, are computed:

Sub-task 2a. Assuming a sparse matrix representation, computation and storage of the combined local transition matrix, $\mathbf{M}_{bd}$, takes $O(\sum_{1 \leq l \leq K} \|G_l\|)$ time and space. Note that (while the matrix $\mathbf{M}_{bd}$ is not re-usable, unless the same set of seeds are provided) the constituting matrices $\mathbf{M}_1$ through $\mathbf{M}_K$ are re-usable.

Sub-task 2b. With a sparse representation, computation and storage of the compensation matrix takes $O(K \times max\_in\_degree \times \|V\| + (\|E\| - \sum_{1 \leq l \leq K} \|E_l\|))$ time and space:

1. Row/column indexing: This takes $O(\sum_{1 \leq l \leq K} \|V_l\|)$ time.

2. Identification of common nodes (i.e., equivalence classes): To locate the common nodes and to create the equivalence classes, we need to go over each node once and see if the node occurs in which of the remaining $K - 1$ localities. Thus, assuming a hash-based implementation, this step takes $O(\sum_{1 \leq l \leq K} \|V_l\|)$ to identify the equivalence classes.

3. Identification of outgoing boundary edges: In order to identify the outgoing boundary edges, we go over the nodes in V_1 through V_K and check if their

outgoing edges are to a node within the same locality or not. If not, we check whether it is to a node within V^+ or not; if it is to a node in V^+, then the edge is labeled as an outgoing boundary edge among localities, otherwise, it is labeled as an outgoing boundary edge to G^-. Assuming that the nodes are labeled with their equivalence classes in the previous step, the cost of this operation is $O(\sum_{1 \leq l \leq K} \sum_{v \in V_l} out(v))$.

Note that, while the sub-task as a whole is not re-usable when the seed set changes, the part of the work involving identification of the outgoing edges from an individual locality is re-usable.

4. Identification of incoming boundary edges from G^-: In order to identify the incoming boundary edges from G^-, we go over the nodes in V_1 through V_K and check if their incoming edges are from a node marked with an equivalence class label. If not, the edge is from a node in G^-. The cost of this operation is $O(\sum_{1 \leq l \leq K} \sum_{v \in V_l} in(v))$.

   Note that, while the sub-task as a whole is not re-usable when the seed set changes, the part of the work involving identification of the incoming edges into a single individual locality from nodes outside of the locality is re-usable.

5. Compensation for the common nodes: Once the $\|V^+\|$ equivalence classes are identified, the edges in the localities' incoming edges need to be rerouted (at most K times), leading to $O(K \times \sum_{1 \leq l \leq K} \|E_l\|)$ time cost in the worst case.

6. Compensation for the outgoing boundary edges: This step involves considering once each outgoing boundary edge. Since all necessary information can be collected during the earlier identification pass (Subtask 2b. 3), the worst case time complexity of this operation is the same as that of the corresponding

identification step.

7. Compensation for the incoming boundary edges: This step involves considering once each incoming boundary edge identified earlier. For each vertex, v, with one or more incoming edges, we create an edge whose weight captures the out-degrees of all corresponding external source nodes. Assuming that all nodes in the graph have been annotated with their out-degrees during a pre-processing step, the worst-case time complexity is the same as that of the corresponding identification step (Subtask 2b. 4).

8. Compensation for the edges in G^-: In the first look, it appears that this step cannot be executed without considering all nodes in V^-. However, this is not true: First of all, assuming that we know $\|V\|$, we can compute $\|V^-\|$ using $\|V\|$ and $\|V^+\|$. Secondly, the term $\sum_{v \in V^-} out(G^-, v)/out(G, v)$ can be rewritten as

$$\sum_{v \in V} \frac{out(G,v)}{out(G,v)} - \sum_{v \in V^+} \frac{out(G^+,v)}{out(G,v)} - \sum_{\langle v \rightarrow v_j \rangle \in (inbound(G^+) \cup outbound(G^+))} \frac{1}{out(G,v)},$$

where $inbound(G^+)$ and $outbound(G^+)$ are the incoming and outgoing edges to G^+, both of which have been computed in earlier steps. Also, the first term is simply $\|V\|$. Thus, this step can be computed using only local information, in worst-case time complexity the same as the identification steps (Subtask 2b. 3 and Subtask 2b. 4).

Sub-task 3. Next, the $\mathbf{Q}_{bd}^{-1}$ matrix is obtained. The execution cost of this step is $O(\sum_{1 \leq l \leq K} matrix\_inversion\_cost(\mathbf{M}_l))$. There exists a $O(n^{2.373})$ algorithm for matrix inversion [113], where $n \times n$ is the dimensions of the input matrix. Thus, we can rewrite the execution cost as $O(\sum_{1 \leq l \leq K} \|V_l\|^{2.373})$. Assuming a sparse matrix representation, we need $O(\sum_{1 \leq l \leq K} \|V_l\|^2)$ space to store the resulting matrix $\mathbf{Q}_{bd}^{-1}$. Note that the work in this sub-task is, again, entirely re-usable.

Sub-task 4. Next, the compensation matrix, $\mathbf{M}_0$ is decomposed. While exact matrix decomposition is expensive, we use highly efficient approximate low-rank (r) decomposition [106], which leverages sparsity of $\mathbf{M}_0$, where r is the selected rank.

Sub-task 5. The matrix, $\mathbf{\Lambda}$, is obtained. The matrix multiplications and inversions in this step take $O(r^{2.373} + r \times \|V^+\|^2 + r^2 \times \|V^+\|)$ time, where r is the selected rank.

Sub-task 6. Finally, $\vec{\phi}_{apx}$ of PPR scores is computed through matrix multiplications in $O(r \times \|V^+\|^2 + r^2 \times \|V^+\|)$ time.

Summary. This cost analysis points to the following advantages of the LR-PPR: First of all, computation is done using only local nodes and edges. Secondly, most of the results of the expensive sub-tasks 1, 2, and 3 can be cached and re-used. Moreover, costly matrix inversions are limited to the smaller matrices representing localities and small matrices of size $r \times r$.

It is important to note that various subtasks have complexity proportional to $\|V^+\|^2$, where $\|V^+\| = \sum_{1 \le l \le K} \|V_l\|$. While in theory the locality V_l can be arbitrarily large, in practice we select localities with a bounded number of nodes; i.e., $\forall_{1 \le l \le K}, \|V_l\| \le L$ for some $L \ll \|V\|$.

3.4 Optimizations

The LR-PPR scheme involves: (a) initialization (where localities are identified and the local transition and compensation matrices are computed); (b) local transition matrix inversion, and (c) compensation matrix decomposition, $r \times r$ matrix inversion, and PPR computation. As mentioned above, tasks for (a) and (b) are cacheable and re-usable, whereas decomposition needs to be executed in query time. In this section, we discuss various optimization and parallelization opportunities.

3.4.1 Locality Selection

In the initialization phase of the algorithm, the first task is to identify localities for the given seed nodes (if they are not already identified and cached). A locality graph consists of a set of graph nodes that are nearby or otherwise related to a seed node. Note that localities can be distance-constrained or size-constrained. Common definitions include h-hop neighborhoods [14, 27, 110, 115, 120], reachability neighborhoods [27], cluster/partition neighborhoods [35, 57, 84], or hitting distance neighborhoods [24, 82]. One straight-forward way to identify the locality of a seed node n is to perform breadth-first search around n to locate the closest L nodes in linear time to the size of the locality. Alternatively, one can use neighborhood indexing algorithms, such as `INI` [62], to identify the neighborhood of a given node in a way that captures topological characteristics (e.g., density of the edges) of the underlying graph.

3.4.2 Caching

As described above LR-PPR algorithm supports caching and re-use of some of the intermediary work. Sub-tasks 1 and 2 result 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. Sub-task 3, on the other hand, 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. Once the cache-space is full, we need to either push the cached inverted matrices into

the secondary storage or drop some existing cached results from the memory. For cache replacement, any frequency-based or predictive cache-replacement policy can be used.

3.4.3 Parallelization Opportunities

Sub-task 1, which involves identifying localities of the seeds, is highly parallelizable: each seed can be assigned to a different processing unit; and the locality search can be parallelized through graph partitioning. If being leveraged, the `INI` algorithm (which relies on hash signatures) is highly parallelizable through signature partitioning [62]. Sub-task 2, which involves construction of the local transition matrices and the compensation matrix is also parallelizable. Different localities and edges can be mapped to different servers for parallel processing. Sub-task 3, which involves matrix inversion of the local transition matrices is also parallelizable: different local matrices can be assigned to different processors; moreover, each matrix inversion itself can be parallelized [91]. Sub-task 4 involves decomposition of the compensation matrix $\mathbf{M}_0$. Since $\mathbf{M}_0$ is sparse, this step can also be parallelized effectively [45]. Finally, Sub-tasks 5 and 6 involve matrix multiplications and inversions. As discussed above, matrix inversion operation can be parallelized. Similarly, there are well-known classical algorithms for parallelizing matrix multiplication [44].

3.5 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 6.1 provides overviews of the four 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

Table 3.1: Data Sets

Data Set	Overall Graph Characteristics		Locality Graph Characteristics	
	# nodes	# edges	# nodes per neighborhood	# edges per neighborhood
Epinions	~76K	~500K	from ~200 to ~2000	from ~10K to ~75K
SlashDot	~82K	~870K	from ~700 to ~5000	from ~10K to ~75K
WikiTalk	~2.4M	~5M	from ~700 to ~6000	from ~10K to ~75K
LiveJournal	~4.8M	~69M	from ~900 to ~6000	from ~10K to ~75K

Data Set	Seeds	
	# seeds	seed distances (hops)
Epinions	2-3	3-4
SlashDot	2-3	3-4
WikiTalk	2-3	3-4
LiveJournal	2-3	3-4

seeds and the distances between the seeds (thereby varying the overlaps among seed localities). We also considered seed neighborhoods (or localities) of different sizes.

Most of Experiments were carried out using a 4-core Intel Core i5-2400, 3.10GHz, machine with 1024 KB L2 cache size, 6144 KB L3 cache size, 8GB memory, and 64-bit Windows 7 Enterprise. For some experiments that were required for the large size data set, 8-core Intel Core i7-4770, 3.40 GHz machine with 32.0 GB Memory and 1024 L2Cache and 8192 Cache size was used. Codes were executed using Matlab 7.11.0(2010b). All experiments were run 10 times and averages are reported.

3.5.1 Alternative Approaches

In this section, we consider the following approaches to PPR computation:

- **Global PPR:** This is the default approach where the entire graph is used for PPR computation. We compute the PPR scores by solving the naive PPR equation.

- **FastRWR:** This is an approximation algorithm, referred to as NB_LIN in [106]. 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. Naturally, the number of partitions impacts the execution time, query time memory usage, as well as approximation quality. As shown in Table 3.2, in our experiments, to be fair against FastRWR, we selected the number of its partitions in a way that minimizes its execution time and memory and maximizes its quality. This table shows the FastRWR performance for different data sets and configurations; the bold entries correspond to the high accuracy low time and memory configuration selected for the experiments in this section. For LiveJournal data set, even with large number of partition, the pre-computational stage could not be finished for 'out of memory.' As 8GB memory machine, 32GB memory memory machine also got the same 'out of memory' error for different number of partitions. Note that, especially 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 (8GB) and this negatively impacts accuracy.

- **GMRES-PPR:** This is a recent scalable algorithm on computing PPR scores [79].

Data Set	# part.	Time (sec.)		Top-10	Memory
		Disk I/O	In-Memory	Sp. Correl.	(MB)
Epinions	3	18.02	0.58	0.96	1547
~76K nodes	**40**	**0.22**	**0.04**	**0.97**	**178**
~500K edges	400	0.15	0.03	0.95	140
	1000	0.16	0.02	0.95	138
SlashDot	3	Out of memory in Q_1^{-1} calculation			
~82K nodes	10	0.79	0.23	0.96	616
~870K edges	**40**	**0.40**	**0.08**	**0.96**	**302**
	400	0.27	0.05	0.92	244
	1000	0.28	0.04	0.95	250
WikiTalk	3	Out of memory in Q_1^{-1} calculation			
~2.4M nodes	40	Out of memory in Q_1^{-1} calculation			
~5M edges	200	Out of memory in Q_1^{-1} calculation			
	400	24.03	17.60	0.86	1454
	1000	**16.75**	**15.15**	**0.87**	**1429**
LiveJournal	1000	Out of memory in $\hat{\Lambda}$ calculation			
~4.8M nodes	3000	Out of memory in $\hat{\Lambda}$ calculation			
~69M edges	5000	Out of memory in $\hat{\Lambda}$ calculation			

Table 3.2: FastRWR Performance for Different Data Sets and Configurations

The algorithm is a GMRES based algorithm to calculate PPR values with exploiting the structure of a graph. We compare PPR results with our proposed algorithms' results in execution time and correlation for given seeds. In this experiment, we used d=10 for the bag size.

- **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 3.2.2. Once the localized transition matrix is created, the PPR scores are computed by solving the naive PPR equation.

- **LR-PPR:** This is the locality sensitive and re-use promoting algorithm described in detail in Section 3.2.4.

In the experiments, we set the restart probability, β, to 0.15 for all approaches.

3.5.2 Evaluation Measures

We consider three key 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.

- **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

Table 3.3: Summary of Execution Time Results for Different Configurations on 10K Seed Localities

Data set	Seeds		Merged Network		Execution Time (sec.)				
	# seeds	# hops	Avg # nodes	Avg # edges	Global PPR	Fast RWR	GMRES-PPR	L-PPR	LR-PPR
Epinions	2	3	~0.7K	~17K	26.44	0.20	0.12	0.05	0.03
~76K nodes	2	4	~0.6K	~15K	28.06	0.21	0.12	0.05	0.04
~500K edges	3	3	~0.7K	~19K	30.40	0.22	0.12	0.07	0.04
	3	4	~0.8K	~20K	30.36	0.22	0.12	0.17	0.05
SlashDot	2	3	~1.3K	~15K	21.56	0.34	0.20	0.08	0.07
~82K nodes	2	4	~1.9K	~17K	21.96	0.34	0.18	0.08	0.07
~870K edges	3	3	~1.8K	~19K	22.25	0.35	0.18	0.10	0.09
	3	4	~2.5K	~25K	22.54	0.35	0.19	0.15	0.10
WikiTalk	2	3	~4.1K	~19K	677.32	17.18	0.39	0.23	0.21
~2.4M nodes	2	4	~4.8K	~20K	741.08	16.51	0.40	0.29	0.26
~5M edges	3	3	~4.4K	~24K	709;35	16.71	0.42	0.34	0.31
	3	4	~5.2K	~29K	763.10	16.61	0.41	0.37	0.21
LiveJournal	2	3	~2.0K	~19K	-	-	-	0.16	0.17
~4.8M nodes	2	4	~0.9K	~20K	-	-	-	0.24	0.22
~69M edges	3	3	~3.0K	~30K	-	-	-	0.21	0.19
	3	4	~1.0K	~30K	-	-	-	0.26	0.18

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.

3.5.3 Results and Discussions

Proposed Algorithms (L-PPR and LR-PPR) vs. FastRWR vs. GMRES-PPR

Tables from Table 3.3 to Table 3.8 presents experiment results for FastRWR, GMRES-PPR, L-PPR, and LR-PPR on different different size of locality graphs.

For the execution time, First of all, all four algorithms are much faster than

Table 3.4: Summary of Accuracy Results for Different Configurations on 10K Seed Localities

Data set	Seeds		Merged Network		Top-10% Correl. (vs. Global PPR)			
	# seeds	# hops	Avg # nodes	Avg # edges	Fast RWR	GMRES-PPR	L-PPR	LR-PPR
Epinions	2	3	~0.7K	~17K	0.954	0.826	0.990	0.988
~76K nodes	2	4	~0.6K	~15K	0.959	0.825	0.992	0.993
~500K edges	3	3	~0.7K	~19K	0.958	0.823	0.991	0.986
	3	4	~0.8K	~20K	0.958	0.823	0.987	0.985
SlashDot	2	3	~1.3K	~15K	0.921	0.810	0.984	0.958
~82K nodes	2	4	~5.7K	~125K	0.922	0.818	0.987	0.977
~870K edges	3	3	~1.8K	~19K	0.921	0.813	0.973	0.973
	3	4	~2.5K	~25K	0.921	0.818	0.982	0.974
WikiTalk	2	3	~4.1K	~19K	0.868	0.853	0.957	0.983
~2.4M nodes	2	4	~4.8K	~20K	0.871	0,854	0.994	0.984
~5M edges	3	3	~4.4K	~24K	0.866	0.852	0.986	0.988
	3	4	~5.2K	~29K	0.855	0.852	0.973	0.964

Table 3.5: Summary of Memory Usage Results for Different Configurations on 10K Seed Localities

Data set	Seeds		Merged Network		Memory usage(MB)			
	# seeds	# hops	Avg # nodes	Avg # edges	Fast RWR	GMRES-PPR	L-PPR	LR-PPR
Epinions	2	3	~2.2K	~90K			0.63	4.40
~76K nodes	2	4	~3.0K	~99K	178.3	8.55	0.71	5.69
~500K edges	3	3	~2.7K	~108K			0.96	7.30
	3	4	~3.5K	~120K			1.03	8.09
SlashDot	2	3	~5.9K	~117K			0.64	4.40
~82K nodes	2	4	~5.7K	~125K	302.1	12.16	1.43	16.66
~870K edges	3	3	~7.1K	~141K			2.08	27.92
	3	4	~7.2K	~159K			2.17	23.36
WikiTalk	2	3	~5.7K	~102K			5.66	26.74
~2.4M nodes	2	4	~5.8K	~100K	1429.0	20.97	5.51	31.44
~5M edges	3	3	~6.3K	~101K			8.82	40.46
	3	4	~6.7K	~103K			8.49	76.08
LiveJournal	2	3	~2.0K	~19K			1.70	23.55
~4.8M nodes	2	4	~0.9K	~20K	-	-	3.19	17.96
~69M edges	3	3	~3.0K	~30K			3.25	37.97
	3	4	~6.7K	~103K			3.64	22.71

Table 3.6: Summary of Execution Time Results for Different Configurations on ~75K Seed Localities

Data set	Seeds		Merged Network		Execution Time (sec.)				
	# seeds	# hops	Avg # nodes	Avg # edges	Global PPR	Fast RWR	GMRES-PPR	L-PPR	LR-PPR
Epinions	2	3	~2.2K	~90K	26.44	0.21	0.12	0.37	0.14
~76K nodes	2	4	~3.0K	~99K	27.58	0.22	0.12	0.51	0.20
~500K edges	3	3	~2.7K	~108K	27.30	0.21	0.12	0.58	0.26
	3	4	~3.5K	~120K	27.90	0.22	0.12	0.76	0.36
SlashDot	2	3	~5.9K	~117K	21.79	0.35	0.20	0.70	0.53
~82K nodes	2	4	~5.7K	~125K	21.85	0.35	0.18	0.78	0.42
~870K edges	3	3	~7.1K	~141K	21.74	0.36	0.18	1.12	0.95
	3	4	~7.2K	~159K	22.93	0.38	0.19	1.39	0.83
WikiTalk	2	3	~5.7K	~102K	681.08	16.28	0.39	0.75	0.37
~2.4M nodes	2	4	~5.8K	~100K	693.44	16.22	0.40	0.73	0.37
~5M edges	3	3	~6.3K	~101K	701.34	16.32	0.42	0.75	0.37
	3	4	~6.7K	~103K	706.26	16.34	0.41	0.78	0.36
LiveJournal	2	3	~7.9K	~144K	-	-	-	1.66	0.83
~4.8M nodes	2	4	~2.9K	~149K	-	-	-	1.06	0.32
~69M edges	3	3	~9.8K	~207K	-	-	-	3.05	1.01
	3	4	~4.8K	~213K	-	-	-	2.63	0.57

Table 3.7: Summary of Accuracy Results for Different Configurations on ~75K Seed Localities

Data set	Seeds		Merged Network		Top-10% Correl. (vs. Global PPR)			
	# seeds	# hops	Avg # nodes	Avg # edges	Fast RWR	GMRES-PPR	L-PPR	LR-PPR
Epinions	2	3	~2.2K	~90K	0.963	0.823	0.997	0.990
~76K nodes	2	4	~3.0K	~99K	0.960	0.824	0.998	0.990
~500K edges	3	3	~2.7K	~108K	0.967	0.826	0.998	0.990
	3	4	~3.5K	~120K	0.967	0.825	0.997	0.991
SlashDot	2	3	~5.9K	~117K	0.955	0.816	0.973	0.990
~82K nodes	2	4	~5.7K	~125K	0.943	0.816	0.965	0.983
~870K edges	3	3	~7.1K	~141K	0.957	0.815	0.971	0.990
	3	4	~7.2K	~159K	0.958	0.815	0.976	0.986
WikiTalk	2	3	~5.7K	~102K	0.868	0.851	0.958	0.944
~2.4M nodes	2	4	~5.8K	~100K	0.870	0.848	0.930	0.929
~5M edges	3	3	~6.3K	~101K	0.877	0.852	0.937	0.927
	3	4	~6.7K	~103K	0.869	0.851	0.976	0.967

Table 3.8: Summary of Memory Usage Results for Different Configurations on ~75K Seed Localities

Data set	Seeds		Merged Network		Memory usage(MB)			
	# seeds	# hops	Avg # nodes	Avg # edges	Fast RWR	GMRES-PPR	L-PPR	LR-PPR
Epinions	2	3	~2.2K	~90K	178.3	8.55	2.9	36.3
~76K nodes	2	4	~3.0K	~99K			3.1	55.2
~500K edges	3	3	~2.7K	~108K			4.6	57.6
	3	4	~3.5K	~120K			4.7	77.7
SlashDot	2	3	~5.9K	~117K	302.1	12.16	5.0	228.1
~82K nodes	2	4	~5.7K	~125K			4.9	172.8
~870K edges	3	3	~7.1K	~141K			7.6	325.9
	3	4	~7.2K	~159K			7.2	256.0
WikiTalk	2	3	~5.7K	~102K	1429.0	20.97	15.5	114.5
~2.4M nodes	2	4	~5.8K	~100K			16.2	120.7
~5M edges	3	3	~6.3K	~101K			24.0	211.6
	3	4	~6.7K	~103K			28.7	197.5
LiveJournal	2	3	~7.9K	~144K	-	-	10.99	322.87
~4.8M nodes	2	4	~2.9K	~149K			8.24	68.10
~69M edges	3	3	~9.8K	~207K			15.12	374.91
	3	4	~4.8K	~213K			13.48	138.25

Global PPR. As Table 3.3, Table 3.4, and Table 3.5, when the seed locality graph size is small, L-PPR and LR-PPR significantly outperform than FastRWR and GMRES-PPR. The major effect on L-PPR and LR-PPR execution time is not the entire graph size but the size of merged network. If the locality graphs are small, L-PPR and LR-PPR can be calculated very effectively. When the locality When graph size is relatively large as Table 3.6, Table 3.7, and Table 3.8, in small data sets (Epinions and Slashdot) FastRWR and GMRES-PPR work slightly faster than L-PPR and LR-PPR as expected. In large data sets (WikiTalk), however, both L-PPR and LR-PPR significantly outperform FastRWR and LR-PPR takes less time than GMRES-PPR in terms of query processing efficiency. Though WikiTalk graph size is larger than Slashdot, the WikiTalk execution time takes less than Slashdot because the number of nodes and edges in the merged network is less. On LiveJournal data set,

because of 'out of memory,' we could not get Global PPR and FastRWR results. For GMRES-PPR, the preprocessing time takes much longer than it is reported, the preprocessing could not be finished. We received the tree-decomposition source code from the author, but tree-decomposition and LU decomposition did not be completed as expected on same experimental setup as they did.

In terms of accuracy, the proposed locality sensitive techniques, L-PPR and LR-PPR, constantly outperform FastRWR and GMRES-PPR and the accuracy gap is still especially large in large data sets, such as WikiTalk. This is because, for FastRWR, it tries to approximate the whole graph, whereas the proposed algorithms focus on the relevant localities. As also discussed in Section 3.5.1, 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. Note that Figure 3.8 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 and GMRES-PPR suffers significantly when we consider larger portions of the merged locality graph.

Figure 3.9 compares in further detail the execution times, accuracies, and amounts of data read by L-PPR, LR-PPR, FastRWR, and GMRES-PPR from the cache per query as a function of the size of the merged locality network for different seeds and target locality sizes of the Epinions data set. As the figure re-confirms, L-PPR and LR-PPR provide significantly higher accuracies than other algorithms. LR-PPR needs more space than L-PPR to fetch the cached localities for reuse, but it uses this memory effectively to significantly reduce the execution time. The figure also re-confirms the execution time results presented in Table 3.3 and Table 3.6: as the figure

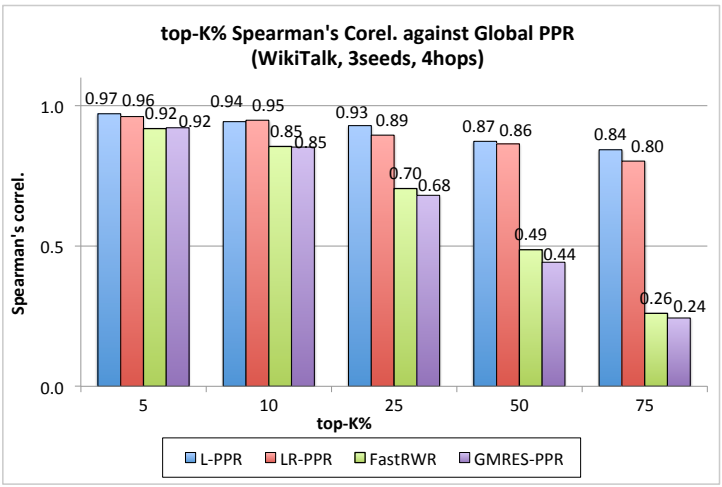
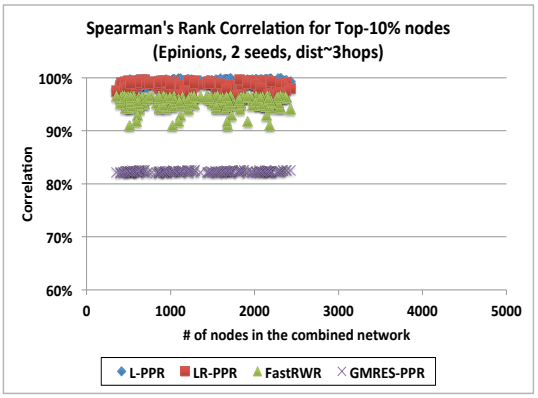


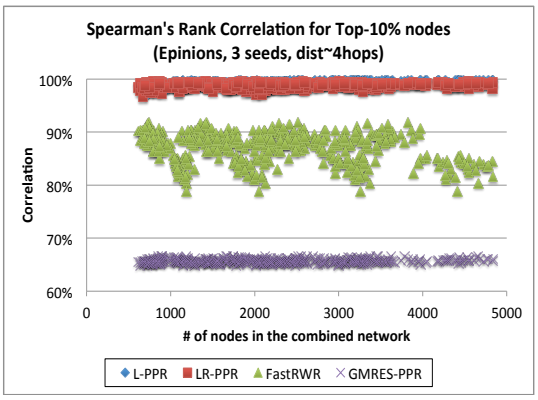
Figure 3.8: Accuracies of L-PPR, LR-PPR, FastRWR, and GMRES-PPR Against the Global PPR for Different Numbers of Target Nodes

shows, the time cost increases for all 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. In the case of the Epinions data set shown in this figure, FastRWR works slightly faster than LR-PPR for large numbers of seeds and larger neighborhoods; however, this comes with a significant loss in accuracy and also higher memory usage than L-PPR and LR-PPR. Note that, since FastRWR does not scale as well as L-PPR and LR-PPR with the overall graph size, this slight execution time advantage of FastRWR also disappears in the case of large graphs like WikiTalk (as presented in from Table 3.3 to Table 3.8 and summarized in Figure 3.10). On GMRES-PPR, it takes more compared to L-PPR and LR-PPR with small number of nodes and less with large number of nodes with small amount of memory usage. This time advantage also disappears because the accuracy is much less than L-PPR and LR-PPR.

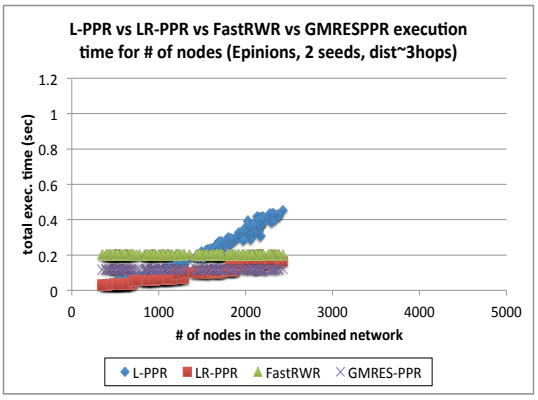
The results for the SlashDot data set (which have similar graph structure as the Epinions data set; see Table 6.1) are similar to the Epinions results and, hence, presented in the Appendix. The WikiTalk data set however has a different structure and, thus, we also present the execution times, accuracies, and amounts of data read



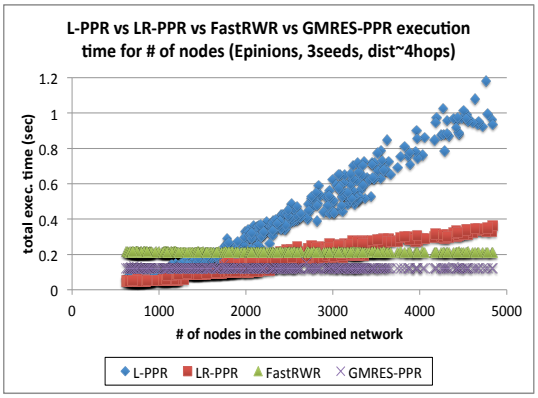
(a) Accuracies for 2 seeds, ~ 3 hops



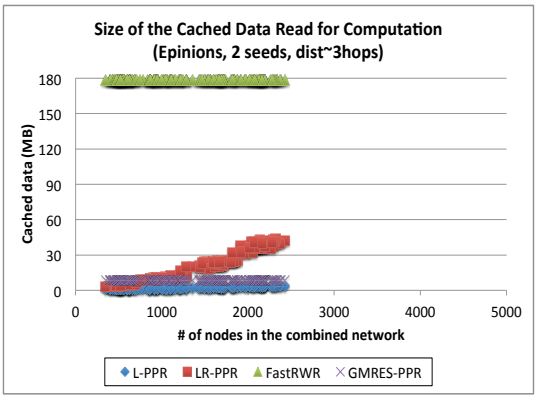
(b) Accuracies for 3 seeds, ~ 4 hops



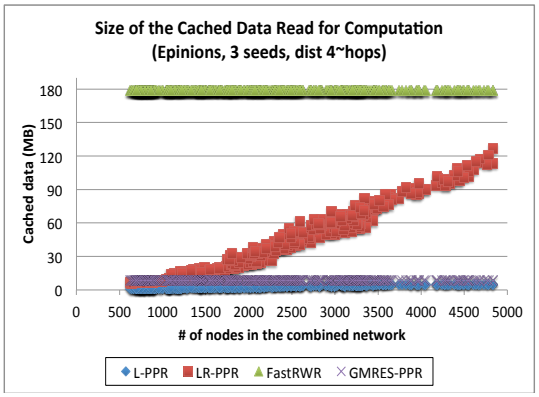
(c) Exec. times for 2 seeds, ~ 3 hops



(d) Exec. times for 3 seeds, ~ 4 hops



(e) Data for 2 seeds, ~ 3 hops



(f) Data for 3 seeds, ~ 4 hops

Figure 3.9: Performances of L-PPR, LR-PPR, FastRWR, and GMRES-ppr on the Size of the Combined Localities Network (Epinion Data Set)

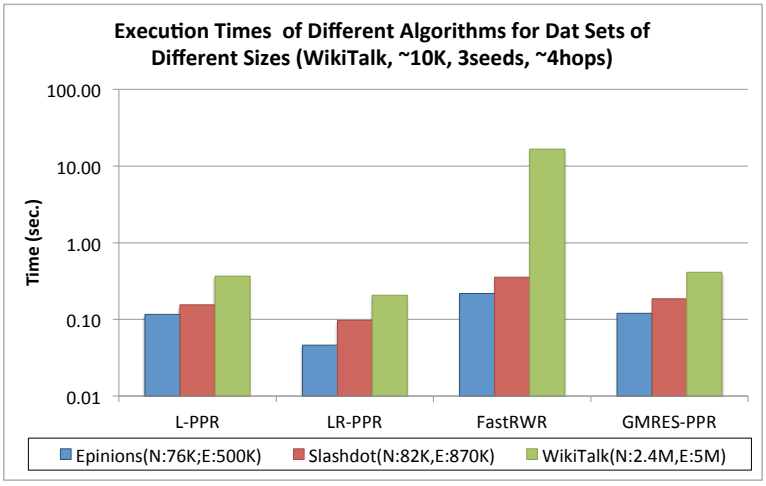
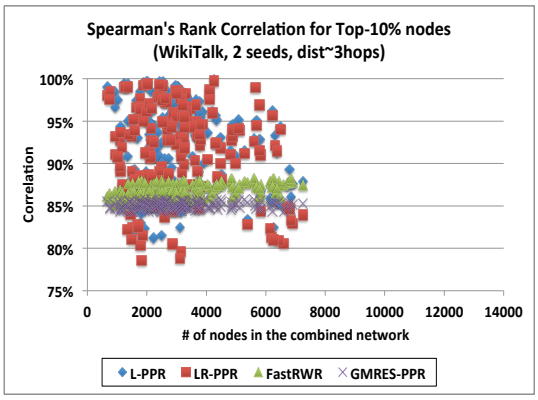


Figure 3.10: Execution Times of the Algorithms L-PPR, LR-PPR, FastRWR, and GMRES-PPR for Different Data Sets of Varying Sizes

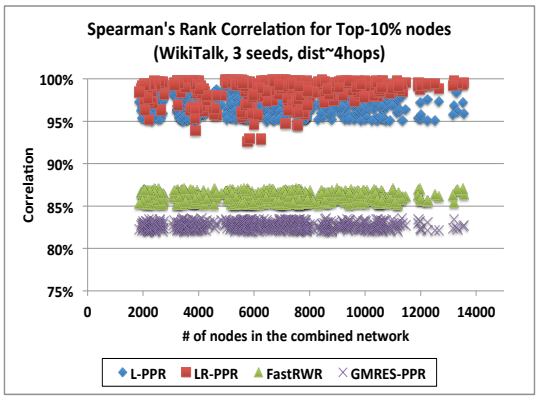
by L-PPR, LR-PPR, FastRWR, and GMRES-PPR. for the WikiTalk data set in Figure 3.11. The most important thing to recognize when comparing Figures 3.9 (for the Epionions data set) and 3.11 (for the WikiTalk data set) is that when the graph is larger (i.e., for the WikiTalk data set), the execution time gains of L-PPR and LR-PPR relative to other algorithsm are even more pronounced. Similarly, as the problem size gets larger (e.g., WikiTalk data, 3 seeds, ~ 4 hops), the accuracy gains of L-PPR and LR-PPR relative to FastRWR and GMRES-PPR also become even more significant. This re-confirms that the proposed locality-sensitive (and re-use promoting) techniques provide not only better scalabilities, but also better accuracies than existing algorithms. We also present the execution times and amounts of data read for LiveJournal data set in the Appendix.

Detailed Studies of L-PPR and LR-PPR

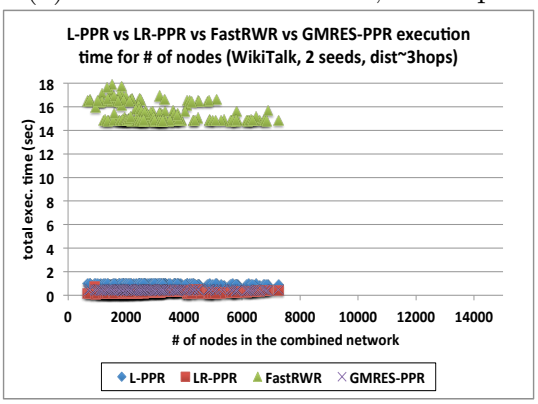
As we see in Figure 3.8 and tables from Table 3.3 to Table 3.8, locality-sensitive and re-use promoting LR-PPR constantly outperforms only locality-sensitive L-PPR (~ 1.5× to 2×), while returning almost as accurate results. Figure 3.12 further investigates



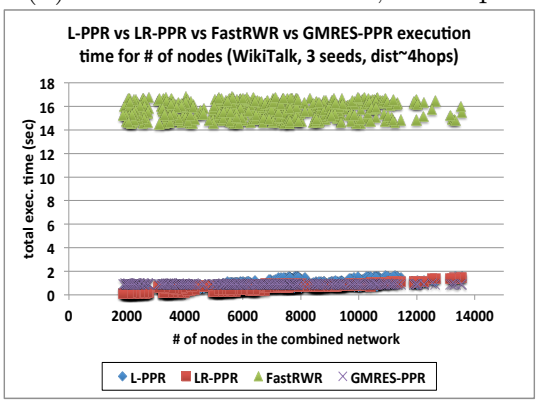
(a) Accuracies for 2 seeds, ~ 3 hops



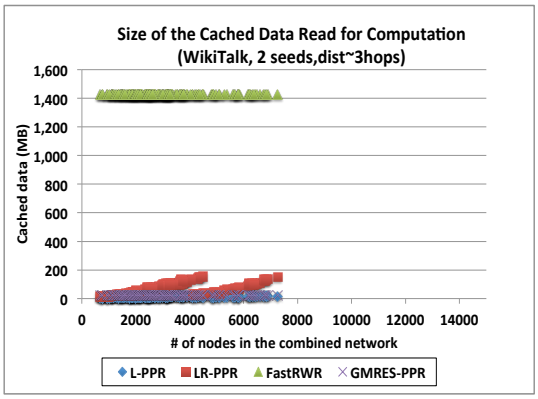
(b) Accuracies for 3 seeds, ~ 4 hops



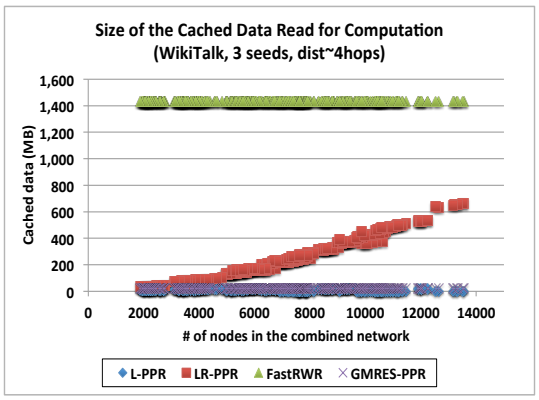
(c) Exec. times for 2 seeds, ~ 3 hops



(d) Exec. times for 3 seeds, ~ 4 hops

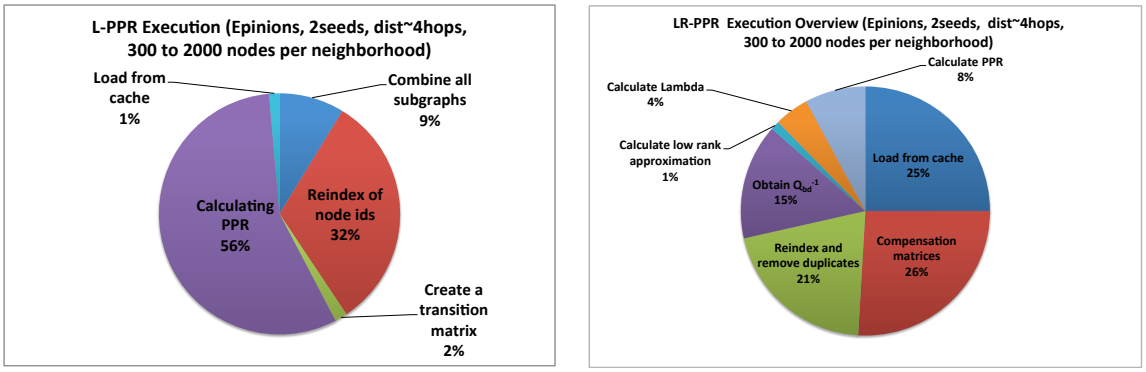


(e) Data for 2 seeds, ~ 3 hops



(f) Data for 3 seeds, ~ 4 hops

Figure 3.11: Performances of L-PPR, LR-PPR, and FastRWR on the Size of the Combined Localities Network (WikiTalk Data Set)



(a) L-PPR (b) LR-PPR

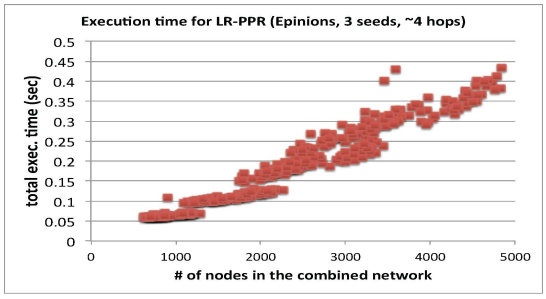
Figure 3.12: Distribution of the Execution Times for L-PPR and LR-PPR for the Epinions Data Set

how the execution times of L-PPR and LR-PPR are distributed among their sub-tasks. As predicted in Section 3.2.4, LR-PPR spends significant portions of its time in loading data from the cache, reindexing nodes, and creating compensation matrices. Creating the low-rank approximation of $\mathbf{M}_0$, computing the matrix $\mathbf{\Lambda}$, and solving for PPR scores take relatively little time. Therefore, significant gains in time can be obtained by parallelizing and further optimizing the initial steps of the LR-PPR algorithm (as discussed in Section 3.4.3).

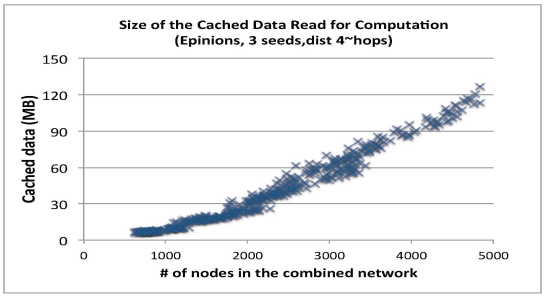
Figure 3.13 shows the execution times, accuracy, and amount of data read by LR-PPR from the cache per query as a function of the size of the merged locality network. As the figure shows, the execution time (Figure 3.13(a)) tracks the amount of data brought into the memory (Figure 3.13(b)), whereas the accuracy is relatively constant (Figure 3.13(c)).

Impact of the Boundary Edges on the Performances of L-PPR and LR-PPR

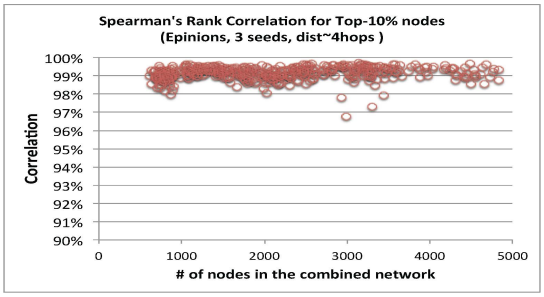
Recall from Section 3.2.2, Figures 3.5 and 3.6, that the merged graph represents nodes outside of the seed localities using a single combined node, which is then connected



(a) Execution times for LR-PPR



(b) Data read into the buffer for LR-PPR



(c) Accuracy for LR-PPR

Figure 3.13: Performance of LR-PPR as a Function of the Size of the Combined Localities Network (Epinion Data Set, 3 Seeds, ~4 Hops)

to the nodes in the seed localities, with outgoing and incoming boundary edges. Figure 3.14 shows the impact of the amount of edges at this boundary. As the figure shows, for a fixed merged locality graph size, the larger the number of boundary edges, the higher the execution times for both L-PPR and LR-PPR; moreover, the larger the merged graph, the faster the increase in the cost. However, the figure also shows that LR-PPR is much less affected from the boundary edges than the basic L-PPR.

Figure 3.15 confirms the impact of the boundary edges on a second data set. As we have seen in tables from Table 3.3 to Table 3.8, for the SlashDot data set, LR-PPR shows a slightly different behavior than for Epinions and WikiTalk data sets: while LR-PPR still outperforms basic L-PPR, the difference is smaller under some configurations. Figure 3.15(a) and (b) explain the reason in terms of the ratio of the

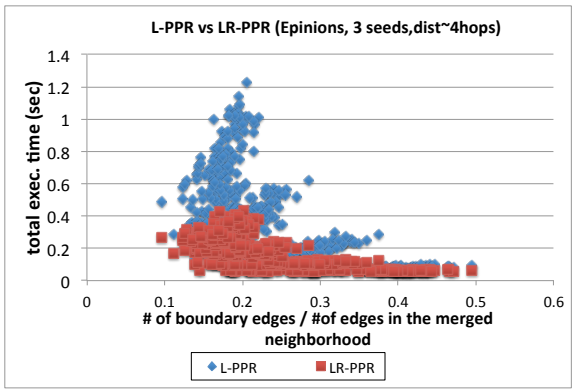
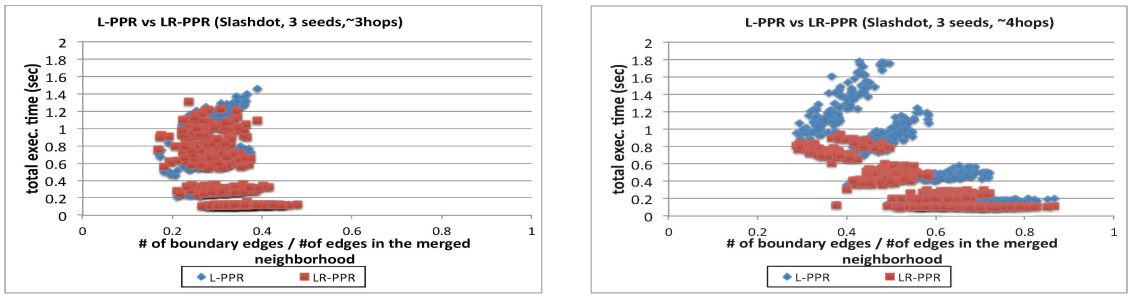


Figure 3.14: The Impact of the Ratio of the Boundary Edges on the Execution Time for L-PPR and LR-PPR (Epinions, 3 Seeds, with Distance ~ 4 Hops)



(a) effect of boundary (3 seeds, ~3 hops) (b) effect of boundary (3 seeds, ~4 hops)

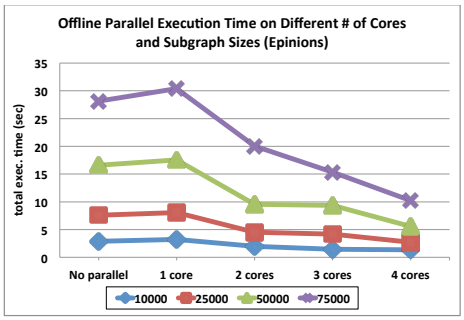
Figure 3.15: Impact of the Boundary Edges for the SlashDot Data Set (3 Seeds)

boundary edges: in the SlashDot data set, when the seeds are close (i.e, when localities overlap significantly), the boundary edges are relatively few and the impact of the boundary edges are similar for both LR-PPR and L-PPR; when the seeds are further away, on the other hand there are more boundary edges and LR-PPR's effectiveness in dealing efficiently with the boundary edges becomes more pronounced. Thus, since the accuracy is not affected and stays high for both LR-PPR and L-PPR, the ratio of the boundary edges in the merged graph can be used as an indicator for when to use LR-PPR and when to simply leverage basic locality-sensitive L-PPR.

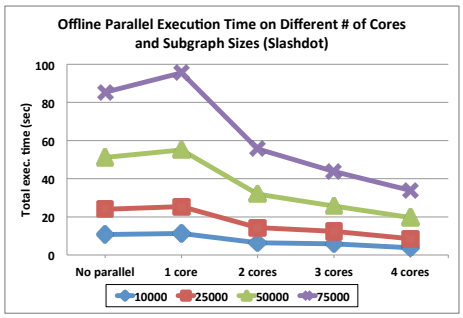
Parallelization of The Off-line Process

As we see in Section 3.4.3, there are multiple opportunities that tasks can be parallelized. One of them is parallelizing the Sub-Task 1 which is generating locality graphs and calculating $\mathbf{Q}_h^{-1}$ processes for each seed node. Figure 3.16 shows the execution time of Sub-Task 1 on no-parallelization and different number of cores with parallelization for four seeds. 'no-parallel' means that all cores are used for the calculation without parallelization and '1-4 cores' means the used number of cores on the parallelization. When the number of core is 1, the execution time takes more than no-parallelization. For other cases whose number of cores are larger than 1, the execution time was decreased significantly. The degree of dropping rate on the large size locality graph is larger than the small size one. All execution time dropping are in same pattern on different data sets and locality graphs. We ran these experiments on the machine with 8GB memory, but for the LiveJournal data set, the results was not static. Even the four cores with parallelization takes longer for the computation. In this case, This is because our experimental machine does not have enough cache size with 1024 KB L2 cache and the 6144 KB L3 cache. We tried run the parallelization on a machine with 32.0 GB Memory and 1024 L2 cache and 8192 cache size and as shown in Figure 3.16 (d), the result show that the pattern follows the same patterns as other data set results.

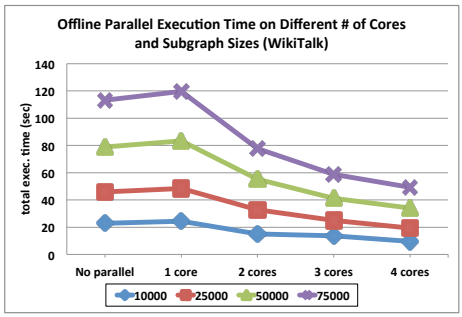
Figure 3.17 compares in further detail the execution times, accuracies, and amounts of data read by L-PPR, LR-PPR, FastRWR, GMRES-PPR from the cache per query as a function of the size of the merged locality network for different seeds and target locality sizes of the SlashDot data set. Since the SlashDot and Epinions Data sets are similar (Table 6.1), the results in Figure 3.17 are also similar to the results for the Epinions data set presented in Section 3.5.3, Figure 3.9.



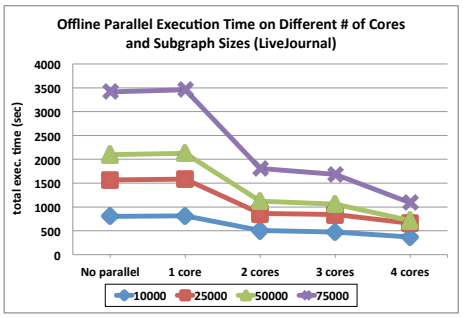
(a) Exec. times for Epinions data set



(b) Exec. times for Slashdot data set



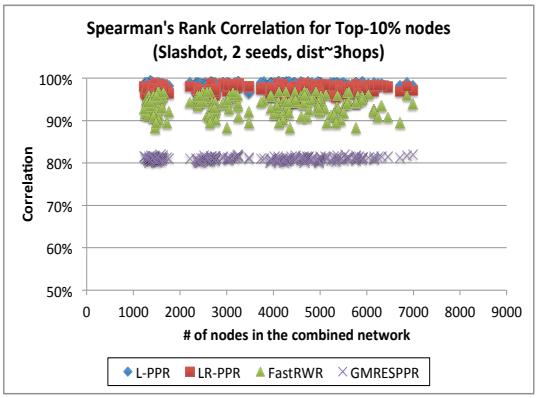
(c) Exec. times for WikiTalk data set



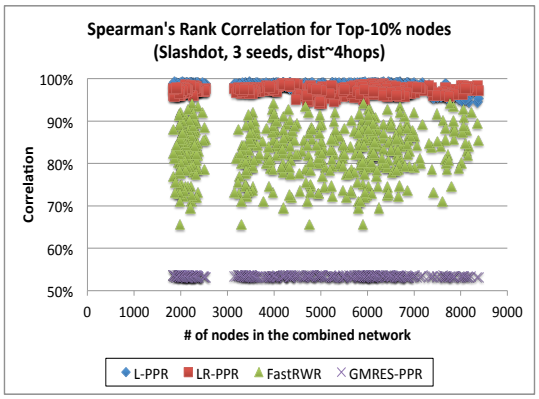
(d) Exec. times for LiveJournal data set

Figure 3.16: Offline Parallelization Execution Time of LR-PPR Generating Locality Graphs and Calculating $\mathbf{Q}_h^{-1}$ for Four Seeds on Different Number of Cores

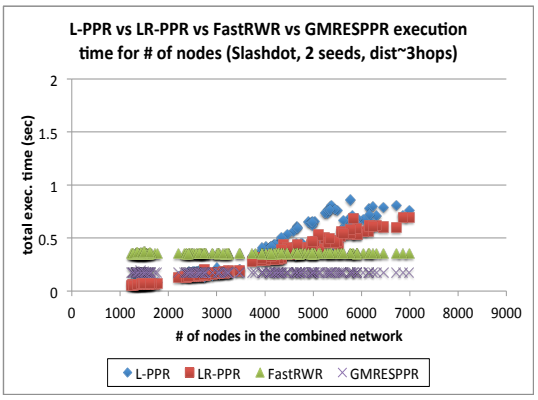
Figure 3.18 also shows and compares the execution times and amounts of data read by L-PPR and LR-PPR from the cache per query as a function of the size of the merged locality network for different seeds and target locality sizes of the LiveJournal data set. Note that we could not compare the accuracies because we could not compute Global PPR. The difference from other datasets is that the number of nodes in 3 hops is larger than the number of nodes in 4 hops. Because we generated our locality graph The results shows that it follows the same pattern as other data sets' results on the execution time and the size of cached data.



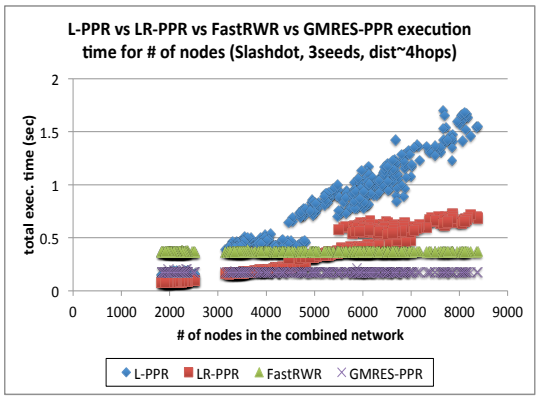
(a) Accuracies for 2 seeds, ~ 3 hops



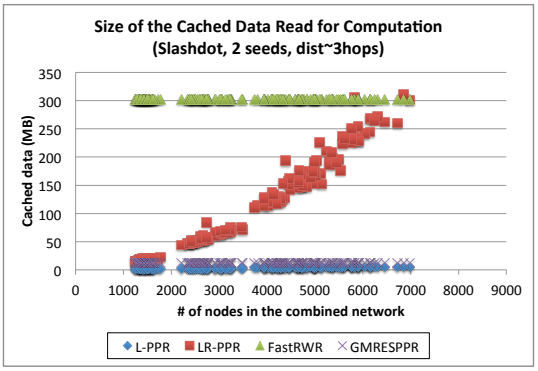
(b) Accuracies for 3 seeds, ~ 4 hops



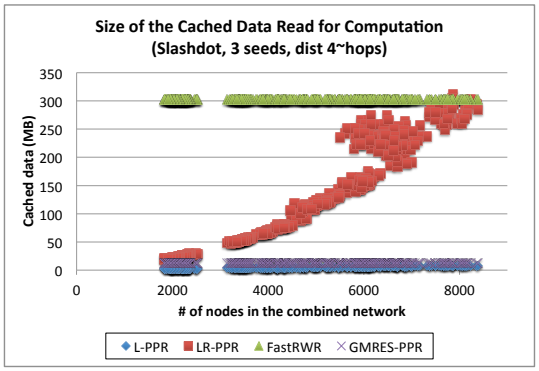
(c) Exec. times for 2 seeds, ~ 3 hops



(d) Exec. times for 3 seeds, ~ 4 hops

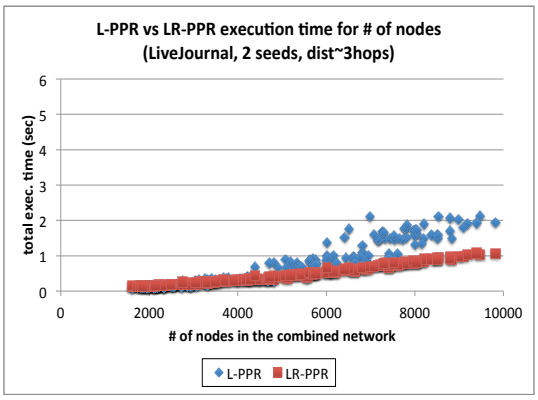


(e) Data for 2 seeds, ~ 3 hops

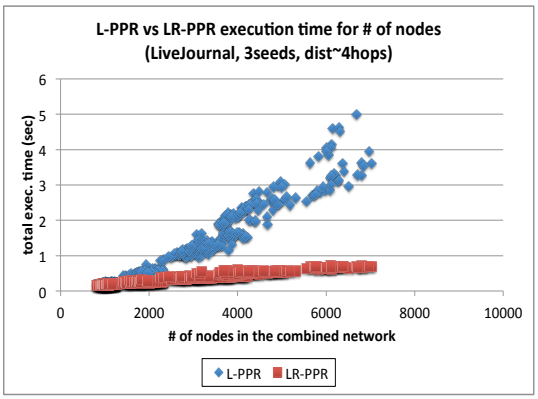


(f) Data for 3 seeds, ~ 4 hops

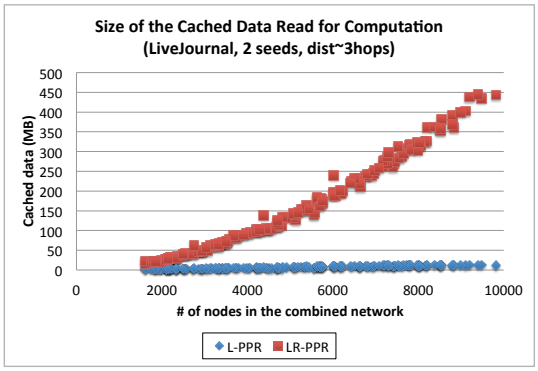
Figure 3.17: Performances of L-PPR, LR-PPR, and FastRWR on the Size of the Combined Localities Network (SlashDot Data Set)



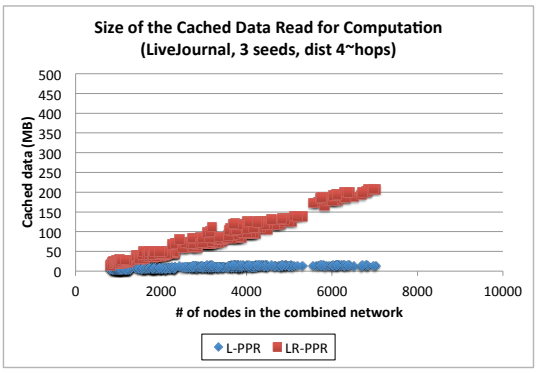
(a) Exec. times for 2 seeds, ~ 3 hops



(b) Exec. times for 3 seeds, ~ 4 hops



(c) Data for 2 seeds, ~ 3 hops



(d) Data for 3 seeds, ~ 4 hops

Figure 3.18: Performances of L-PPR, LR-PPR, and FastRWR on the Size of the Combined Localities Network (LiveJournal Data Set)

Chapter 4

IMPACT NEIGHBORHOOD INDEXING IN DIFFUSION GRAPHS

4.1 Introduction

A graph neighborhood consists of a set of graph nodes that are nearby or otherwise related to each other. Common definitions include h-hop neighborhoods [14, 27, 110], reachability neighborhoods [27], cluster/partition neighborhoods [57], hitting distance neighborhoods [24, 82], or random walks based proximity measures [19, 11, 23, 105, 99].

Impact neighborhoods are fundamentally different from h-hop [14, 27, 110] and hitting-distance neighborhoods [24, 82, 95], both of which place (direct or indirect) limits on the number of steps. In contrast, impact neighborhood depends on the efficiency with which the nodes and the edges propagate information. In probabilistic definitions of reachability neighborhoods [27], entire messages can be lost at nodes or edges of the graph with some probability. In this sense, our definition of impact neighborhood is related to probabilistic reachability neighborhoods. The key difference, however, is that the definition of impact neighborhood allows multiple imperfect (or weakened) messages received at a node to be recombined to strengthen (or reinforce) its impact.

4.1.1 Contributions and Structure of the Chapter

The structure of this chapter is as follows: In Section 4.2, I introduce the key concepts, leading to the problem definition in Section 4.2.3. Intuitively, I associate to each node in the network a random, binary string that acts as the node's fingerprint.

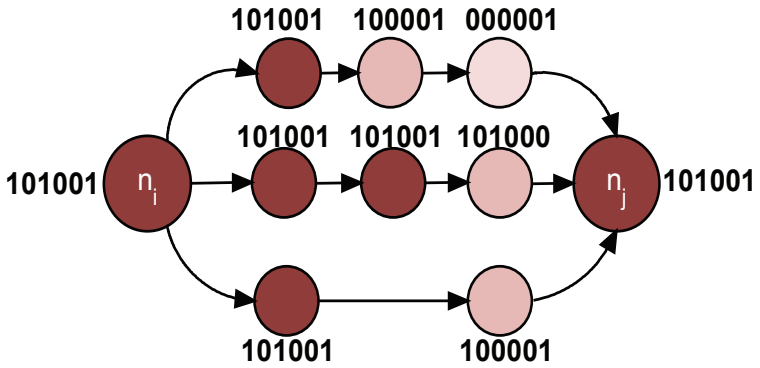


Figure 4.1: The Fingerprint of Node n_i Propagates in the Graph, Subject to Bit Erasures

The impact propagation within the network is modeled as transmissions of these fingerprints in the form of signatures. Random bit erasures are introduced to model the decay in the network and signature composition is used to model reinforcement. The impact n has on m is modeled as the likelihood that n's fingerprint is correctly transmitted (i.e., propagates) from n to m.

In Section 4.3, I present the outline of the basic `INI` algorithm to compute zero-erasure neighborhoods (ZENs) and impact neighborhoods (for a given impact radius, r). `INI` propagates fingerprints in the network subject to bit-erasures, modeling decay. During query time, impact neighborhoods are identified by querying the network nodes for the query node's fingerprint. In this section, I also highlight key efficiency and effectiveness challenges, including communication, processing, and space costs and potential false positives. In Section 4.4, I focus on the reduction of communication and processing costs through the use of combined signatures that eliminate the need for each node n_i to explicitly propagate the fingerprint of each node n_j. In Section 4.5, I introduce the concept of "noise" in combined signatures. Such noise (remnants of partially erased signatures) may lead to false positives and increase processing, communication, and space costs. I thus propose a novel grid-signature scheme which significantly reduces the noise in the system, thereby improving accuracy. In

Section 4.6, I discuss how to re-use an index structure originally created for impact radius r, for identifying impact neighborhoods with radius different from r. I follow this with a discussion, in Section 4.7, of implementation details and parallelization opportunities for `INI`. I evaluate the proposed algorithms for querying impact neighborhoods in Section 4.8. Experiment results show that impact neighborhoods can be quickly and effectively identified using `INI` algorithms.

4.2 Key Concepts

As briefly discussed in the introduction, we associate each node in the network with a random, binary fingerprint.

Definition 1 (Node Fingerprint) *Let $G(V, E)$ be a graph. Each node $n_i \in V$ has a fingerprint which is a b-length bit-string, with c bits set to 1.*

4.2.1 Propagation

Propagation is modeled as transmissions of fingerprints within the graph; random bit erasures are introduced to model the decay in the network. We say that a node n impacts another node m, if n's fingerprint reaches m intact (Figure 4.1).

Definition 2 (Propagation efficiency) *Let $G(V, E)$ be a graph and let $n_i, n_j \in V$ be two nodes on the network, with an edge from n_i to n_j. Let σ_i be a b-length bit-string, with c bits set to 1, on node n_i. We say that information/influence/impact propagates from n_i to n_j with $\mathcal{E}$ efficiency if one of the c non-zero bits of σ_i may be erased (i.e., set to 0) during the transmission from n_i to n_j with erasure probability, $p_e = 1 - \mathcal{E}$.*

Propagation and erasure characteristics of the network are taken to be consistent over sufficiently long periods of time. In other words, if σ_i is transmitted from n_i to

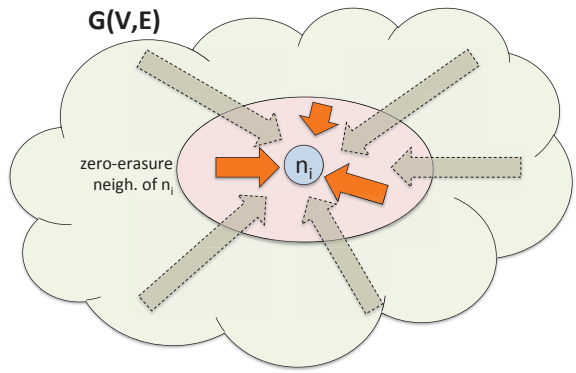


Figure 4.2: Node n_i Receives the Fingerprints of the Nodes Within Its zero-erasure Neighborhood Intact

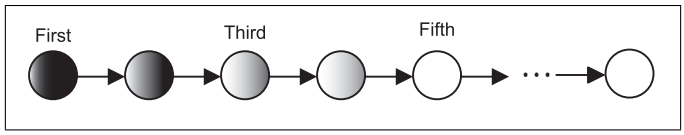
n_j multiple times and if any bit is erased the first time, then the same bit is erased during each and every transmission of the message: We refer to this as the persistency property of the information propagation network.

Definition 3 (Persistent Erasure Mask) *Let $G(V, E)$ be a graph and let $n_i, n_j \in V$ be two nodes on the network, with an edge from n_i to n_j. A persistent erasure mask corresponding to this edge is a b-length bit-string marking the positions of bit erasures.*

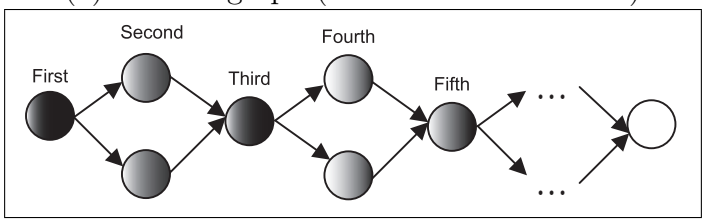
4.2.2 Reinforcement

As shown in Figure 4.1, when there are multiple paths from n to m, a given bit of n's fingerprint has more opportunities for reaching m intact.

Definition 4 (Propagation with Reinforcement) *Let $G(V, E)$ be a graph and let $n_j \in V$ be a node on the network. Let $in(n_j) \subseteq V$ be a subset of nodes with edges towards n_j. The reinforced copy of the message, $\sigma_{i,j}$, originating at node $n_i \in V$, received at node n_j is $\sigma_{i,j} = \bigvee_{n_h \in in(n_j)} \sigma_{i,h}$, where $\sigma_{i,h}$ is the copy of the message σ_i at node n_h and $\bigvee$ is the bit-wise or operation (due to erasures $\sigma_{i,h}$ can differ from σ_i).*



(a) a linear graph (without reinforcement)



(b) a graph with reinforcement

Figure 4.3: Propagation with Reinforcement: Darker Shaded Nodes Have higher Probability of Receiving the Message Intact

4.2.3 Zero-Erasure Neighborhoods (ZENs)

The zero-erasure neighborhood of n is the set of nodes that receive n's fingerprint intact. (Figure 4.2).

Definition 5 (Zero-erasure neighborhood) *Let $G(V, E)$ be a graph with efficiency $\mathcal{E}$ and let $n_i \in V$ be a node on the network. Let also σ_i be a b-length bit-string, with c bits set to 1. The zero-erasure neighborhood, $Z_{\mathcal{E}}(n_i)$ of n_i is a subset of V:*

$$Z_{\mathcal{E}}(n_i) = \{n_j \mid (\sigma_{i,j} = \sigma_i) \wedge (n_j \in V)\}.$$

Note that how far information propagates from a given source node depends on the propagation efficiency as well as the degree of reinforcements enabled by the connectivity. Compare for example the two graphs in Figure 4.3: information propagates further in the second graph due to reinforcements. It is important to note that, in Figure 4.3(b), the node in the "third" layer is, in some sense, nearer to source node than the two nodes in the "second" layer, because it has a better likelihood of receiving the source's fingerprint intact.

ZENs in Linear Graphs

Due to reinforcements, the relationship between the size of the zero-erasure neighborhood (ZEN) and the topology of the graph is complex. Therefore, we first study ZENs on linear graphs , as in Figure 4.3(a), which do not provide opportunities for reinforcement.

Definition 6 (Zero-Erasure Radius in Linear Graphs) *Let $S(V, E)$ be a linearly-structured (infinite) graph, such that $n_0 \in V$ has no incoming edges and there is an edge from node $n_i \in V$ to $n_{i+1} \in V$ (and there are no other edges in E). Let $\mathcal{E}$ be the propagation efficiency of the network and let σ_0 be a b-length bit-string, with c bits set to 1, on n_0. The distance to which σ_0 is* **expected** *to propagate without any erasures of 1 bits is referred to as the zero-erasure radius (or ZE-radius, r_{ze}) of $Z_{\mathcal{E}}(n_0)$.*

The following theorem relates the zero-erasure radius, r_{ze}, and the efficiency, $\mathcal{E}$, of a (linearly-structured) network.

Theorem 1 (Zero-Erasure Radius in Linear Graphs) *Let $S(V, E)$ be a linearly-structured (infinite) graph, such that $n_0 \in V$ has no incoming edges and there is an edge from node $n_i \in V$ to $n_{i+1} \in V$ (and there are no other edges in E). Let $\mathcal{E}$ be the propagation efficiency of the network and let σ_0 be a b-length bit-string, with c bits set to 1, on n_0. The zero-erasure propagation radius, r_{ze}, is less than or equal to $\frac{1}{p_e}$, where $p_e = 1 - \mathcal{E}$ is the erasure probability; i.e. $r_{ze} \leq \frac{1}{p_e} < r_{ze} + 1$.*

We can restate this theorem in the form of a constraint on $\mathcal{E}$:

$$\textbf{Radius Constraint}: \quad 1 - \frac{1}{r_{ze}} \leq \mathcal{E} < 1 - \frac{1}{r_{ze} + 1}.$$

A corollary of this formulation is that the zero-erasure radius r_{ze} does not need to be an integer; we can talk about non-integer radii, such as 2.3, meaning that the expected number of hops information will propagate without errors is 2.3.

ZENs with Reinforcement

Since, as we have seen in Figure 4.1, there can be gaps in the zero-erasure neighborhood of a given node, instead of attempting to directly measure the radius of a zero-erasure neighborhood in a graph with reinforcement, we associate a linear-equivalent radius: , reflecting the underlying propagation efficiency:

Definition 7 (Linear-Equivalent Radius) *Let $G(V, E)$ be a graph with propagation efficiency $\mathcal{E}$ and $n_i \in V$ be a node on the graph. Let also σ be a b-length bit-string, with c bits set to 1. Let $Z_{\mathcal{E}}(n_i)$ be the zero-erasure neighborhood of n_i on G. The corresponding linear-equivalent zero-erasure radius of $Z_{\mathcal{E}}(n_i)$ is defined by two bounds, $r_{le,\perp}$ and $r_{le,\top}$:*

$$r_{le,\perp} = \frac{1}{1-\mathcal{E}} - 1 \quad \textit{and} \quad r_{le,\top} = \frac{1}{1-\mathcal{E}}.$$

Intuitively, the definition of the linear-equivalent radius of $Z_{\mathcal{E}}(n_i)$ reflects the observation that nodes that are in $Z_{\mathcal{E}}(n_i)$ should have a similar chance of receiving the fingerprint of n_i intact as its r_{le}-hop neighbors would have on a linearly structured graph.

4.2.4 Impact Neighborhoods

Let us be given a graph $G(V, E)$, a node n, and a target (linear-equivalent) radius, r, and asked to identify the nodes within the zero-erasure neighborhood defined by of node n. and the radius, r, this alternative formulation of the problem requires leveraging of the relationship between the (linear equivalent) network radius and the propagation efficiency formulated in Theorem 1. In particular, We use Theorem 1 to identify two bounds, $\mathcal{E}_{\perp}$ and $\mathcal{E}_{\top}$, on the required propagation efficiency corresponding to radius, r:

$$\mathcal{E}_{\perp}(r) = 1 - \frac{1}{r} \text{ and } \mathcal{E}_{\top}(r) = 1 - \frac{1}{r+1}.$$

Given these we define the impact neighborhood with (linear equivalent) radius, r, as follows:

Definition 8 (r-Radius Impact Neighborhood) *The r-radius impact neighborhood, $N(r, n_i)$, of n_i is the neighborhood defined by the propagation efficiency, $\mathcal{E}_\perp(r)$; i.e., $N(r, n_i) = Z_{\mathcal{E}_\perp(r)}(n_i)$.*

In other words, the r-radius impact neighborhood is the neighborhood defined by the lowest possible propagation efficiency corresponding to the target (linear-equivalent) radius, r.

4.3 Impact Neighborhood Indexing (INI): Overview and Challenges

In this section, I propose an off-line zero-erasure (or impact) neighborhood indexing (`INI`) algorithm for querying zero-erasure impact neighborhoods. Given an input graph, $G(V, E)$ and a target radius, r, the off-line zero-erasure (or impact) neighborhood indexing (`INI`) algorithm creates a signature-based index structure, by selecting a corresponding erasure rate, p_e, and by propagating the node signatures in the network with this erasure rate. In this section, we first present the outline of the basic `INI` algorithm and highlight the key efficiency and effectiveness challenges, including communication, processing, and space costs and potential false positives.

4.3.1 Outline of the Basic INI Process

Let us be given a graph $G(V, E)$, a node n, and a propagation efficiency $\mathcal{E}$. The basic `INI` algorithm consists of three steps.

Step 1 (Initialization): In its very first step, the `INI` algorithm associates to each node n_i in the graph an (almost) unique fingerprint, σ_i, which is a b-length bit-string, with c random bits set to 1. To prevent collisions, b and c need to be selected carefully.

Since with b bits, c of which are set to 1, we can represent $\binom{b}{c}$ unique node fingerprints, b and c need to satisfy the following uniqueness constraint:

$$\textbf{Uniqueness Const.}: \quad \prod_{h=1}^{|V-1|} \frac{\binom{b}{c} - h}{\binom{b}{c}} \sim 1.$$

Intuitively, this constraint implies that b and c need to be selected in a way that allows sufficient diversity in fingerprints (i.e., $b \gg c$). (In the basic version of the algorithm) at this step, we also associate to each node n_i an erasure mask to be applied to the fingerprint of another node n_j if that node's fingerprint propagates over n_i.

Step 2 (Propagation): Next, the `INI` algorithm propagates these fingerprints within the graph, subject to erasures reflecting the underlying network efficiency, $\mathcal{E}$. At each step, each node receives fingerprints from its incoming edges, applies its own signature and the corresponding erasure mask to each fingerprint, and forwards the updated fingerprints. A key theorem states that cycles in the network have no effect on how far information propagates:

Theorem 2 (Cycle-Agnosticity of Propagation) *Let $G(V, E)$ be a graph and let $n_i, n_j \in V$ be two nodes in V. Let σ_i be a b-length bit-string, with c bits set to 1, on node n_i. Let also $G'(V, E')$ be an acyclic subset of G which preserves all simple paths in G from n_i to all other vertices $n_h \in V$. Then, if $\sigma_{i,j}$ is the copy of the message σ_i at node n_j on G and $\sigma'_{i,j}$ is the copy at node n_j on G', then $\sigma'_{i,j} = \sigma_{i,j}$.*

The proof follows from the persistency property of the erasures (See Definition 3). Since propagation is not affected by the cycles, it follows that the impact neighborhoods (that depend on propagation) are also immune to cycles. `INI` terminates as soon as the bit-strings propagation within the network reaches a fixed-point.

Step 3 (Indexing): After propagation, each node, n_i, aggregates all fingerprints it received into a combined signature, $\sigma_{comb}(n_i)$. These are which are then stored and indexed using signature indexing, such as [119].

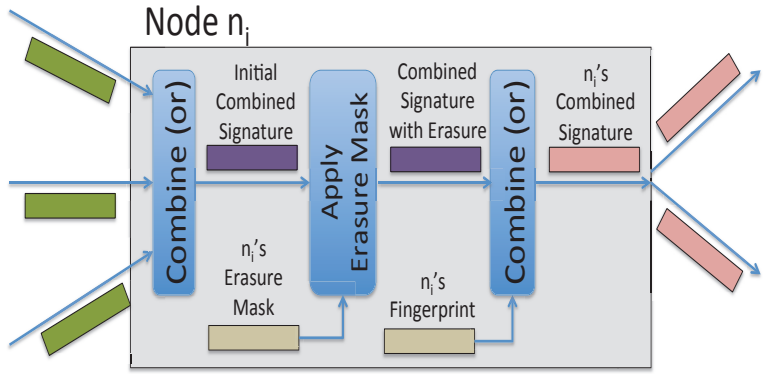


Figure 4.4: Combining Signatures for Reduced Cost

Query Processing: Once the `INI` index is created, a query for the zero-erasure neighborhood, $Z_{\mathcal{E}}(n_i)$, of n_i is answered by searching the node fingerprint, σ_i, of the query node within the combined signature, $\sigma_{comb}(n_j)$, of each node, n_j in the graph:

$$Z_{\mathcal{E}}(n_i) = \left\{ n_j \;\middle|\; (\sigma_{comb}(n_j) \bigwedge \sigma_i) = \sigma_i \right\},$$

where $\bigwedge$ is bitwise-and operation. This search can be performed using any signature search algorithm [119].

4.4 Reducing Costs

Obviously, propagating individual fingerprints until a fixed-point is achieved and maintaining at each node a different erasure mask for each and every other node would be very inefficient. As shown in Figure 4.4, we associate to each node a single erasure mask. At each iteration, each node combines all incoming signatures into a single combined signature, σ_{comb} (by "or"ing the bit-strings) and applies this single erasure mask to this combined signature.

Let us assume that we use a combined signature, σ_{comb}, integrating a set, $\mathcal{S}$, of b-length signatures, each with c non-zero bits. If we erase one non-zero bit of σ_{comb} with probability p_{comb}, then the chance that the erased bit corresponds to one of the c non-zero bits of $\sigma \in \mathcal{S}$, is $\frac{c}{b}$. Hence, given p_{comb}, we can compute the single bit erasure

probability, p_{ind}, for individual signatures, $\sigma \in \mathcal{S}$, as $p_{ind} = p_{comb} \times \frac{c}{b}$. Therefore, if the erasure rate in the network for the individual signatures is $p_{ind} = p_e = 1 - \mathcal{E}$, we need to erase one bit from the combined signature, σ_{comb}, with probability,

$$\textbf{Singlebit Erasure Const.}: \quad p_{comb} = (1 - \mathcal{E}) \times \frac{b}{c}$$

to match the erasure rate of the individual signatures. Note that, as long as $(1-\mathcal{E}) \times \frac{b}{c} \leq 1$, this constraint can be used for computing the single bit erasure rate for the combined signature. However, if the value $(1-\mathcal{E}) \times \frac{b}{c}$ is larger than 1, we need to erase more than 1 bit. Thus, we achieve the necessary degree of erasure by erasing $k > 1$ random bits from the combined signature with p^* probability instead. In that case, the probability that exactly $l > 1$ one bits have been erased on a given individual bit string will be

$$p_{ind}(l) = p^* \binom{k}{l} \prod_{i=0}^{l-1} \frac{c-i}{b-i} \prod_{j=0}^{k-l-1} \frac{b-c-j}{b-l-j}.$$

Given this, the expected number of bit erasures is $X_{ind} = \sum_{l=1}^{c} l \times p_{ind}(l)$. This implies that the values of p^* and k have to be selected such that the expected number of one bits erased during propagation is close to $1-\mathcal{E}$; in other words $X_{ind} = (1+\epsilon) \times (1-\mathcal{E})$, for an error rate ϵ very close to 0. Note that, since $b \gg c$, we can simplify the computation of p^* and k as follows: Let us assume that we erase k bits out of the overall b bits; we can write the probability, $\mathcal{E}$, that none of the c non-zero bits will be erased as $(1 - p^*) + p^* \left(1 - \frac{k}{b}\right)^c$. In our experiments presented in Section 4.8, we set $p^* = 1$ (i.e., perform erasure in each and every step of propagation) and thus pick k such that

$$\textbf{Multibit Erasure Const.}: \quad k \simeq b \times \left(1 - \mathcal{E}^{\frac{1}{c}}\right).$$

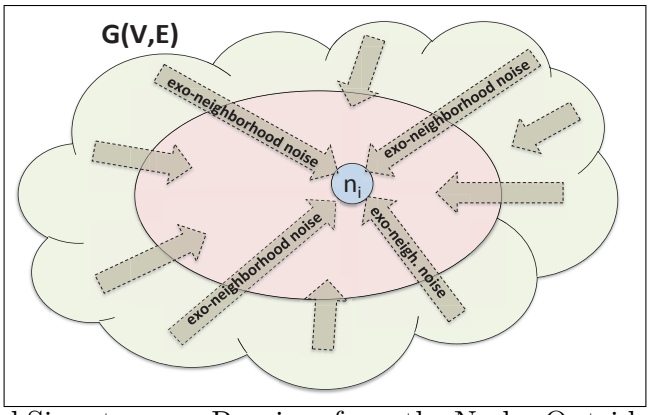


Figure 4.5: Partial Signatures n_i Receives from the Nodes Outside of Its Zero-erasure Neighborhood Makes up the Exo-neighborhood Noise on n_i

4.5 Reducing False Positives

While the search for neighborhood nodes using the combined signatures can be implemented efficiently, there may be false positives in the query results. Bits remaining in the partially erased signatures of the nodes outside the zero-erasure neighborhood (i.e., exo-neighborhood noise) Figure 4.5) may contribute to false positives. False positives can also arise when the fingerprints of two or more nodes inside the zero-erasure neighborhood may combine in a way that matches the fingerprint of a node which is in reality not in the neighborhood – the in-neighborhood noise.

Let us be given a graph $G(V, E)$ and node $n_i \in V$ on which we are measuring the exo-neighborhood noise. Intuitively, the likelihood, $p_{exo}(n_i)$, of exo-neighborhood noise on n_i depends on the number of nodes outside the zero-erasure neighborhood of n_i and how far they are located from n_i. Here, we first treat exo-neighborhood noise along with the in-neighborhood noise to obtain an upper bound on the false positive rate. Then, in Section 4.5.2, we discuss how to reduce exo-neighborhood noise.

4.5.1 Signature Length and False Positives

Let us consider the combined signature, $\sigma_{comb}(n_i)$, at node n_i. The combined signature is composed of $m = |Z_{\mathcal{E}}(n_i)|$ signatures from the nodes within the zero-erasure neighborhood of n_i and also the exo-neighborhood noise from outside the neighborhood. Remember from Section 4.2, that each of the m node signatures contributing to $\sigma_{comb}(n_i)$ is a b-length bit-string, with c random bits set to 1. Thus, we can compute the probability of a given bit in the combined signature, $\sigma_{comb}(n_i)$, being set to 1 as follows: $p_{bitset}(n_i) \sim 1 - \left((1 - p_{exo}(i)) \times e^{\frac{-mc}{b}}\right)$. Now, let us consider a query q to identify the zero-erasure neighborhood of n_i. Let us assume that we are given an upper-bound on the false positive rate ρ_{fp}. We can formalize the false positive rate constraint as $\left(1 - \left((1 - p_{exo}(n_i)) \times e^{\frac{-cm}{b}}\right)\right)^c \leq \rho_{fp}$. Since $Z_{\mathcal{E}}(n_i) \subseteq V$, we remove dependency on m by tightening this constraint as $\left(1 - \left((1 - p_{exo}(n_i)) \times e^{\frac{-c|V|}{b}}\right)\right)^c \leq \rho_{fp}$. Intuitively, this constraint treats all the nodes in V as if they are in the zero-erasure neighborhood. Thus, to prevent double counting of the exo-neighborhood noise, we can set $p_{exo}(i)$ to 0. As a result, the above constraint can be simplified to

$$\textbf{False Positive Const.}: \quad \left(1 - e^{\frac{-c|V|}{b}}\right)^c \leq \rho_{fp}.$$

Note that this is a pessimistic constraint and will in practice result in unnecessarily large values of b.

4.5.2 Grid-Signatures

Due to the probabilistic nature of the erasures and the possible existence of exo-neighborhood noise in the system, in individual runs, the propagation may differ from the predicted distance. Thus, we need to tighten the probabilistic spread of the propagation. We achieve this by associating to each node, $n_i \in V$, not a single combined signature, but a set, $\Sigma_{comb}(n_i)$, of $u \times v$ signatures, logically arranged into

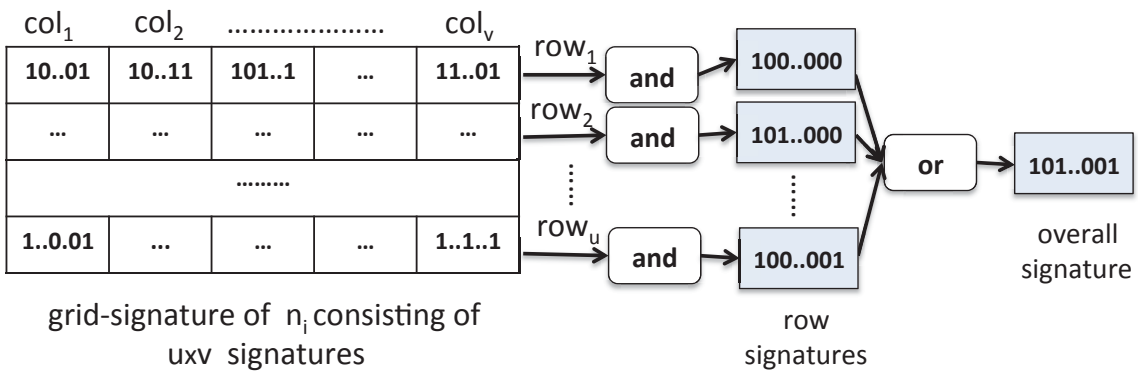


Figure 4.6: Grid-signatures

grid consisting of u rows, each with v signatures[1]. (Figure 4.6). Intuitively, the combined signature at each grid-cell corresponds to an independent run of the `INI` propagation algorithm and there is a different erasure mask corresponding to each grid-cell (i.e., run). Let n_i be a node in the graph and let $\Sigma_{comb}(n_i)$ consist of $u \times v$ combined signatures at node n_i:

- Each row of $\Sigma_{comb}(n_i)$ represents a conjunction: Let $R_h \subseteq \Sigma_{comb}(n_i)$ be a row of v (column) signatures ($1 \leq h \leq u$). A bit in the row signature corresponding to R_h is said to be set only if it is also set in all v (column) signatures. Thus, if the erasure probability corresponding to the single bit-signature is p, then the overall erasure probability of the row signature is $p_{row} = 1 - (1-p)^v$.

- $\Sigma_{comb}(n_i)$ represents a disjunction of its rows: If a bit is set in the row signature corresponding to any row $R_1, \ldots, R_u$, then it is also set in the signature corresponding to $\Sigma_{comb}(n_i)$. Thus, the erasure corresponding to the $u \times v$ grid-signature is $p_{overall} = p_{\Sigma}^{u} = (1 - (1-p)^v)^u$.

[1] A similar multi-hashing technique is leveraged for nearest neighboring searching in high dimensional spaces [41]

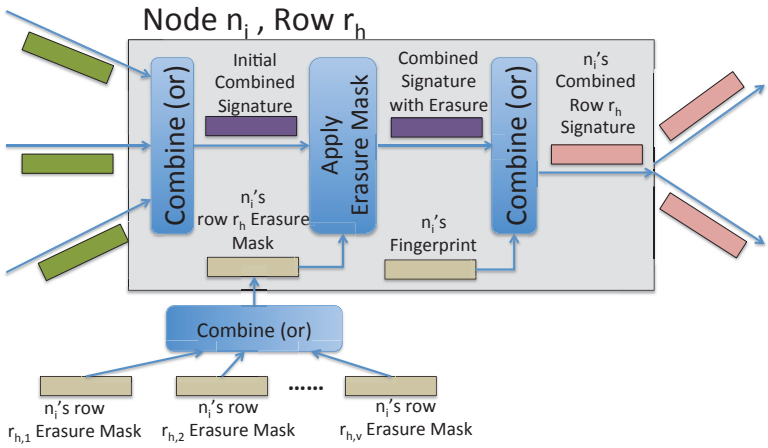
Based on this, given a target erasure probability, p_e, if u, v, and p are selected in such a way that

$$\textbf{Erasure Equivalence Const.}: \quad p_e = (1-(1-p)^v)^u,$$

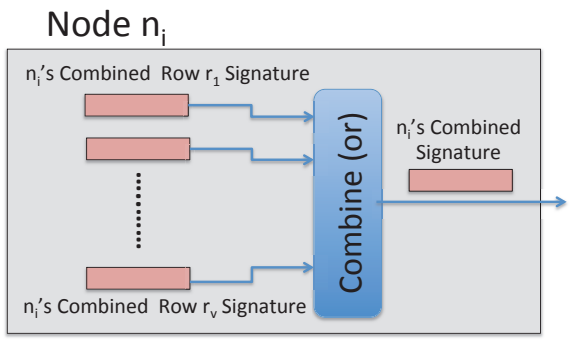
then the expected propagation distance with erasure probability p when using a grid-signature with u rows and v columns will be the same as the original expected propagation distance with erasure probability p_e without using a grid-signature. As we see next, having $u > 1$ and $v > 1$ helps the `INI` indexing scheme reduce the false positive rates as well as (perhaps counter-intuitively) the overall space, communication, and processing costs.

Reductions in the False Rates

Let p_{ok} be the probability of a zero-erasure neighborhood node being successfully located using a single combined signature, whereas p_{fp} be the probability of a node being returned as a false positive. When using a grid-signature corresponding to $u \times v$ combined signatures as described above, the probability of a neighborhood node being successfully located becomes $p'_{ok} = 1-(1-(p_{ok})^v)^u$. Similarly, the probability of a node outside of the neighborhood being returned as a false positive becomes $p'_{fp} = 1-(1-(p_{fp})^v)^u$. Note that the function $1-(1-(p)^v)^u$ has a separating effect on the probabilities: relatively large values of p are pushed even higher, whereas values that are close to 0 are pulled even lower. As a consequence, assuming that initially we have $p_{fp} < p_{ok}$, the separating nature of the function $1-(1-(p)^v)^u$, ensures that $p'_{fp} \ll p'_{ok}$: in other words, using u rows and v columns of bit-signatures helps bring the likelihood of false positives down, without affecting the radius.



(a) Construction of a row-signature (propagation step)



(b) Construction of a combined grid-signature (post-propagation)

Figure 4.7: Creation of Grid-signatures

Reductions in Operating Costs

In practice, we do not perform $u \times v$ individual propagations for each cell in the grid for each node. Instead, for each row, we create a single combined row erasure mask, with erasure probability $p_{row} = 1 - (1 - p)^v$, recording the combined positions of erasures for each row and apply this row erasure mask on the signatures at each step (Figure 4.7). Since the distance to which each bit can propagate is reduced, combined row signatures of the nodes may carry lesser exo-neighborhood noise, and the propagation step may require fewer iterations to reach a fixed-point.

4.6 Index Reuse

In applications where the propagation efficiency is variable or we are interested in different impact radii, however, we may want to re-use an existing `INI` index created specifically for a given $\mathcal{E}$ or r for a different efficiency, $\mathcal{E}'$, or radius, r'. In this section, we propose two different schemes for allowing the re-use of `INI` indexes: bit masking and ε-erasure.

4.6.1 Bit Masking

Remember from Section 4.2.3 that the (linear-equivalent) propagation radius, r, is the largest integer, less than or equal to $\frac{1}{p_e}$, where $p_e = 1 - \mathcal{E}$ is the erasure probability of the graph:

$$r \leq \frac{1}{p_e} < r + 1.$$

$r \leq (1/p_e) < r + 1$. This inequality implies that, in order to increase the radius of the impact neighborhood to $r' = \kappa \times r$, for $\kappa > 1$, we need to reduce the erasure probability by a corresponding factor of κ.

However, since the erasure rate is a design parameter of the `INI` index, we cannot change its value freely (without re-indexing the entire graph). But, we can alter the effective rate of erasure by masking one or more random bits in the combined signatures during query processing.

Consider b-length node fingerprints, with c random bits set to 1. Let us assume that, during query processing, we mask (i.e., ignore) x bits of the query node's fingerprint (and the combined signatures in the database). Then, the probability with which a bit-erasure is detected will drop by a factor $\frac{b}{b-x}$. This implies that, if we want to increase the radius of the zero-erasure neighborhood by $\kappa > 1$, we need to ignore x

bits such that

$$\textbf{Bit Masking Const.}: \quad \kappa \geq \frac{b}{b-x}.$$

Note that bit masking carries some risk: bits ignored during retrieval may contribute to an increase in the false positive rate. When x bits are ignored, assuming that $b \gg c$, the upper bound on the false positive rate (discussed in Section 4.5.1) becomes

$$\sim \left(1 - e^{\frac{-c|V|}{b-x}}\right)^c.$$

An alternative way to increase the radius of the impact neighborhood is to relax the matching constraint and look for ε-erasure neighborhoods, where $\varepsilon > 0$. We discuss this next.

4.6.2 $\mathcal{E}$-Erasure Neighborhoods

Allowing a 1-bit erasure between the query node's fingerprint and the combined signatures would (roughly) double the effective radius of the 1-erasure neighborhood. Similarly, allowing 2-bit erasures would (roughly) triple the effective radius of the neighborhood. To see why, remember from Section 4.4 that, given b-length node fingerprints, with c random bits set to 1, the relationship between the erasure rate, p_{comb}, on the combined signatures and the erasure rate on the individual node signatures is as follows: $p_{ind} = p_{comb} \times (c/b)$. Therefore, the radius, r_0, of the zero-erasure neighborhood (i.e., before the 1st bit erasure) is

$$r_0 \leq \frac{b}{p_{comb} \times c} < r_0 + 1.$$

Now let us consider the 1-erasure neighborhoods. Let r_1 denote the (linear-equivalent) radius of the 1-erasure neighborhood and let $d_1 = (r_1 - r_0)$ be the number of hops between the 1st and 2nd bit erasures. Since one of the non-zero bits has already been zeroed due to the 1st erasure, the individual erasure probability, $p_{ind,1}$, in the range

between the 1st and 2nd bit erasures becomes $p_{comb} \times ((c-1)/b)$. Hence, we have $d_1 = r_0 \times (c/c-1)$. In other words,

$$r_1 = r_0 + d_1 = r_0 \times \left(1 + \frac{c}{c-1}\right).$$

It is easy to see that we can generalize this to the radius of any ε-erasure neighborhood: More generally, the impact radius of the ε-erasure neighborhood of a given node can be calculated as

$$\varepsilon - \textbf{Erasure Const.}: \quad r_h = r_0 \times \left(\sum_{i=0}^{\varepsilon} \frac{c}{c-i}\right),$$

where $d(= d^0)$ r_0 is the radius of the zero-erasure neighborhood.

Note that while the use ε-erasure neighborhoods will also impact the false positive rates, the increase is likely to be lower than when using bit-masking. While the proof of this is outside of the scope of this chapter, it is easy to see why this is the case: Bit-masking simply ignores x randomly selected bit positions in the query; thereby effectively reducing the signature length from b to $b-x$ bits. Since there are many more 0s in nodes' combined signatures than there are 1s, this has the unwelcome impact of reducing the discriminating power of these 0s during retrieval. The ε-erasure approach, on the other hand, does not reduce the effective length of the signatures: For example, in the case of 1-erasure neighborhoods, in addition to the original query signature where c out of b bits are set to 1, we simply perform searches for query signatures where only $c-1$ out of the original c bits are set to 1. Since, for each query, we still use all b signature bits, this does not necessarily reduce the discriminating power provided by the signature length.

4.7 Implementation Discussions

The `INI` indexing scheme consists of three major phases: (Step 1) initialization (where fingerprints and erasure masks are created); (Step 2) propagation; and (Step

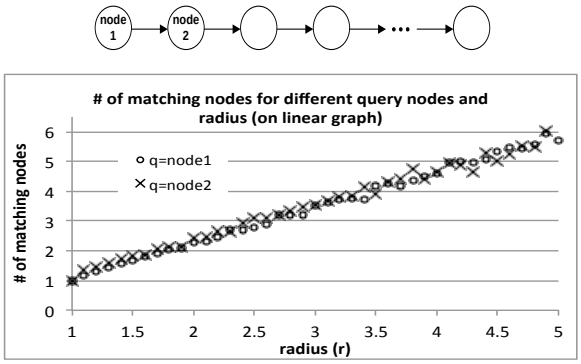
3) aggregation and indexing:

The initialization step (Step 1) is easy to parallelize by (randomly) assigning each node in the graph onto a server. Since node fingerprints and erasure masks have only few 1s but many 0s (i.e., $c \ll b$), it is more effective to use a storage scheme which only records positions of 1s in the signature and erasure masks. In our implementation of `INI`, we rely on the JavaEWAH package [71] to maintain all the signatures in the compressed form and perform the necessary logical operations on them in the compressed domain.

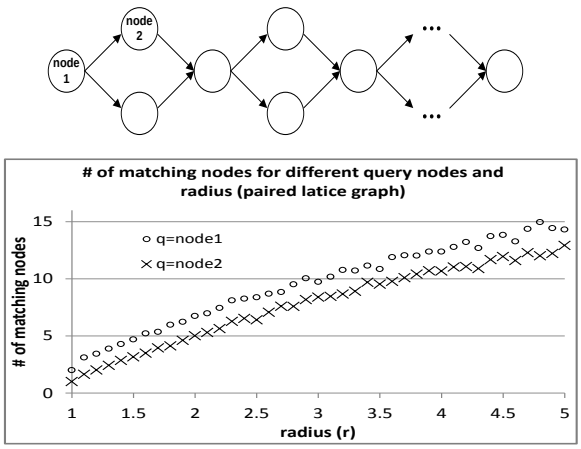
The propagation step (Step 2) can be parallelized in multiple ways: First of all, (a) large-scale graph computing frameworks, such as Pregel [80] and Hama [98], which rely on the bulk synchronous parallel (BSP) model, can help parallelize the propagation process. Secondly, (b) the b length signatures can be divided into div partitions and each partition can be propagated independently (possibly in parallel) from the others. Finally, (c) when using the grid-signature approach, each row-signature can be propagated independently (possibly in parallel) from the others.

The aggregation step (Step 3) can also be parallelized similarly to Step 1 by (randomly) assigning each node in the graph onto an available server and performing the aggregation on that server. If the b length signature itself is split into div partitions, then each signature partition can be aggregated independently (possibly in parallel) from the others.

Once the index is created, the search for the zero-erasure neighborhood of a query node can be performed efficiently using any signature search algorithm [119]. Moreover, the search for a query node's fingerprint is trivially parallelizable by splitting the b bit positions of the signatures into div partitions and assigning each partition to a different server. Also, since bit-erasures are related to Hamming errors, in the case of ε-erasure neighborhood searches, algorithms for string searching with Hamming



(a) neighbor sizes on a linear graph for different query nodes



(b) neighbor sizes on a lattice graph for different query nodes

Figure 4.8: Simple Graph Topologies: (a) in a Linear Graphand (b) in a Lattice Graph

errors, including [41, 68], can be used for identifying initial candidates.

4.8 Experiments

Data sets. First, small linearly structured graphs are used for assessing whether the proposed impact neighborhood schemes work as expected in controlled settings. Then, real data sets are used for observing the effectiveness and efficiency of the algorithm in larger networks: the sparse "co-authorship network science" data set, with 1591 nodes and 2744 edges is obtained from [84]. The dense "online student

INI Performance				Indexing time (seconds)					
				Erasure generation (per row, per div)		Propagation (per row, per div)		Aggregation (per div)	
network	nodes	edges	r	u1:v1	u3:v3	u1:v1	u3:v3	u1:v1	u3:v3
Co-authors (sparse)			1.5	0.814	1.528	0.124	0.101	N/A	0.013
(div = 1)	1591	2744	2	0.546	1.430	0.121	0.091	N/A	0.012
			3	0.326	0.889	0.119	0.088	N/A	0.007
Students (dense)			1.5	1.163	2.209	3.011	2.978	N/A	0.038
(div = 1)	1899	59835	2	0.793	2.035	3.000	2.897	N/A	0.041
			3	0.472	1.414	2.935	2.804	N/A	0.039
P2P (sparse)			1.5	56.9	59.2	83.1	70.0	N/A	3.790
($u1 : v1 \rightarrow div$ = 10)	36682	88328	2	43.1	54.2	60.9	46.8	N/A	3.951
($u3 : v3 \rightarrow div$ = 20)			3	28.2	35.7	50.2	32.0	N/A	4.164

INI Performance				Querying time (seconds)		Storage per node (KB)			
						Erasure signature (per row, per div)		Prop. signature (per row, per div)	
network	nodes	edges	r	u1:v1	u3:v3	u1:v1	u3:v3	u1:v1	u3:v3
Co-authors (sparse)			1.5	0.009	0.010	1.288	1.290	0.208	0.184
(div = 1)	1591	2744	2	0.011	0.010	1.287	1.289	0.214	0.195
			3	0.007	0.010	1.285	1.288	0.217	0.206
Students (dense)			1.5	0.020	0.014	1.532	1.533	1.497	1.497
(div = 1)	1899	59835	2	0.020	0.013	1.531	1.533	1.497	1.497
			3	0.022	0.013	1.529	1.532	1.497	1.497
P2P (sparse)			1.5	0.091	0.074	2.912	1.478	2.823	1.418
($u1 : v1 \rightarrow div$ = 10)	36682	88328	2	0.126	0.114	2.912	1.478	2.825	1.408
($u3 : v3 \rightarrow div$ = 20)			3	0.144	0.127	2.912	1.478	2.825	1.432

Table 4.1: Index Creation and Querying times, and Memory Usage for INI

community" data set, with 1899 nodes and 59835 edges, is from [89]. The "peer-to-peer network" data set, with 36682 nodes and 88328 edges, is from [72].

Default parameters. For all experiments, the signature length, b, and the number, c, of bits set to 1 are selected using the worst-case formulation in Section 4.5 such that the false positive rate is ≤ 0.05. Also, as Section 4.2, the default probability of erasure, p_e, is set to $1/r$, where r is the target radius.

Implementation. The code has been implemented using Java. We run our experiments on a Intel Core i5-2400 CPU @ 3.1GHz with 8GB of RAM (of which 1GB was allocated as the Java heap size). During propagation, all data structures are maintained in memory; when all signatures do not fit into memory of a single server, they are divided into multiple independent partitions, each of which fits into memory, and run in parallel. Similarly, different rows of a grid signature are processed in parallel. Once the propagation is over, the aggregated signatures are indexed in a bit-sliced manner.

4.8.1 Verification of INI's General Properties

In this subsection, we first confirm the key ideas on graphs with simple structures: this allows us to directly observe the effects of the various design decisions and parameter settings. Experiments for the "online student community" sets are run 50 times; for the larger "peer-to-peer network" data set, each experiment is run 10 times.

Relationship between Erasure and Target Radius. In the first experiments, we aimed to verify the relationship between the target neighborhood radius and the erasure rate described in Section 4.4. For these experiments, we have used simple graphs with ~ 30 nodes each. The linear graph (Figure 4.8(a)) is the simplest graph with no reinforcement; thus it provides the best verification tool. The simple lattice graph (Figure 4.8(b)) helps observe the impact of influence reinforcement in the graph. The charts in the figure show the average number of matches for 1000 runs.

Signature Length. Figure 4.9 confirms that using smaller signatures than the signature length, b (192 bits in these experiments), obtained using the worst-case formulation in Section 4.5, may lead to an increase in the number of false positives (though the false positives stay small even with up to 50% drop in signature length).

Grid-Signatures. In Figure 4.10, we study the impact of increasing the number of

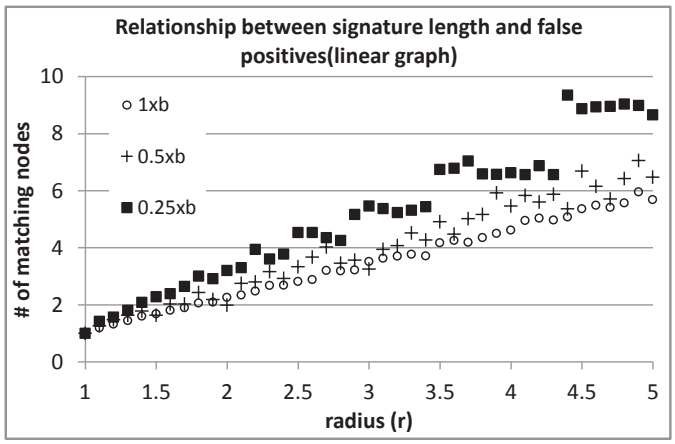
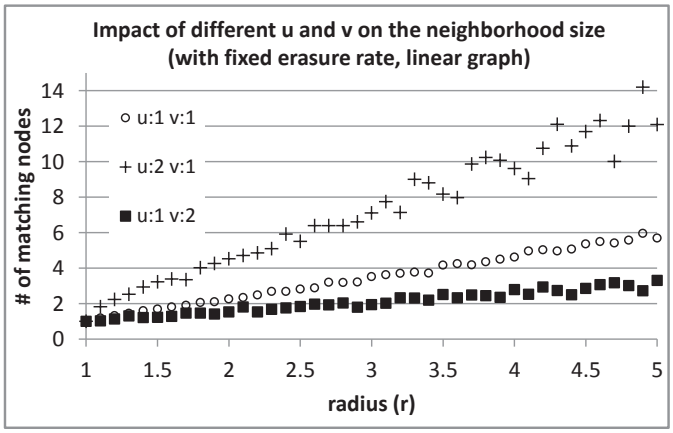


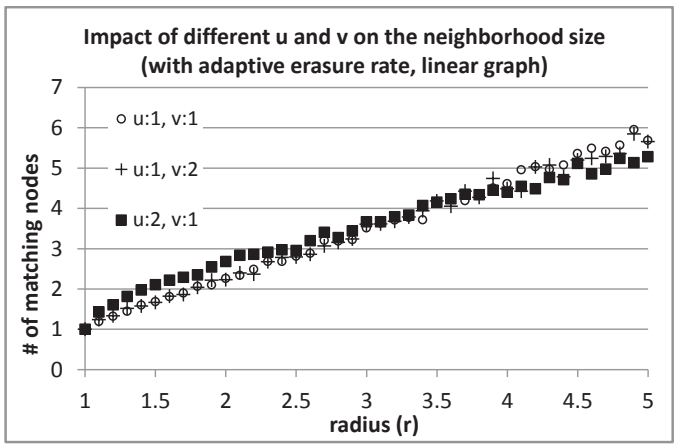
Figure 4.9: Using Smaller Number of Signature Bits Than b (Computed in Section 4.5) Results in False Positives

rows and columns. As can be seen in Figure 4.10(a) naively increasing the number of grid columns results in misses, whereas increasing the number of rows results in false positives. However, in line with the discussion in Section 4.5.2, one can adjust the erasure rate to match the target radius Figure 4.10(b). As shown in Figure 4.11, when using only one signature per node ($u = 1, v = 1$), there is a large variance (misses or false positives) in the number of nodes retrieved in individual runs. When using a grid-signature ($u = 2, v = 2$), on the other hand, the variance in the number of matches is greatly reduced (with the mean being around the target radius – 3 in this example) as predicted in Section 4.5.2.

Index Reuse Strategies. As discussed in Section 4.3, the proposed algorithm creates an index structure for a given target effective distance d by selecting a corresponding erasure rate, p_e. This implies that a given index structure can only be used for a single neighborhood radius. However, as we have seen in Section 4.6, it is possible to revise the effective distance of an already existing technique using bit masking or by relying on ε-erasure. in matching. Figures 4.12 and 4.13 confirm that both bit masking and ε-erasure relaxation techniques can be leveraged for index reuse, but ε-erasure approach leads to more precise distance revisions.



(a) with fixed erasure probability



(b) with adaptive erasure probability

Figure 4.10: (a) A Fixed Erasure May Lead to Drops in Accuracy, Whereas (b) Adaptive Erasure Helps Improve Accuracy

4.8.2 Efficiency of INI on Real Graphs

Table 4.1 shows the indexing and query execution times for networks of different densities of connectivities. The table shows that (a) the number of nodes impacts the amount of computation that needs to be performed for each iteration, whereas the number of edges impacts how far the information travels (i.e., how many iterations are needed); (b) the use of grid signatures increases erasure signature generation time, but (especially in larger graphs and larger impact radii) it also reduces the propagation

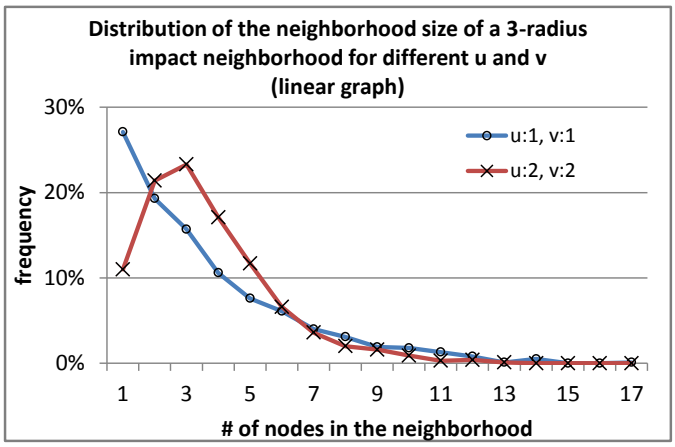


Figure 4.11: Distribution of Matching Nodes for a Target Radius of 4 on a Linear Graph (1000 Runs).

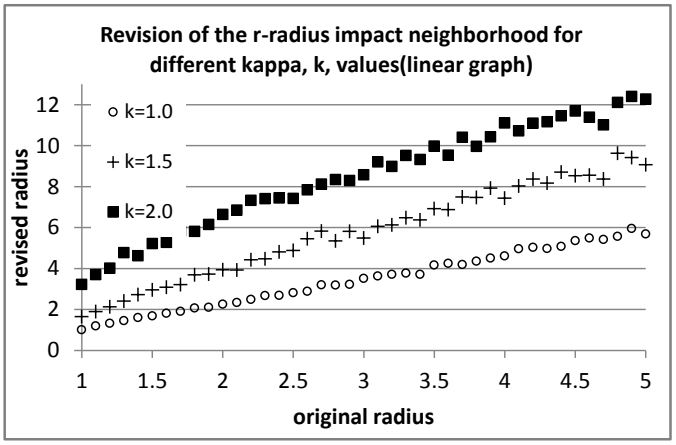


Figure 4.12: Index Re-use with Bit-masking

times (by eliminating noise, which may contribute to additional propagation work); (c) querying times are very fast, even for large graphs and different radii; (d) as the graphs grow in size and/or become dense, the lengths of the signatures grow – the growth in signature length can be limited by using multi-row grid signatures (which eliminate noise, thus the number of bits to be encoded); and (e) the per-server memory footprint of the index creation process can be kept under control by using multiple divisions, processed in parallel.

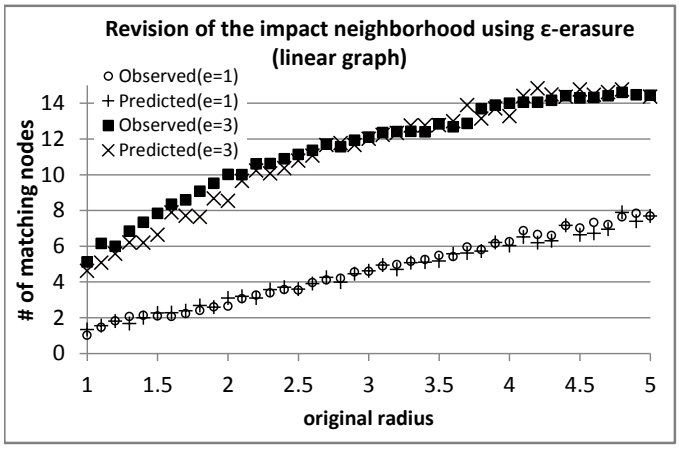


Figure 4.13: Index Re-use Through ε-erasure (ε = 1 and ε = 3): predicted and observed number of matches are well aligned

Chapter 5

PAGERANK REVISITED: RELATIONSHIP BETWEEN NODE DEGREES AND NODE SIGNIFICANCES

5.1 Introduction

In recent years, there has been considerable interest in measuring the significance of a node in a graph and relatedness between two nodes in the graph, as if measured accurately, these can be used for supporting many knowledge discovery, search, and recommendation tasks [9, 19, 23, 36, 103]. The significance of a node in a given graph often needs to reflect the topology of the graph. Measures like the betweenness measure [112] and the centrality/cohesion [15], help quantify how significant any node is on a given graph based on the underlying graph topology. The betweenness measure [112], for example, quantifies whether deleting the node would disconnect or disrupt the graph. Centrality/cohesion [15] measures quantify how close to a clique the given node and its neighbors are. Other authority, prestige, and prominence measures [11, 17, 15] quantify the significance of the node through eigen-analysis or random walks, which help measure how reachable a node is in the graph.

5.1.1 Tight Coupling of PageRank Scores of Nodes and their Degrees

Let us consider an undirected graph $G(V, E)$. Given a node, $v \in V$, the ranking score of v in PageRank measure is decided by two distinct facts which are a) significance of neighbors and b) number of neighbors. Intuitively, the more significant the neighbors of a node are, the higher its likelihood to be also significant. Secondly, even if the neighbors are not all significant, a larger number of neighbor implies that the

Data Set	Listener Graph (Friendship edges, Last.fm)	Article Graph (co-author edges, DBLP)	Movie Graph (co-contributor edges, DBLP)
Correlation between PageRank and Degree	0.988	0.997	0.848

Table 5.1: Spearman's Rank Correlation Between the Node Degree Ranks and The Node Ranks' Based on PageRank Scores for Various Data Graphs

node, v, is well-connected and, thus, likely to be structurally important. Therefore, in theory, these two factors should complement each other. In practice, however, the PageRank formulation described above implies that there is a very tight coupling between the degrees of the nodes in the graph and their PageRank scores (see Table 5.1).

Problem I: When a Large Node Degree Does Not Indicate High Node Significance

In this chapter, I highlight (and experimentally show) that, in many applications, node degree and node significance are in fact inversely related and that the tight- coupling between node degrees and PageRank scores might be counter-productive in generating accurate recommendations.

Example 1 *Consider, for example, a recommendation application where a movie graph, consisting of* `movie` *and* `actor` *nodes, is used for generating movie recommendations. In this application, the first factor (significance of neighbors) clearly has a positive contribution: a movie with good actors is likely to be a good movie and an actress playing in good movies is likely to be a good actress. On the other hand, the second factor (number of neighbors) may in fact be a negative contributor to node significance: the fact that an actor has played in a large number of movies may be a sign*

that he is a non-discriminating ('B movie') actor, whereas an actress with relatively fewer movies may be a more discriminating ('A movie') actress.

As we see in Section 5.3, this observation turns out to be true in many applications, where (a) acquiring additional edges has a cost that is correlated with the significance of the neighbor (e.g. the effort one needs to invest to a high quality movie) and (b) each node has a limited budget (e.g. total effort an actor/actress can invest in his/her work).

Problem II: When PageRank Does Not Sufficiently Account for Contributions of Degrees

The mismatch between PageRank and node significance is not limited to the cases where node degrees are inversely related to the node significance. As we see in Section 5.3, there are other scenarios where PageRank may, in fact, fail to sufficiently account for the contribution of the node degrees to their significances.

5.1.2 PageRank Revisited: De-coupling Node Significance from Node Degrees

As it was discussed above, one key shortcoming of the conventional PageRank scores is that they are often tightly coupled with the degrees of the graph nodes and in many applications the relationship between the significance of the node and its degree in the underlying network may not be as strong as is implied by PageRank-based measure. By this problem, it could return poor results when the connection or edge degree is not related to the expectation.

To address these challenges, in this chapter, I propose degree de-coupled PageRank (D2PR) techniques to improve the effectiveness of PageRank based knowledge discovery and recommendation systems. These techniques suitably penalize or (if

needed) boost[1] the transition strength based on the degree of a given node to adapt the node significances based on the network and application characteristics.

In Sections 5.2, I introduce the proposed network-adaptive degree-decoupled PageRank techniques and after that, I evaluate the proposed techniques in Section 5.3.

5.2 Degree De-Coupled PageRank

The key difficulty of de-coupling node degrees from the PageRank scores is that the definition of the PageRank, based on random walk transitions, is inherently dependent on the number of transitions available from one node to the other. As it was mentioned above, the more ways there are to reach into a node, the higher will be its PageRank score.

5.2.1 Desideratum

Therefore, to de-couple the PageRank score from node degrees, we need to modify the transition matrix. In particular, for each node v_i in the graph, we would like to be able to control the transition process with a single parameter (p), such that

- if $p \ll -1$, transitions from node v_i are $\sim 100\%$ towards the neighbor with the highest degree,
- if $p = -1$, transition probabilities from node v_i are proportional to the degrees of its neighbors,
- if $p = 0$, the transition probabilities mirror the standard PageRank probabilities (assuming undifferentiated neighbors),

[1]In this context, de-coupled does not necessarily imply de-correlated. In fact, D2PR can boost correlation between node degree and PageRank if that is required by the application.

- if $p = 1$, transition probabilities from node v_i are inversely proportional to the degrees of its neighbors,
- if $p \gg 1$, transitions from node v_i are $\sim 100\%$ towards the neighbor with the lowest degree.

In other words, the transition function should de-couple the transition process from node-degrees and penalize or boost the contributions of node degrees in the transition process, as needed.

5.2.2 Degree De-coupling Transition Matrix

In this subsection, we will consider degree de-coupling of the transition matrix as implied by the above desideratum.

Undirected Unweighted Graphs

Let $G = (V, E)$ be an undirected and unweighted graph. Let α also be a given residual probability parameter, and $deg(v)$ be a function which returns the number of edges on the node v. We represent degree de-coupled PageRank (D2PR) scores in the form of a vector

$$\vec{d} = \alpha \mathbf{T}_D \vec{d} + (1 - \alpha)\vec{t},$$

where $\vec{t}$ is the teleportation vector, such that $\vec{t}[i] = \frac{1}{\|V\|}$ for all i and $\mathbf{T}_D$ is a degree de-coupled transition matrix,

$$\mathbf{T}_D(j, i) = \frac{deg(v_j)^{-p}}{\sum_{v_k \in neighbor(v_i)} deg(v_k)^{-p}}, \tag{5.1}$$

where

- $\mathbf{T}_D(j, i)$ denotes the degree de-coupled transition probability from node v_i to

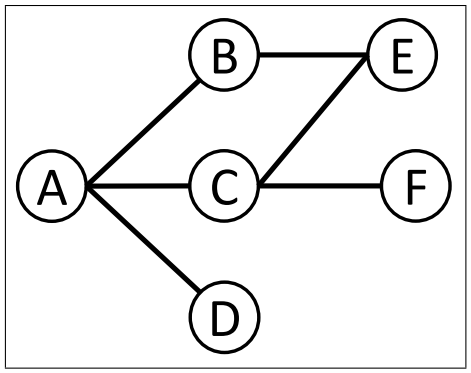
node v_j over an edge $e_{ij} = [v_i \rightarrow v_j]$ when there exists at least one edge between two nodes,

- $neighbor(v_i)$ is the set of all neighbors of the source node, v_i, and
- $p \in \mathbb{R}$ is a degree de-coupling weight.

Intuitively, the numerator term, $deg(v_j)^{-p}$, ensures that the edge incoming to v_j is weighted by its degree: if $p > 0$, then its degree negatively impacts (reduces) transition probabilities into v_j, if $p < 0$ then its degree positively impacts (boosts[2]) transition probabilities into v_j, and if $p = 0$, we obtain the standard PageRank formulation without degree de-coupling. In other words, the transition function satisfies our desideratum of de-coupling the transition process from node-degrees and penalizing or boosting the contributions of node degrees on-demand. Note that, since all transitions from the node v_i are degree de-coupled individually based on the degrees of their destinations, the denominator term, $\sum_{v_k \in neighbor(v_i)} deg(v_k)^{-p}$, ensures that the transition probabilities from node v_i add up to 1.0. Note also that when there is no edge between node v_i and v_j, $\mathbf{T}_D(j,i) = 0$ and, consequently, the term $\mathbf{T}_D(j,i)$ is not affected by the degree de-coupling process.

Example 2 *Figure 5.1 shows how the random walk probabilities are differentiated in a degree de-coupled transition matrix on a sample graph where a node A has three neighbors, B (with degree* 2*), C (with degree* 3*), and D (with degree* 1*). In conventional PageRank, the transition probabilities from node A to all its neighbor nodes are equal to* 0.33*. In degree de-coupled PageRank (D2PR), however, the value of* p *is used for explicitly accounting for the impact of node degree on the transition probabilities: When* $p = 2$*, the transition probabilities from A to its neighbors are 0.18, 0.08, and*

[2]In fact, a similar function was used in [29] to quickly locate nodes with higher degrees in a given graph.



Dest. v_j	*deg.* (v_j)	Transition probability from A to its neighbors v_j		
		$p = 0$	2	−2
B	2	0.33	0.18	0.29
C	3	0.33	0.08	0.64
D	1	0.33	0.74	0.07

(a) A sample graph (b) Transition probabilities from A

Figure 5.1: Transition Probabilities from Node $v_i = A$ to All Its Neighbors v_j in Different p Values

node id	node degree	Ranks of the graph nodes for different de-coupling weights (p)				
		−4	−2	0	2	4
53608	883	1	1	69	5549	6793
351	739	2	12	425	1992	1935
...	...	...	...	...	...	...
79538	1	7661	7545	4149	195	182
79917	1	7793	7790	7522	2443	2043

Table 5.2: Ranks of Graph Nodes of Different Degrees on a Sample Graph for Different De-coupling Weights, p

0.74, which penalizes nodes which have larger degrees, whereas when $p = -2$, D2PR boosts the transition probabilities to large degree nodes leading to transition probabilities 0.29, 0.64, and 0.07, respectively. ◇

This example shows that, in degree de-coupled PageRank (D2PR), as we also see in Table 5.2, the value of p can be used to penalize ($p > 0$) or boost ($p < 0$) transition probabilities based on the degree of the destination, v_j.

Directed Unweighted Graphs

The semantics of degree de-coupling is slightly different in directed graphs. In particular, edges incoming to v_i often do not require a particular effort from v_i to establish and hence are often out of the control of v_i, but indicate a certain degree of interestingness, usefulness, or authority as perceived by others. The same is not true for edges outgoing from v_i; in particular, a vertex with a large number of outgoing edges may either indicate a potential hub or simply indicate a non-discerning connection maker. The distinction between these two situations gains importance especially in applications where establishing a new connection has a non-negligible cost to the source node and, thus, a large number of outgoing edges may indicate either (a) a very strong participant to the network or (b) a very poor participant with a large number of weak linkages.

Let $G = (V, E)$ be a directed graph and for the simplicity of the discussion, without any loss of generality, let us assume that G is unweighted. Let us also be given a residual probability parameter, α and let $outdeg(v)$ be a function which returns the number of outgoing edges from the node v. The degree de-coupled PageRank (D2PR) scores can be represented in the form of a vector $\vec{d}$, $\vec{d} = \alpha \mathbf{T}_D \vec{d} + (1 - \alpha)\vec{t}$, where $\vec{t}$ is the teleportation vector, such that $\vec{t}[i] = \frac{1}{\|V\|}$ for all i and

$$\mathbf{T}_D(j, i) = \frac{outdeg(v_j)^{-p}}{\sum_{[v_i \to v_k] \in out\_edges(v_i)} outdeg(v_k)^{-p}},$$

where $\mathbf{T}_D(j, i)$ denotes the degree de-coupled transition probability from node v_i to node v_j over an edge $e_{ij} = [v_i \to v_j]$, $out\_edges(v_i)$ is the set of out-going edges from the source node, v_i, and $p \in \mathbb{R}$ is a degree de-coupling weight.

Example 3 *Figure 5.2 (a) in Section 5.3 provides an example illustrating the correlations between the degree de-coupled PageRank (D2PR) scores and external evidence*

Data	Graph	# of nodes	# of edge	Average node degree	Standard deviation of node degrees	Median standard deviation of neighbors' degrees
IMDB	movie-movie	191,602	4,465,272	23.30	51.86	2.89
	actor-actor	32,208	2,493,574	77.42	67.15	114.41
DBLP	article-article	8,808	951,798	108.06	171.25	309.92
	author-author	47,252	310,250	6.57	8.89	6.39
Last.fm	listener-listener	1,892	25,434	13.44	17.31	22.37
	artist-artist	17,626	2,640,150	149.79	299.66	998.53
Epinions	commenter-commenter	6,703	2,395,176	425.05	438.97	609.39
	product-product	13,384	2,355,460	175.99	224.12	202.78

Table 5.3: Data Sets and Data Graphs

for different values of p for some application: here, the higher the correlation, the better resulting ranking reflects the application semantics. As we see in this example, which we will investigate in greater detail in Section 5.3, the optimal de-coupling weight is not always $p = 0$ as implied by the conventional PageRank measure. In this particular case, for example, the correlation between D2PR and external evidence of significance is maximized when the de-coupling weight, p, is equal to 0.5, implying that in this application a moderate degree of penalization based on the node degrees is needed to align PageRank scores and application semantics. ◇

Weighted Graphs

Once again, the semantics of degree de-coupling need to be reconsidered for weighted graphs. Let $G = (V, E, w)$ be a directed, weighted graph, where $w(e)$ is a function which returns the weight of the edge associated with edge e. It is important to note that, in such a graph, the weight of an edge can 1) indicate the strength of the connection between two nodes (thus positively contributing to the significance of the destination node); and at the same time and 2) contribute to the degree of a node as a multiplier (thus positively or negatively contributing to the node significance

depending on the degree-sensitivity of the application). In other words, given an edge $e_{ij} = [v_i \rightarrow v_j]$, from node v_i to node v_j, the transition probability from v_i to v_j can be written as

$$\mathbf{T}(j,i) = \beta \mathbf{T}_{conn\_strength}(j,i) + (1-\beta)\mathbf{T}_D(j,i),$$

where

$$\mathbf{T}_{conn\_strength}(j,i) = \frac{w(v_i \rightarrow v_j)}{\sum_{[v_i \rightarrow v_h] \in out\_edges(v_i)} w(v_i \rightarrow v_h)},$$

accounts for the connection strength (as in the conventional PageRank) whereas $\mathbf{T}_D$ is a degree de-coupled transition matrix,

$$\mathbf{T}_D(j,i) = \frac{\Theta(v_j)^{-p}}{\sum_{[v_i \rightarrow v_k] \in out\_edges(v_i)} \Theta(v_k)^{-p}},$$

such that, $\mathbf{T}_D(j,i)$ denotes the degree de-coupled transition probability from node v_i to node v_j over an edge $e_{ij} = [v_i \rightarrow v_j]$, $p \in \mathbb{R}$ is a degree de-coupling weight, and

$$\Theta(v) = \sum_{[v \rightarrow v_h] \in out\_edges(v)} w(v \rightarrow v_h).$$

Note that, above, β controls whether accounting for the connection strength or degree de-coupling is more critical in a given application. In Section 5.3, we will study the impact of degree de-coupling in weighted graphs for different scenarios.

5.3 Case Studies

In this section, we present case studies assessing the effectiveness of the degree de-coupling process and the relationship between the degree de-coupling weight p and recommendation accuracy for different data graphs.

5.3.1 Setup

For all experiments, the degree de-coupling weight, p, is varied between -4 and 4 with increments of 0.5. The residual probability, α, is varied between 0.5 and 0.9,

with default value chosen as 0.85. We also varied the β parameter, which controls whether accounting for the connection strength or degree de-coupling is more critical in a given application, between 0.0 and 1.0, with the default value set to 0 (indicating full decoupling).

Datasets

Four real data sets are used for the experiments. Each data set is used to create two distinct data graphs and corresponding ratings data. Table 5.3 provides further details about the various graphs created using these four data sets. These recommendation tasks based on these data graphs are detailed below:

• For the **IMDB** [50] data set, we created (a) a movie-movie graph, where movie nodes are connected by an edge if they share common contributors, such as actors, directors, writers, composers, editors, cosmetic designers, and producers and (b) an actor-actor graph based on whether two actors played in the same movie. **Applications:** For this data set, we consider applications where movies are rated by the users: thus, we merged the IMDB data with the MovieLens 10M [47] data (based on movie names) to identify user ratings (between 1 and 5) for the movies in the graph. We consider the (a) average user rating as the significance of the movies in the movie-movie graph and (b) average user rating of the movies played in as the significance of the actors in the actor-actor graph.

• For the **DBLP** [103] data set, we constructed (a) an article-article graph where scientific articles were connected to each other if they shared a co-author and (b) an author-author graph based on co-authorship. **Applications:** (a) In the article-article graph, the number of citations to an article is used to indicate its significance. Similarly, (b) in the author-author graph, average number of citations to an author's papers is used as his/her significance.

• For the **Last.fm** [21], we constructed (a) a listener-listener graph, where the nodes are Last.FM listeners and undirected edges reflect friendship information among these listeners. We also constructed (b) an artist-artist graph based on shared listeners. **Applications:** (a) In the listener-listener graph, we considered the total listening activity of a given listener as his/her significance. (b) In the artist-artist graph, the umber of times an artist has been listened is considered as his/her significance.

• For the **Epinions** [102]: We constructed (a) a commenter-commenter graph based on the products on which two individuals both commented and (b) a product-product graph based on shared commenters. **Applications:** (a) For the nodes on the commenter-commenter graph, the number of trusts the commenter received from others is used as his/her commenter significance. (b) For each product in the product-product graph, its average rating by the commenters is used as its node significance.

5.3.2 Measures

In this section, our goal is to observe the impact of different D2PR degree de-coupling weights on the relationship between D2PR rankings and application specific significance measures for the above data sets[3]. We also aim to verify whether de-coupling weights can also be used to improve recommendation accuracies.

In order to measure the relationship between the degree de-coupled PageRank (D2PR) scores and the application-specific node significance, we used 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 the D2PR ranks of the nodes in the graph

[3]In this chapter, we are not proposing a new PageRank computation mechanism. Because of this (and since the focus is not improving scalability of PR), we do not report execution times and compare our results with other PageRank computation mechanisms.

and their application-specific significances. Here, x are rankings by D2PR and y are significances for an application and $\bar{x}$ and $\bar{y}$ are averages of two values.

5.3.3 Impact of De-Coupling in Different Applications (Unweighted Graphs)

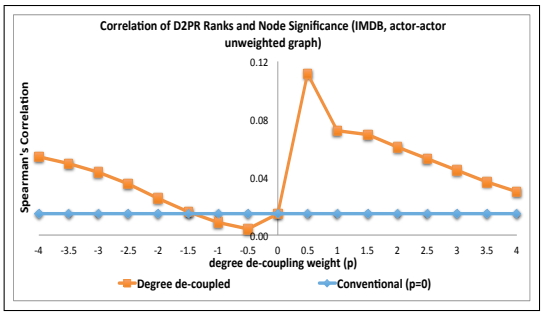
In this subsection, we present results that aim to assess D2PR under the settings described above. For these experiments, the residual probability, α, and the parameter, β, are set to the default values, 0.85 and 0, respectively. In these experiments, we consider only unweighted graphs (we will study the weighted graphs and the impact of parameter β later in Section 5.3.5).

Figures 5.2 through 5.4 include charts showing the Spearman's correlations between the D2PR ranks and application specific node significances for different values of p and for different data graphs. These ./figures/chap5 clearly illustrate that different data graphs require different degrees of de-coupling[4] to best match the application specific node significance criterion.

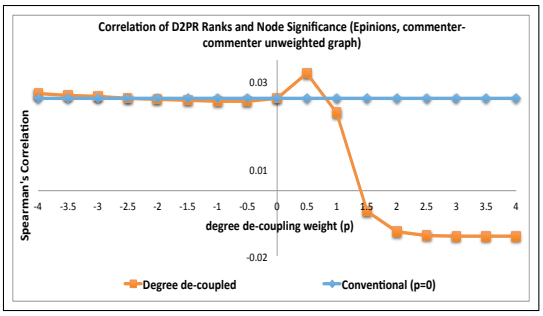
Application Group A: When Degree Penalization Helps

The actor-actor (based on common movies) and commenter-commenter (based on common products) graphs have highest correlation at $p = 0.5$, with the correlations dropping significantly when the degrees are over-penalized (i.e., when $p \gg 0.5$). The Epinions product-product graph (based on common commenters, Figure 5.2(c)) also provides the highest correlations with $p > 0$, but behaves somewhat differently from the other two cases: the correlations stabilize and do not deteriorate significantly when degrees are over-penalized, indicating that the need for degree penalization is especially critical in this case: this is due to the fact that, the larger the number of comments a product has, the more likely it is that the comments are negative

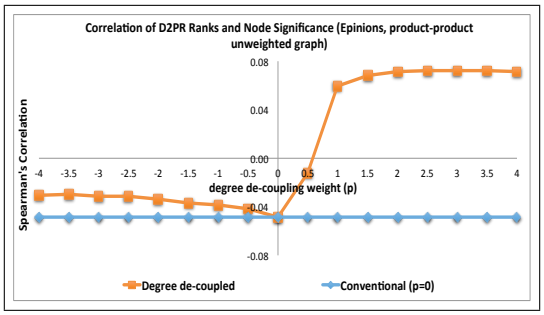
[4]Degree penalization or degree-based boosting



(a) IMDB (actor-actor)



(b) Epinions (commenter-commenter)

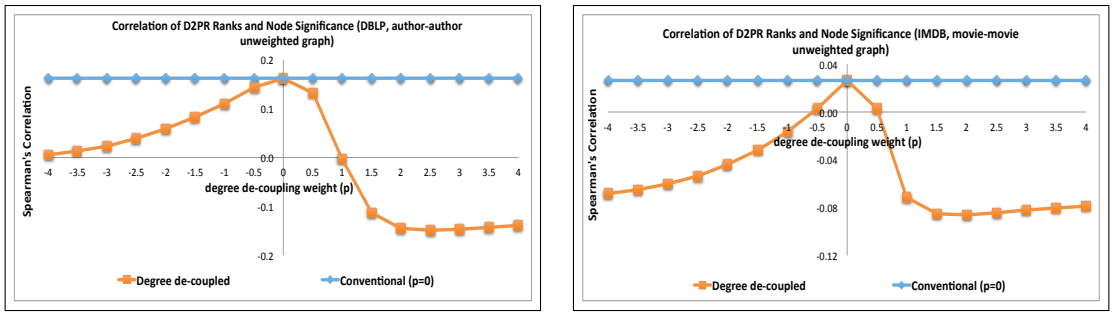


(c) Epinions (product-product)

Figure 5.2: Application Group A: $p > 0$ is Optimal (i.e., Node Degrees Need to Be Penalized)

(Figure 5.5). In fact, we see that, among the three graphs, this is the only graph where the traditional PageRank (with $p = 0$) leads to **negative correlations** between node ranks and node significances.

These results indicate that actors who have had many co-actors, commenters who commented on products also commented by many others, or products which received comments from individuals who also commented on many other products are not good candidates for transition during random walk. This aligns with our expectation that, in applications where each new movie role or comment requires additional effort, high degree may indicate lower per-movie or per-comment effort and, hence, lower significance.



(a) DBLP (author-author) (b) IMDB (movie-movie)

Figure 5.3: Application Group B: $p = 0$ Is Optimal

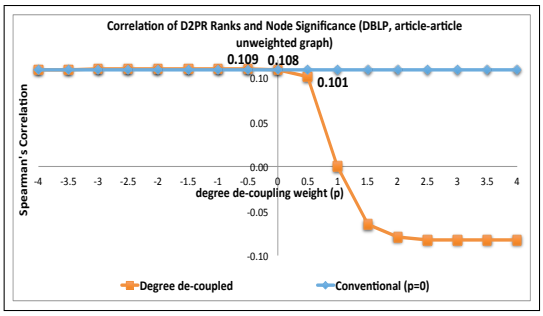
Application Group B: When Conventional PageRank is Ideal

Figure 5.3 shows that, for movie-movie (based on common actors) and author-author (based on common articles) graphs, the peak correlation is at $p = 0$ indicating that the conventional PageRank which gives positive weight to node degree, is appropriate.

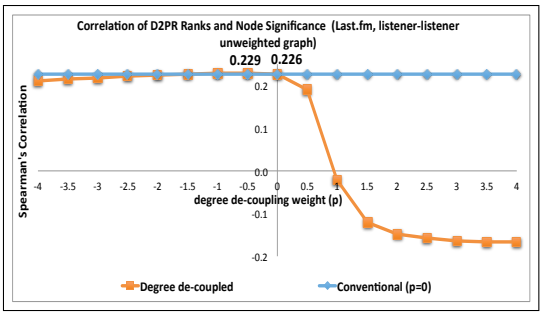
This perhaps indicates that movies with a lot of actors tend to be big-budget products and that authors with a large number of co-authors tend to be experts with whom others want to collaborate. Note that, in these applications, additional boosting, with $p < 0$, negatively affects the correlation, indicating that the relationship between node degree and significance is not very strong (Figure 5.5). The quick change when $p < 0$ is because, as we see in Table 5.3, median standard deviations of neighbors' degrees are low; i.e., degrees of neighbors of a node are comparable: there is no dominant contributor to $\mathbf{T}_D(j, i)$ in Equation 5.1 (Section 5.2) and, thus, the transition probabilities are sensitive to changes in p, when $p < 0$.

Application Group C: When Degree Boosting Helps

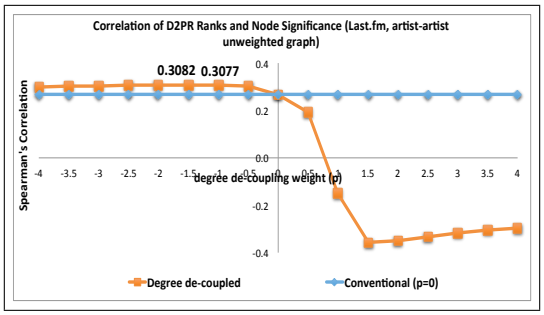
Figure 5.4 shows that there are scenarios where additional boosting based node degrees provides some benefits. The article-article (based on common authors), listener-listener (based on common artists), and artist-artist (based on common listeners)



(a) DBLP (article-article)



(b) Last.fm (listener-listener)



(c) Last.fm (artist-artist)

Figure 5.4: Application Group C: $p < 0$ Is Optimal (i.e., Node Degrees Need to Be Boosted)

graphs reach their peaks around $p \sim -1$, indicating that these also benefit from large node degrees though improvements over $p = 0$ are slight.

A significant difference between applications in Group B and Group C is that, for $p < 0$, the correlation curve is more or less stable. This is because, as we see in Table 5.3, in these graphs median standard deviations of neighbors' degrees are high: in other words, for each node, there is a dominant neighbor with a high degree and this neighbor has the highest contribution to $\mathbf{T}_D(j, i)$; thus, the rankings are not very sensitive to p, when $p < 0$.

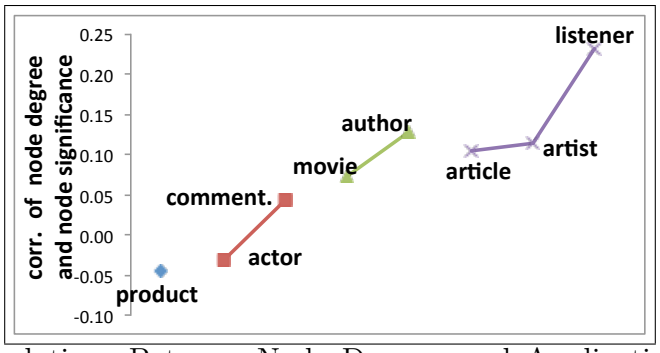


Figure 5.5: Correlations Between Node Degrees and Application specific Significances for Different Data Graphs

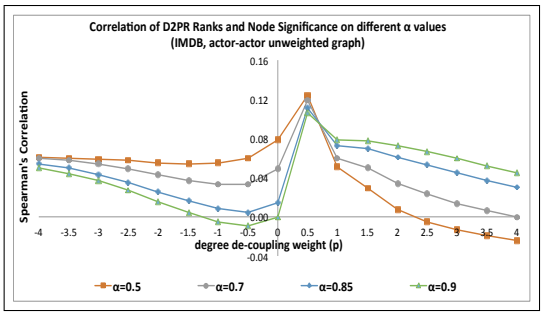
Summary: Correlations between Node Degrees and Application Specific Significances

The experiments reported above show that degree de-coupling is important as different applications, even on the same data set, may associate different semantics to node degrees and the conventional PageRank scores are too tightly coupled with node degrees to be effective in all scenarios. Figure 5.5, which plots correlations between node degrees and application specific significances for different data graphs, re-confirms that the ideal value of the p is related to the usefulness of the node degree in capturing the application specific definition of node significance.

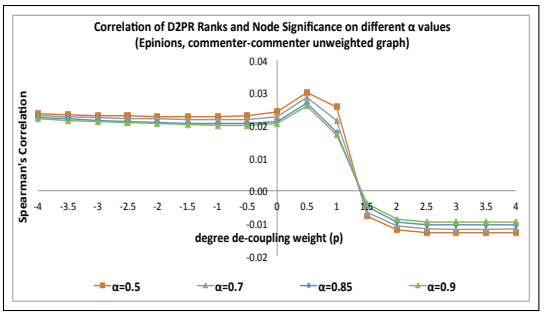
5.3.4 Relationship between α and $\mathbf{p}$

In Figures 5.6 through 5.8, we investigate the relationship between the value α and the degree de-coupling parameter p for different application types. Here we use the default value, 0, for the parameter β and present the results for unweighted graphs (the results for the weighted graphs are similar).

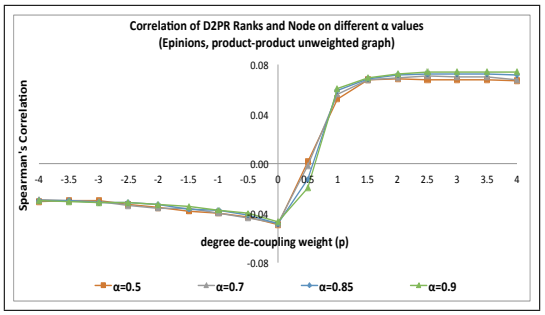
First thing to notice in these ./figures/chap5 is that the grouping of the applications (into those where, respectively, $p > 0$, $p = 0$, or $p < 0$ is useful) is preserved when different values of α are considered.



(a) IMDB (actor-actor)



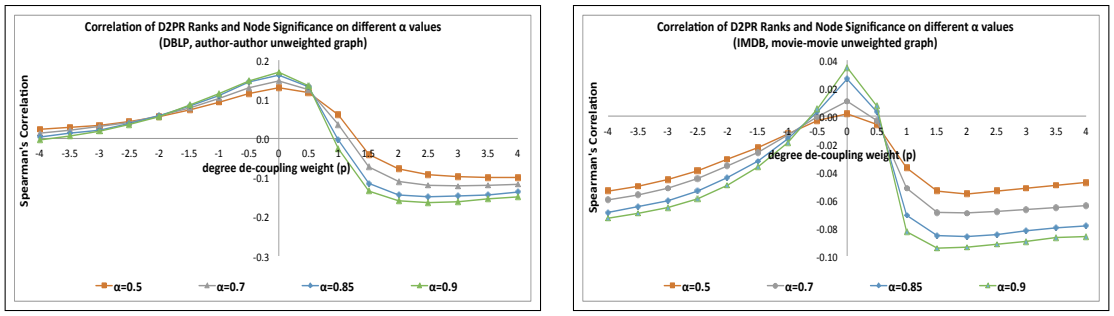
(b) Epinions (commenter-commenter)



(c) Epinions (product-product)

Figure 5.6: Relationship Between p and α, for Application Group A, Where $p > 0$ Is Optimal (i.e., Degrees Need to Be Penalized)

Figure 5.6 studies the impact of the value of α in application group A, where degree penalization helps ($p > 0$). As we see here, for the IMDB actor-actor (Figure 5.6(a)) and Epinions commenter-commenter (Figure 5.6(b)) graphs, having a lower value of α (i.e., lower probability of forward movement during the random walk) provides the highest possible correlations between D2PR ranks and node significance (with the optimal value of p being ~ 0.5 independent of the value of α). This indicates that in these graphs, it is not necessary to traverse far during the random walk. Interestingly, though, when degrees are over-penalized (i.e., $p \gg 0$), smaller values of α start leading to worse correlations, indicating that (while not being optimal) severe penalization of node degrees helps make random traversals more useful than random jumps. As we have already observed in Figure 5.2(c), the Epinions product-product graph (Figure 5.6(c)) behaves somewhat differently from the other two cases



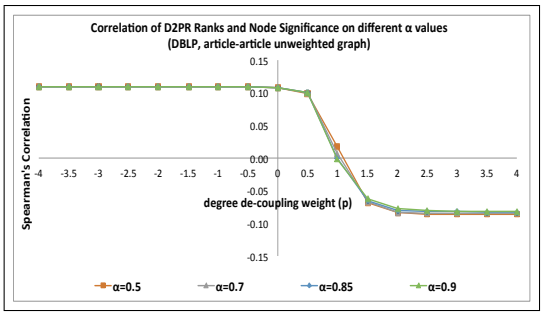
(a) DBLP (author-author) (b) IMDB (movie-movie)

Figure 5.7: Relationship Between p and α, For Application Group B, Where $p = 0$ Is Optimal

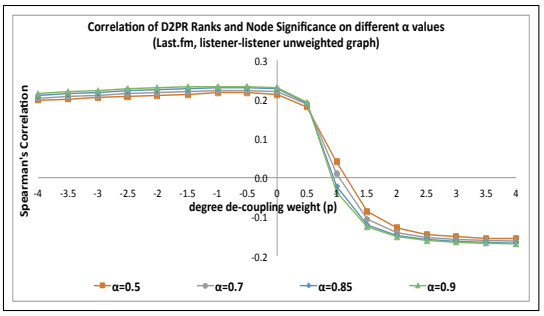
where degree penalization ($p > 0$) leads to larger correlations: in this case, unlike the other two graphs, the highest possible correlations between D2PR ranks and node significance are obtained for large values of α, indicating that this application benefits from longer random walks (though the differences among the correlations for different α values are very small).

Figure 5.7 shows that the pattern is different for application group B, where conventional PageRank is ideal ($p = 0$): in this case, having a larger value of α (i.e., larger probability of forward movement during the random walk) provides the highest correlations between ranks and significance. Interestingly, in these applications, when $p \ll 0$ or $p \gg 0$, higher probabilities of random walk traversal (i.e., larger α) stop being beneficial and lower values of α lead to larger correlations. This re-confirms that, for these applications, $p \sim 0$ leverages the random walk traversal the best.

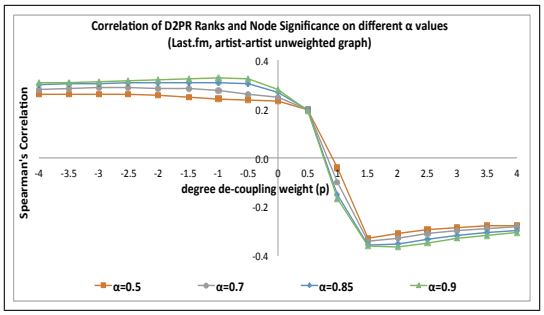
As we see in Figure 5.8, in application group C, where degree boosting helps ($p < 0$), it is also the case that larger values of α (i.e., larger probabilities of forward transitions during the random walk) provides the highest correlations between node ranks and significance. On the other hand, in these applications, $p \sim 0.5$ serves as a balance point where the value of α stops being relevant; in fact, for $p > 0.5$ the higher values of α stops being beneficial and lower values of α lead to larger correlations. This



(a) DBLP (article-article)



(b) Last.fm (listener-listener)



(c) Last.fm (artist-artist)

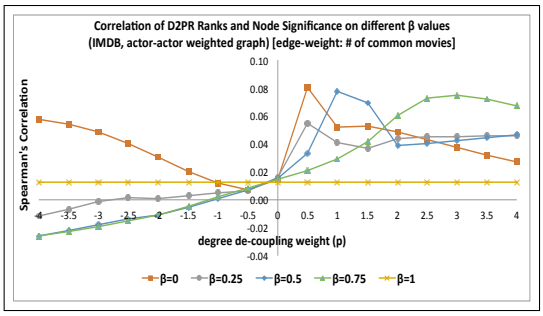
Figure 5.8: Relationship Between p and α, for Application Group C, Where $p < 0$ Is Optimal (i.e., Node Degrees Need to Be Boosted)

re-confirms that smaller values of p (which provides degree boosting) help leverage the random walk traversal the best.

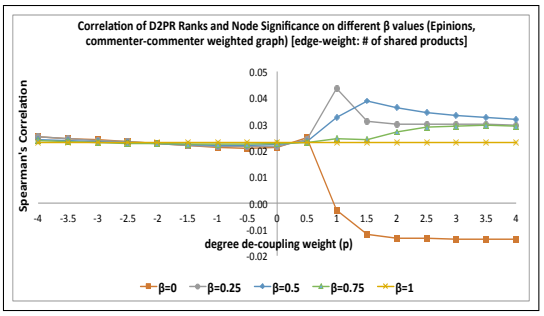
5.3.5 Relationship between β and **p** *in Weighted Graphs*

Finally, in Figures 5.9 through 5.11, we investigate the relationship between the value β (which controls whether accounting for the connection strength or degree de-coupling is more critical in a given application) and the degree de-coupling parameter p for different application types. Here we use the default value, 0.85, for the parameter α and present the results for weighted graphs:

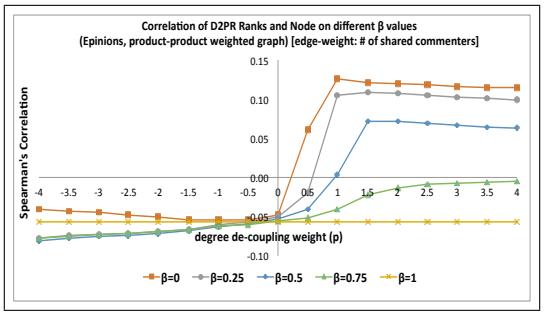
Figure 5.9 depicts the impact of the value of the parameter β in application group A, where degree penalization helps ($p > 0$). As we see here, for all three weighted graphs, performing degree penalization (i.e., $\beta < 1.0$) provides better rank-significance



(a) IMDB (actor-actor)



(b) Epinions (commenter-commenter)

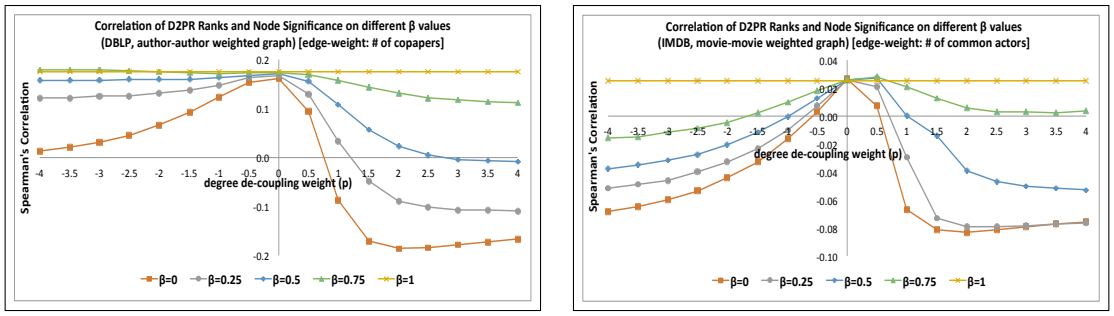


(c) Epinions (product-product)

Figure 5.9: Relationship Between p and β, for Application Group A, Where $p > 0$ Is Optimal (i.e., Node Degrees Need to Be Penalized)

correlation than relying solely on the connection strength (i.e., $\beta = 1.0$). Note that the value of β impacts the optimal value of degree penalization parameter p: the more weight is given to connection strength (i.e., the greater β is), the larger is the optimal value of p.

Figure 5.10 shows that, for applications in group B, where $p \sim 0$ is ideal, when the connection strength is given significantly more weight than degree de-coupling (i.e., $\beta \sim 0$), we observe high rank-significance correlations. Interestingly however, for the movie-movie graph (where the edge weights denote common actors) the highest correlations are obtained not with $p = 0$, but with $p = 0.5$ and $\beta = 0.75$, indicating that degree penalization is actually beneficial in this case: movies that share large numbers of actors with other movies are likely to be B-movies, which are not good candidates for transitions during the random walk.

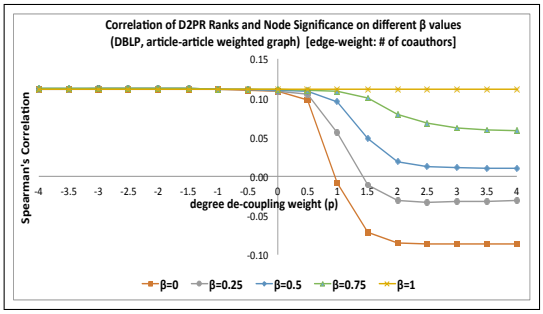


(a) DBLP (author-author) (b) IMDB (movie-movie)

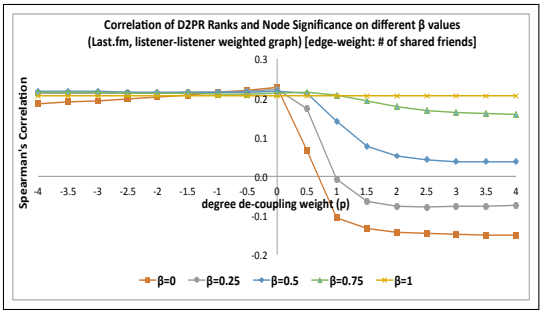
Figure 5.10: Relationship Between p and β, for Application Group B, Where $p = 0$ Is Optimal

Figure 5.11 shows that in application group C, where degree boosting ($p < 0$) helps, giving more weight to connection strength (i.e., $\beta \sim 1.0$) is a good, but not necessarily the best strategy. In fact, in these graphs, the highest overall correlations are obtained with $\beta = 0$ or $\beta = 0.25$, indicating that degree de-coupling is beneficial also in these cases. Interestingly, (unlike the case with the unweighted listener-listener graph, where the best correlation was obtained when $p < 0$) for the weighted version of the listener-listener graph (where edge weights denote the number of shared friends), when $\beta = 0$ through 0.5, $p = 0$ provides the highest correlation and when $\beta = 0.75$, $p = 0.5$ provides the highest correlation – these indicate that listeners who have large numbers of shared friends with others are good candidates for random walk.

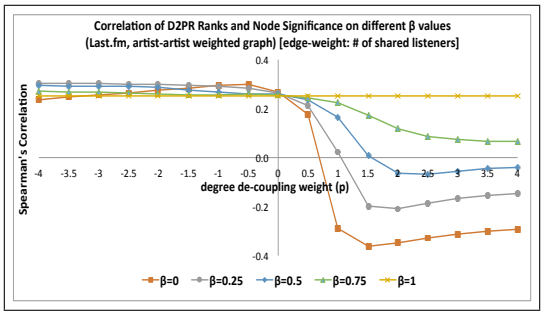
Note that a key observation from the above results is that the conventional PageRank, based on connection strength (i.e., $\beta = 1.0$), is not always the best strategy for the applications considered.



(a) DBLP (article-article)



(b) Last.fm (listener-listener)



(c) Last.fm (artist-artist)

Figure 5.11: Relationship Between p and β, for Application Group C, Where $p < 0$ Is Optimal (i.e., Node Degrees Need to Be Boosted)

Chapter 6

PERSONALIZED PAGERANK IN UNCERTAIN GRAPHS WITH MUTUALLY EXCLUSIVE EDGES

6.1 Introduction

Measures of node ranking are used in many web and social media based prediction and recommendation applications [19, 54, 76, 100, 97]. Due to wide-spread use of graphs in analysis, mining, and visualization of interconnected data, in the literature there are several ways to rank nodes in a graph ranking, including the well known personalized PageRank (PPR) measure [11, 23], which weights the nodes in a given graph based on their positions relative to a given seed set of nodes.

Despite their effectiveness, the measures become difficult to use in the case when a graph contains uncertainty. Most of measures on node ranking computations are designed assuming that a certain information is given but unfortunately, in many real world applications, it may not be possible to obtain a perfect and intact information (structure) of the graph for various reasons:

- **missing/stale information:** edges and nodes can be missed when there are data crawling problems
- **noise:** the existence of edges can be changed or uncertain when noise is included in measurements
- **privacy:** the data can be modeled with uncertainty because of obfuscating the identity of users for privacy reasons

- **outdatedness:** The webs and social network graphs are continually changed, so it is not possible to know a structure of the network at a time

Measuring node proximity in an uncertain graph is especially difficult, because adding or removing one single edge in a given graph can have a drastic effect on the proximity of nodes in it [30, 32]: addition of just one edge may be sufficient to link two otherwise distant clusters of nodes, thereby significantly altering the proximities of a large number of pairs of nodes in the graph.

Given that, under graph uncertainty, we may end up with different node proximity measurements based on which interpretation of the available graph data we believe, one way to deal with this uncertainty is to attempt to measure expected node proximities, which take into account the likelihood of different interpretations and the node proximity measurements corresponding to each interpretation.

A naive way to deal with this challenge would be to measure expected node proximities by taking into account the likelihoods of different interpretations and the node proximity measurements corresponding to each interpretation:

1. one can first enumerate all possible interpretations (or possible worlds) of the uncertain graph, where each interpretation is a possible certain graph;

2. one can then compute node proximity under each possible world; and

3. finally, one can combine all these node proximity measurements into a single expected proximity value.

It is, however, easy to see that an exhaustive enumeration based approach will quickly become intractable since (as we see in Section 6.2) the number of possible worlds can grow exponentially with the amount of uncertainty in the graph. To tackle this challenge, in this chapter, I propose an efficient Uncertain Personalized PageRank

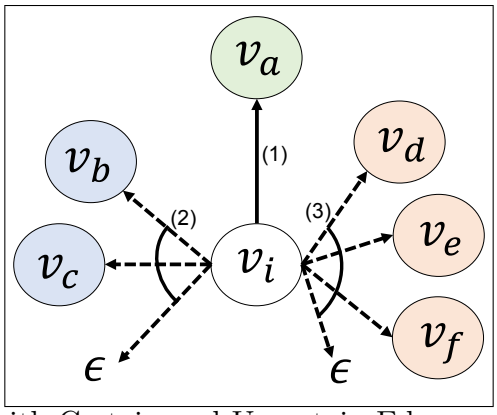


Figure 6.1: A Graph with Certain and Uncertain Edges

(UPPR) algorithm to approximately compute personalized PageRank values on an uncertain graph with edge uncertainties. The proposed UPPR approach avoids enumeration of all possible worlds, yet it is able to achieve comparable accuracy by carefully encoding edge uncertainties in a data structure that leads to fast approximations. Experimental results for different data sets show that UPPR is very efficient in terms of execution time (multiple orders faster than other algorithms with similar accuracy) and its accuracy is close to perfect.

In the next section, I introduce the uncertain graph model relied on in the chapter. In Section 6.3, I discuss alternative "naive" techniques and discuss their individual shortcomings. Then, in Section 6.4, I present the proposed efficient and effective uncertain personalized PageRank (UPPR) technique. I evaluate the various techniques discussed in the chapter in Section 6.5 using several data sets.

6.2 Problem Formulation

Let $G = (V, E)$ be a directed graph with a set, V, of nodes and a set, E, of edges. Conventionally, each edge $e \in E$ is defined using two nodes in the graph: a source node $source(e) \in V$ and a target node $target(e) \in V$. In this chapter, on the other hand, we divide the graph edges into certain and uncertain edges.

Definition 9 (Certain edges) *A certain edge $e_+ \in E$ has a well defined source node, v_{source} and a well defined target node, v_{dest}. We denote this with $source(e_+) = \{v_{source}\}$ and $target(e_+) = \{v_{dest}\}$. Also, we denote the subset of E consisting of all of E's certain edges as E_+.* ◇

In Figure 6.1, $e_{+_{(1)}} = \{\langle v_i, v_a \rangle\}$ is a certain edge from v_i to v_a. Note that, since $\|source(e_+)\| = \|target(e_+)\| = 1$, this edge type does not include any uncertain information. In this chapter, we refer to this certainty as having a unique possible world. Each uncertain edge, on the other hand, can represent multiple possible worlds:

Definition 10 (Uncertain Edges) *An uncertain edge $e_- \in E$ has a well defined source node but does not have a well defined target node.More specifically, we have*

- $source(e_-) \subseteq V$,
- $target(e_-) \subseteq V \cup \{\epsilon\}$ *and* $target(e_-) \neq \{\epsilon\}$, *and*
- $\|source(e_-)\| = 1$ *and* $\|target(e_-)\| > 1$.

Above ϵ denotes a non-existing node. We denote the subset of E consisting of all of E's uncertain edges as E_-. ◇

Figure 6.1 includes two uncertain edges, $e_{-_{(2)}}$ and $e_{-_{(3)}}$ with different degrees. Note that in Figure 6.1, the uncertain edge $e_{-_{(3)}}$ captures a form of uncertainty with mutual exclusion among the edges from v_i to v_d, v_e, or v_f. This uncertainty, however, is independent from the existence uncertainty of $e_{-_{(2)}}$. Therefore, the proposed model allows as a special case the independent existence uncertainty model considered by many of the existing works [13, 28, 60, 75, 118, 116, 93, 121].

6.2.1 Possible Worlds of an Uncertain Edge

Each uncertain edge implicitly defines multiple possible worlds in which different interpretations are valid:

Definition 11 (Possible Worlds of an Edge under Mutual Exclusion Semantics) *Let $e \in E$ be an edge. Let $source(e)$ denote a source node of the edge and let $target(e) \subseteq V \cup \{\epsilon\}$ denote the potential targets of the edge. Given this edge, we define all possible worlds covered by this edge, $pw_{unique}(e)$, under mutual exclusion semantics as*

$$\Big\{\langle v_i, v_j\rangle \mid (v_i = source(e)) \ \wedge \ (v_j \in target(e))\Big\}$$

The possible worlds covered by an uncertain edge consist of all combinations of target nodes; if a target node is potentially non-existent, then it is also a possible world. $\|pw_{unique}(e)\| = \|target(e)\|$ is the number of possible worlds on the edge, e ◇

In the example visualized in Figure 6.1, there are three possible worlds defined by $e_{-(2)}$ $(= \{\langle v_i, v_b\rangle, \langle v_i, v_c\rangle, \langle v_i, \epsilon\rangle\}$ – the last one implying that this edge does not exist) and four possible worlds defined by $e_{-(3)}$ $(= \{\langle v_i, v_d\rangle, \langle v_i, v_e\rangle, \langle v_i, v_f\rangle, \langle v_i, \epsilon\rangle\}$ – again the last one implying that this edge does not exist).

Note that under a more general interpretation, more than one of the potential combinations, implied by the uncertainty encoded in the edge, may be possible in the real world.

Definition 12 (Possible Worlds of an Edge under Multiple Edge Semantics) *Let $e \in E$ be a certain or uncertain edge and $pw_{unique}(e)$ be the corresponding possible worlds covered by this edge under mutual exclusion semantics. Given this edge, we define all possible worlds covered by this edge under multiple edge semantics as all possible non-empty subsets of its target set. Note that, since a possible world containing*

ϵ is equivalent to the world where ϵ has been removed, we have

$$\|pw_{multiple}(e)\| = \begin{cases} 2^{(\|pw_{unique}(e)\|-1)}, & \epsilon \in target(e) \\ 2^{\|(pw_{unique}(e)\|)} - 1, & otherwise \end{cases} \quad \diamond$$

Under these semantics, in the example in Figure 6.1, there would be $2^{(3-1)} = 4$ possible worlds defined by the uncertain edge $e_{-_{(2)}}$ and $2^{(4-1)} = 8$ possible worlds defined by $e_{-_{(3)}}$. For the certain edge $e_{(1)}$, this gives $2^{(1-1)} = 1$ possible world.

6.2.2 Possible Worlds of a Graph

Given the above definitions, we can now define the possible worlds of a graph with uncertainty:

Definition 13 (Possible Worlds of a Graph) *Let $G = (V, E)$ be a directed graph which has a set of nodes V and a set of edges E. For all $e \in E$, let $pw(e)$ denote the possible worlds (under mutual exclusion or multiple edge semantics) of the edge e. We define all possible worlds covered by this graph as the Cartesian product of the possible worlds of edges:*

$$pw(G) = \bigtimes_{e \in E} pw(e). \quad \diamond$$

If we reconsider the example in Figure 6.1, under mutual exclusion semantics, this graph would have $1 \times 3 \times 4 = 12$ possible worlds. In contrast, under the multiple edge semantics, the graph would have $1 \times 4 \times 8 = 32$ possible worlds. Note that, since only uncertain edges have ≥ 2 possible worlds, it is easy to see that the size of the $pw(G)$ grows exponentially in the number of uncertain edges; i.e., $\|pw(G)\|$ is $O(2^{\|E_-\|})$.

6.2.3 PPR under Uncertainty

We now define personalized PageRank under uncertainty.

Definition 14 (Personalized PageRank under Uncertainty) *Let $G(V, E)$ be an uncertain graph. Given a seed set, S, of nodes we can define the personalized PageRank vector, $\overrightarrow{r}$, for graph G as follows:*

$$\overrightarrow{r} = \underset{G_i \in pw(G)}{AVG} PPR(G_i, S),$$

where G_i denotes a possible world implied by the uncertain graph G and $PPR(G_i, S)$ returns a personalized PageRank vector, $\overrightarrow{r}_i$, corresponding to G_i and seed set S. $\diamond$

Intuitively, under the assumption that all possible worlds are equally likely, the above definition of personalized PageRank corresponds to the values of the node scores.

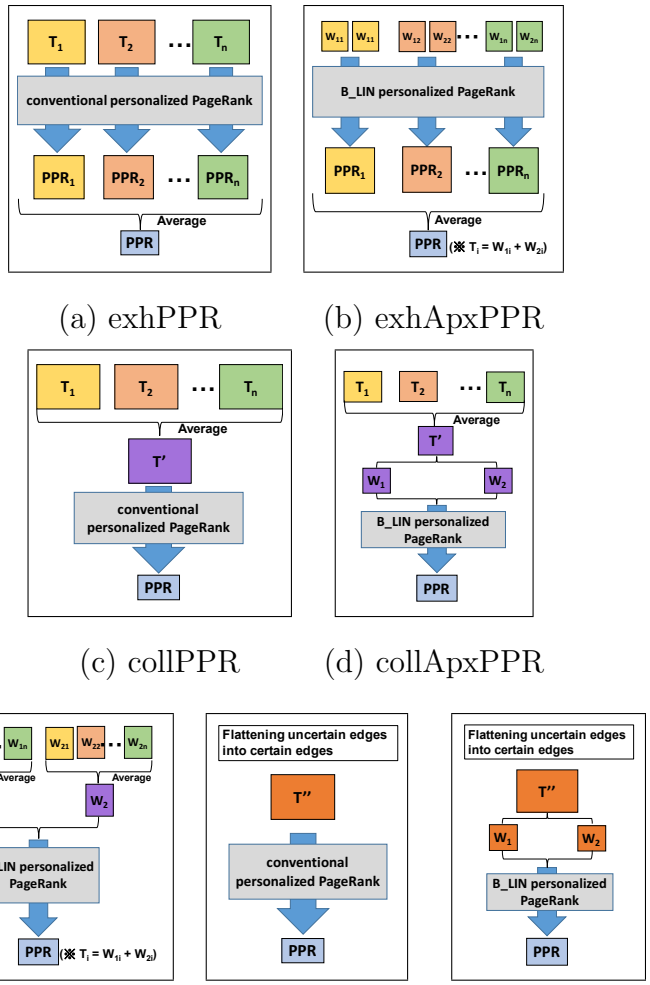
6.3 Naive Approaches

In this section, we present several (naive) approaches for computing PPR values on an uncertain graph (Figure 6.2):

6.3.1 Exhaustive Approaches

The most straightforward way to obtain the PPR values on an uncertain graph is to exhaustively enumerate all possible worlds, compute the PPRs for each possible world, and combine (i.e., average) the results. Obviously this exhaustive approach (exhPPR), visualized in Figure 6.2(a), is likely to be very expensive as it involves potentially exponential number of PPR computations.

One way to alleviate this cost is to rely on a fast approximate PPR technique (such as B_LIN [106], which partitions the given graph into subgraphs and pre-processes intra-partition edges, W_1, and inter-partition edges, W_2, on these subgraphs in a post-processing phase) to obtain PPR scores for each possible world (Figure 6.2(b)). Note that, while this exhaustive approximate approach, which we refer to as exhApxPPR,



(a) exhPPR (b) exhApxPPR

(c) collPPR (d) collApxPPR

(e) collApx2PPR (f) flatPPR (g) flatApxPPR

Figure 6.2: Alternative (Naive) Approaches for Computing PPR values on an Uncertain Graph

is likely to be faster than the basic approach, since it involves exponential number of (approximate) PPR computations, it is still likely to be prohibitively expensive.

6.3.2 Collapsing-based Approaches

Since the major cost of the exhaustive approach is the number of exhaustive PPR computations, one way to reduce the cost would be to enumerate all possible transition matrices corresponding to all possible worlds of the uncertain graph and then collapse

these transition matrices into a single transition matrix by taking their average. After this, we can obtain the final PPR scores either by solving an exact PPR (collPPR, Figure 6.2(c)) or approximate PPR (collApxPPR, Figure 6.2(d)) problem.

Another alternative is to first partition each individual transition matrix of each possible world, G_i, and then collapse the intra-partition, W_{1i}, and inter-partition, W_{2i}, transition matrices for all possible worlds into aninter-partition and an intra-partition matrix to be processed using B_LIN[106] and combined in a post-processing phase. In Figure 6.2(e), we refer to this pre-partitioning based alternative approach as collApx2PPR.

Accuracy Problem with Collapsing: The collapsing based approach can lead to relatively large errors when uncertainty is concentrated around nodes with large PPR scores: Let G be an uncertain graph with two possible worlds with transition matrices, T_1 and T_2, respectively. Given these, we can compute the expected PPR scores as defined in the previous section as

$$\vec{r} = \frac{\vec{r}_1 + \vec{r}_2}{2} = \frac{\alpha\left(T_1\vec{r}_1 + T_2\vec{r}_2\right)}{2} + (1-\alpha)\vec{s},$$

where $\vec{s}$ is the teleportation vector representing the seeds. In contrast, when using the collapsing based approach we instead compute

$$\vec{r}\,' = \alpha\left(\frac{T_1 + T_2}{2}\right)\vec{r}\,' + (1-\alpha)\vec{s}.$$

Given these, the error term, $\vec{e} = \vec{r} - \vec{r}\,'$ can be obtained as

$$\vec{e} = \frac{\alpha\left(T_1\vec{r}_1 + T_2\vec{r}_2\right)}{2} - \alpha\left(\frac{T_1 + T_2}{2}\right)\vec{r}\,'.$$

Assuming that this error term is relatively small; i.e., $\vec{r} \sim \vec{r}\,'$, we can replace $\vec{r}\,'$ with $\vec{r} = (\vec{r}_1 + \vec{r}_2)/2$, to obtain

$$\begin{aligned}\vec{e} &\sim \frac{\alpha\left(T_1\vec{r}_1 + T_2\vec{r}_2\right)}{2} - \alpha\left(\frac{T_1 + T_2}{2}\right)\left(\frac{\vec{r}_1 + \vec{r}_2}{2}\right)\\ &\sim \left(\frac{T1 - T2}{4}\right)\vec{r}_1 + \left(\frac{T2 - T1}{4}\right)\vec{r}_2.\end{aligned}$$

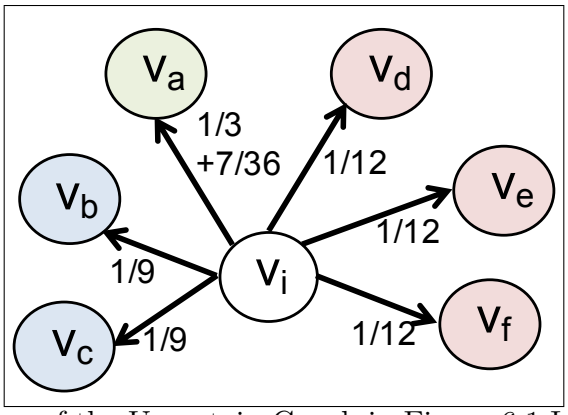


Figure 6.3: Flattening of the Uncertain Graph in Figure 6.1 Into an (Approximate) Certain Graph

In other words, in the collapsing based approach, the error term is especially large when the uncertainties (i.e., differences between the transition matrices of the possible worlds) are concentrated around nodes with large PPR scores.

Execution Time Problem with Collapsing: Since they reduce the number of PPR computations to just one, the collapsing based approaches are likely to be much faster than the exhaustive approach. Nevertheless, since it involves the enumeration of all possible worlds before obtaining the collapsed transition matrix, the cost of this technique is still exponential in the number of uncertain edges.

6.3.3 Flattening-based Approaches

An alternative approach to avoid the enumeration cost of collapsing is to approximate the collapsed transition matrix by constructing it directly from the uncertain graph G by flattening each uncertain edge into certain edges. Let v_i be a node with c outgoing certain edges and u outgoing uncertain edges. To flatten the outgoing edges of a node v_i, we do the following:

1. Each outgoing certain edge is associated with $\frac{1}{c+u}$ transition probability.

2. Each outgoing uncertain edge is also associated with $\frac{1}{c+u}$ transition probability.

Let e_{-} be an uncertain edge among outgoing uncertain edges, with t targets. Each

(a) each non-ϵ target of e_{-} is given a transition probability of $\frac{1}{t} \times \frac{1}{c+u}$

(b) if ϵ is a target for e_{-},

i. if there is c certain edges and u_0 ucertain edges which does not have ϵ, then the corresponding $\frac{1}{t} \times \frac{1}{c+u}$ transition probability is distributed among the c certain edges of v_i and u_0 uncertain edges with $\frac{1}{c} \times \frac{1}{u_0} \times \frac{1}{t} \times \frac{1}{c+u}$.

ii. if the vertex does not have any outgoing certain edges or uncertain edges without ϵ as targets, then the probability is re-distributed among all the nodes with $\frac{1}{\|V\|} \times \frac{1}{c+u}$ in the graph when $\|V\|$ is the number of nodes in G.

For instance, in the example visualized in Figure 6.1, since there are one certain outgoing edge and two uncertain outgoing edges, the probabilities of outgoing edges for v_i would be set as $\frac{1}{3}$ on the edge going to v_a, $\frac{1}{3} \times \frac{1}{3} = \frac{1}{9}$ on the edge going to v_b and v_c, and $\frac{1}{3} \times \frac{1}{4} = \frac{1}{12}$ on the edge going to v_d, v_e, and v_f. Note that, when ϵ is selected for any of the outgoing edges, the only available traversal direction is towards v_a. Therefore, this would lead to an additional transition probability of $\frac{1}{9} + \frac{1}{12} (= \frac{7}{36})$ towards v_a. This is visualized in Figure 6.3. If there is no certain edge from v_i to v_a, $\frac{1}{9} \times \frac{1}{12}$ with ϵ on two uncertain edges is distributed and added to all nodes in the graph.

Once the flattened transition matrix is obtained, we can solve the final PPR scores either using an exact PPR (flatPPR, Figure 6.2(f)) or an approximate PPR (flatApxPPR, Figure 6.2(g)) technique. Note that, while they are likely to be faster than both exhaustive and collapsing-based approaches, flattening-based solutions further

compound the accuracy problems.

6.4 UPPR: Proposed Approach

In this section, we propose an efficient and effective Uncertain Personalized PageRank (UPPR) algorithm to approximately compute personalized PageRank values on an uncertain graph with edge uncertainties. In particular, UPPR avoids enumeration of all possible worlds, yet is able to achieve high accuracy by carefully encoding edge uncertainties in a data structure that leads to good approximations.

6.4.1 Special Case: Two Possible Worlds

Let $G(V, E)$ be an edge uncertain graph as introduced earlier in this chapter. Let us split $G(V, E)$ into two subgraphs: a subgraph, $G_c(V, E_c)$, consisting of certain edges, and a subgraph, $G_u(V, E_u)$, consisting of uncertain edges. Let us first consider the special case where $G_u(V, E_u)$ defines only two possible worlds. In Section 6.4.2, we will generalize this to the case where there may be more than two possible worlds.

Let T_1 and T_2 be transition matrices corresponding to two possible worlds of G. The personalized PageRank values $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ for T_1 and T_2 on seed set, S, given by the user, are defined as

$$\overrightarrow{r_1} = \alpha \mathbf{T_1} \overrightarrow{r_1} + (1-\alpha)\overrightarrow{s}, \quad \text{and} \quad \overrightarrow{r_2} = \alpha \mathbf{T_2} \overrightarrow{r_2} + (1-\alpha)\overrightarrow{s},$$

where α is a residual probability parameter and $\overrightarrow{s}$ is a re-seeding vector such that if a node $v_i \in S$, then $\overrightarrow{s}[i] = \frac{1}{\|S\|}$ and $\overrightarrow{s}[i] = 0$, otherwise. It is easy to see that these two equations can be re-written as follows to solve for $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$:

$$\overrightarrow{r_1} = (1-\alpha)(I - \alpha T_1)^{-1}\overrightarrow{s} \quad \text{and} \quad \overrightarrow{r_2} = (1-\alpha)(I - \alpha T_2)^{-1}\overrightarrow{s}.$$

Given these, as defined in Section 6.2.3, we can compute the expected PPR values

for the edge uncertain graph as

$$\overrightarrow{r} = \frac{1}{2}(\overrightarrow{r_1} + \overrightarrow{r_2}) = \frac{1-\alpha}{2}((I - \alpha T_1)^{-1} + (I - \alpha T_2)^{-1})\overrightarrow{s}.$$

Now, let us split each of the transition matrices, T_1 and T_2, into three parts:

$$T_1 = T_{BL} + T_X + P_1 \text{ and } T_2 = T_{BL} + T_X + P_2,$$

where T_{BL}+T_X corresponds to the certain parts of the graph and P_1 and P_2 correspond to the uncertain edges in each of the two possible worlds. Moreover, let T_{BL} be the block-diagonal matrix, obtained by partitioning the graph into blocks (for example using METIS [57]), and T_X represent (certain) transitions across these partitions.

Note that, in general, we have $|T_{BL}| \gg |T_X|$. As we will see shortly, in this section, we further assume that $|T_X| \gg |P_1|$ and $|T_X| \gg |P_2|$. While this is a common assumption in related work [13], in Section 6.4.5, we discuss how to relax this assumption in cases where the number of uncertain edges involved in each possible world is large. As proposed in [106], assuming that the blocks are sufficiently small, we can efficiently compute $Q_{BL}^{-1} = (I - \alpha T_{BL})^{-1}$ by first computing the inverse matrices of each block and then combining these inverse matrices to obtain Q_{BL}^{-1}, which itself is in block-diagonal form. Moreover, since T_X, P_1, and P_2 are all sparse, we can also efficiently decompose the $T_X + P_1$ and $T_X + P_2$ into

$$T_X + P_1 \simeq U_1 S_1 V_1 \text{ and } T_X + P_2 \simeq U_2 S_2 V_2, \tag{6.1}$$

using a sparse approximate decomposition algorithm, such as [16]. Given these, we can rewrite $\overrightarrow{r} = \overrightarrow{r} = \frac{1}{2}(\overrightarrow{r_1} + \overrightarrow{r_2})$ as

$$\simeq \frac{1-\alpha}{2}\left(\Big(I - \alpha(T_{BL} + U_1 S_1 V_1)\Big)^{-1} + \Big(I - \alpha(T_{BL} + U_2 S_2 V_2)\Big)^{-1}\right)\overrightarrow{s}.$$

Then, by applying the well-known Sherman-Morrison lemma [92] on the term $(I -$

$\alpha(T_{BL} + U_i S_i V_i))^{-1}$, we can reformulate the above equation to obtain

$$\overrightarrow{r} \simeq \frac{1-\alpha}{2}\Big(Q_{BL}^{-1} + \alpha Q_{BL}^{-1} U_1 (S_1^{-1} - \alpha V_1 Q_{BL}^{-1} U_1)^{-1} V_1 Q_{BL}^{-1} + \\ Q_{BL}^{-1} + \alpha Q_{BL}^{-1} U_2 (S_2^{-1} - \alpha V_2 Q_{BL}^{-1} U_2)^{-1} V_2 Q_{BL}^{-1}\Big)\overrightarrow{s}.$$

When we further apply the Sherman-Morrison lemma on the term $(S_1^{-1} - \alpha V_1 Q_{BL}^{-1} U_1)^{-1}$ in the above equation, we obtain

$$(1-\alpha) Q_{BL}^{-1} \overrightarrow{s} \\ + \frac{\alpha(1-\alpha)}{2} Q_{BL}^{-1}\Big(U_1 (S_1 + \alpha S_1 V_1 (Q_{BL} - \alpha U_1 S_1 V_1)^{-1} U_1 S_1) V_1 \\ + U_2 (S_2 + \alpha S_2 V_2 (Q_{BL} - \alpha U_2 S_2 V_2)^{-1} U_2 S_2) V_2\Big) Q_{BL}^{-1} \overrightarrow{s}.$$

This equation can be significantly simplified by introducing the terms $M_1 = U_1 S_1 V_1$ and $M_2 = U_2 S_2 V_2$ (where $M_1 \simeq T_X + P_1$ and $M_2 \simeq T_X + P_2$):

$$\overrightarrow{r} \simeq (1-\alpha)\Bigg(I + \frac{\alpha}{2} Q_{BL}^{-1}\Big((M_1 + M_2) + \alpha\Big(M_1 (Q_{BL} - \alpha M_1)^{-1} M_1 \\ + M_2 (Q_{BL} - \alpha M_2)^{-1} M_2\Big)\Big)\Bigg) Q_{BL}^{-1} \overrightarrow{s}. \tag{6.2}$$

Moreover, relying on the assumption that $|T_{BL}| \gg |T_X| + |P_1|$ and $|T_{BL}| \gg |T_X| + |P_2|$, we can ignore the terms αM_1 and αM_2 in $(Q_{BL} - \alpha M_1)^{-1}$ and $(Q_{BL} - \alpha M_2)^{-1}$ in the above equation and rewrite the rest as

$$\overrightarrow{r} \simeq (1-\alpha)\Bigg(I + \frac{\alpha}{2} Q_{BL}^{-1}\Big((2T_X + P_1 + P_2) + \alpha\Big(2T_X Q_{BL}^{-1} T_X \\ + (P_1 + P_2) Q_{BL}^{-1} T_X + T_X Q_{BL}^{-1} (P_1 + P_2) \\ + P_1 Q_{BL}^{-1} P_1 + P_2 Q_{BL}^{-1} P_2\Big)\Big)\Bigg) Q_{BL}^{-1} \overrightarrow{s}. \tag{6.3}$$

Furthermore, again relying on the assumption that $|T_{BL}| \gg |T_X| \gg |P_1|, |P_2|$, the term $P_1 Q_{BL}^{-1} P_1 + P_2 Q_{BL}^{-1} P_2$ will be negligible next to $(P_1 + P_2) Q_{BL}^{-1} T_X + T_X Q_{BL}^{-1} (P_1 + P_2)$ and thus can be ignored and $\overrightarrow{r}$ can be approximately computed as

$$\overrightarrow{r} \simeq (1-\alpha)\Bigg(I + \frac{\alpha}{2} Q_{BL}^{-1}\Big((2T_X + (P_1 + P_2)) + \alpha\big(2T_X Q_{BL}^{-1} T_X + (P_1 + P_2) Q_{BL}^{-1} T_X + T_X Q_{BL}^{-1}(P_1 + P_2)\big)\Big)\Bigg) Q_{BL}^{-1} \overrightarrow{s}. \tag{6.4}$$

Summary and Key Advantages: This formulation for UPPR has several advantages. First of all, assuming that the blocks are sufficiently small and Q_{BL}^{-1} can be efficiently computed, once Q_{BL}^{-1} is at hand, solving for $\overrightarrow{r}$ using the above equation involves very sparse matrix multiplications (involving T_X and $P_1 + P_2$) and thus can be processed very efficiently (see Section 6.5). A second advantage of the above formulation is that it can be easily extended to any number of possible worlds.

6.4.2 General Case: > 2 Possible Worlds

When we have n possible worlds (i.e., $\overrightarrow{r} = \frac{1}{n}(\overrightarrow{r_1} + ... + \overrightarrow{r_n})$), the UPPR equation (Equation 6.4) can be generalized as

$$\simeq (1-\alpha)\Bigg(I + \frac{\alpha}{n} Q_{BL}^{-1}\Big((nT_X + (P_1 + ... + P_n)) + \alpha\big(nT_X Q_{BL}^{-1} T_X + (P_1 + ... + P_n) Q_{BL}^{-1} T_X + T_X Q_{BL}^{-1}(P_1 + ... + P_n)\big)\Big)\Bigg) Q_{BL}^{-1} \overrightarrow{s}. \tag{6.5}$$

As we see in Section 6.5, this formulation leads to very efficient execution plans, especially because the term $\frac{1}{n}(P_1 + ... + P_n)$ in Equation 6.5 can be obtained (without having to enumerate all possible worlds) directly by computing the ratio of the number of possible worlds in which a given edge exists:

Under mutual exclusion semantics: As we have seen in Section 6.2.1, the possible worlds covered by an uncertain edge consist of all combinations of its target nodes. Under mutual exclusion semantics, only one of the edges implied by the uncertain edge can be valid in the real world. Let v_i be a node which has c outgoing certain edges and u outgoing uncertain edges. If, in a given possible world, some of the u outgoing

uncertain edges map to ϵ, then in that possible world, the transition probabilities for the remaining certain and uncertain edges will be higher. We can use this observation to compute $P_{avg} = \frac{1}{n}(P_1 + ... + P_n)$ as follows:

Let v_j be a target node of an uncertain edge, e_-, with $\|target(e_-)\| = k$. The value of $P_{avg}(j, i)$ can be computed as

$$\frac{1}{k} \times \Big(\sum_{h=0}^{u-1} \Big(\frac{1}{c+u-h} \Big) p(\text{ratio of worlds s.t. } h \text{ of other uncertain edges from } v_i \text{ are } \epsilon) \Big).$$

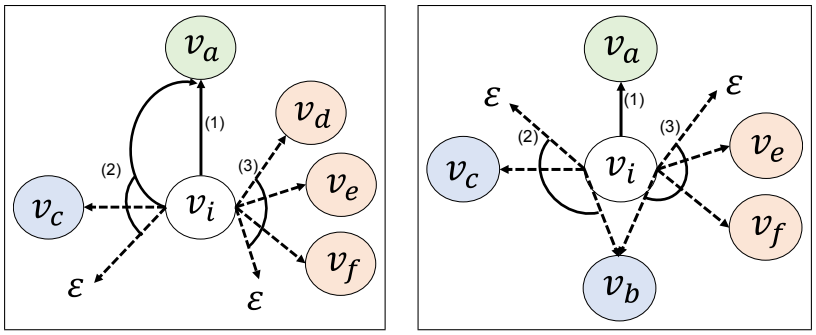
Here, $p()$ denotes the probability that h number of ϵ among all other uncertain edges from v_i are chosen given an event.

Note that, if e_- has ϵ as a target, then the corresponding transition probability, $L = \frac{1}{k} \times p(\epsilon$ on all uncertain edges are selected as targets) has to be redistributed among the outgoing certain edges of the node and outgoing uncertain edges without ϵ. When v_i has c outgoing certain edges and u_0 outgoing uncertain edges, $\frac{L}{c+u_0}$ is distributed to each edge and for uncertain edges, $\frac{1}{\|target(e)\|} \times \frac{L}{c+u_0}$ is added in the cells of P_{avg}. If none exists, then it needs to be redistributed among all nodes in the graph. When $\|V\|$ is the number of nodes in the graph, the value $\frac{L}{\|V\|}$ is added into all rows in the ith column of P_{avg}. This helps random walks to jump randomly to all nodes instead of staying in the node when v_i is a dangling node with no outgoing edge.

Let e_+ be an outgoing certain edge from v_i and let us denote its target as v_j. The transition probability, for e_+, taking into account ϵ transition for the uncertain edges, can be computed as

$$\sum_{h=0}^{u} \Big(\frac{1}{c+u-h} \Big) p(\text{ratio of worlds s.t. } h \text{ of uncertain edges from } v_i \text{ are } \epsilon).$$

However, since e_+ is a certain edge, it belongs to either intra-partition or cross-partition certain edges. Therefore, when we compute the $P_{avg}(j, i)$, we need to compensate for the portion of the transition probability already accounted in T_{BL} or T_X.



(a) uncertain and certain (b) uncertain and uncertain

Figure 6.4: An Example of Multiple Uncertain/Certain Edges with Same Target Nodes

Let $C(j,i)$ denote $T_{BL}(j,i)+T_X(j,i)$; then, the cell $[j,i]$ in P_{avg} has the compensated value

$$\left(\sum_{h=0}^{u}\left(\frac{1}{c+u-h}\right)p(\text{ratio of worlds s.t. } h \text{ of uncertain edges are } \epsilon)\right) - C(j,i).$$

Note that, if v_j is a target for multiple outgoing edges from v_i, all transition probabilities to v_j need to be aggregated. Before the aggregation, we consider v_j as different targets in different edges and compute the cell of $P_{avg}(j,i)$ on each certain or uncertain edge. After individual computations, if there are multiple edges whose targets are same as v_j, cell values of $P_{avg}(j,i)$ on different edges are aggregated by summing all values, $\sum_{v_f \in \{target(e \in E)=v_j\}} P_{avg}(j,f)$. Note that the aggregated node was duplicated by the summation of possible worlds that both nodes, v_j are considered together. Let v_k be a target of an edge e that has v_j as a target. Let ut_j be the number of uncertain edges that have v_j as targets and ns_i be the number of uncertain edges whose sources are v_i and have ϵ as targets excluding e_{-}. For the aggregation, the following value should be added into the cell $P_{avg}(k,i)$:

$$\sum^{2^{ut_j}-1}\left(\sum_{h=0}^{ns_i}\left(\frac{1}{(w-h)(w-h+1)}p(\text{ratio of worlds s.t. } h \text{ of uncertain edges are } \epsilon)\right)\right)$$

where $w = ct_j + ut_j$ when ct_j is the number of certain edges which have v_j as targets.

For the compensation of $P_{avg}(j,i)$, it needs to subtract the summation of added values of $P_{avg}(k,i)$ for all nodes.

For example, in Figure 6.4(a), the transition probability $P_{avg}(a,i)$ can be computed assuming Figure 6.1 and then, we can aggregate them into $P_{avg}(a,i)$ by summation. In this case, there are three possible worlds that both v_a exist with v_d, v_e, and v_f. For v_d, v_e, and v_f, $\frac{1}{2\times3} \times \frac{1}{12}$ is added into $P_{avg}(d,i)$, $P_{avg}(e,i)$, and $P_{avg}(f,i)$ and $3 \times \frac{1}{2\times3} \times \frac{1}{12}$ should be subtracted from $P_{avg}(a,i)$. In Figure 6.4(b), for the same target on uncertain edges, after aggregation of $P_{avg}(b,i)$, $\frac{1}{2\times3} \times \frac{1}{12}$ is added into $P_{avg}(a,i)$ and subtracted from $P_{avg}(b,i)$.

In both cases, to compute, P_{avg}, we need to compute the probability that for h out of a given number of uncertain edges, ϵ will be selected as the target. Let us be given $m = (m_0 + m_1)$ uncertain edges, such that m_0 many do not contain ϵ in the target set and m_1 many do. Let the maximum target size for this latter set of nodes be max_target. Then, we can group the m_1 uncertain edges to max_target many groups where, each group, g_l, consists of uncertain edges with target size l; i.e., $\|g_1\| + \|g_2\| + \ldots + \|g_{max\_target}\| = m_1$. Note that, by definition, any uncertain edge which contains ϵ as a target must also have at least one other node in its target set, $\|g_1\| = 0$.

Given this, we can compute the probability that h out of m uncertain edges will be ϵ as

$$p(h_2 + h_3 + \ldots + = h_{max\_target} = h \;\; s.t. \;\; \forall_{2 \leq l \leq max\_target} \;\; h_l \text{ in } \|g_l\| \text{ edges select } \epsilon).$$

The probability $p(h_l \text{ in } \|g_l\| \text{ edges select } \epsilon)$ is binomially distributed with $B(\|g_l\|, 1/l)$ – i.e., there are $\|g_l\|$ uncertain edges, each serving as an independent trial with $1/l$ success rate for the selection of ϵ among the available targets. Consequently, the probability that h out of m uncertain edges select ϵ as their targets is distributed as a summation of the binomial distributions $B(\|g_2\|, 1/2) + \ldots + B(\|g_{max\_target}\|, 1/max\_target)$.

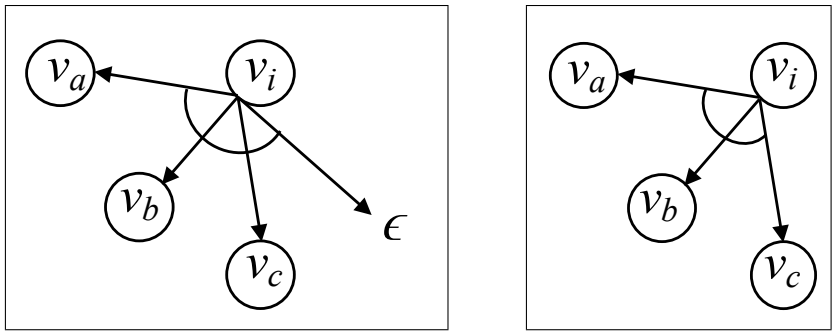
Algorithms to efficiently compute summation of binomial distributions are presented in [18]. They showed how to compute the exact distribution of the summation of binomial distributions. After calculating the distribution of each binomial random variable X_g, it compute $S = P(X_1 + ...X_g = z)$ for all z with less computation by the recurrence relation of the binomial distribution. They claimed that the complexity of this approach is $O((max_z)^2)$ where max_z is the maximum value of S that $P(S = max_z)$ is non-trivial. To reduce the computation, they tried some approximated approaches such as Kolmogorov-type and Pearson curve approximation. Specially Kolmogorov approximation showed fast efficient with less computation time and very least errors. Kolmogorov approximation uses the idea that the moment of the true distribution can be found at least up to some order z and with small z, it can be found easily with less computation of moments. In our computation, the summation of binomial distributions does not usually require high complexity because h is not a large number in real applications, but we can compute the approximated computations by above approaches.

Under multiple edge semantics: In this case, several of the edges implied by a given uncertain edge can be simultaneously valid. Let v_i be a node with c outgoing certain edges and u outgoing uncertain edges. Let v_j be a target node of an outgoing edge, e, from v_i. The value of $P_{avg}(j, i)$ can be computed as

$$\sum_{h=0}^{total\_out} \left(\frac{1}{c+h}\right) \times p\left(\sum_{e \epsilon U} num\_selected\_target\_nodes(e) = h\right),$$

where $total\_out = \sum_{e \epsilon U} \|target(e)/\{\epsilon\}\|$ and $num\_selected\_target\_nodes(e)$ is the number of nodes selected as outgoing targets for e in a given possible world (if ϵ is the only target selected, then $num\_selected\_target\_nodes(e) = 0$). Again, all v_i to v_j transitions need to be aggregated.

Note that, similarly with the case of mutual exclusion semantics, for certain edges,



(a) Uncertain edge with ϵ (b) Uncertain edge without ϵ

Figure 6.5: An Example of an Uncertain Edge with and without ϵ

we need to compensate for transition probabilities already accounted in T_{BL} or T_X. Also, if v_i does not have neither any certain edges nor any uncertain edges without ϵ as target, the transition probability for the case where all uncertain edges select ϵ as target needs to be distributed among all nodes in the graph. As we saw in the mutual exclusive case, $P_{avg}(j,i)$ to ϵ target is divided by the number of nodes in the graph, $|V|$ and added to all rows in the ith column of P_{avg}.

To compute P_{avg} using the above equation, we need to compute the probability $p\left(\sum_{e \in U} num\_selected\_target\_nodes(e) = h\right)$. Once again, this can be achieved by representing the distribution as a sum of binomial-like distributions: intuitively, if e is an uncertain edge with ϵ, then the probability that t many non-ϵ targets are selected can be represented in the form of a binomial with $2^{(\|target(e)\|-1)}$ many trials and 1/2 success rate. If, on the other hand, e is an uncertain edge without ϵ, the probability that t many targets are selected can be represented in the form of a binomial with $2^{\|target(e)\|}$ many trials and $\frac{1}{2}$ success rate. In the latter case, however, we need to correct for the situation where $t = 0$. This is because, under multiple edge semantics, for an uncertain edge without ϵ, the selected target nodes must include at least one node in the graph; thus, t cannot take the value of 0. The problem leads to non-binomial distribution because of non-existence of ϵ. To adjust this case, we, at first,

can compute the summation of the binomial distributions on the case of an uncertain edge which has $\|target\| + 1$ without ϵ when $\|target\|$ is the number of targets on the uncertain edge. After that, we multiply $\frac{2^{(\|target\|-1)}}{2^{\|target\|}}$ to get the summation of the binomial-like distribution for the case with ϵ. Figure 6.5 shows possible cases of an uncertain edge with 3 actual targets with ϵ and without ϵ. For Figure 6.5(a), there are 8 possible worlds (2^3) and we can compute with the binomial with $B(8, \frac{1}{2})$. For Figure 6.5(b), without ϵ, we cannot compute the binomial with $B(7, \frac{1}{2})$. We can use the binomial distribution on Figure 6.5(a) at first and multiply $\frac{2^3}{2^3-1}$ To make the binomial-like distribution on the case of Figure 6.5(b).

6.4.3 Accuracy of UPPR

The UPPR equation (Equation 6.5) captures the underlying uncertainty in a way that leads to minimal approximation errors under the assumption $|T_{BL}| \gg |T_X| \gg |P_*|$. In particular, the UPPR process has three specific sources for potential errors, each of which is minimized under these, generally valid, assumptions:

- <u>The first potential source of error</u> is the decomposition of $T_X + P_*$ into $U_* S_* V_*$ using an approximate algorithm, such as [16], that relies on the sparsity of the edges that cross partitions and of the uncertain edges (see Equation 6.1).

- <u>The second source of error</u> is the assumption that the terms αM_1 and αM_2 are negligible relative to the rest of the terms in Equation 6.2; this again relies on the assumption that T_X and P_* that contribute to M_* are both sparse matrices.

- <u>The third source of error</u> is the assumption that the term $P_1 Q_{BL}^{-1} P_1 + P_2 Q_{BL}^{-1} P_2$ in Equation 6.3 is negligible relative to $(P_1 + P_2) Q_{BL}^{-1} T_X + T_X Q_{BL}^{-1} (P_1 + P_2)$.

Note that all three potential sources of error are minimized when $|T_{BL}| \gg |T_X| \gg |P_*|$. While the fact that whether $|T_{BL}| \gg |T_X|$ holds or not depends on the type of graph

and the partitioning algorithm used, whether $|T_X| \gg |P_*|$ or not depends on the amount of uncertain edges in the graph.

In Section 6.4.5, we discuss how to relax the assumption, $|T_X| \gg |P_*|$, in cases where there are significant number of uncertain edges in the graph rendering $|P_*|$ relatively dense, using a hybrid strategy.

6.4.4 Efficient Computation of UPPR Scores

Here we show that the UPPR equation (Equation 6.5) leads to very efficient execution plans. To see this, let us first partition the UPPR equation into 6 subcomponents:

$$\begin{aligned}\overrightarrow{r} =& \frac{1}{n}(\overrightarrow{r_1} + ... + \overrightarrow{r_n}) \simeq \underbrace{(1-\alpha)Q_{BL}^{-1}\overrightarrow{s}}_{(1)} + \underbrace{\alpha(1-\alpha)Q_{BL}^{-1}T_X Q_{BL}^{-1}\overrightarrow{s}}_{(2)} \\ &+ \underbrace{\frac{\alpha(1-\alpha)}{n}Q_{BL}^{-1}(P_1 + ... + P_n)Q_{BL}^{-1}\overrightarrow{s}}_{(3)} \\ &+ \underbrace{\alpha^2(1-\alpha)Q_{BL}^{-1}T_X Q_{BL}^{-1}T_X Q_{BL}^{-1}\overrightarrow{s}}_{(4)} \\ &+ \underbrace{\frac{\alpha^2(1-\alpha)}{n}Q_{BL}^{-1}(P_1 + ... + P_n)Q_{BL}^{-1}T_X Q_{BL}^{-1}\overrightarrow{s}}_{(5)} \\ &+ \underbrace{\frac{\alpha^2(1-\alpha)}{n}Q_{BL}^{-1}T_X Q_{BL}^{-1}(P_1 + ... + P_n)Q_{BL}^{-1}\overrightarrow{s}}_{(6)}.\end{aligned}$$

It is important to note that each of the six subcomponents above contains an extremely sparse re-seeding vector $\overrightarrow{s}$. Moreover, Q_{BL}^{-1} is a block diagonal matrix and T_X and P_* are all sparse. Consequently, each of the terms can be computed, right to left, through efficient vector-matrix multiplications.

For example, the subcomponent (2) can be computed from right to left with the

following sequence of efficient operations:

$$\underbrace{Q_{BL}^{-1}}_{|V|\times|V|}\underbrace{\overrightarrow{s}}_{|V|\times 1} \rightarrow \underbrace{T_X}_{|V|\times|V|}\underbrace{Q_{BL}^{-1}\overrightarrow{s}}_{|V|\times 1} \rightarrow \underbrace{Q_{BL}^{-1}}_{|V|\times|V|}\underbrace{T_X Q_{BL}^{-1}\overrightarrow{s}}_{|V|\times 1}$$

$$\rightarrow \alpha(1-\alpha)\underbrace{Q_{BL}^{-1}T_X Q_{BL}^{-1}\overrightarrow{s}}_{|V|\times 1}.$$

Moreover, since the terms $(P_1 + ... + P_n)$, $Q_{BL}^{-1}\overrightarrow{s}$, $T_X Q_{BL}^{-1}\overrightarrow{s}$, and $Q_{BL}^{-1}T_X Q_{BL}^{-1}\overrightarrow{s}$ occur in multiple subcomponents, they can be cached and reused – once these terms are cached, the rest of the computations for the six subcomponents can be executed in parallel. Note further that several of the terms above can be cached and reused for the same uncertain graph with different seed vectors or even graphs with the same certain, but different uncertain components (to carry out hypothetical, if-then type of analyses).

6.4.5 Hybrid Computation in the Presence of Large Numbers of Uncertain Edges

As we have discussed in the previous section, the accuracy of the proposed UPPR technique relies on the assumption that $|T_{BL}| \gg |T_X| \gg |P_*|$. In particular, whether $|T_X| \gg |P_*|$ or not depends on the amount of uncertain edges in the graph: UPPR is likely to be highly effective and efficient if the number of uncertain edges in the graph is relatively small. In contrast, as we have seen in Section 6.3.2, the collapsing (and similarly flattening) based techniques may lead to large errors if the uncertain edges are concentrated around nodes with large PPR scores.

Here we note that we can leverage these two observations to deal with graphs with large numbers of uncertain edges. The idea is to eliminate uncertain edges in the graph, relying on the highly efficient flattening technique, away from the seed nodes of the graph (which are likely to have large PPR scores) and only maintain uncertain edges in the neighborhoods of the seed nodes. Consequently, errors due

to flattening are minimized as this technique is utilized only in regions with less likelihood of producing high PPR scores; errors due to UPPR are also minimized, especially in large graphs, as the numbers ($|P_*|$) of uncertain edges in possible worlds that UPPR has to deal with have been reduced relative to the rest of the graph.

6.5 Experiments

In this section, we present the results of the experiments assessing the efficiency and effectiveness of the algorithms presented in this chapter. We ran experiments on a 16-core CPU Nehalem Node with 64 GB RAM. All codes were implemented in Matlab and run using Matlab R2013b.

6.5.1 Datasets and Setup

Table 6.1 provides an overview of the four data sets [74], with different numbers of nodes and edges, and graph-partitions, considered in the experiments. The graph partitions are obtained using METIS [57].

Table 6.2 details the volumes of uncertainty we have experimented with for the results reported in this section. Here, the "degree of uncertainty" refers to the number of target nodes on each uncertain edge it represents and the "edge semantics" describes "mutual exclusion" and "multiple edge" semantics. These together define the number of possible worlds corresponding to a given uncertain edge. To obtain uncertain graphs with the specifications in the table, we select random edges in the original graph and render them uncertain by augmenting destinations with random nodes.

We further assume that the uncertain edges are located on the seeds (as discussed in Sections 6.3.2 and 6.4.5, uncertain edges further away from the seeds can be flattened into the certain parts of the transition matrix).

Data	# of nodes	# of edges	# of partitions
ego-Facebook	4,039	88,234	3
Wiki-Vote	7,115	103,689	3
web-NotreDame	325,729	1,497,134	50
web-BerkStan	685,230	7,600,595	500

Table 6.1: Data Sets

6.5.2 Alternative Approaches

In this section, in addition to UPPR (presented in Section 6.4), we considered all alternative approaches discussed in Section 6.3. As a further baseline, we also consider a Monte Carlo-based solution (which starts from the seed nodes, and samples random walks of a given length) and BEAR [97], a recent PPR computation algorithm, which originally does not take uncertainty into account. For uncertainty, we use the flattened transition matrix for the transition matrix and compute PPR values. In the experiments, without loss of generality, we set the residual probability parameter, α to 0.85. To compare different algorithms, we consider both efficiency (i.e., PPR computation time) and accuracy (in terms of the correlations of PPR rankings for the nodes that are ranked top-50 by the exhaustive technique, exhPPR).

6.6 Results and Discussions

We start the discussion of the results by considering efficiency and accuracy of the various algorithms on the Facebook data set, for different degrees of uncertainty in the graph.

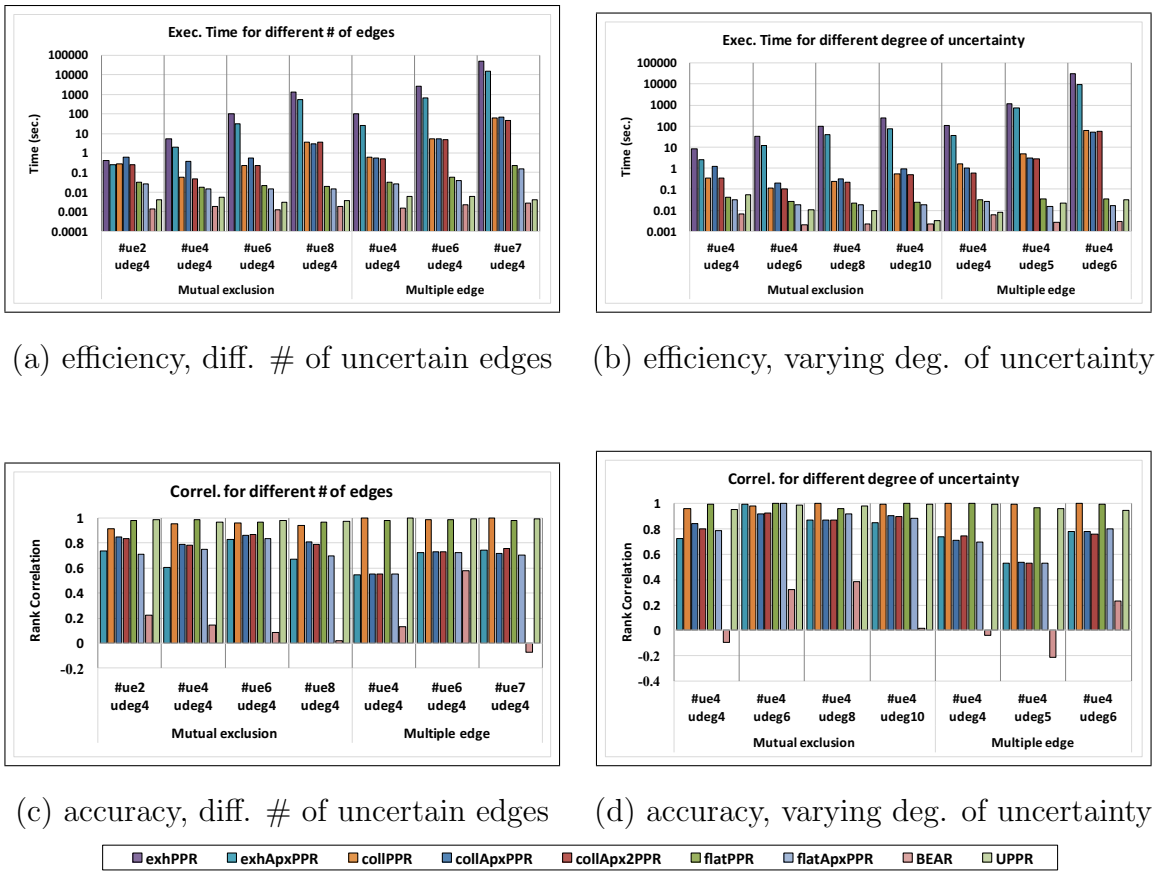
Impact of the Degree of Uncertainty. Figures 6.6(a) and (b) show the execution times of different algorithms, as the overall number of uncertain edges and degree of

	# of uncertain edges	degree of edge uncertainty	edge semantics	# of possible worlds
different # of uncertain edges	2	4	mut.excl. (multiple)	16-64
	4			256-4,096
	6			4,096-262,144
	8(7)			65,536-2,097,152
different degree of edge uncertainty	4	2	mut.excl. (multiple)	16-16
		4		256-4,096
		6(5)		1,296-65,536
		8(6)		4,096-1,048,576
		10		10,000

Table 6.2: Uncertainty Scenarios

uncertainty in the graph are increased.

As we see in the figure 6.6, exhaustive and collapsing-based approaches (which need to enumerate the possible worlds) quickly become infeasible as the number of possible worlds increases. While flattening-based approaches are reasonably fast and scale better than the exhaustive and collapsing-based approaches, they are 1 or 2 order slower than UPPR. *BEAR* takes less time than *UPPR* for PPR computation but the difference between them is negligible. Figures 6.6(c) and (d) confirm that execution time savings on UPPR do not come with any drop in accuracy – UPPR provides similar (or in some cases better) accuracy to the two collapsing- and flattening- based approaches, collPPR and flatPPR, that rely on direct computation of PPR from the transition matrix, even though it uses an approximate solution for PPR. As expected, the accuracy of BEAR is very poor compared to *UPPR* and the accuracy is not stable and affected by the amount of uncertainty. Other techniques such as collApxPPR, collApx2PPR, and flatApxPPR that similarly solve PPR approximately,



(a) efficiency, diff. # of uncertain edges

(b) efficiency, varying deg. of uncertainty

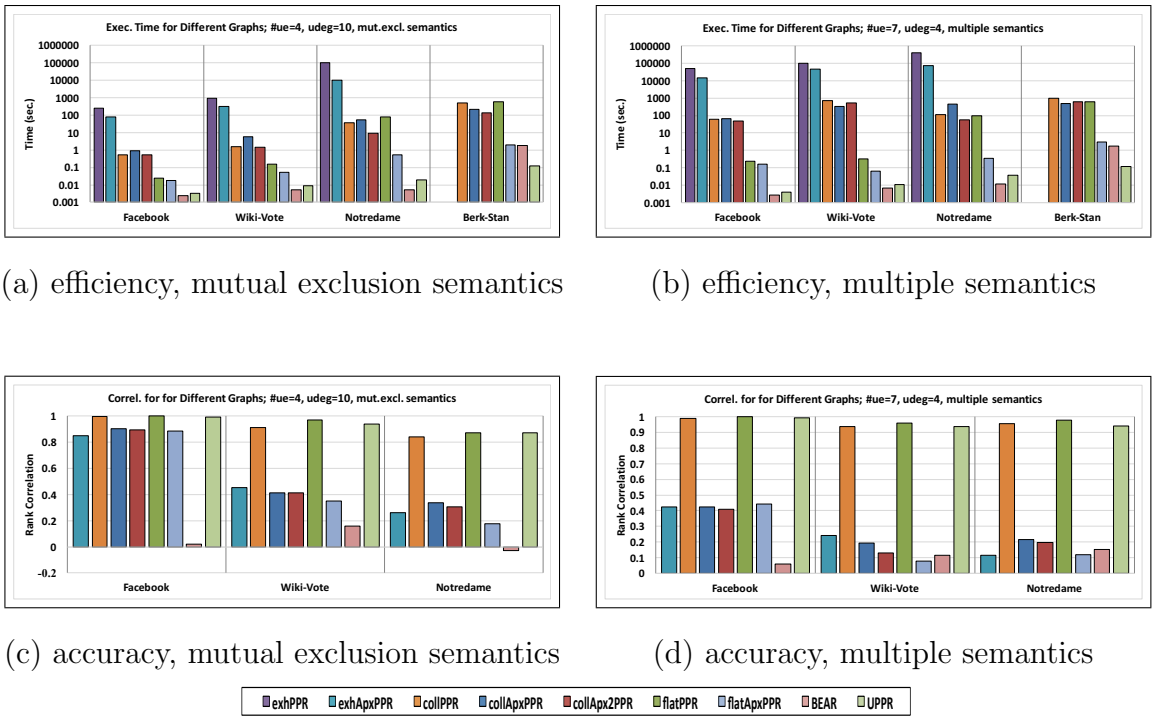
(c) accuracy, diff. # of uncertain edges

(d) accuracy, varying deg. of uncertainty

Figure 6.6: Results on the Facebook Data Set, for Different Amount of Uncertainty with Different Edge Semantics

relying on a sparse approximation method, all have significantly degraded accuracies. This indicates that, by carefully accounting for the sources of errors, UPPR is able to achieve high accuracies (~1.0) efficiently (~0.01 seconds) and avoids accuracy pitfalls that other schemes are not able to handle effectively.

UPPR vs. Monte Carlo Method. Additionally, we consider a Monte Carlo (MC) based alternative to UPPR. [78] notes that (in regular graphs) for estimating PPR values close to a desired threshold δ (where δ is the expected PPR score; i.e., $1/|V|$, where $|V|$ is the number of nodes), a Monte Carlo based algorithm would need $O(1/(\delta \times \rho^2)) = O(|V|/\rho^2)$, samples of length, $geometric(\frac{1}{1-\alpha})$, where ρ is the relative error and $1-\alpha$ is the teleportation rate. This means that, when we seek high accuracy,



(a) efficiency, mutual exclusion semantics

(b) efficiency, multiple semantics

(c) accuracy, mutual exclusion semantics

(d) accuracy, multiple semantics

Figure 6.7: Results in Different Graphs of Different Sizes

Monte Carlo based solutions may be prohibitive [78]. Indeed, for the Facebook data set, with ~ 4000 nodes, to have 95% accuracy, we would need $4000/0.05^2 = 1,600,000$ random walk samples (of length $\geq \lceil \frac{1}{0.15} \rceil = 7$, since we set α to 0.85).

In Table 6.3, we report the accuracy comparison for a more modest target error rate of 0.15, which leads to ~ $150K$, random walks – note that, even in this modest case, taking $150K$ random walk samples is more expensive (65 seconds in Matlab) to compute than UPPR (~0.01 seconds). In the table, we see that for top-100 to top-500 results, Monte Carlo, is able to match the target accuracy in the presence if mutual exclusion semantics; but fails to do so when all nodes are considered. In the presence of multiple edge semantics, MC is able to match the target error rate only when top-500 results are considered and the results are very poor for top-100 nodes, even with larger number of samples, with longer lengths. Note that UPPR is able to achieve significantly higher accuracy (for top-100, top-500, as well as for all nodes),

Edge type	# of random walks	Length of random walks	Top 100 acc.	Top 500 acc.	All nodes acc.
Mutual exclusion semantics (#ue=4, #udeg=10)	**UPPR**		**0.987**	**0.966**	**0.994**
	150K Monte Carlo	8	0.797	0.947	0.579
		10	0.824	0.945	0.576
		20	0.846	0.942	0.574
		30	0.843	0.943	0.573
	300K Monte Carlo	8	0.798	0.928	0.572
		10	0.823	0.928	0.567
		20	0.843	0.927	0.562
		30	0.846	0.924	0.563
Multiple edge semantics (#ue=5, #udeg=4)	**UPPR**		**0.994**	**0.995**	**0.999**
	150K Monte Carlo	8	0.198	0.921	0.673
		10	0.198	0.919	0.670
		20	0.203	0.918	0.666
		30	0.203	0.916	0.667
	300K Monte Carlo	8	0.148	0.905	0.660
		10	0.138	0.900	0.655
		20	0.145	0.900	0.648
		30	0.145	0.898	0.647

Table 6.3: UPPR Vs. MC Method on the Facebook Graph

very cheaply (~ 0.01 seconds for this data set as shown in Figure 6.6).

Different Data Sets and the Impact of the Graph Size. In the experiments reported in Figure 6.7, we compare the efficiency and effectiveness of the various algorithms we presented in the chapter for graphs of different sizes. The figure reports results for two sample uncertainty complexities: Figures 6.7(a) and (c) report execution time and rank correlation for a scenario with mutual exclusion semantics, whereas Figures 6.7(b) and (d) consider a scenario with multiple edge semantics. As we see in this figure, the proposed UPPR method is scalable, not only in terms of

the possible worlds of the graph, but also the graph size. While the closest algorithms to UPPR in terms of efficiency and scalability, flatApxPPR and BEAR, suffer significantly from accuracy degradations, UPPR provides very high (mostly close to perfect) accuracy in all cases considered in this section.

Here, we do not present the accuracy results for the largest Berk-Stan data set as the cost of performing the exhaustive enumeration needed to obtain the accuracy ground-truth is prohibitive on this data set. However, the results show that UPPR provides very good accuracy, while its execution time is minimally effected by graph size. In fact, on the largest data set, UPPR is even faster than the BEAR baseline, while providing significantly better accuracy.

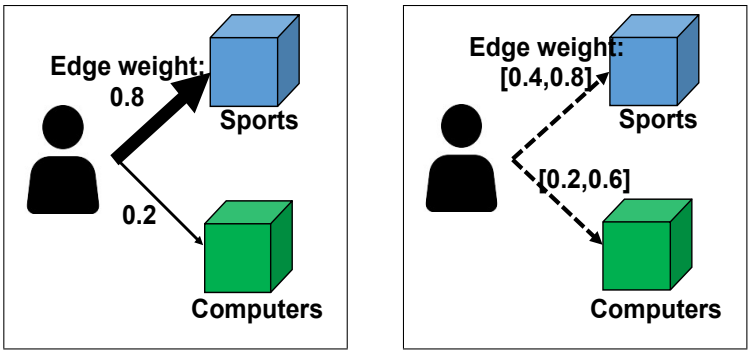
Chapter 7

PERSONALIZED PAGERANK IN UNCERTAIN GRAPHS WITH UNCERTAIN EDGE WEIGHTS

7.1 Introduction

The node proximity between two nodes in a graph-structured application shows how much they are nearby or related to each other. The popular measure for node proximity includes Random-walk based definitions such as hitting time [24, 82] and personalized PageRank (PPR) score [11, 19, 20, 54, 64, 97, 106]. Instead of the number of hops or distances between two nodes, these consider the density of edges in a graph. This takes into account how tightly connected two nodes are and argues that nodes which have many paths between them can be considered more related.

Despite the effectiveness of PPR on measuring node proximities, there are certain situations when the performance of PPR measure is not guaranteed. Possible uncertainties in the input graph make it difficult to compute PPR, since PPR method is designed with the assumption that all information of edges in a graph is certain with known scalar values. The problem is that, in many real applications, uncertainty happens due to various reasons, such as lack of information, noise in data collection, or privacy issues[2, 3, 13, 61]. There are various uncertain types on edges in a graph. One uncertain type is the uncertainty of edge existence in a graph as discussed in Chapter 6. In this situation, the edges in the graph exists probabilistically and, in many existing works prior to our own, the existence probabilities of individual edges are assumed to be independent from each other [13, 33, 60, 61, 93, 118]. Most of works compute the node proximity[33] or find k-nearest neighbors[93, 118] with probabilistic



(a) certain (b) interval valued uncertain

Figure 7.1: Interval Edge Weights When a User Have Different Degree of Interests

computations.

In this chapter, I note that another common uncertain edge type is when the uncertainty exists on the edge weights in the graph. For instance, edge weights may need to be represented by a list of scalar numbers, by a probability distribution, or intervals with min/max value. For example, let us consider a user who has two interests, such as sports and computers, as shown in Figure 7.1. In certain graphs, the relative strength between these two interests would be represented as scalar valued weights as in Figure 7.1.(a). In reality, however, data collected over time and from different sources and contexts, may provide interest values that are not scalars, but interval of values representing different evidences. Computer and road networks are examples: load (or other factors, such as noise or cost) on networks may be variable and analysis of such networks may need to reflect such variability.

In this chapter, I consider edge weights with interval values. When edge weights are interval valued, it is hard to use the basic PPR equation since that equation requires scalar valued inputs, based on the assumption that the edge weight is certain. A possible solution would be to take multiple samples from each interval edge weight, compute PPR scores for all possible worlds using the combinations of sampled values, and return an average of PPR score capturing all considered alternative scores. This,

however, requires an exponential work to compute scores. To tackle this challenge, in this chapter, I consider two alternative approaches: Interval Personalized PageRank with Mean (IPPR-M) and Interval Personalized PageRank with Integration (IPPR-I) to compute PPR values when edge weights are uncertain with interval values. IPPR-I provides optimal solutions, yet is faster than the sampling approach. IPPR-M, provides approximated solutions and is faster than both sampling and IPPR-I. Nevertheless, as I will show in this chapter, when the networks are "well mixed", IPPR-M is also as effective as IPPR-I.

In the following section, I first formally define the problem of PPR computation for an interval valued uncertain graph. In Section 7.4, I present the optimal solution, IPPR-I, and then present the approximate solution, IPPR-M. In Section 7.6, I evaluate IPPR-M and IPPR-I under different data sets and different scenarios to understand under which scenarios the cheaper IPPR-M can be used instead of the relatively more expensive IPPR-I.

7.2 Problem Formulation

7.2.1 Interval Edges and Interval Graphs

Let $G(V, E, W)$ be a graph, where V is a set of nodes, E is a set of edges, and W is a set of weights on edges E. There are two kind of values of a weight $w_{ij} \in W$ on an edges $e_{ij} \in E$ whose a source node is $v_i \in V$ and $v_j \in V$.

Definition 15 (Interval Valued Edge Weight) *An interval weight w_{ij} on e_{ij} between $v_i \in V$ and $v_j \in V$ is written as $[w_{ij_*}, w^*_{ij}]$ where w_{ij_*} is the minimum value and w^*_{ij} is the maximum value of the interval value.*

Definition 16 (Scalar Valued Edge Weight) *Scalar valued edge weight is a special case of interval weights, with $w_{ij_*} = w^*_{ij}$ that the minimum value and the maximum*

value are the same.

Given this, an interval edge e_{ij} is defined as follows:

Definition 17 (Interval Edge) *An interval edge $e_{ij} \in E$ is an interval valued edge, with a non-scalar interval weight w_{ij}.* [1].

Definition 18 (Interval Graph) *An interval graph G_I is a graph which has at least one interval edge.*

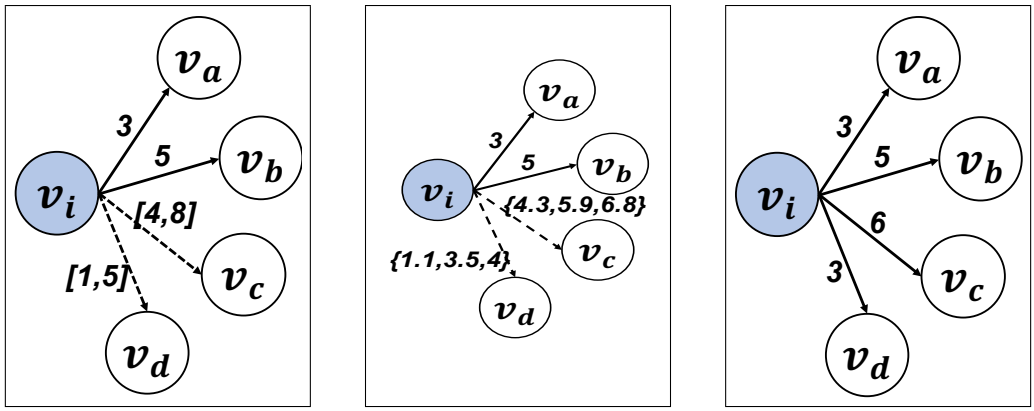
Figure 7.2(a) shows an example of a graph with interval edges. From a node v_i, there are four outgoing edges with different edge weights: $w_{ia} = 3, w_{ib} = 5, w_{ic} = [4, 8]$, and $w_{id} = [1, 5]$. For edges from v_i to v_a and v_b, edge weights are scalar valued. For edges from v_i to v_c and v_d, edge weights are interval valued.

7.2.2 Naive Approach: PPR Computation with Sampling

One approach to approximately compute PPR values in an interval graph is to generate a subset of the "possible worlds", by sampling edge weights within the intervals associated to the edges. In this approach, we consider the edge weights as sampled values by selecting d values randomly in each interval weight.

Definition 19 (d-Sample Graph) *Let $G = (V, E, W)$ be an interval graph which has a set of nodes, V, and a set of edges, E, with interval valued edge weights, W: given an edge from v_i to v_j, the corresponding weight w_{ij} has an interval value, $[w_{ij_*}, w^*_{ij}]$. A d-sample graph, $G' = (V, E, W')$ is a graph which has same V and E as G but has weights, W', where each $w'_{ij} \in W'$ is a set of d scalar values, $w'_{ij} = \{w'_{ij(1)}, \ldots, w'_{ij(d)}\}$ such that $w_{ij*} \leq w'_{ij(k)} \leq w^*_{ij}$.*

[1] When the interval contains 0, this represents the special case of non-existence of the edge. We do not consider this case in this chapter since it is already discussed in Chapter 6



(a) Interval weights (b) Sampled edge weights (c) Scalar weights

Figure 7.2: Examples of Interval-valued/Sampled/Scalar-valued Graphs

Figure 7.2(b) shows an example of 3-sample graph sampled from Figure 7.2(a). On the uncertain edge e_{ic}, 3 values have been randomly selected from the interval $[4, 8]$; similarly on edge e_{id}, 3 values have been are randomly selected.

When an edge $e_{ij} \in E$ has d scalar values as the edge weight w'_{ij}, there are d possible worlds $pw(e_{ij})$ for the edge. When we consider a graph with sampled edges, all possible worlds covered by this graph are defined as $\prod_{e \in E} pw(e)$ which is the product of the possible worlds of edges. Therefore, if there are $\|E_I\|$ interval edges in the graph and the sampling rate is d, there would be $d^{\|E_I\|}$ scenarios to be evaluated to compute the expected value of the PPR scores. It is easy to see that this approach quickly becomes unfeasible as it requires one to compute PPR scores for exponentially many possible worlds.

7.2.3 Personalized PageRank in an Interval Graph

Given the above definitions, I now define personalized PageRank in an interval graph. Given the above definitions, we now define personalized PageRank in an interval graph.

Definition 20 (Personalized PageRank in an Interval Graph) *Let* $G_I(V, E, W)$

be an interval graph. Given a seed set, S, of nodes we can define the personalized PageRank vector $\vec{r}$ as

$$\vec{r} = \alpha T_I \vec{r} + (1 - \alpha)\vec{e},$$

where T_I denotes a normalized transition matrix generated from G_I with interval values, α is a residual probability, and $\vec{e}$ is a seed vector if $v_i \in S$, then $\vec{s}[i] = \frac{1}{\|S\|}$ and $\vec{s}[i] = 0$, otherwise.

7.3 Interval Personalized PageRank with Integration (IPPR-I)

In this section, I first introduce transition matrices for interval graphs and propose an Interval Personalized PageRank with Integration (IPPR-I) method to compute optimal PPR scores in graphs with interval valued edge weights.

7.3.1 Interval Transition Matrix for Interval Graphs

Let $G_I = (V, E, W)$ be an interval graph, with $\|V\|$ nodes and $\|E\|$ edges. Each edge, $e_{ij} \in E$, between two nodes, $v_i \in V$ and $v_j \in V$, has a weight, w_{ij}, which can be a scalar or an interval, $[w_{ij_*}, w^*_{ij}]$. To compute PPR scores of this interval graph, we need to build a transition matrix T_I corresponding to the interval weighted graph G_I.

Case I: A Node with the Number of Outgoing Interval Edges = 1

Let v_i be a node which has a single outgoing interval edge to v_k with an edge weight, $[w_{ik_*}, w^*_{ik}]$. The transition matrix, T_I, needs to include normalized probabilities for outgoing edges, summing to 1. We compute transition probabilities for the outgoing edges of v_i as follows:

- **No other outgoing edges:** If the only outgoing edge from v_i is the interval valued edge from v_i to v_k, then the random walk has to go over this edge independent of the specific interval weights. Therefore, the transition probability

$T[k,i]$ is set as 1. In other words, since there is only one outgoing edge, the transition probability value is 1 and the interval value, $[w_{ik_*}, w^*_{ik}]$ does not affect the transition (again assuming that $w_{ik_*} > 0$).

- **With other (scalar valued) outgoing edges:** Let w_{ij} be a scalar edge weight on an edge, e_{ij}, from v_i to v_j and sw_i be sum of all outgoing scalar valued edge weights from v_i. In this case, the integrated weight, $I[j,i]$ can be computed as

$$\int_{w_{ik_*}}^{w^*_{ik}} \frac{w_{ij}}{sw_i + x} dx = w_{ij} \times \ln(sw_i + x)\Big|_{w_{ik_*}}^{w^*_{ik}} = w_{ij} \times (\ln(sw_i + w^*_{ik}) - \ln(sw_i + w_{ik_*}))$$
$$= w_{ij} \times \ln(\frac{sw_i + w^*_{ik}}{sw_i + w_{ik_*}}),$$

where x is a variable corresponding to the possible scalar values the interval edge can take. Intuitively, the integral considers all possible values that the interval valued edge can take and normalizes the transition probability along the edge from v_i to v_j accordingly. Similarly, the integrated weight, $I[k,i]$ can be computed with

$$\int_{w_{ik_*}}^{w^*_{ik}} \frac{x}{sw_i + x} dx = (x - sw_i \times \ln(sw_i + x))\Big|_{w_{ik_*}}^{w^*_{ik}}$$
$$= (w^*_{ik} - sw_i \times \ln(sw_i + w^*_{ik})) - (w_{ik_*} - sw_i \times \ln(sw_i + w_{ik_*}))$$
$$= w^*_{ik} - w_{ik_*} + sw_i \times \ln(\frac{sw_i + w_{ik_*}}{sw_i + w^*_{ik}}).$$

Given these integrated weights, we can obtain the transition probabilities by normalizing the corresponding column in the transition matrix such that the sum of entries is equal to 1.0:

$$T[j,i] = \frac{I[j,i]}{I[j,i] + I[k,i]} \text{ and } T[k,i] = \frac{I[k,i]}{I[j,i] + I[k,i]}.$$

For example, in Figure 7.3, integrated weights on the two outgoing edges can be computed as follows:

$$I[a,i] = \int_1^5 \frac{3}{3+x} dx = 3 \times (\ln(3+5) - \ln(3+1)) \approx 2.079,$$

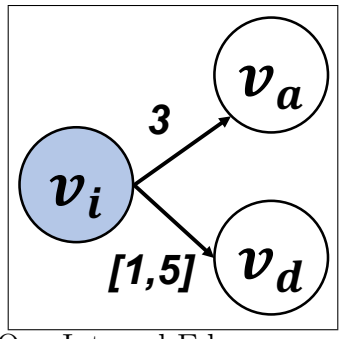


Figure 7.3: An Example of One Interval Edge

$$I[d,i] = \int_1^5 \frac{x}{3+x} dx = (5 - 3 \times \ln(3+5)) - (1 - 3 \times \ln(3+1)) \approx 1.921.$$

After the normalization process, we get the transition probabilities, $T[a,i] = 0.5199$ and $T[d,i] = 0.4801$. Note that, though the mean of the interval $[1,5]$ on w_{di} is same as the scalar value w_{ai}, $T[d,i]$ is less than $T[a,i]$. This means that replacing the interval edge weights simply with the means of the intervals (as I will consider in Section 7.5) may not give the optimal solution.

Case II: A Node with the Outgoing Number of Interval Edges = 2

When the node has more than one outgoing interval edges, we need multiple integrals each corresponding to one of the interval edges. For instance, let us consider a node v_i with two outgoing interval valued edges, (one to v_k with $[w_{ik_*}, w^*_{ik}]$ interval weight and v_l with $[w_{il_*}, w^*_{il}]$ interval weight). Let us assume that the sum of all scalar valued outgoing weights for v_i is equal to sw_i. In this case, the integrated weight, $I[j,i]$ from node v_i to node v_j with scalar valued weight, w_{ij}, can be computed as

$$\int_{y=w_{il_*}}^{w^*_{il}} \int_{x=w_{ik_*}}^{w^*_{ik}} \frac{w_{ij}}{sw_i + x + y} dx\, dy = w_{ij} \times \left(\left(-y + (w^*_{ik} + x + y) \ln(w^*_{ik} + x + y) \right) \Big|_{w_{ik_*}}^{w^*_{ik}} \right) \Big|_{w_{il_*}}^{w^*_{il}}$$

Similarly, the integrated weight, $I[k,i]$, corresponding to the interval valued edge v_i to v_k is computed as

$$\int_{y=w_{il_*}}^{w_{il}^*} \int_{x=w_{ik_*}}^{w_{ik}^*} \frac{x}{sw_i + x + y}\, dx\, dy$$
$$= \frac{1}{4} \times \left(y \times (2w_{ik}^* + 2x + y) - 2 \times ({w_{ik}^*}^2 - x^2 + 2w_{ik}^* y + y^2) \times \ln(w_{ik}^* + x + y)\right)\Big|_{w_{ik_*}}^{w_{ik}^*}\Big|_{w_{il_*}}^{w_{il}^*}.$$

To obtain the actual transition probabilities, we need to normalize the corresponding integrated weights in each column of the transition matrix.

For example, if we consider the example graph fragment in Figure 7.2(a) which shows a node with two outgoing interval weighted edges, we can compute the integrated weights as follows:

$$I[b,a] = \int_{y=4}^{8} \int_{x=1}^{5} \frac{3}{8 + x + y}\, dx\, dy = 2.8502$$

$$I[c,a] = \int_{y=4}^{8} \int_{x=1}^{5} \frac{5}{8 + x + y}\, dx\, dy = 4.7503$$

$$I[d,a] = \int_{x=1}^{5} \int_{y=4}^{8} \frac{y}{8 + x + y}\, dy\, dx = 5.6249.$$

$$I[e,a] = \int_{y=4}^{8} \int_{x=1}^{5} \frac{x}{8 + x + y}\, dx\, dy = 2.7747.$$

After the normalization, we can get the transition probabilities as $T[b,a] = 0.1781$, $T[c,a] = 0.2969$, $T[d,a] = 0.3516$, and $T[e,a] = 0.1734$.

General Case

When the number of interval weighted edges from a node v_i is more than 2, it requires same number of multiple integrals as the number of interval edges.

Given an interval graph $G_I(V,E,W)$, let $v_i \in V$ be a node that have different type of outgoing edges: Let $R = \{R_1, \ldots, R_m\}$ denote the set of scalar valued outgoing edges and $N = \{N_1, \ldots, N_n\}$ denote the set of interval valued outgoing edges.

Given this, the integrated weight for a scalar valued edge from node v_i to node v_j is defined as

$$I[j,i] = \underbrace{\int_{w_{in*}}^{w_{in}^*} \dots \int_{w_{i1*}}^{w_{i1}^*}}_{\text{n}} \frac{weight(R_j)}{\sum_{u=0}^{m} weight(R_u) + \underbrace{(x_1 + \dots + x_n)}_{\text{n}}} \underbrace{dx_1 \dots dx_n}_{\text{n}}$$

whereas the integrated weight for an interval weighted edge from node v_i to node v_h is defined as

$$I[k,i] = \underbrace{\int_{w_{in*}}^{w_{in}^*} \dots \int_{w_{i1*}}^{w_{i1}^*}}_{\text{n}} \frac{x_k}{\sum_{u=0}^{m} weight(R_u) + \underbrace{(x_1 + \dots + x_n)}_{\text{n}}} \underbrace{dx_1 \dots dx_n}_{\text{n}},$$

Once again, these integrated weights need to be normalized to 1.0 to obtain the transition probabilities for the outgoing edges from v_i.

Note that for any graph with a maximum outgoing degree, d_{max}, we need to pre-compute the $2 \times d_{max}$ closed-form formulas that can be used to obtain the integrated weights and outgoing transition probabilities for any node in the graph.

7.3.2 Interval Personalized PageRank with Integration (IPPR-I)

Given the interval normalized transition matrix $\mathbf{T_I}$, it is straight-forward to compute interval personalized PageRank (IPPR-I) scores in the form of a vector

$$\vec{r} = \alpha \mathbf{T_I} \vec{r} + (1 - \alpha)\vec{e},$$

where α is a residual probability, $\mathbf{T_I}$ is an interval weighted normalized transition matrix, and $\vec{e}$ is a re-seeding vector, such that given a set of seeds S, if $v_i \in S$, then $\vec{e}[i] = \frac{1}{\|S\|}$ and $\vec{e}[i] = 0$, otherwise when S is a set of seeds.

7.4 Interval Personalized PageRank with Mean (IPPR-M)

One approach to obtain approximate PPR scores in an interval graph is to use the mean of interval weights. Instead of integrating interval values $[w_{ij_*}, w_{ij}^*]$ as

described in the previous section, we could flatten the interval values into scalars w'_{ij}, by averaging the minimum value and the maximum value $\frac{w_{ij*}+w^*_{ij}}{2}$ and compute conventional PPR scores with these flattened scalar values. Figure 7.2(c) shows an example of mean of weights from Figure 7.2(a). In this approach, w_{ia} and w_{id} are the same and treated with same transition probabilities As we see in Section 7.6, this approach provides approximate PPR scores very efficiently. The key question, then is under what conditions such an approximation can be effective. In Section 7.6, we experimentally study the conditions in which this approach works well.

7.5 Mixing Factor and Localized Graph

As I described in previous sections, IPPR-I is an optimal solution to compute PPR scores accurately and IPPR-M is an approximate solution for relatively fast computation in an interval graph. In this section, I will describe how to make a choice between IPPR-I and IPPR-M for the better performance. t first, I define low-mixing factor as follows:

Definition 21 (Low-Mixing Factor) *We call a graph low-mixing factor if*

- *the structure is tree like,*
- *interval edges have similar values, and*
- *number of outgoing interval edges are similar.*

A tree-structured graph such as Binary tree [85] and taxonomic/hierarchical tree [25] is an example of low-mixing factor. From seed nodes, the connections are spread into the network with less number of commons nodes between neighbors. A small-world network [109] is another example of low-mixing factor graphs. In Figure 7.4(a), it shows a binary tree-structured graph spreading from a leftmost node to other nodes

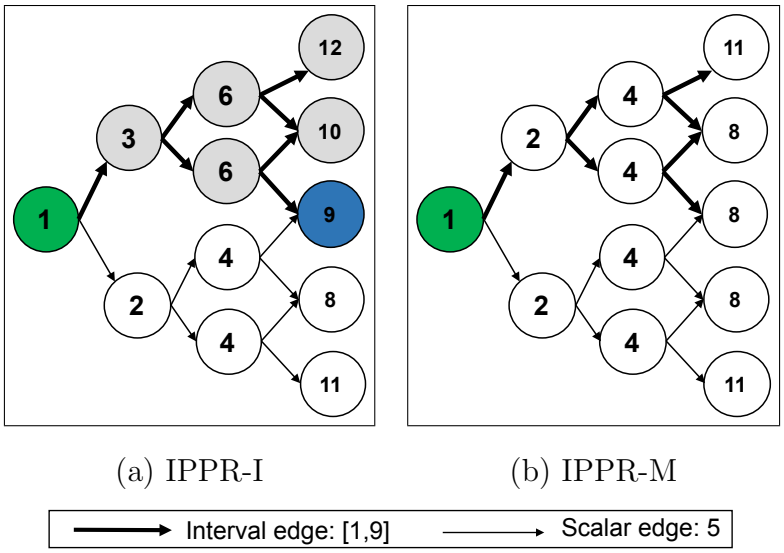


Figure 7.4: An Example of Rankings of IPPR-R and IPPR-M

with less number of common nodes. If a graph is randomly generated, there is a high chance that the mixing factor of the graph is high. The local clustering coefficient that quantifies how much its neighbors are connected can be a mixing factor. If it is low, the mixing factor is low and otherwise, the mixing factor is high.

When a graph is with low-mixing factor, the accuracy of IPPR-M is lower than IPPR-I and for high-mixing factor, the accuracy of IPPR-M is close to that of IPPR-I. When there are mixture of incoming scalar valued edges and incoming interval valued edges to a node, the scalar valued edges cancel the impact of incoming interval valued edges out because they have higher probability than interval valued edges. The combinations of these edges make the interval values close to the average of the intervals. Therefore, when the mixing factor is high, the transition matrix and PPR scores of IPPR-M and IPPR-I are similar, so it is better to use IPPR-M for same accuracy with fast execution time.

For example, in Figure 7.4, the mixing factor is low since the graph is a binary-tree structure with very less number of common nodes among neighbors. In the circle of nodes, it shows the PPR ranking scores of nodes for IPPR-I and IPPR-M when the

leftmost node is a seed node (with Green color). The thick lines are interval valued edges with $[1-9]$ and the thin lines are scalar valued edges with 5. As discussed in Section 7.3.1, since the probabilities of interval valued edges are less than those of scalar valued edges, the rankings of nodes which have only incoming interval edges are lower than the nodes which have only incoming scalar edges. The impact of interval valued edges becomes less when a node has both incoming interval valued edges and scalar valued edges. On the right most layer, The node (with blue color) whose ranking is 9 in (Figure 7.4)(a) has both an incoming interval valued edge and an incoming scalar valued edge. The combination of these edges lifts its ranking score and makes the ranking close to the ranking in IPPR-M (Figure 7.4)(b). Therefore, If a graph has a high-mixing factor, it would be better to use IPPR-M instead of IPPR-I with almost same accuracy and faster execution. In Section 7.6, we evaluate the impact of different mixing factors and show when IPPR-I or IPPR-M is suitable for better accuracy.

When the graph has a high-mixing factor, it is better to use IPPR-M instead of IPPR-I, but instead of it, we can identify parts of a graph with low-mixing factor and limit IPPR-I computation only to those parts as we described in Chapter 3. Given an interval graph, we can compute the localities of the seed nodes and check whether each individual locality has high or low mixing factor, and use IPPR-I or IPPR-M appropriately for each locality.

7.6 Experimental Evaluation

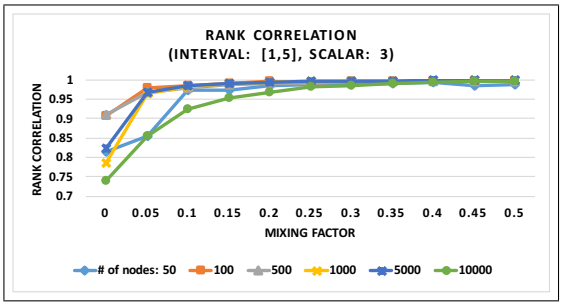
In this section, I will present results of experiments assessing the efficiency and effectiveness of IPPR-M and IPPR-I algorithms.

parameter	values
# of nodes	50, 100, 500, 1000, 5000, 10000
scalar & interval weights	3&[1,5], 50&[1,99], random&random([1,99])
number of out-degree for a node	4, 6, 8, random
% of interval edges for a node	50%, random
mixing factor	0, 0.5, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5

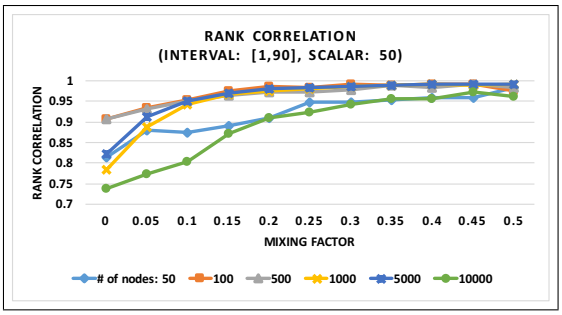
Table 7.1: Data and Parameters

7.6.1 Datasets and setup

Table 7.1 shows the overview of data and parameters. For all experiments, we start a graph which is a Tree-like random graph. The basic setting of the experiments is when the number of outgoing degree of nodes is 6 and half of edges is a set of scalar edges with 50 and the rest is a set of interval edges with [1,99]. I will change the numbers and show how the performance of PPR-M and PPR-I is changed. Here is how to generate a tree-like random graph. At first, given the number of nodes and the degree of a node, from a seed node, I made connections to nodes with the degree with setting half of edges are scalar value weights and the rest are interval value weights. Using Breath-first search, I continuously add more nodes until the number of nodes in the graph is larger than the given number of nodes. Since the graph is tree-structured, the number of edges is the number of nodes - 1. Given the graph, I added more edges with mixing factors and increased the number of edges randomly. When mixing factor is 0.1, we add 10% more edges with random edge weight type (scalar/interval) into the graph. The increase of mixing factors means the increase of clustering coefficient. The residual probability, α is set with default value chosen as 0.85.



(a) interval range [1,5]



(b) interval range [1,90]

Figure 7.5: Rank Correlation Results on Different Range of Intervals

7.6.2 Measure

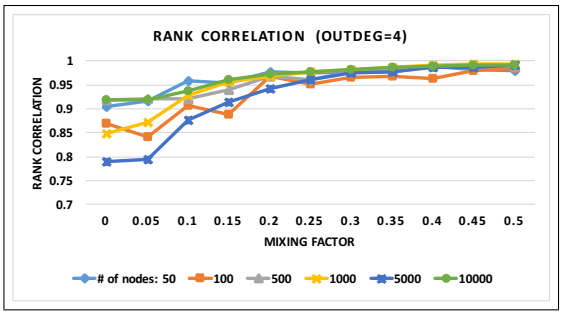
In this experiment, I consider the accuracy and efficiency. For the accuracy, I report the Spearman' rank correlation

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

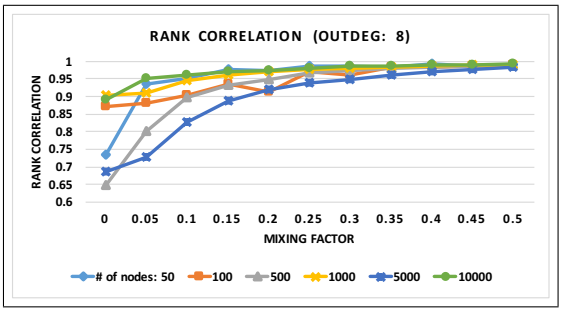
between the rankings of IPPR-M and IPPR-I. For efficiency, I check the execution time of IPPR-M and IPPR-I for different parameters.

7.6.3 Results and Discussions

Figure 7.5 shows the impact of range of interval edges. The results on large range of intervals show that the degree of increase of the correlation is lower than results of small range of intervals. Given small range of intervals, the mixing factor brings more significant impact on rank correlation. In Figure 7.5(a), when the mixing factor is 0.5, all correlation results became more than 0.95 and in Figure 7.5(b), after 0.3 mixing factor, correlation results are more than 0.95 which is much slower than small range results. This result shows that it is better to use IPPR-I than IPPR-M if the range of interval weights is large. Additionally, when the mixing factor is larger, the impact of interval edges becomes same as scalar edges. For example, in Figure 7.5(a), when the mixing factor becomes 0.2, the rank correlation is more than 0.95 which



(a) Degree of nodes: 4



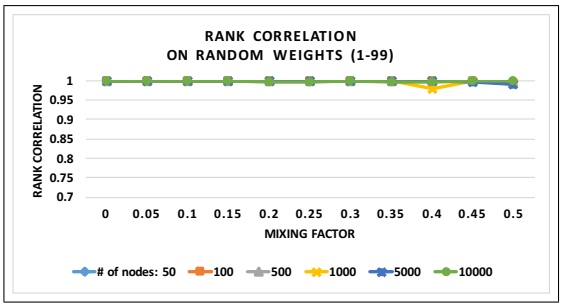
(b) Degree of nodes: 8

Figure 7.6: Rank Correlation Results on Different Outgoing Degrees of Nodes

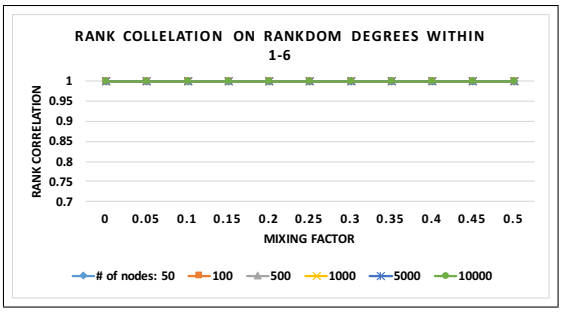
means that rankings of PPR-M is almost same as rankings of PPR-I.

Figure 7.6 shows how the number of degree of nodes affect IPPR-M and IPPR-I. When the outdegree is large and the graph structure is Tree-like, the correlation became much lower than the case of small outdegree. The degree of increase of correlation values as the mixing factor is increased is similar between small outdegree and large outdegree but when the cluster coefficient is low, it rather uses IPPR-I than IPPR-M for the better accuracy.

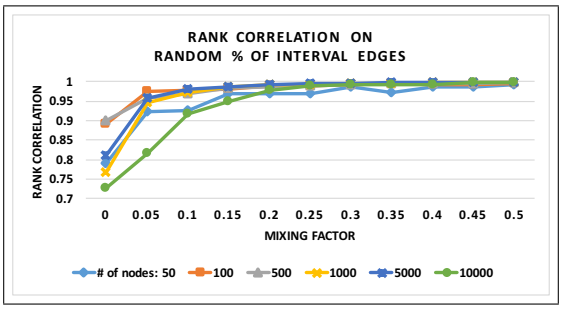
Figure 7.7 shows results of randomly selected cases. When the edge weights are random, the correlation stays close to 1 on all of mixing factors. This means that using PPR-M is a better selection when the variance of edge weights in a graph is high. Figure 7.7(b) shows that the high variance of out-degrees also depolarize the strength of interval edges. In both cases, PPR-I is a good choice in all mixing factors. In opposite to (a) and (b), for the random % of interval edges in a graph, Figure 7.7(c) shows that randomness on the number of interval edges is an exceptional case for the correlation. This is for the general case of interval edges with random percentage selection of outgoing edges on nodes. The shape of results are similar to Figure 7.5(a) with fast increase of correlations as the mixing factor is larger. The results show that the mixing factor has same impact in rank correlation as the small range of interval weights.



(a) Random edge weights

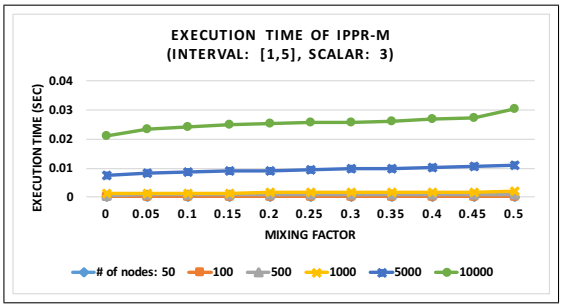


(b) Random out-degrees

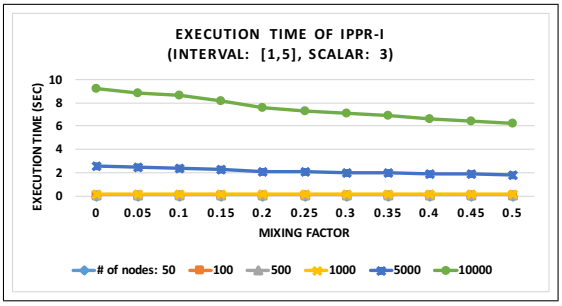


(c) random % of interval edges

Figure 7.7: Rank Correlation Results on Random Weights, Random Out-degrees, Random % of Interval Edges



(a) execution time of IPPR-M



(b) execution time of IPPR-I

Figure 7.8: Results of Execution on IPPR-M and IPPR-I

Figure 7.8(a) and (b) are the results of execution times of IPPR-M and IPPR-I. They show that the execution time of IPPR-M is much lower than that of IPPR-I. I do not report execution times on other experiments because the execution times for all different parameter settings stay consistent and are almost same as Figure 7.8.

The results show that we need to choose PPR-M or PPR-I based on the cluster coefficient, the variance of weights, and the variance of outdegrees.

Chapter 8

CONCLUSIONS

In this chapter, I briefly recap the contributions in this thesis. Node proximity measures are commonly used for quantifying how nearby or otherwise related to two or more nodes on a graph are. Node significance measures are applied to find the importance of nodes in a graph. PageRank is one of the most widely-used random-walk based methods for measuring node importance and has been used in a variety of application domains. Personalized PageRank is an alternative approach to find the relativeness and closeness of nodes related to the seed set given by the user. Despite its performance, there are some challenges such as 1) scalability issues in the large graph and 2) accuracy issues without considering on the relationship between the significance of the node and its degree. Additionally, it fails to compute ranking scores given an 3) uncertain graph with the exponential combinations of possible worlds.

8.1 Locality-Sensitive, Re-use Promoting, Approximate Personalized PageRank

For the scalability challenge, in Chapter 3, I 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 com-

puted 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, with negligible impacts on accuracy.

8.2 Impact Neighborhood Indexing in Diffusion Graphs

For the locality selection approach, in Chapter 4, I proposed a propagation and erasure based impact neighborhood indexing (`INI`) algorithm for efficiently identifying the neighborhood of a given node with formally defined the concept of zero-erasure and r-radius impact neighborhoods. I also proposed various optimization techniques to reduce false positives and improving storage and execution time efficiency of the algorithm.

8.3 Degree Decoupled PageRank

In Chapter 5, for the accuracy challenge, I noted that in many applications the relationship between the significance of the node and its degree in the underlying network may not be as strong (or as weak) as implied by PageRank-based measures. As I have experimentally shown in the section, in some applications, the significance of the node may even be negatively correlated with the node degree and in such applications a naive application of PageRank or personalized PageRank may return poor results. I proposed degree de-coupled PageRank (D2PR) technique to improve the effectiveness of PageRank based knowledge discovery and recommendation tasks. I first showed how to choose the penalty degree for penalizing or boosting the random-walk probability using the correlation between PageRank ranking scores and expected values on a given network and then showed how to calculate rankings using the penalty degree. Evaluations on different data graphs and recommendation tasks have confirmed that degree de-coupling would be an effective way to match application

specific node significances and improve recommendation accuracies using PageRank based approaches.

8.4 Uncertain Personalized PageRank

In Chapter 6, for the uncertainty of edge existence, I presented an uncertain edge model with mutual exclusion and multiple edge selections in uncertain graphs. While there are several ways to naively extend existing personalized PageRank computation techniques to graphs with uncertain edges, these either lead to large degrees of errors or are very expensive to compute in practice. I, therefore, proposed a novel Uncertain Personalized PageRank (UPPR) algorithm to approximately compute personalized PageRank values on such graphs. Experiments confirmed that the proposed technique has very high accuracy and is multiple-orders faster than available algorithms that can provide comparable accuracy.

8.5 Interval Personalized PageRank

For the challenge of uncertainty on edge weights as interval values, in Chapter 7, I explained that how much it is difficult to compute the ranking scores with interval weights on edges in a graph. It can be computed by interval matrix computation of PPR equations with sampling on interval ranges of edge weights but it requires to compute all possible worlds of combinations of edge weights. It leads to some problems on complexity, execution time, and low accuracies. To overcome the problem, I proposed Interval Personalized PageRank with Integrals (IPPR-I) algorithm which computes optimal PPR scores with integral calculations on interval edge weights. $IPPR-I$ returns accurate PPR scores without approximation but it requires execution time for integral computations. I also presented an efficient Interval Personalized PageRank with Mean (IPPR-M) which is an approximate personalized PageRank al-

gorithm that computes the scores quickly with mean of interval values in a graph. The experimental results shows when the cheaper and approximate $IPPR-M$ can be used instead of relatively expensive but accurate $IPPR-M$.

Chapter 9

FUTURE WORKS

In this chapter, I discuss my future research directions.

9.1 SVD Decomposition of an Interval Valued Matrix

In linear algebra, matrix decomposition is a factorization of a matrix into a product of matrices. In the real world, it is not feasible to calculate the matrix computations when the size of data matrices are very large, so decomposing matrices into some low-order canonical forms helps to compute and analyze the data with the inherent characteristic and structure of matrices. The problem is that, when a matrix has interval values, it is difficult to decompose matrices holding low error rates.

Let M be an interval matrix. If we split an interval valued M into two scalar valued matrices M_l and M_h, consisting of minimum and maximum values respectively, we can then seek decompositions

$$M_l = US_lV \text{ and } M_h = US_hV.$$

The problem is that, in general, it may not be possible to find left-singular vectors and right-singular vectors with the same vector of M_l and M_h. Therefore, we can instead seek

$$M_l = U_lS_lV_l \text{ and } M_h = U_hS_hV_h$$

such that $U_l \sim U_h$ and $V_l \sim V_h$ with

$$M_l \sim (\frac{U_l + U_h}{2})S_l(\frac{V_l + V_h}{2}) \text{ and } M_h \sim (\frac{U_l + U_h}{2})S_h(\frac{V_l + V_h}{2}).$$

Algorithm 1 Interval Matrix Decomposition

1: Let M be an interval matrix which is partitioned into two matrices M_l and M_h, consisting of minimum and maximum values respectively.

2: Let $\{\langle U_{l,1}, S_{l,1}, V_{l,1}\rangle, \ldots, \langle U_{l,n}, S_{l,n}, V_{l,n}\rangle\}$ be the n left-singular vectors, rectangular diagonal matrix, and right singular vectors pairs for M_l.

3: Let $\{\langle U_{h,1}, S_{h,1}, V_{h,1}\rangle, \ldots, \langle U_{h,m}, S_{h,m}, V_{h,m}\rangle\}$ be the m left-singular vectors, rectangular diagonal matrix, and right singular vectors pairs for M_h.

4: Let $k = min(n, m)$.

5: Find a mapping $\mu_i = \langle \mu_{l,i}, \mu_{h,i}\rangle$ such that $\sum_{1\le i\le k} \|U_{l,\mu_{l,i}} - U_{h,\mu_{h,i}}\|$ is minimum.

6: Given this mapping, for the first k pairs of M are

$$U'_i = avg\left(U_{l,\mu_{l,i}}, U_{h,\mu_{h,i}}\right) \ , \ V'_i = avg\left(V_{l,\mu_{l,i}}, V_{h,\mu_{h,i}}\right), \text{ and}$$

$$S'_i = [min(S_{l,\mu_{l,i}}, S_{h,\mu_{h,i}}), max(S_{l,\mu_{l,i}}, S_{h,\mu_{h,i}})]$$

7: If $k = n$ then, the next $m - n$ pairs are

$$U'_i = U_{h,i} \ , \ V'_i = V_{h,i}, \text{ and } S'_i = [min(0, S_{h,i}), max(0, S_{h,i})]$$

where $U_{h,i}$, $V_{h,i}$, and $S_{h,i}$ are the unmatched vectors and eigenvalues of M_h.

8: If $k = m$, then, the next $n - m$ pairs are

$$U'_i = U_{l,i} \ , \ V'_i = V_{l,i}, \text{ and } S'_i = [min(0, S_{l,i}), max(0, S_{l,i})]$$

where $U_{l,i}$, $V_{l,i}$, and $S_{l,i}$ are the unmatched vectors and eigenvalues of M_l.

Given these, we have

$$S' = [min(S_l, S_h), max(S_l, S_h)] \ , \ U' = \frac{U_l + U_h}{2}, \text{ and } V' = \frac{V_l + V_h}{2}$$

which U', S', and V' are the decomposed matrices from an interval matrix M.

Based on the above approach, Algorithm 1 shows steps how to decompose an interval matrix. At step 1, given an interval matrix M, it is separated into two

matrices which have scalar valued matrices M_l and M_h for minimum and maximum values of M. At step 2 and 3, we compute the matrix decomposition on each matrix with basic SVD matrix decomposition. After finding the minimum low ranks of two matrices (step 4), in step 5, we use a matching algorithm to minimize the difference between left-singular vectors, U_l and U_h. In step 6, we get the decomposed matrices U', S', and V' with averaging matched $U_{l,\mu_{l,i}}$ and $U_{h,\mu_{h,i}}$, averaging matched $V_{l,\mu_{l,i}}$ and $V_{h,\mu_{h,i}}$, and finding the minimum eigenvalue and the maximum eigenvalue between $S_{l,\mu_{l,i}}$ and $S_{h,\mu_{h,i}}$. This approach keeps the interval values only in S' and U' and V' have only scalar values in the matrices.

9.2 Matrix Inverse on an Interval Valued Matrix

I, first, explain interval matrix arithmetics and propose an efficient algorithm how to compute an inverse matrix of an interval-valued Matrix.

9.2.1 Interval Matrix Arithmetic

Let M be an interval matrix whose values are defined by interval. $M(i,j)$ is an interval value and can be defined as $[a_*, a^*]$ where a_* is the minimum value and a^* is the maximum value. $span([a_*, a^*])$ is an integer that is the range of interval value and computed by $(a^* - a_*)$.

For given $M_a(i,j) = [a_*, a^*]$ and $M_b(i,j) = [b_*, b^*]$ of two interval matrix M_a and M_b,

- the addition of interval values in two interval matrices:

$$[a_*, a^*] + [b_*, b^*] = [a_* + b_*, a^* + b^*]$$

- the subtraction of interval values:

$$[a_*, a^*] - [b_*, b^*] = [a_* - b_*, a^* - b^*]$$

- the multiplication of interval values:

$$[a_*, a^*] \times [b_*, b^*] = [min(a_* \times b_*, a_* \times b^*, a^* \times b_*, a^* \times b^*),$$

$$max(a_* \times b_*, a_* \times b^*, a^* \times b_*, a^* \times b^*)]$$

When the value is scalar such as $M_a(i,j) = a = a_* = a^*$, the multiplication is defined as

$$[a, a] \times [b_*, b^*] = [min(a \times b_*, a \times b^*), max(a \times b_*, a \times b^*)].$$

9.2.2 Matrix Inverse in a Diagonal Interval Valued Matrix

Let S be a $k \times k$ diagonal interval matrix, where the entries in the diagonal may have interval values and the rest of the entries are 0. Let assume that the entries in the diagonal are non-negative numbers.

We seek a $k \times k$ diagonal matrix, S^{-1}, such that

$$S\, S^{-1} = \tilde{I},$$

where $\tilde{I}$ is a $k \times k$ interval valued matrix, approximately equal to the identity matrix. More specifically, for all $1 \leq i \leq k$, we have $\tilde{I}(i,i) = [1 - \epsilon_i, 1 + \epsilon_i]$, where $0 \leq \epsilon_i \leq 1$. We solve for S^{-1} as follows: Let $S(i,i) = [s_{i*}, s_i^*]$ and $S^{-1}(i,i) = [\sigma_{i*}, \sigma_i^*]$. We seek σ_{i*} and σ_i^* values that minimize the value of ϵ_i subject to the constraints:

$$s_{i*} \times \sigma_{i*} = 1 - \epsilon_i \ , \ s_i^* \times \sigma_i^* = 1 + \epsilon_i \ , \ 0 \leq \epsilon_i \leq 1, \text{ and } \sigma_{i*} \leq \sigma_i^*.$$

We can apply $s_{i*} \times \sigma_{i*} = 1 - \epsilon_i$ and $s_i^* \times \sigma_i^* = 1 + \epsilon_i$ to $\sigma_{i*} \leq \sigma_i^*$.

$$\frac{1 - \epsilon_i}{s_{i*}} \leq \frac{1 + \epsilon_i}{s_i^*}$$

This equation can be rewritten as

$$(\frac{1}{s_{i*}} + \frac{1}{s_i^*})\epsilon_i \geq \frac{1}{s_{i*}} - \frac{1}{s_i^*}.$$

Using this equation, we get the following equation,

$$0 \leq \frac{s_i^* - s_{i*}}{s_{i*} + s_i^*} \leq \epsilon_i \leq 1.$$

This equation shows that ϵ_i is minimum when it is equal to $\frac{s_i^* - s_{i*}}{s_{i*} + s_i^*}$, and this case works when $\sigma_{i*} = \sigma_i^*$. Therefore, after inverting the interval matrix S whose elements contain interval values, elements in interval inverse matrix S^{-1} has only scalar values where $\sigma_{i*} = \sigma_i^*$. Additionally, when $\sigma_{i*} = \sigma_i^* = \sigma_i$, the equations,

$$s_{i*} \times \sigma_i = 1 - \epsilon_i \text{ and } s_i^* \times \sigma_i = 1 + \epsilon_i,$$

are used to get σ_i as follows,

$$\sigma_i = \frac{2}{s_{i*} + s_i^*} = inv(\frac{s_{i*} + s_i^*}{2}).$$

Using this equation, we can easily get the inverse matrix and the elements become scalar values.

REFERENCES

[1] Adar, E., L. Adamic *et al.*, "Tracking information epidemics in blogspace", in "Web intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM international conference on", pp. 207–214 (IEEE, 2005).

[2] Adar, E. and C. Re, "Managing uncertainty in social networks.", IEEE Data Eng. Bull. **30**, 2, 15–22 (2007).

[3] Aggarwal, C. C., *Managing and Mining Uncertain Data* (Springer Publishing Company, Incorporated, 2009).

[4] Akiba, T., C. Sommer and K.-i. Kawarabayashi, "Shortest-path queries for complex networks: exploiting low tree-width outside the core", in "Proceedings of the 15th International Conference on Extending Database Technology", pp. 144–155 (ACM, 2012).

[5] Anagnostopoulos, A., R. Kumar and M. Mahdian, "Influence and correlation in social networks", in "Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 7–15 (ACM, 2008).

[6] Avrachenkov, K. and N. Litvak, "The effect of new links on google pagerank", Stochastic Models **22**, 2, 319–331 (2006).

[7] Avrachenkov, K., N. Litvak, D. Nemirovsky and N. Osipova, "Monte carlo methods in pagerank computation: When one iteration is sufficient", SIAM Journal on Numerical Analysis **45**, 2, 890–904 (2007).

[8] Avrachenkov, K., N. Litvak, D. Nemirovsky, E. Smirnova and M. Sokol, "Quick detection of top-k personalized pagerank lists", in "Algorithms and Models for the Web Graph", pp. 50–61 (Springer, 2011).

[9] Bahmani, B., A. Chowdhury and A. Goel, "Fast incremental and personalized pagerank", Proceedings of the VLDB Endowment **4**, 3, 173–184 (2010).

[10] Bakshy, E., B. Karrer and L. A. Adamic, "Social influence and the diffusion of user-created content", in "Proceedings of the 10th ACM conference on Electronic commerce", pp. 325–334 (ACM, 2009).

[11] Balmin, A., V. Hristidis and Y. Papakonstantinou, "Objectrank: Authority-based keyword search in databases", in "Proceedings of the Thirtieth international conference on Very large data bases-Volume 30", pp. 564–575 (VLDB Endowment, 2004).

[12] Becchetti, L., C. Castillo, D. Donato, S. Leonardi and R. Baeza-Yates, "Using rank propagation and probabilistic counting for link-based spam detection", in "Proc. of WebKDD", vol. 6 (2006).

[13] Boldi, P., F. Bonchi, A. Gionis and T. Tassa, "Injecting uncertainty in graphs for identity obfuscation", Proceedings of the VLDB Endowment **5**, 11, 1376–1387 (2012).

[14] Boldi, P., M. Rosa and S. Vigna, "Hyperanf: Approximating the neighbourhood function of very large graphs on a budget", in "Proceedings of the 20th international conference on World wide web", pp. 625–634 (ACM, 2011).

[15] Borgatti, S. P., C. Jones and M. G. Everett, "Network measures of social capital", Connections **21**, 2, 27–36 (1998).

[16] Brand, M., "Fast online svd revisions for lightweight recommender systems.", in "SDM", pp. 37–46 (SIAM, 2003).

[17] Brin, S. and L. Page, "The anatomy of a large-scale hypertextual web search engine", in "Proceedings of the Seventh International Conference on World Wide Web 7", pp. 107–117 (Elsevier Science Publishers B. V., 1998).

[18] Butler, K. and M. Stephens, "The distribution of a sum of binomial random variables", Tech. rep., STANFORD UNIV CA DEPT OF STATISTICS (1993).

[19] Candan, K. S. and W.-S. Li, "Using random walks for mining web document associations", in "Knowledge Discovery and Data Mining. Current Issues and New Applications", pp. 294–305 (Springer, 2000).

[20] Candan, K. S. and W.-S. Li, "Reasoning for web document associations and its applications in site map construction", Data & Knowledge Engineering **43**, 2, 121–150 (2002).

[21] Cantador, I., P. Brusilovsky and T. Kuflik, "2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011)", in "Proceedings of the 5th ACM conference on Recommender systems", RecSys 2011 (ACM, New York, NY, USA, 2011).

[22] Cao, H., K. S. Candan and M. L. Sapino, "Skynets: Searching for minimum trees in graphs with incomparable edge weights", in "Proceedings of the 20th ACM international conference on Information and knowledge management", pp. 1775–1784 (ACM, 2011).

[23] Chakrabarti, S., "Dynamic personalized pagerank in entity-relation graphs", in "Proceedings of the 16th international conference on World Wide Web", pp. 571–580 (ACM, 2007).

[24] Chen, M., J. Liu and X. Tang, "Clustering via random walk hitting time on directed graphs.", in "AAAI", vol. 8, pp. 616–621 (2008).

[25] Chen, W., W. Fang, G. Hu and M. W. Mahoney, "On the hyperbolicity of small-world and treelike random graphs", Internet Mathematics **9**, 4, 434–491 (2013).

[26] Chen, W., C. Wang and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks", in "Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 1029–1038 (ACM, 2010).

[27] Cohen, E., E. Halperin, H. Kaplan and U. Zwick, "Reachability and distance queries via 2-hop labels", SIAM Journal on Computing **32**, 5, 1338–1355 (2003).

[28] Collins, J. B. and S. T. Smith, "Network discovery for uncertain graphs", in "Information Fusion (FUSION), 2014 17th International Conference on", pp. 1–8 (IEEE, 2014).

[29] Cooper, C., T. Radzik and Y. Siantos, "A fast algorithm to find all high degree vertices in graphs with a power law degree sequence", in "Algorithms and Models for the Web Graph", pp. 165–178 (Springer, 2012).

[30] Csáji, B. C., R. M. Jungers and V. D. Blondel, "Pagerank optimization by edge selection", Discrete Applied Mathematics **169**, 73–87 (2014).

[31] De Choudhury, M., Y.-R. Lin, H. Sundaram, K. S. Candan, L. Xie and A. Kelliher, "How does the data sampling strategy impact the discovery of information diffusion in social media?", ICWSM **10**, 34–41 (2010).

[32] De Kerchove, C., L. Ninove and P. Van Dooren, "Maximizing pagerank via outlinks", Linear Algebra and its Applications **429**, 5, 1254–1276 (2008).

[33] Du, L., C. Li, H. Chen, L. Tan and Y. Zhang, "Probabilistic simrank computation over uncertain graphs", Information Sciences **295**, 521–535 (2015).

[34] Eiron, N., K. S. McCurley and J. A. Tomlin, "Ranking the web frontier", in "Proceedings of the 13th international conference on World Wide Web", pp. 309–318 (ACM, 2004).

[35] Feige, U., M. Hajiaghayi and J. R. Lee, "Improved approximation algorithms for minimum-weight vertex separators", in "Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing", pp. 563–572 (ACM, 2005).

[36] Fercoq, O., "Pagerank optimization applied to spam detection", in "Network Games, Control and Optimization (NetGCooP), 2012 6th International Conference on", pp. 127–134 (IEEE, 2012).

[37] Fogaras, D., B. Rácz, K. Csalogány and T. Sarlós, "Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments", Internet Mathematics **2**, 3, 333–358 (2005).

[38] Fouss, F., A. Pirotte, J.-M. Renders and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation", Knowledge and data engineering, ieee transactions on **19**, 3, 355–369 (2007).

[39] Fujiwara, Y., M. Nakatsuji, M. Onizuka and M. Kitsuregawa, "Fast and exact top-k search for random walk with restart", Proceedings of the VLDB Endowment **5**, 5, 442–453 (2012).

[40] Fujiwara, Y., M. Nakatsuji, T. Yamamuro, H. Shiokawa and M. Onizuka, "Efficient personalized pagerank with accuracy assurance", in "Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 15–23 (ACM, 2012).

[41] Gionis, A., P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing", in "VLDB", vol. 99, pp. 518–529 (1999).

[42] Gleich, D. and M. Polito, "Approximating personalized pagerank with minimal use of web graph data", Internet Mathematics **3**, 3, 257–294 (2006).

[43] Guille, A., H. Hacid, C. Favre and D. A. Zighed, "Information diffusion in online social networks: A survey", ACM SIGMOD Record **42**, 2, 17–28 (2013).

[44] Gunnels, J., C. Lin, G. Morrow and R. Van De Geijn, "A flexible class of parallel matrix multiplication algorithms", in "Parallel Processing Symposium, 1998. IPPS/SPDP 1998. Proceedings of the First Merged International... and Symposium on Parallel and Distributed Processing 1998", pp. 110–116 (IEEE, 1998).

[45] Gupta, A., G. Karypis and V. Kumar, "Highly scalable parallel algorithms for sparse matrix factorization", IEEE Transactions on Parallel and Distributed Systems **8**, 5, 502–520 (1997).

[46] Gupta, M., A. Pathak and S. Chakrabarti, "Fast algorithms for topk personalized pagerank queries", in "Proceedings of the 17th international conference on World Wide Web", pp. 1225–1226 (ACM, 2008).

[47] Harper, F. M. and J. A. Konstan, "The movielens datasets: History and context", ACM Transactions on Interactive Intelligent Systems (TiiS) **5**, 4, 19 (2016).

[48] Haveliwala, T. H., "Topic-sensitive pagerank", in "Proceedings of the 11th international conference on World Wide Web", pp. 517–526 (ACM, 2002).

[49] Hickey, T., Q. Ju and M. H. Van Emden, "Interval arithmetic: From principles to implementation", Journal of the ACM (JACM) **48**, 5, 1038–1068 (2001).

[50] IMDB, "Imdb", `http://www.imdb.com/`, [Online; accessed 2013] (1990).

[51] Ishii, H. and R. Tempo, "Computing the pagerank variation for fragile web data", SICE Journal of Control, Measurement, and System Integration **2**, 1, 1–9 (2009).

[52] Ishii, H. and R. Tempo, "Fragile link structure in pagerank computation", in "Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on", pp. 121–126 (IEEE, 2009).

[53] Jeh, G. and J. Widom, "Simrank: a measure of structural-context similarity", in "Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 538–543 (ACM, 2002).

[54] Jeh, G. and J. Widom, "Scaling personalized web search", in "Proceedings of the 12th international conference on World Wide Web", pp. 271–279 (ACM, 2003).

[55] Jin, R., L. Liu, B. Ding and H. Wang, "Distance-constraint reachability computation in uncertain graphs", Proceedings of the VLDB Endowment **4**, 9, 551–562 (2011).

[56] Kamvar, S. D., T. H. Haveliwala, C. D. Manning and G. H. Golub, "Extrapolation methods for accelerating pagerank computations", in "Proceedings of the 12th international conference on World Wide Web", pp. 261–270 (ACM, 2003).

[57] Karypis, G. and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs", SIAM Journal on scientific Computing **20**, 1, 359–392 (1998).

[58] Kempe, D., J. Kleinberg and É. Tardos, "Maximizing the spread of influence through a social network", in "Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 137–146 (ACM, 2003).

[59] Kempe, D., J. Kleinberg and É. Tardos, "Influential nodes in a diffusion model for social networks", in "Automata, languages and programming", pp. 1127–1138 (Springer, 2005).

[60] Khan, A., F. Bonchi, A. Gionis and F. Gullo, "Fast reliability search in uncertain graphs.", in "EDBT", pp. 535–546 (2014).

[61] Khan, A. and L. Chen, "On uncertain graphs modeling and queries", Proceedings of the VLDB Endowment **8**, 12, 2042–2043 (2015).

[62] Kim, J. H., K. S. Candan and M. L. Sapino, "Impact neighborhood indexing (ini) in diffusion graphs", in "Proceedings of the 21st ACM international conference on Information and knowledge management", pp. 2184–2188 (ACM, 2012).

[63] Kim, J. H., K. S. Candan and M. L. Sapino, "Lr-ppr: Locality-sensitive, re-use promoting, approximate personalized pagerank computation", in "Proceedings of the 22nd ACM international conference on Conference on information & knowledge management", pp. 1801–1806 (ACM, 2013).

[64] Kim, J. H., K. S. Candan and M. L. Sapino, "Locality-sensitive and re-use promoting personalized pagerank computations", Knowledge and Information Systems **47**, 2, 261–299 (2016).

[65] Kim, J. H., K. S. Candan and M. L. Sapino, "Pagerank revisited: On the relationship between node degrees and node significances in different applications", in "Proceedings of the 19th International Conference on Extending Database Technology Workshops", (ACM, 2016).

[66] Kim, J. H., X. Chen, K. S. Candan and M. L. Sapino, "Hive open research network platform", in "Proceedings of the 16th International Conference on Extending Database Technology", pp. 733–736 (ACM, 2013).

[67] Kim, J. H., M.-L. Li, K. S. Candan and M. L. Sapino, "Personalized pagerank in uncertain graphs with mutually exclusive edges", in "Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval", (ACM, 2017).

[68] Kim, J. W., K. S. Candan and J. Tatemura, "Efficient overlap and content reuse detection in blogs and online news articles", in "Proceedings of the 18th international conference on World wide web", pp. 81–90 (ACM, 2009).

[69] Kleinberg, J. M., "Authoritative sources in a hyperlinked environment", Journal of the ACM (JACM) **46**, 5, 604–632 (1999).

[70] Langville, A. N. and C. D. Meyer, *Google's PageRank and beyond: The science of search engine rankings* (Princeton University Press, 2011).

[71] Lemire, D., C. Moon, D. McIntosh, R. Becho, C. Ranger, V. Zenz and O. Kaser, "Javaewah", `https://github.com/lemire/javaewah`, [Online; accessed 2012] (2014).

[72] Leskovec, J., J. Kleinberg and C. Faloutsos, "Graph evolution: Densification and shrinking diameters", ACM Transactions on Knowledge Discovery from Data (TKDD) **1**, 1, 2 (2007).

[73] Leskovec, J., A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen and N. Glance, "Cost-effective outbreak detection in networks", in "Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 420–429 (ACM, 2007).

[74] Leskovec, J. and A. Krevl, "{SNAP Datasets}:{Stanford} large network dataset collection", (2015).

[75] Li, R.-H., J. X. Yu, R. Mao and T. Jin, "Efficient and accurate query evaluation on uncertain graphs via recursive stratified sampling", in "Data Engineering (ICDE), 2014 IEEE 30th International Conference on", pp. 892–903 (IEEE, 2014).

[76] Liben-Nowell, D. and J. Kleinberg, "The link-prediction problem for social networks", journal of the Association for Information Science and Technology **58**, 7, 1019–1031 (2007).

[77] Liu, Q., Z. Li, J. Lui and J. Cheng, "Powerwalk: Scalable personalized pagerank via random walks with vertex-centric decomposition", in "Proceedings of the 25th ACM International on Conference on Information and Knowledge Management", pp. 195–204 (ACM, 2016).

[78] Lofgren, P., "Efficient algorithms for personalized pagerank", arXiv preprint arXiv:1512.04633 (2015).

[79] Maehara, T., T. Akiba, Y. Iwata and K.-i. Kawarabayashi, "Computing personalized pagerank quickly by exploiting graph structures", Proceedings of the VLDB Endowment **7**, 12, 1023–1034 (2014).

[80] Malewicz, G., M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser and G. Czajkowski, "Pregel: a system for large-scale graph processing", in "Proceedings of the 2010 ACM SIGMOD International Conference on Management of data", pp. 135–146 (ACM, 2010).

[81] Mathieu, F. and L. Viennot, "Local aspects of the global ranking of web pages", in "6th International Workshop on Innovative Internet Community Systems (I2CS)", pp. 493–506 (2006).

[82] Mei, Q., D. Zhou and K. Church, "Query suggestion using hitting time", in "Proceedings of the 17th ACM conference on Information and knowledge management", pp. 469–478 (ACM, 2008).

[83] Myers, S. A., C. Zhu and J. Leskovec, "Information diffusion and external influence in networks", in "Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 33–41 (ACM, 2012).

[84] Newman, M. E., "Finding community structure in networks using the eigenvectors of matrices", Physical review E **74**, 3, 036104 (2006).

[85] Nguyen, V. and C. Martel, "Analysis and models for small-world graphs", in "Proceedings of Symposium on Discrete Algorithms, ACM-SIAM", vol. 16 (2005).

[86] Nikolakopoulos, A. N. and J. D. Garofalakis, "Ncdawarerank: a novel ranking method that exploits the decomposable structure of the web", in "Proceedings of the sixth ACM international conference on Web search and data mining", pp. 143–152 (ACM, 2013).

[87] Niu, X., L. Li and K. Xu, "Digrank: Using global degree to facilitate ranking in an incomplete graph", in "Proceedings of the 20th ACM international conference on Information and knowledge management", pp. 2297–2300 (ACM, 2011).

[88] Olsen, M., "Maximizing pagerank with new backlinks", in "Algorithms and Complexity", pp. 37–48 (Springer, 2010).

[89] Opsahl, T. and P. Panzarasa, "Clustering in weighted networks", Social networks **31**, 2, 155–163 (2009).

[90] Palmer, C. R., P. B. Gibbons and C. Faloutsos, "Anf: A fast and scalable tool for data mining in massive graphs", in "Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 81–90 (ACM, 2002).

[91] Pease, M. C., "Matrix inversion using parallel processing", Journal of the ACM (JACM) **14**, 4, 757–764 (1967).

[92] Piegorsch, W. W. and G. Casella, "Erratum: inverting a sum of matrices", SIAM Review **32**, 3, 470–470 (1990).

[93] Potamias, M., F. Bonchi, A. Gionis and G. Kollios, "K-nearest neighbors in uncertain graphs", Proceedings of the VLDB Endowment **3**, 1-2, 997–1008 (2010).

[94] Rump, S., "INTLAB - INTerval LABoratory", in "Developments in Reliable Computing", edited by T. Csendes, pp. 77–104 (Kluwer Academic Publishers, Dordrecht, 1999), `http://www.ti3.tuhh.de/rump/`.

[95] Sarkar, P., A. W. Moore and A. Prakash, "Fast incremental proximity search in large graphs", in "Proceedings of the 25th international conference on Machine learning", pp. 896–903 (ACM, 2008).

[96] Shakarian, P., M. Broecheler, V. Subrahmanian and C. Molinaro, "Using generalized annotated programs to solve social network diffusion optimization problems", ACM Transactions on Computational Logic (TOCL) **14**, 2, 10 (2013).

[97] Shin, K., J. Jung, S. Lee and U. Kang, "Bear: Block elimination approach for random walk with restart on large graphs", in "Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data", pp. 1571–1585 (ACM, 2015).

[98] Siddique, K., Z. Akhtar, E. J. Yoon, Y.-S. Jeong, D. Dasgupta and Y. Kim, "Apache hama: An emerging bulk synchronous parallel computing framework for big data applications", IEEE Access **4**, 8879–8887 (2016).

[99] Song, H. H., T. W. Cho, V. Dave, Y. Zhang and L. Qiu, "Scalable proximity estimation and link prediction in online social networks", in "Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference", pp. 322–335 (ACM, 2009).

[100] Su, J., A. Sharma and S. Goel, "The effect of recommendations on network structure", in "Proceedings of the 25th International Conference on World Wide Web", pp. 1157–1167 (International World Wide Web Conferences Steering Committee, 2016).

[101] Tang, J., S. Chang, C. Aggarwal and H. Liu, "Negative link prediction in social media", in "Proceedings of the Eighth ACM International Conference on Web Search and Data Mining", pp. 87–96 (ACM, 2015).

[102] Tang, J., H. Gao and H. Liu, "mtrust: discerning multi-faceted trust in a connected world", in "Proceedings of the fifth ACM international conference on Web search and data mining", pp. 93–102 (ACM, 2012).

[103] Tang, J., J. Zhang, L. Yao, J. Li, L. Zhang and Z. Su, "Arnetminer: extraction and mining of academic social networks", in "Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 990–998 (ACM, 2008).

[104] Tong, H. and C. Faloutsos, "Center-piece subgraphs: problem definition and fast solutions", in "Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 404–413 (ACM, 2006).

[105] Tong, H., C. Faloutsos and Y. Koren, "Fast direction-aware proximity for graph mining", in "Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 747–756 (ACM, 2007).

[106] Tong, H., C. Faloutsos and J.-Y. Pan, "Fast random walk with restart and its applications", (2006).

[107] Wang, Y. and J. Chu, "Use noisy link analysis to improve web search", in "Proceedings of the 20th ACM conference on Hypertext and hypermedia", pp. 377–378 (ACM, 2009).

[108] Watts, D. J. and P. S. Dodds, "Influentials, networks, and public opinion formation", Journal of consumer research **34**, 4, 441–458 (2007).

[109] Watts, D. J. and S. H. Strogatz, "Collective dynamics of small-worldnetworks", nature **393**, 6684, 440–442 (1998).

[110] Wei, F., "Tedi: efficient shortest path query answering on graphs", in "Proceedings of the 2010 ACM SIGMOD International Conference on Management of data", pp. 99–110 (ACM, 2010).

[111] Weng, L., J. Ratkiewicz, N. Perra, B. Gonçalves, C. Castillo, F. Bonchi, R. Schifanella, F. Menczer and A. Flammini, "The role of information diffusion in the evolution of social networks", in "Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 356–364 (ACM, 2013).

[112] White, D. R. and S. P. Borgatti, "Betweenness centrality measures for directed graphs", Social Networks **16**, 4, 335–346 (1994).

[113] Williams, V., "Breaking the coppersmith-winograd barrier. unpublished manuscript", (2011).

[114] Wu, Y. and L. Raschid, "Approxrank: Estimating rank for a subgraph", in "Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on", pp. 54–65 (IEEE, 2009).

[115] Xiao, Y., W. Wu, J. Pei, W. Wang and Z. He, "Efficiently indexing shortest paths by exploiting symmetry in graphs", in "Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology", pp. 493–504 (ACM, 2009).

[116] Yuan, Y., L. Chen and G. Wang, "Efficiently answering probability threshold-based shortest path queries over uncertain graphs", in "International Conference on Database Systems for Advanced Applications", pp. 155–170 (Springer, 2010).

[117] Zhang, L., T. Qin, T.-Y. Liu, Y. Bao and H. Li, *N-step PageRank for web search* (Springer, 2007).

[118] Zhu, K., W. Zhang, G. Zhu, Y. Zhang and X. Lin, "Bmc: an efficient method to evaluate probabilistic reachability queries", in "International Conference on Database Systems for Advanced Applications", pp. 434–449 (Springer, 2011).

[119] Zobel, J., A. Moffat and K. Ramamohanarao, "Inverted files versus signature files for text indexing", ACM Transactions on Database Systems (TODS) **23**, 4, 453–490 (1998).

[120] Zou, L., L. Chen and M. T. Özsu, "Distance-join: Pattern match query in a large graph database", Proceedings of the VLDB Endowment **2**, 1, 886–897 (2009).

[121] Zou, Z., J. Li, H. Gao and S. Zhang, "Mining frequent subgraph patterns from uncertain graph data", IEEE Transactions on Knowledge and Data Engineering **22**, 9, 1203–1218 (2010).