Research Square

Preprints are preliminary reports that have not undergone peer review. They should not be considered conclusive, used to inform clinical practice, or referenced by the media as validated information.

FPPR: Fast Pessimistic (dynamic) PageRank to Update PageRank in Evolving Directed Graphs on Network Changes

Suman Kundu (suman@iitj.ac.in)
Indian Institute of Technology Jodhpur

Rohith Parjanya Pashikanti
Indian Institute of Technology Jodhpur

Research Article

Keywords: Dynamic Network, Randomized Algorithm, Link Sensitivity Index, Approximate Visit, Dynamic PageRank

Posted Date: April 4th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1512145/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

FPPR: Fast Pessimistic (dynamic) PageRank to Update PageRank in Evolving Directed Graphs on Network Changes

Rohith Parjanya Pashikanti[1†] and Suman Kundu[1*†]

[1]Cognitive and Social Analytics Lab, Department of Computer Science and Engineering, Indian Institute of Technology Jodhpur, NH 62, Surpura Bypass Rd, Karwar, Jodhpur, 342030, Rajasthan, India.

*Corresponding author(s). E-mail(s): suman@iitj.ac.in;
Contributing authors: parjanya.1@iitj.ac.in;
†These authors contributed equally to this work.

Abstract

The paper presents a new algorithm FPPR which updates PageRanks of a directed network after topological changes in the graphs. The algorithm is capable of regenerating scores on node and link addition/deletion. The changes in the expected value of random surfers are used for updating the scores of the newly added nodes as well as the impacted chain where the nodes/links are added or removed. The complexity of the algorithm for $\boldsymbol{k}$ new node addition is $\boldsymbol{\mathcal{O}(k \times d_{avg}^{(k)})}$ where $\boldsymbol{d_{avg}^{(k)}}$ is the average degree of $\boldsymbol{k}$ nodes added. On the other hand for node deletion the complexity is $\boldsymbol{\mathcal{O}(|V_s| + |E_s|)}$ where $\boldsymbol{V_s}$ and $\boldsymbol{E_s}$ the set of nodes and edges updated using Selective Breath First Update. Extensive experiments have been performed on different synthetic and real-world networks. The experimental result shows that the rank generated by the proposed method is highly correlated with that of the recalculation on changes using the benchmark Power Iteration algorithm.

Keywords: Dynamic Network, Randomized Algorithm, Link Sensitivity Index, Approximate Visit, Dynamic PageRank

1 Introduction

Ranking search results of any web search is an important task. PageRank [18] is one of the pioneer algorithms to rank web pages. There were only a few pages in the initial days of the internet. So, static page ranking algorithms were sufficient. However, as the WWW (World Wide Web) started growing, the calculation of PageRank became more and more complex and computationally challenging. With the growth of the internet, many subnetworks appeared and PageRank provides different values to those networks. Thus PageRank is not limited to web search only. These subnetworks may have dynamic characteristics. Dynamic networks are those networks where nodes and links get added or deleted with time. Today's internet is full of dynamic networks. For example, the Twitter retweet network, where the addition of nodes (retweets) happen frequently, the following/followers network in Twitter/Instagram, where addition and deletion of nodes, links take place frequently, Twitter mentions network changes with each Tweet post, the citation network, where research papers get added over time etc. Calculating PageRank in such a dynamic network is an important and challenging research problem. The trivial way to find PageRank in dynamic networks is to run the static PageRank methods after every update in the network. This is time-consuming and non-sustainable for rapid updates in the network.

The real-time use cases like finding the top-k popular products in an online shopping cart or finding top-k spreaders of news considering reshare network etc., will need high-speed computation of PageRank. PageRank is also used to get suggestions to users with other accounts to follow in Twitter [9]. PageRank also finds applications in network outside internet. For example, identify new possible drug targets in proteins [11] in biochemistry; predicting how many pedestrians/vehicles visit the individual places, or streets [12] etc.

The PageRank was initially proposed in [18]. It is the classical algorithm for ranking web pages. Over the years, many variations of PageRank [4, 15, 16, 21, 23, 28] had been evolved. Randomized algorithms [2, 4, 21, 23, 28] for approximating PageRanks were also introduced. These algorithms are called Monte Carlo based algorithms. Randomized Monte Carlo algorithms provide a reasonable estimation of PageRank. All these algorithms are designed for static networks. There are methods in the literature to solve the problem of dynamic PageRank [5, 7, 10, 15, 17, 25] as also. In [5, 7, 15], a subset of graph is chosen and PageRank is computed on that subset using static methods for any updates. In [10, 17, 25] random walk model is used, where random walk segments are adjusted in case of updates. However, these algorithms could not provide satisfying results either in terms of accuracy or computational complexity, especially for Big Data Networks. Hence, a simple dynamic PageRank algorithm is critical to address the above problems.

This paper proposes a simple algorithm, namely, Fast Pessimistic dynamic PageRank (FPPR) to approximate the PageRank on change in network topology. Different topology changes considered are (i) adding a new node/link to the network and (ii) deleting a node/link from the network. The present work

is an extension of our initial results [19] where only node addition was considered. The proposed algorithm uses the expected value of a random surfer to re-calculate the score for changes in the topology. For example, when the new node or link is added, the score is calculated by adding the scores contributed by inlinks to the new node or target node in the case of link addition and estimating to what extent the chain may contribute to the score of the node. The same method is used to update existing nodes through the outlinks of the newly added node. FPPR uses Selective Breadth First Update (SBFU) for node deletion. SBFU updates the score of nodes and links traversed during the process with the appropriate deductions calculated by the expected values. Furthur for link deletion, FPPR performs a local update adjusting the PageRank score of the target node only. The proposed algorithm takes $\mathcal{O}(k \times d_{avg}^{(k)})$ time complexity for k node or link addition. Here $d_{avg}^{(k)}$ is the average degree of the k nodes added to the graph. Time complexity for node deletion and link deletion is $\mathcal{O}(V_s + E_s)$ and $\mathcal{O}(1)$ respectively, where V_s is the set of nodes which are updated, E_s is the set of edges associated with V_s. Further, FPPR takes only $\mathcal{O}(|V|)$ additional space, specifically, $4 \times |V|$ space in worst case for node addition. For node deletion additional E space is required. Then FPPR takes $\mathcal{O}(|V + E|)$ additional space. The experiments are performed over several synthetic and real-world networks considering different dynamic behavior of the network. Random graph generators from the networkX library are used specifically to test FPPR for link addition and link deletion. The experimental results show that the ranking of the proposed method is highly correlated with the benchmark Power Iteration based recalculation and better than Fast Incremental PageRank(FIPR) [25]. The FPPR is also tested simulating a real-world graph growth and decay, for which the proposed method has better performance than FIPR [25]. In summary major contribution of the paper are

1. We propose FPPR algorithm for dynamic network which takes less computation and space. The proposed algorithm can estimate the PageRank for both node/link addition and deletion.
2. We experimentally show that the updated page ranks are highly correlated with those of Power Iteration (PI) method.
3. We showed with experiments that the proposed algorithm works with evolving networks for both growing and decaying cases.

The paper is organized as: Section 2 describes the graph model, Section 3 provides a brief literature review, the proposed FPPR method and its rationale are presented in Section 4, experiments performed and corresponding results are reported in Section 5. Finally, Section 6 describes the conclusions of the research work.

2 The Model

In the paper a directed network is represented with graph $G(V, E)$ where V is set of nodes in the network and $E = V \times V$ is the set of edges. The graph is

directed i.e., $e(u, v) \neq e(v, u)$. Also, we assume that there is no self loop in the graph.

Symbols used throughout the paper is provided in Table 1 for the reader's reference.

Table 1 Symbol Table

Symbol	Remarks
G	graph
V	set of vertices
n	number of nodes
E	set of edges
u, v	nodes
A	HITS athority score
H	HITS hub score
$\Gamma_{in}(u)$	inbound neighbours of u
$\Gamma_{out}(u)$	outbound neighbours of u
ρ	probability of random surfer to restart a new walk
P	transition probability matrix
π	PageRank vector
π_u	PageRank of node u
δ	convergence threshold of PI method
c	$1 - \rho$
$d_{out}(u)$	outdegree of u i.e. $\lvert\Gamma_{out}(u)\rvert$
d^k_{avg}	average outdegree of k nodes
$AV(u)$	approximate visits of u
$AV(u, v)$	ontribution of score of node u to v
$edgeWT(u, v)$	edge weight between u, v
ρ_{n+}	probability of adding node
ρ_{l+}	probability of adding link
ρ_{n-}	probability of deleting node
ρ_{l-}	probability of deleting link
d_{avg}	average degree of the graph
V_s	the set of nodes that are updated
E_s	the set of edges concerning k_s
R	the number of random walk simulations

3 Related Work

Since the inception of WWW in 1970s, the size of internet is increasing in rapid scale. There is a need for ranking of pages to find relevant information from it. This led to the invention of search engines to make web search possible for any given user query. With the increase in size rank of the pages need updates. The early search engines use algorithms like HITS [13] and PageRank [18]. Over the years there has been many methods to get PageRanks, which can be divided into (i) Classical PageRank, (ii) Static Monte Carlo based methods and (iii) PageRank for dynamic networks.

3.1 HITS (Hypertext Induced Topic Search)

The web is considered to have hubs, pages with good links and Authorities. HITS [13] uses the authority score $A(x)$ and the hub score $H(x)$. The procedure initiates with $A(x) = 1$ and $H(x) = 1$ for each node x in the graph. Then the repeated updates take place as follows.

for all $x \in E$ **do**
  $A(x) \leftarrow \sum_{u \in \Gamma_{in}(x)} H(u)$
end for
for all $x \in E$ **do**
  $H(x) \leftarrow \sum_{u \in \Gamma_{out}(x)} A(u)$
end for

where $\Gamma_{in}(x)$ and $\Gamma_{out}(x)$ are the inbound neighbours and outbound neighbours of x respectively. The process continues until convergence. HITS penalizes the hub, which points to poor authorities resulting in biased rankings.

Many variations of the HITS has been proposed in the literature. In the G-HITS [26] algorithm the random surfer is used to evaluate the HITS score. The random surfer alternately follows links in the forward and backward directions and occasionally restarts the walk at a randomly chosen node. Addresing the problem that HITS algorithm ignores textual content but focuses itself only on the structure of the web, I-HITS [27] is introduced. It assigns different weights to edges according to the query-induced similarity and popularity of web pages.

3.2 PageRank

PageRank [18] defines the importance of web pages based on the link structure of the web. PageRank is inspired by the eigenvector centrality measure. The PageRank of any node u is calculated based on the set of nodes ($\Gamma_{in}(u)$) that point to u (backward links) and the set of nodes ($\Gamma_{out}(u)$) that u points to (forward links). Then the ranking of any node u (π_u) is given by, $\pi_u = \sum_{b \in \Gamma_{out}(u)} \frac{\pi_b}{|\Gamma_{out}(b)|}$. The PageRank for nodes with no hyperlinks is given by $\frac{1}{\text{Total number of nodes}}$.

In simple terms, it uses a random surfer model where the random surfer clicks out-links with probability $1 - \rho$ and terminates its walk to start a new walk at a random page with probability ρ. The PageRank transistion matrix (P^*) is as follows

$$P^* = (1 - \rho)P\pi + \rho\frac{1}{n}I \tag{1}$$

where I is unit square matrix, n is number of nodes in the network and P is transition probability matrix in which each entry is $\frac{1}{d_{out}(u)}$ when $(u, v) \in E$ and 0 otherwise. Here $d_{out}(u) = |\Gamma_{out}(u)|$ is the out degree of u. The simple psuedo code for calculating PageRank vector (π) is described below.

while $\delta > \varepsilon$ **do**
  $\pi^{(i+1)} = \pi^{(i)} P^*$
  $\varepsilon = |\pi^{(i+1)} - \pi^{(i)}|$

end while

The δ is the convergence threshold provided by the user and $(i+1)$ is the current iteration. The converged $\pi^{(i+1)}$ is final PageRank vector.

3.3 Static Monte Carlo based PageRank Algorithms

The Monte-Carlo methods are used to approximate clasical PageRank [18]. In [21], it is stated that PageRank is nothing but a finite-state Markov Chain and there exists an eigenvalue 1. Hence $\pi = P^*\pi$ and π is the final rank vector. Replacing P^* with (Eq.1) gives the following equation. It is interpreted in [28] as the distribution of all the random walks ending at each node.

$$\pi = \frac{(1-\rho)}{n}[1][I-(\rho)P]^{-1} \tag{2}$$

There are many variations evolved [2, 4, 21, 23, 28] from the above formulation for approximating PageRank (π). All these methods majorly follow four different strategies.

1. **Monte Carlo end-point with a random start**: In this type of methods, simulation of N number of random walks initiated at any random node in the graph. The final PageRank of any node u is calculated as total number of random walk terminating at u, divided by the total random walks. Final rank of u is defined as $\pi_u = \frac{[\text{random walk termination at u}]}{\text{N}}$.
2. **Monte Carlo end-point with cyclic start**: Here, simulation of N number of random walks initiated at every node u in the graph with equal number of simulations m. The final PageRank of any node u is defined as $\pi_u = \frac{[\text{terminations at u}]}{N \times m}$.
3. **Monte Carlo complete path**: These methods involve simulation of N number of random walks initiated at every node in the graph with equal number of simulations m. The PageRank of any node u is defined as $\pi_u = \frac{[visits_u]}{\Sigma_{u=1}^{n} visits}$, where n is total number of nodes.
4. **Monte Carlo complete path stopping at the dangling nodes**: It is similar to the Monte Carlo complete path but the random walk stops at the dangling node, a node with no outlinks. In this method, R number of random walks are simulated starting from each node, and the random walk terminates at dangling nodes. The PageRank of any node u is defined as $\pi_u = \frac{[visits_u]}{\Sigma_{u=1}^{n} visits}$, where n is total number of nodes.

It is shown in [23] that out of all the mentioned methods, the last strategy shows better performance in terms of execution time.

3.4 PageRank for Dynamic Networks

Dynamic networks are those networks where nodes, links get added or deleted dynamically in real time. This is more practical in today's web 2.0 applications. Many algorithms for the dynamic network were proposed in the literature. These are broadly classified into two categories:

1. **Aggregation algorithms** [5, 7, 15]: In this type of method, the algorithm carefully finds the subset of the graph in the vicinity of the updated node or edge, and other parts of the graph are assumed to be supernodes. This gives the smaller graph and then computes the PageRank using static methods. The disadvantages of this approach include the accuracy, approximation error and slower execution time. Accuracy in this type of algorithm purely depends on the choice of the subset chosen [3]. The approximation error can also accumulate over time. It involves high aggregation computation resulting in a slower execution time.
2. **Monte Carlo based algorithms** [17, 25]: On the other hand, Monte Carlo based approaches use the theory of Markov Chains [10]. In this method, the algorithm needs to store all the random walks made along with the visits that each random walk contributes to each node. The random walk segment is adjusted only if its path has an updated node or edge [17, 25]. Monte Carlo methods have two major issues. First, high space usage as the graph evolves. Along with graph evolution, random walks that need to be re-initiated also get lengthy. Second, the random walk segment to be removed and the simulated new random walk segment follows the same distribution but are actually different segments. Hence it brings errors.

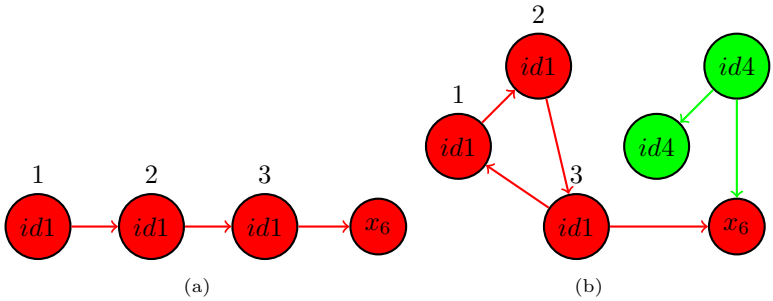


Fig. 1 Examples

4 Proposed Fast Pessimistic dynamic PageRank for Evolving Directed Graphs (FPPR)

In this section, we present the FPPR algorithm, which is capable of recalculating the PageRanks of a directed network upon the topological changes in the network.

4.1 FPPR algorithm for Node addition [19] and Link addition

When a new node is added, FPPR calculates the expected score of random surfers considering the static graph, i.e., if the Monte Carlo method is executed on the graph after the changes in the network, what would be the expected score generated by the random walks. We calculate this score in two phases. We will explain this process through examples shown in Fig. 1. Let 'x6' be a newly added node. The expected value of (considering $\rho = 0.2$) of random walks through outlink reaching 'x6' considering a static Monte Carlo simulation is 80%. This is what we would like to calculate in the first phase. We call this score as Approximate Visits defined in Definition 1. We assume the incoming link is linear in the second phase. Accordingly, we defined Link Sensitivity Index in Definition 2. This linearity assumption is taken as a pessimistic way. For example, in Fig. 1(a), R random walks starting from the node (1) holding chain (all the connected nodes with same ID) ID 'id1' would have reached 'x6' in 3 hops, i.e., the expected number of visits is $R * (0.8)^3$. Similarly, the next node (2) must have gone 'x6' in 2 hops (the expected visit is $R*(0.8)^2$), and so on. All these contribute to the score of the new node. This linearity assumption will provide correct expectation value for chain 'id1' in Fig. 1(b) even though it is a loop in the network. Note that the same formulation may not be valid for chain 'id4' in Fig. 1(b). However, the algorithm thinks pessimistic that either if the chain is linearly contributing or it is a loop. In such a case, the algorithm is prone to error. For example in Fig. 1(b), chain 'id4' doesnt have a loop but still the algorithms assumes the chain is linear/has a loop and accordingly calculates the PageRank scores of 'x6'. However, we expect this error will be very small. A similar kind of process is adopted in addressing the outlink from the new node as well.

A new link addition in a network would only modify the score of the target node. When a new link is added between source and target, then the above discussed procedure is followed on the target node to estimate the PageRank of the target node.

Definition 1 (Approximate Visits (AV))**.** *AV estimates the contribution of the incoming links to the newly added node that a random walk might have used if the random walk is executed from the scratch. If nodes have bidirectional edges, a max of approximate visits is considered for both nodes. It is calculated as:*

$$AV(v) \leftarrow \sum_{u \in \Gamma_{in}(v)} \frac{1}{d_{out}(u)} * (1-\rho) * (AV(u)) \tag{3}$$

Here, $\Gamma_{in}(.)$ *and* $d_{out}(.)$ *return the set of incoming neighbors and out degree respectively. AV(u,v) is the contribution of score of node u to v defined as:*

$$AV(u,v) \leftarrow \frac{1}{d_{out}(u)} * (1-\rho) * (AV(u)) \tag{4}$$

Definition 2 (Link Sensitivity Index (LSI))**.** *LSI defines the extent one node affects the whole link chain it belongs. We assume the link chain is linear and the LSI is mathematically defined by:*

$$LSI \leftarrow c + c(c)^1 + c(c)^2 + c(c)^3 + ... = \frac{c(1-(c)^n)}{(1-c)} \tag{5}$$

Here, $c = 1 - \rho$ is the probability that a random surfer moves forward and n is the length of the chain. In our experiment we took $c = 0.8$.

FPPR does not keep track of all the random walk segments or aggregations. FPPR takes only $4 \times |V|$ space for node/link addition. That is, space complexity is $\mathcal{O}(|V|)$. For a network with billions of nodes, this provides a significant improvement.

The Algorithm 1 & 2 shows the steps of FPPR algorithm for node and link addition. The $linkID$ of a node is the ID given to a node to identify the chain to which it belongs. In other words, $linkID$ keeps track of all links or distinct chains in the graph. One may note that a lower $linkID$ is assigned to a node if multiple chains go through that node. This convention will help resolving the conflicts. As we are keeping track of every link(chain) in the graph, we store the length of each $linkID$.

4.1.1 FPPR algorithm for Node deletion

When a node is deleted, FPPR opts for a Selective Breadth First Update (SBFU) approach (Algorithm 3) from the deleted node. As part of node deletion, the score contribution from the source node to the target node is used to update the changes in the target node to approximate PageRank. These values can be stored as edge weight parameter during graph evolution itself. In Algorithm 1 Lines 8, 24 stores $AV(u, v)$ as edge weight parameter. The $AV(u, v)$ is removed from the target node (v) in the first level of SBFU, and from the next level, the edge weight $(AV(u, v))$ and PageRank score of the nodes are updated accordingly. Note that the algorithm SBFU puts only selected nodes, nodes for which incoming edge weights have been updated, into the queue.

4.1.2 FPPR algorithm for Link deletion

For link deletion, the simple local update is used in FPPR. The $AV(u, v)$ value is removed from the target node. This local update causes approximation error but doesn't seem significant in the experimental results.

5 Experiments and Results

5.1 Dataset

Different Experiments for comparing the performance of proposed FPPR have been conducted on synthetically generated networks as well as real-world networks. These experiments consider node/link addition/deletion. Both the Random Network and Barabasi-Albert [1] networks are used for testing FPPR

Algorithm 1 Calculate the approximate PageRank of the newly created node

Input: new node u, $R = 1000$
for all $v \in \Gamma_{in}(u)$ **do**
    Assign linkID to the node as described in text
    $linkID[v].length \leftarrow linkID[v].length + 1$
    $temporary_Var \leftarrow R \times LSI(n = linkID[v].length)$ comment: LSI calculated by Eqn. 5
    $linkSensitivityIndex \leftarrow \max(linkSensitivityIndex, temporary_Var)$
    $approxVisits \leftarrow AV(v)$ comment: Using Eqn. 3
    $edgeWT(v, u) \leftarrow AV(v, u)$ comment: Using Eqn. 4
end for
$approxVisits \leftarrow approxVisits + R + linkSensitivityIndex$
$finalResult[u] \leftarrow approxVisits$, $totalVisits \leftarrow totalVisits + approxVisits$
$approxVisits, linkSensitivityIndex \leftarrow 0$
for all $v \in \Gamma_{out}(u)$ **do**
    if u has bidirectional edge with v **then**
        replace both nodes with maximum visits
    else
        Assign link ID to the node as described in text
        $linkID[u].length \leftarrow linkID[u].length + 1$
        $temporary_Var \leftarrow R \times LSI(n = linkID[u].length)$ comment: LSI calculated by Eqn. 5
        $approxVisits \leftarrow AV(u)$ comment: Using Eqn. 3
        $finalResult[v] \leftarrow approxVisits + linkSensitivityIndex$
        $totalVisits \leftarrow totalViists + approxVisits + linkSensitivityIndex$
    end if
    $edgeWT(u, v) \leftarrow AV(u, v)$ comment: Using Eqn. 4
end for
Output: $finalResult \div totalVisits$

Algorithm 2 Calculate the approximate PageRank when new link is added

Input: new link (source u, target v), $R = 1000$
Assign linkID to the target node as described in text
$linkID[v].length \leftarrow linkID[v].length + 1$
$temporary_Var \leftarrow R \times LSI(n = linkID[u].length)$ comment: LSI calculated by Eqn. 5
$approxVisits \leftarrow AV(u)$ comment: Using Eqn. 3
$edgeWT(u, v) \leftarrow AV(u, v)$ comment: Using Eqn. 4
$finalResult[v] \leftarrow approxVisits + linkSensitivityIndex$
$totalVisits \leftarrow totalViists + approxVisits + linkSensitivityIndex$
Output: $finalResult \div totalVisits$

Algorithm 3 Selective Breadth First Update for node deletion

Input: $visited, queue, delNode, finalResult, totalVisits$
$visited.append(delNode)$
$queue.append(delNode)$
while $queue$ **do**
    update the linkID for only delNode with any of the incoming node's linkID
    $m = queue.pop()$
    for all $neighbour \in \Gamma_{out}(m)$ **do**
        if $neighbour$ is not in $visited$ **then**
            $linkID[neighbour] = linkID[m]$
            update the $linkID[neighbour].length$ and $linkID[delNode].length$
            $previousEdgeWt \leftarrow edgeWt(m, neighbour)$
            $currentEdgeWt \leftarrow AV(m, neighbour)$
            $changedEdgeWt \leftarrow previousEdgeWt - currentEdgeWt$
            if $changedEdgeWt > 0$ **then**
                $totalVisits \leftarrow totalVisists - changedEdgeWt$
                $finalResult[neighbour] \leftarrow finalResult[neighbour] - changedEdgeWt$
                $edgeWt(m, neighbour) \leftarrow currentEdgeWt$
                $queue.append(neighbour)$
            end if
        end if
        $visited.append(neighbour)$
    end for
end while
Output: $totalVisits, finalResult$

Algorithm 4 FPPR for link deletion

Input: $source, target$
$weight \leftarrow edgeWt(source, target)$
$totalVisits \leftarrow totalVisits - weight$
$finalResul[target] \leftarrow finalResult[target] - weight$
update the linkID of the target as discribed in the text
Output: $totalVisits, finalResult$

for node addition and deletion. For node deletion, upto 50 percent of the nodes chosen randomly are deleted. The random graph generators in Networkx library like Erdos-Renyi [8], GNM graphs [14] are used for experimenting FPPR for link addition and deletion. For link deletion, upto 10 percent of the randomly chosen links are deleted. The real-world networks are Wiebo reshare

network of [6] and reptilia-tortoise network, aves-weaver network, mammalia-voles network from the NetworkRepository [20]. The salient features of these graphs are presented in Table 2.

As the Erdos-Renyi random graph does not support growth, we use Algorithm 5 to generate the Random network. In order to get the dynamic character, we started with one node and added each node according to their creation for Barabasi-Albert and Random networks.

Table 2 Dataset used for different experiments

Experiment for Node Addition						
Name	$\|V\|$	$\|E\|$	Min $\|\Gamma_{in}\|$	Max $\|\Gamma_{in}\|$	Min $\|\Gamma_{out}\|$	Max $\|\Gamma_{out}\|$
Weibo1 [6]	40	66	0	12	0	6
Weibo2 [6]	206	206	0	45	0	2
Weibo3 [6]	759	780	0	201	0	3
Weibo4 [6]	817	901	0	28	0	297
Random51	51	93	0	5	0	7
Random126	126	233	0	5	0	8
Random201	201	386	0	7	0	9
Random276	276	543	0	10	0	7
BA1 [1]	55	154	0	35	0	3
BA2 [1]	130	376	0	61	0	3
BA3 [1]	205	604	0	101	0	3
BA4 [1]	280	829	0	135	0	3
Experiment for Node Deletion						
Name	$\|V\|$	$\|E\|$	Min $\|\Gamma_{in}\|$	Max $\|\Gamma_{in}\|$	Min $\|\Gamma_{out}\|$	Max $\|\Gamma_{out}\|$
Randdel51	51	582	0	42	0	26
Randdel201	201	326	0	7	0	9
Randdel651	651	950	0	7	0	12
Randdel1001	1001	1490	0	7	0	10
BAd1 [1]	55	154	0	40	0	3
BAd2 [1]	205	604	0	95	0	3
BAd3 [1]	505	1504	0	229	0	3
BAd4 [1]	1005	3004	0	338	0	3
Experiment for Link Addition & Link Deletion						
Name	$\|V\|$	$\|E\|$	Min $\|\Gamma_{in}\|$	Max $\|\Gamma_{in}\|$	Min $\|\Gamma_{out}\|$	Max $\|\Gamma_{out}\|$
ER1 [8]	100	2034	11	29	11	29
ER2 [8]	200	5987	16	45	0	53
ER3 [8]	300	11209	26	56	0	79
ER4 [8]	500	13921	16	42	0	120
GNM1 [14]	150	100	0	3	0	3
GNM2 [14]	150	1000	2	14	1	16
GNM3 [14]	500	200	0	3	0	4
GNM4 [14]	500	1729	0	10	0	10
Experiment for Real World Simulation						
Name	$\|V\|$	$\|E\|$	Min $\|\Gamma_{in}\|$	Max $\|\Gamma_{in}\|$	Min $\|\Gamma_{out}\|$	Max $\|\Gamma_{out}\|$
ReW1 [20]	445	1357	0	21	0	17
ReW2 [20]	1218	3697	0	25	0	24
ReW3 [20]	136	375	0	16	0	16

Algorithm 5 Random graph generator

```
Input: number of nodes N comment: single node '0' exist in the graph
for all x ∈ Γ_in(1, N) do
    add x to Graph G
    indegree ← random[0, 1]
    if indegree then
        G.addEdge(random[0, x − 1], x)
    end if
    outdegree ← random[0, x − 1]
    shuffle.list[0, ..x − 1]
    while outdegree do
        G.addEdge(x, list[outdegree])
        outdegree−−
    end while
end for
```

5.2 Comparing Methods

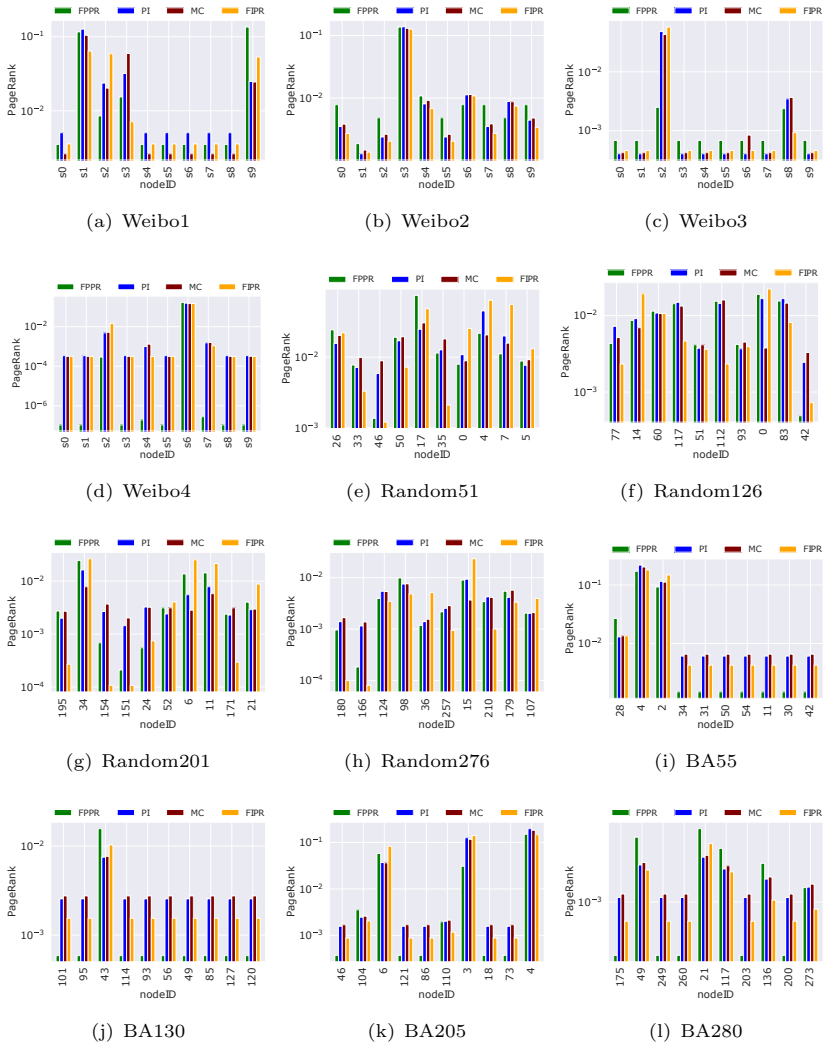
The proposed methods have been compared with the following methods.

- **PowerIteration (PI)** [18]: We consider this method as a benchmark in our experiments. In the experiment, we restarted PI algorithm and recalculated PageRank for the whole graph on appropriate (depending upon experiments) node/link addition or deletion.
- **Static Monte Carlo (MC)** [23]: Static Monte-Carlo method is the method of approximate PageRanks of the network using Monte Carlo method. We implemented the version of the complete path with dangling nodes. Similar to PI, we recalculate the PageRanks on network changes in appropriate steps. The number of random walk is considered in the experiment is 1000.
- **Fast Incremental PageRank on Dynamic Networks (FIPR)** [25]: The method is designed for dynamic networks and proposed in 2019. As our algorithm is designed for dynamic networks, we included this method as a related research. Parameter R is set to 16 in the experiments.

5.3 Comparing Parameters

All comparing algorithms were executed on different graphs. As expected, the absolute values of PageRanks by different algorithms of a node are different. The results for 10 random nodes for all the datasets with respect to node addition are shown in Fig. 2 for reference. Hence, comparing different algorithms in terms of absolute values of the PageRank is not fare. Therefore the Spearman's rank correlation coefficient [22] is used to compare the ranking of the proposed FPPR with that of the comparing methods. The Spearman correlation between two vectors will be high when observations have a similar rank between the two variables and it will be lower otherwise. A value of it in between 0.8 to 1

is considered to be strongly correlated [24]. Apart from Spearman Rank correlation coefficient, the comparing parameters include (i) Change in Spearman Rank correlation coefficient over time, (ii) Execution time.



(a) Weibo1 (b) Weibo2 (c) Weibo3

(d) Weibo4 (e) Random51 (f) Random126

(g) Random201 (h) Random276 (i) BA55

(j) BA130 (k) BA205 (l) BA280

Fig. 2 PageRank for Randomly Sampled 10 nodes of different algorithms.

5.4 Results

5.4.1 FPPR for node addition

Accuracy: The results of Spearman's ranking coefficient for all nodes in the network for node addition are shown in the Fig. 3. It is evident from the result that the proposed FPPR is highly correlated with the benchmark PI method. Median and mean Spearman's correlation with PI method for all the experiments performed are 0.96 and 0.94 respectively. Further the proposed method is equal or better than FIPR for all the network except Weibo2. Note that MC and PI are highly correlated as both are recalculated over the full graph once a new node is added to the graph.

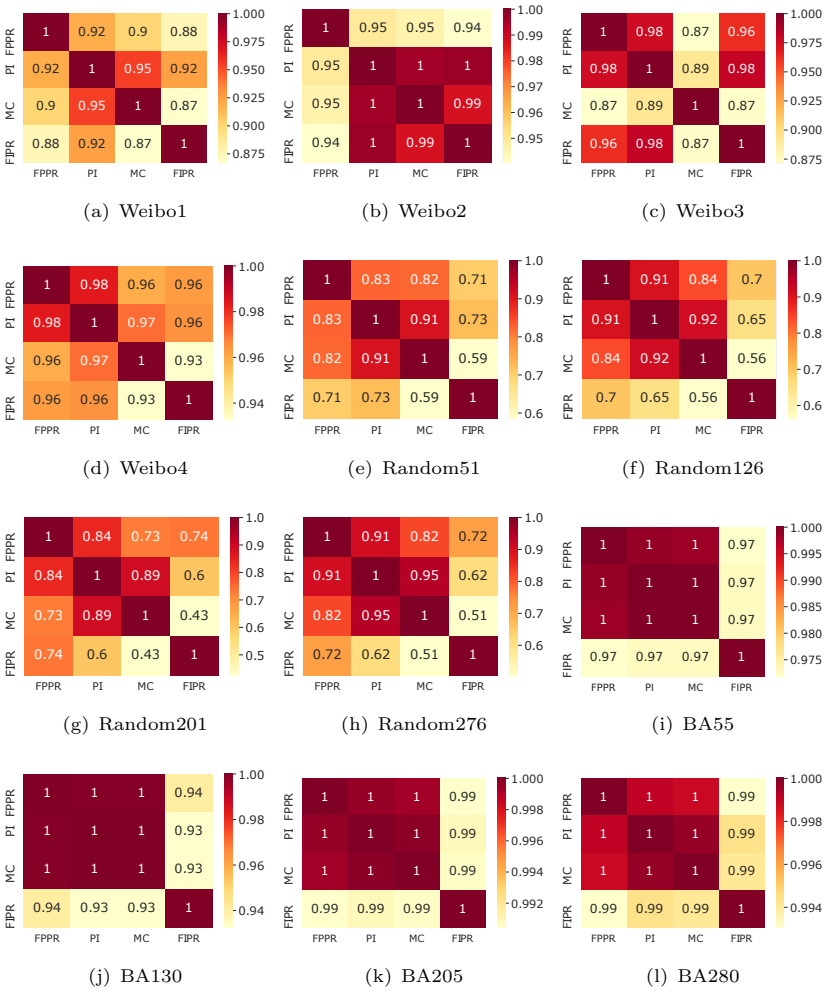
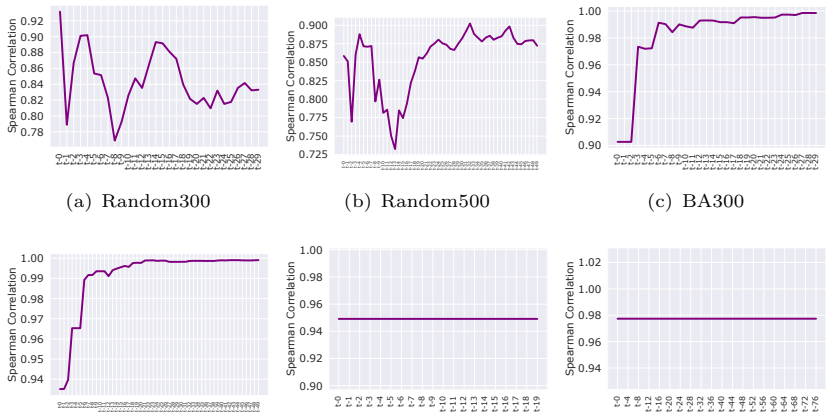


Fig. 3 Spearman Correlation between all four approaches over the different data sets for node addition

Spearman's rank correlation with changes in network: As part of experiment, we would like to see how the Spearman's correlation coefficient changes with addition of nodes. We recalculate Spearman's correlation coefficient of the proposed algorithm against the benchmark Power Iteration for the addition of each 10 nodes. The result is plotted in Fig. 4 for 6 data sets. The value dipped around 26% for both random networks, while the Barabasi-Albert network shows consistent improvement in the value of Spearman's correlation coefficient. For both real-world networks, the values are constant because, in general, real-world retweet networks have few influential nodes (hubs) responsible for the tweet's diffusion resulting in high page rank scores throughout the evolution of the graph.

Execution Time: We checked the overall execution time of different comparing algorithms and found that the proposed algorithm is much faster than all the methods we experimented with for all data sets with respect to node addition. The comparison is presented in the Fig. 5.



(a) Random300 (b) Random500 (c) BA300

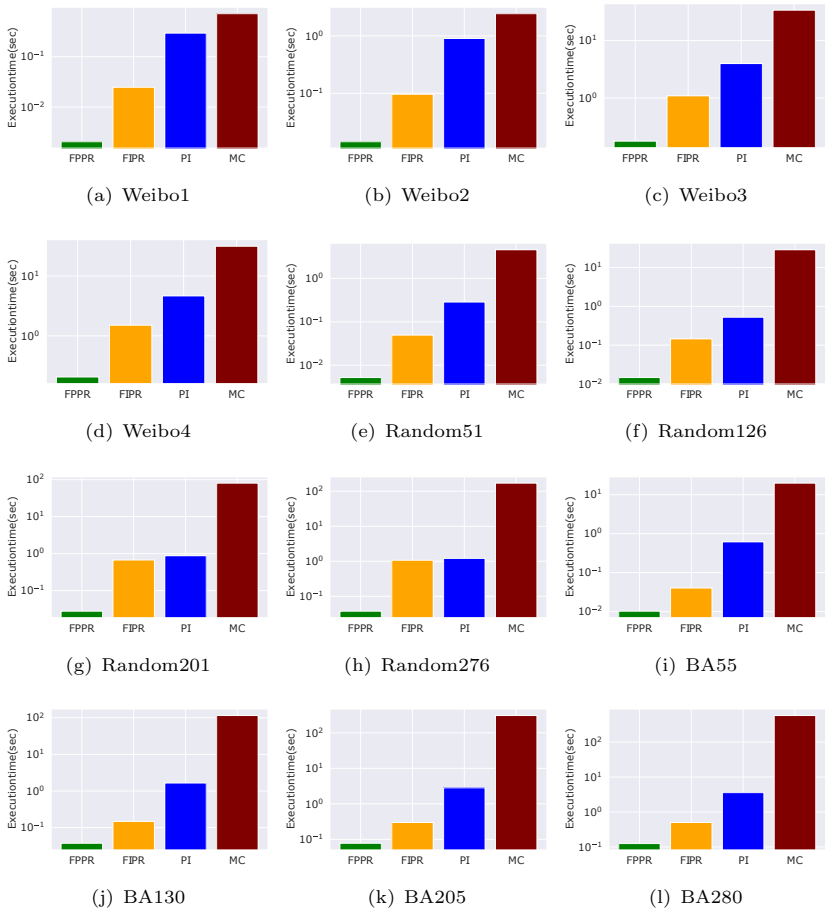
(d) BA500 (e) Weibo2 (f) Weibo4

Fig. 4 Spearman Correlation of FPPR VS PI over time. Each time tick denotes addition of 10 nodes in the network.

5.4.2 FPPR for node deletion

Accuracy: For testing FPPR deletion accuracy, we deleted 50 percent of the randomly chosen nodes from the network and plotted the heatmap of Spearman Rank correlation in Fig. 6 similar to node addition. It has been observed that FPPR is performing well with respect to benchmark PI method. Median and mean Spearman's correlation with PI method for all the experiments performed for node deletion are 0.89 and 0.88 respectively. FPPR is also observed to be equal or better than FIPR.

Spearman's rank correlation with changes in network: The Spearman rank correlation coefficient is recorded after regular intervals and plotted in Fig. 7. It is transparent that there is a gradual dip in the Spearman Rank correlation. The reason for the downhill is that for deletion, FPPR uses Selective Breadth First Update (SBFU). In SBFU, the breadth first traversal is used, marking visited nodes. In such a case, one node is visited only once, i.e., only one update is possible on each neighboring node. For the scenario shown in the Fig.8(a), considering the deletion of the node x, there is a need to update the score of the neighbors of x more than once because the scores of $n1, n2, n3$ are updated accordingly as x deleted. Still, there is a link between $n2, n3$ and $n3, n1$, which must be crawled. This needs further update of score in node $n3$ and $n1$ as shown in Fig.8(b). SBFU cannot update the neighboring nodes more than once, accumulating errors over time.



(a) Weibo1 (b) Weibo2 (c) Weibo3

(d) Weibo4 (e) Random51 (f) Random126

(g) Random201 (h) Random276 (i) BA55

(j) BA130 (k) BA205 (l) BA280

Fig. 5 Comparing plots of execution time of different algorithms for node addition

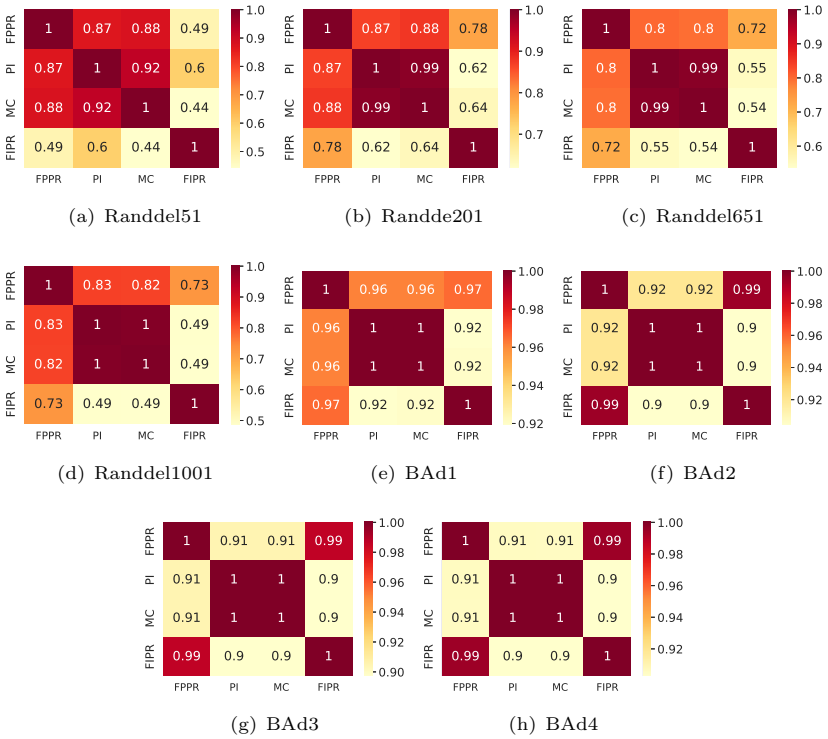


Fig. 6 Spearman Correlation between all four approaches over the different data sets after 50% node deletion

Execution Time: The execution time for all the comparing methods with respect to node deletion is plotted in Fig.9. The FPPR takes a little higher or equal execution time as FPPR's deletion time depends on the set of updated nodes along with its associated edges.

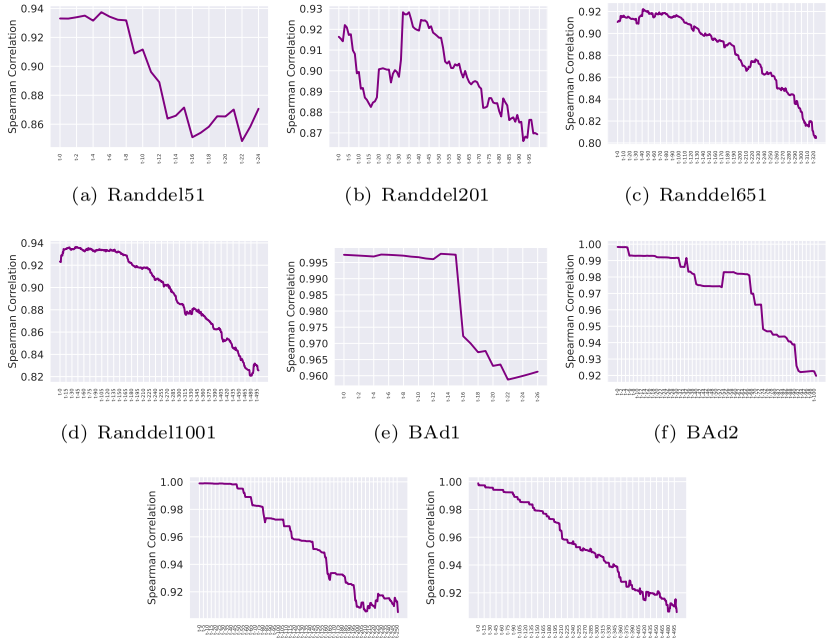
5.4.3 FPPR for link addition and deletion

Accuracy: The accuracy for the link addition is shown in the Fig. 10 ((a) to (h)). Median and mean Spearman's correlation with PI method for all the experiments performed are 0.97 and 0.94 for link addition, 0.95 and 0.91 for link deletion, respectively. This shows that the proposed FPPR is highly correlated with the benchmark PI method. The proposed FPPR is also performing reasonably good in comparison with FIPR except for GNM2, GNM3 graphs.

As a part of the experiment, 10 percent of the randomly chosen links are deleted from the graph, and PageRank scores are computed with respect to all comparing algorithms. The accuracy for link deletion is depicted in the Fig. 10 ((i) to (p)). When coming to the deletion, FPPR performs better than all

the comparing methods. Note that all the graphs used for the link addition and deletion experiment Monte Carlo method (MC) had poor performance as the graphs generated are highly dense.

Spearman's rank correlation with changes in network: The change in the spearman rank correlation is recorded in regular intervals (after addtion or deletion of every 10 links) is shown in the Fig. 11 ((a) to (h)) & ((i) to (p)). For link addition, the FPPR spearman rank correlation with PI has minute fluctuations and gradually decreases. For link deletion, there are sharp transitions in the graph. This is due to the local update of the FPPR for link deletion.

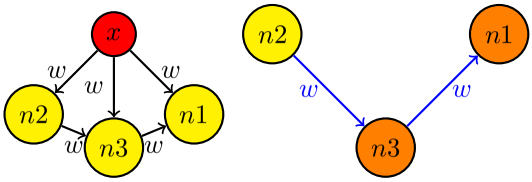


(a) Randdel51 (b) Randdel201 (c) Randdel651

(d) Randdel1001 (e) BAd1 (f) BAd2

(g) BAd3 (h) BAd4

Fig. 7 Spearman Correlation of FPPR VS PI over time. Each time tick denotes deletion of 10 nodes in the network.



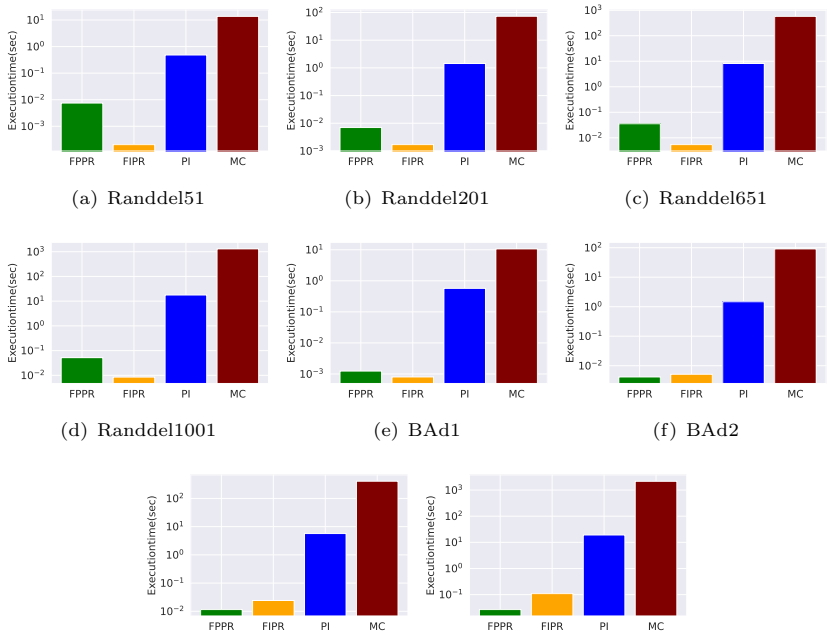
(a) (b)

Fig. 8 Examples

Execution Time: The execution time of the proposed FPPR is much faster than all the comparing methods for link addition and deletion. For link addition, the graphs ER2, ER3, ER4, GNM2 have greater than or equal FIPR execution times when compared to the benchmark PI method.

5.5 Computation Complexity

Computation complexities for all the operations, node addition, node deletion, link addition, link deletion of different algorithms along with the space complexity are shown in Table 3. Thus, the worst-case complexity of the proposed algorithm for node and link addition is $\mathcal{O}(k \times d_{avg}^{k})$, where k is the number of nodes/links added to the network. Whereas the worst-case complexity of the proposed algorithm for node deletion, link deletion is $\mathcal{O}(|V_s| + |E_s|)$ and $\mathcal{O}(1)$ respectively, where V_s is the set of nodes that are updated, E_s is the edges associated with those updated edges.



(a) Randdel51 (b) Randdel201 (c) Randdel651

(d) Randdel1001 (e) BAd1 (f) BAd2

(g) BAd3 (h) BAd4

Fig. 9 Comparing plots of execution time of different algorithms for node deletion

Table 3 Computation complexities and space complexities of different PageRank algorithms.

Algorithm	Addition Time complexity	Deletion Time complexity	Space complexity
Power Iteration	$\Omega\frac{(kn^2)}{1(1-\rho)}$	$\Omega\frac{(kn^2)}{1(1-\rho)}$	$\mathcal{O}(n)$
Monte-Carlo method	$\Omega\frac{(knR)}{(\rho)}$	$\Omega\frac{(knR)}{(\rho)}$	$\mathcal{O}(n)$
FIPR	$O\frac{(knR)}{(\lvert E\rvert)}$	$O\frac{(knR)}{(\lvert E\rvert)}$	$\mathcal{O}(nR)$
Our Method	$\mathcal{O}(k \times d_{avg}^{k})$	$\mathcal{O}(\lvert V_s\rvert + \lvert E_s\rvert)$	$\mathcal{O}(\lvert V\rvert + \lvert E\rvert)$

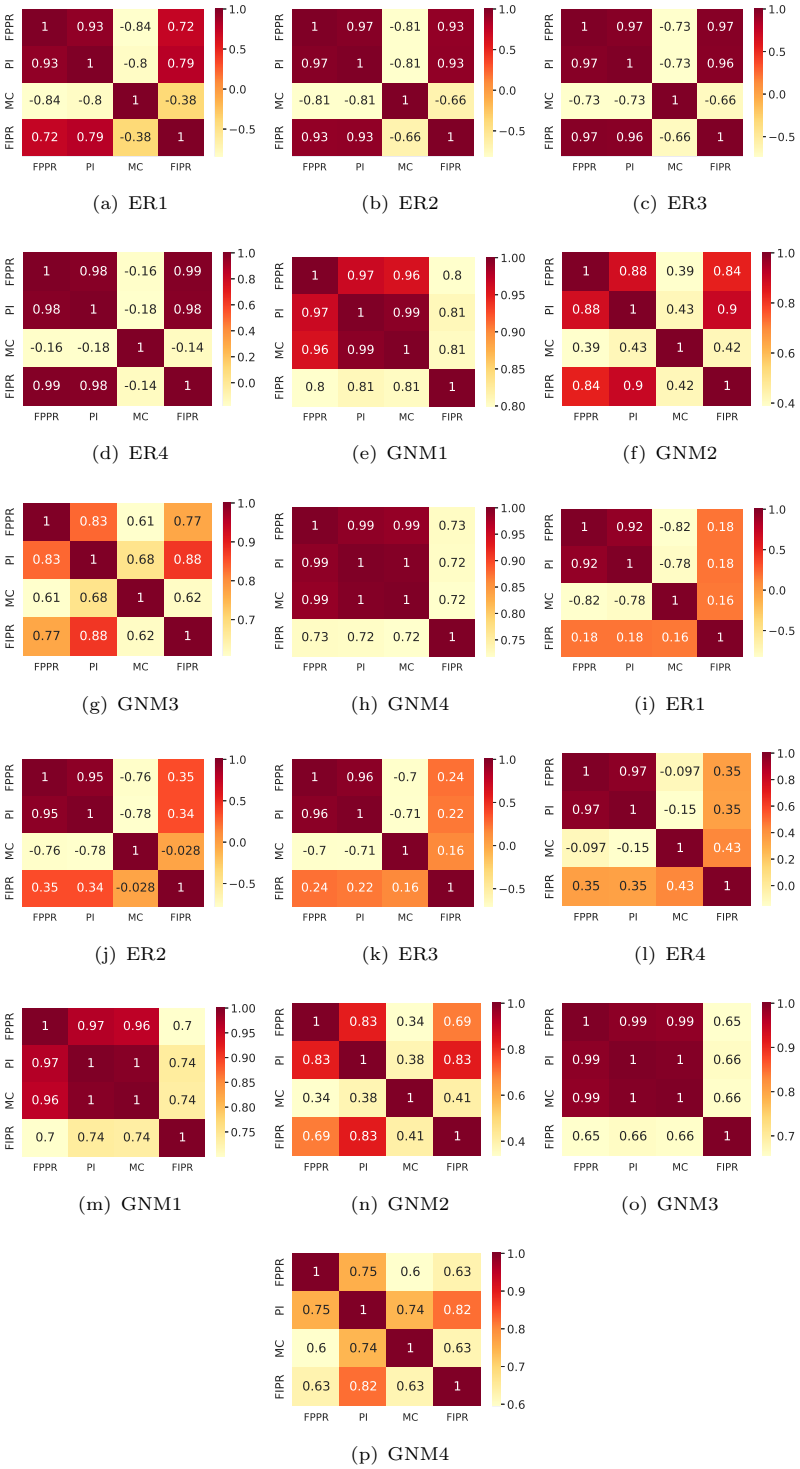
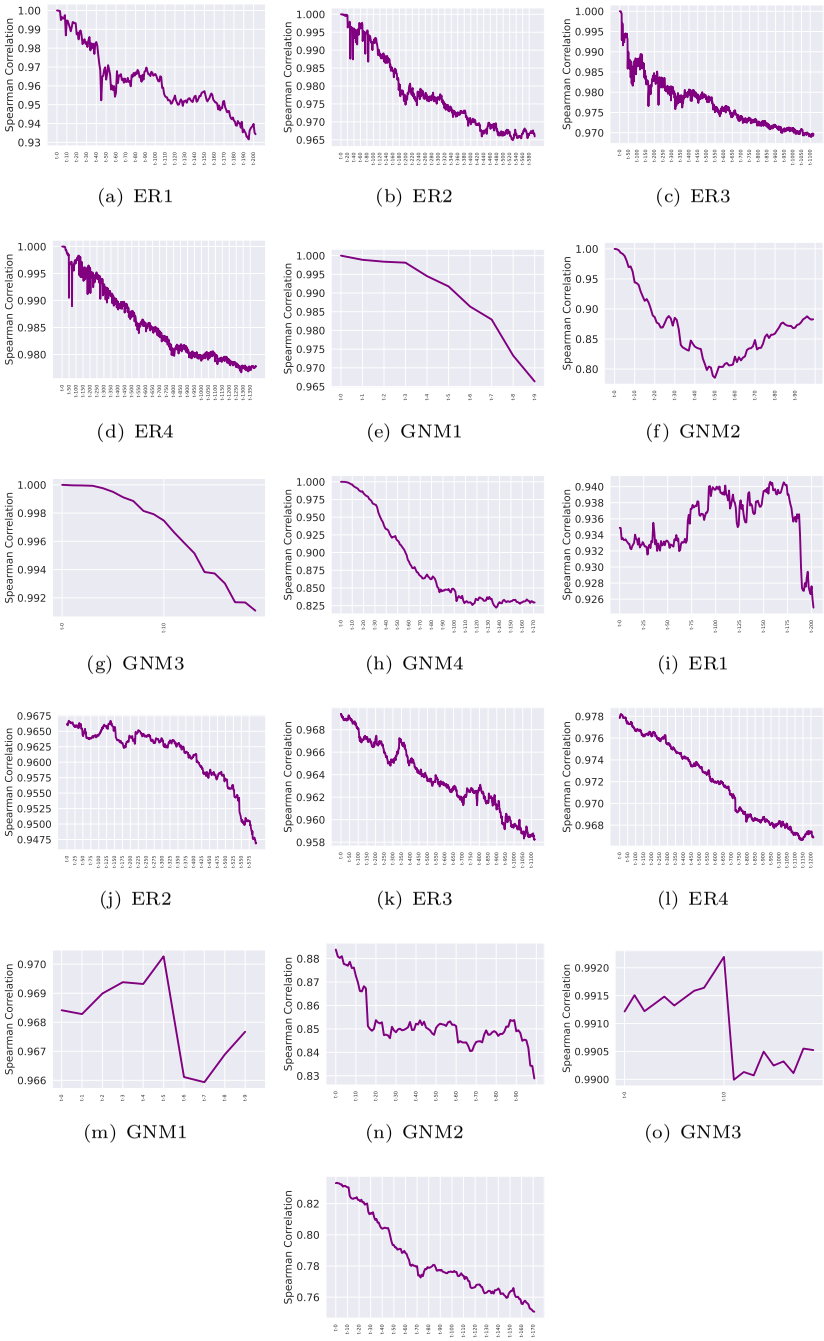


Fig. 10 Spearman Correlation between all four approaches over the different data sets for Link addition and deletion



(a) ER1 (b) ER2 (c) ER3

(d) ER4 (e) GNM1 (f) GNM2

(g) GNM3 (h) GNM4 (i) ER1

(j) ER2 (k) ER3 (l) ER4

(m) GNM1 (n) GNM2 (o) GNM3

(p) GNM4

Fig. 11 Spearman Correlation of FPPR VS PI over time. Each time tick denotes addition of 10 links in the network.

Algorithm 6 Real World graph simulation

```
Input: number of operations K
while K do
    addnodeProb ← random.uniform(0, 1)
    addlinkProb ← random.uniform(0, 1)
    delnodeProb ← random.uniform(0, 1)
    dellinkProb ← random.uniform(0, 1)
    if addnodeProb < ρ_{n+} then
        algo.(1)
        K−−
    end if
    if addlinkProb < ρ_{l+} then
        algo.(1)
        K−−
    end if
    if delnodeProb < ρ_{n−} then
        algo.(3)
        K−−
    end if
    if dellinkProb < ρ_{l−} then
        algo.(4)
        K−−
    end if
end while
```

5.6 FPPR for real world graph simulation (growth and decay)

Eventually, real-world graph simulators (Algorithm 6) are used to test FPPR in both graph decay and graph growth. In this real-world graph simulation, all four operations, node addition, node deletion, link addition, and link deletion, have been incorporated. All the four mentioned operations are performed based on four independent probabilities ($\rho_{n+}, \rho_{l+}, \rho_{n-}, \rho_{l-}$). We tried to capture graph growth and decay by adjusting the independent probabilities. We used them in our experiments to check the accuracy of FPPR concerning other mentioned PageRank algorithms. The setup used for graph growth and decay is as follows. For growing graph, $\rho_{n+} = 0.2, \rho_{l+} = 0.2, \rho_{n-} = 0.01, \rho_{l-} = 0.01$ and for deacaying graph, $\rho_{n+} = 0.1, \rho_{l+} = 0.01, \rho_{n-} = 0.3, \rho_{l-} = 0.2$.

Accuracy: The accuracy test is performed on various real world datasets as shown in Table 2 and also on the graphs generated by real world simulator (Algorithm 6). Both the graph growth and decay are tested for accuracy. For graph growth and decay, the number of operations (node add, node delete, link add, link delete) are set to be 50, 100, 500. For growth, FPPR is in good correlation with respect to benchmark PI method. Median and mean

Spearman's correlation with the PI method for all the real-world graph growth simulation experiments are 0.90 and 0.89. For decay, FPPR is in reasonably good correlation concerning PI method with median and mean values of 0.8 and 0.75. In all the dataset graphs, FPPR performed better than the FIPR. The results are shown in the Fig. 13 ((a) to (f)) for graph evolution, Fig. 13 ((g) to (l)) for graph decay.

Spearman's rank correlation with changes in network: The changes in the spearman rank correlation is captured in regular intervals (after every 10 operations) for both graph growth and decay. The results are shown in the Fig. 14 ((a) to (f)) for node growth, Fig. 14 ((g) to (l)) for link growth and Fig. 15 ((a) to (f)) for node decay, Fig. 15 ((g) to (l)) for link decay. The graph represents time in x-axis,change in Spearman rank coefficient of all the comparing methods with respect to benchmark PI method in y-axis using a bar plot and number of nodes/links in z-axis. In the graph growth phase, it is evident that the proposed FPPR is performing better than or equal with respect to FIPR. FPPR is even performing equally with respect to the static MC method for the graphs RWS1, RWS2, ReW1, ReW3. FPPR had a gradual dip in Spearman rank correlation in the graph decay phase but performed better than or equal with FIPR.

Execution Time: The proposed FPPR has faster execution times concerning for all the comparing methods in both graph growth and decay phases. The results are presented in Fig. 16 ((a) to (f)) for graph growth and in Fig. 16 ((g) to (l)) for graph decay.

6 Conclusion

The present paper proposed a new algorithm for calculating PageRank for dynamic directed graphs. The algorithm estimates the PageRank concerning node addition, link addition, node deletion, link deletion. We showed through experimental results that the results of the proposed algorithm is highly correlated with that of the benchmark Power Iteration method. In particular, the minimum correlation in ranking found for addition of node, deletion of node, addition of link and deletion of link are 0.83, 0.80, 0.83 and 0.75 respectively. The proposed FPPR is also shown to performs better in ranking than the state-of-the-art FIPR algorithm for all different topological changes in the network. The execution time is significantly faster while providing more acceptable results.

While the PageRanks of growing and sinking networks with proposed FPPR method shows comparable and better results against FIPR, the execution time for the proposed algorithm are very less. The proposed algorithm performs better in terms of execution time and accuracy for all the operations except the node deletion case.

References

[1] Réka Albert and Albert-László Barabási. Statistical Mechanics of Complex Networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[2] K Avrachenkov, 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.

[3] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast incremental and personalized PageRank. *Proceedings of the VLDB Endowment*, 4(3):173–184, 2010.

[4] L. A. Breyer. Markovian page ranking distributions: Some theory and simulations. Citeseer, 2002.

[5] Steve Chien, Cynthia Dwork, Ravi Kumar, Daniel R Simon, and D Sivakumar. Link Evolution: Analysis and Algorithms. *Internet Mathematics*, 1:277–304, 2004.

[6] Yuwei Chuai and Jichang Zhao. Anger makes fake news viral online. "https://arxiv.org/abs/2004.10399", 2020.

[7] Prasanna Desikan, Nishith Pathak, Jaideep Srivastava, and Vipin Kumar. Incremental page rank computation on evolving graphs. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, WWW '05, page 1094–1095, New York, NY, USA, 2005. Association for Computing Machinery.

[8] P. Erdös and A. Rényi. *On the evolution of random graphs*, pages 38–82. Princeton University Press, 2011.

[9] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. Wtf: The who to follow service at twitter. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 505–514, New York, NY, USA, 2013. Association for Computing Machinery.

[10] Valerie Isham and E. Seneta. Non-Negative Matrices and Markov Chains. *Journal of the Royal Statistical Society. Series A (General)*, 146(2):202, 1983.

[11] Gábor Iván and Vince Grolmusz. When the web meets the cell: using personalized pagerank for analyzing protein interaction networks. *Bioinformatics*, 27(3):405–407, 2011.

[12] Bin Jiang. Ranking spaces for predicting human movement in an urban environment. *International Journal of Geographical Information Science*,

23(7):823–837, 2009.

[13] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[14] Donald E Knuth. *Art of computer programming, volume 2: Seminumerical algorithms*. Addison-Wesley Professional, 2014.

[15] Amy N Langville and Carl D Meyer. Updating PageRank with iterative aggregation. In *Proc. of Thirteenth International World Wide Web Conference*, pages 1124–1125, New York, 2004.

[16] R Lempel and S Moran. Stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks*, 33(1):387–401, 2000.

[17] Qun Liao, ShuangShuang Jiang, Min Yu, Yulu Yang, and Tao Li. Monte carlo based incremental pagerank on evolving graphs. In Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon, editors, *Advances in Knowledge Discovery and Data Mining*, pages 356–367, Cham, 2017. Springer International Publishing.

[18] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *World Wide Web Internet And Web Information Systems*, 54(1999-66):1–17, 1998.

[19] Rohith Parjanya and Suman Kundu. Fppr: Fast pessimistic pagerank for dynamic directed graphs. In Rosa Maria Benito, Chantal Cherifi, Hocine Cherifi, Esteban Moro, Luis M. Rocha, and Marta Sales-Pardo, editors, *Complex Networks & Their Applications X*, pages 271–281, Cham, 2022. Springer International Publishing.

[20] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 4292–4293. AAAI Press, 2015.

[21] Ozlem Salehi. *PageRank algorithm and Monte Carlo methods in PageRank Computation*. PhD thesis, Bogazici University, 2007.

[22] C Spearman. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, 15(1):72, 1904.

[23] Brian Vargas. *Exploring PageRank Algorithms : Power Iteration & Monte Carlo Methods*. PhD thesis, California State University, San Marcos, 2020.

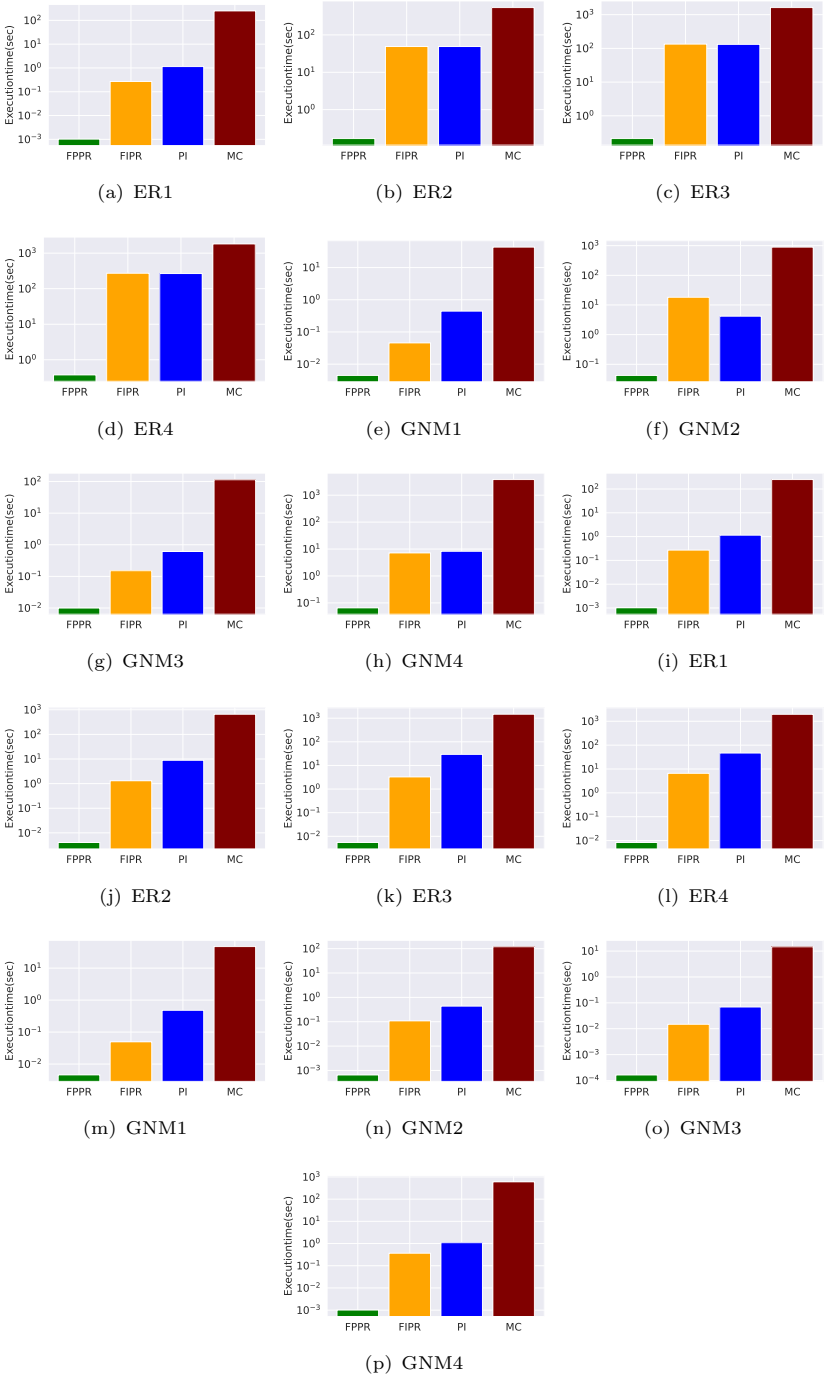
[24] Jerrold H Zar. Spearman Rank Correlation. In *Encyclopedia of Biostatistics*. John Wiley & Sons, Ltd, 2005.

[25] Zexing Zhan, Ruimin Hu, Xiyue Gao, and Nian Huai. Fast incremental pagerank on dynamic networks. In Maxim Bakaev, , Flavius Frasincar, , and In-Young Ko, editors, *Proc. of International Conference on Web Engineering*, volume 11496 LNCS, pages 154–168, Cham, 2019. Springer International Publishing.

[26] Xianchao Zhang, Xinyue Liu, Liguo Zhang, and Hong Yu. G-hits: A link analysis algorithm based on gravitation model. In *The First International Symposium on Data, Privacy, and E-Commerce (ISDPE 2007)*, pages 149–151, ChengDu, China, 2007. IEEE.

[27] Xianchao Zhang, Hong Yu, Cong Zhang, and Xinyue Liu. An improved weighted hits algorithm based on similarity andpopularity. In *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, pages 477–480, Iowa City, IA, 2007. IEEE.

[28] Zhi Zhou. *Evaluation of Monte Carlo Method in PageRank*. PhD thesis, University of Missouri-Columbia, 2015.



(a) ER1 (b) ER2 (c) ER3

(d) ER4 (e) GNM1 (f) GNM2

(g) GNM3 (h) GNM4 (i) ER1

(j) ER2 (k) ER3 (l) ER4

(m) GNM1 (n) GNM2 (o) GNM3

(p) GNM4

Fig. 12 Comparing plots of execution time of different algorithms for Link addition

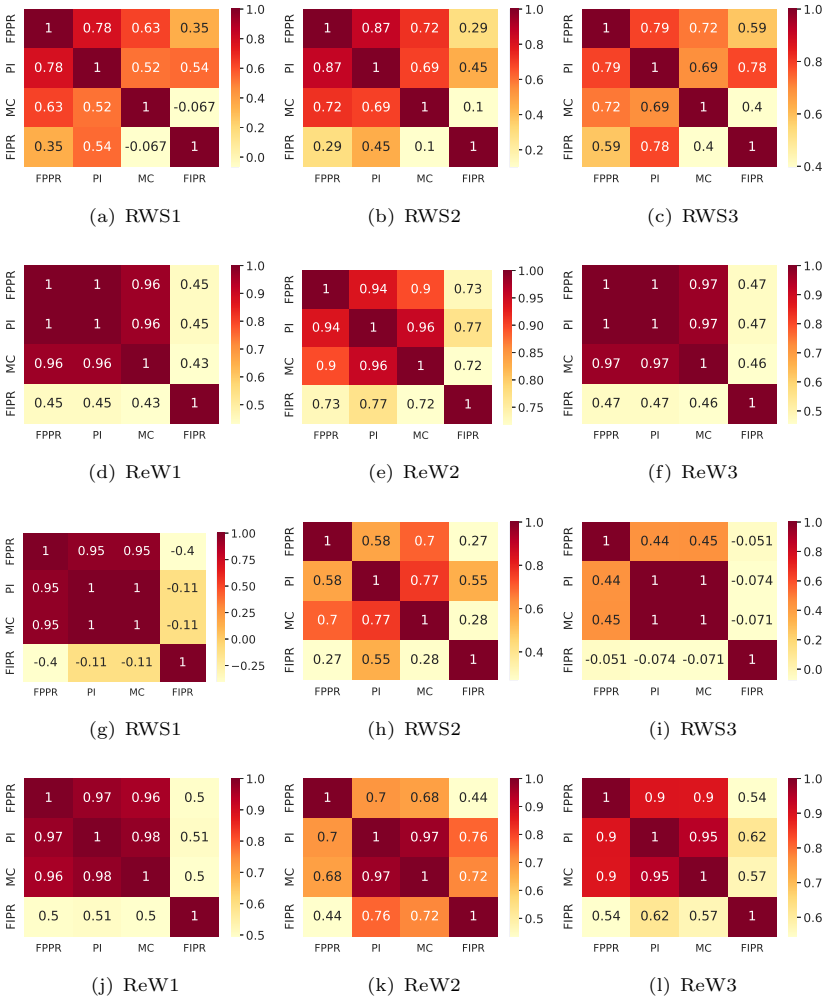
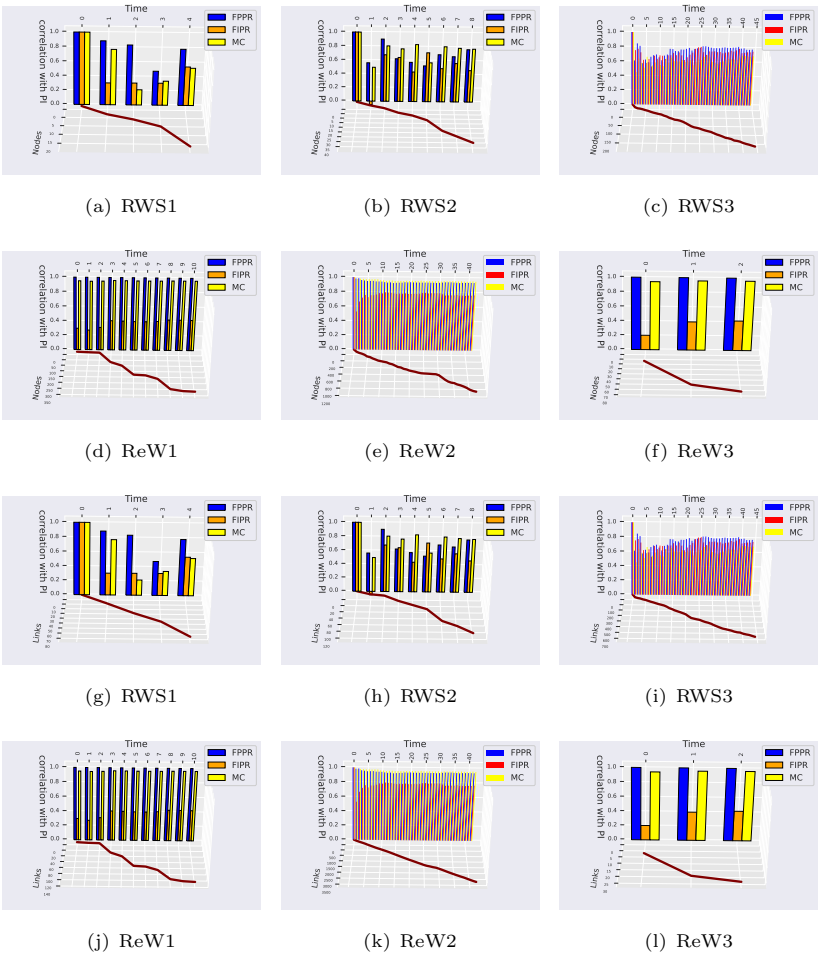


Fig. 13 Spearman Correlation between all four approaches over the different data sets for RW growth and decay



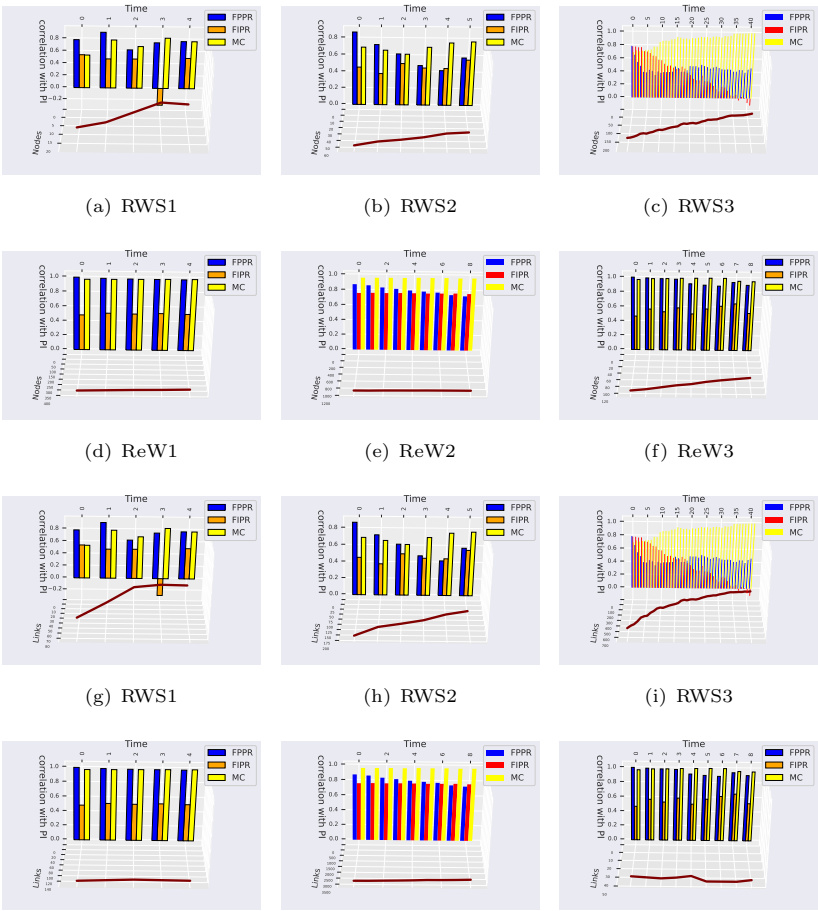
(a) RWS1 (b) RWS2 (c) RWS3

(d) ReW1 (e) ReW2 (f) ReW3

(g) RWS1 (h) RWS2 (i) RWS3

(j) ReW1 (k) ReW2 (l) ReW3

Fig. 14 Spearman Correlation of FPPR VS PI over time along with node and link growth



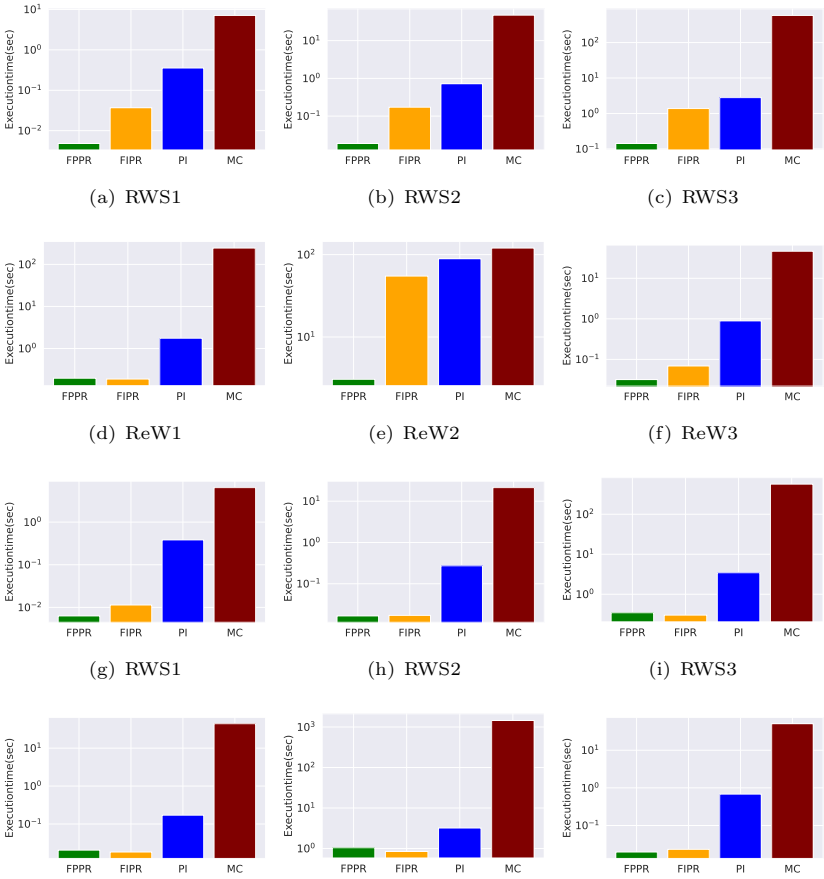
(a) RWS1 (b) RWS2 (c) RWS3

(d) ReW1 (e) ReW2 (f) ReW3

(g) RWS1 (h) RWS2 (i) RWS3

(j) ReW1 (k) ReW2 (l) ReW3

Fig. 15 Spearman Correlation of FPPR VS PI over time along with node and link decay



(a) RWS1 (b) RWS2 (c) RWS3

(d) ReW1 (e) ReW2 (f) ReW3

(g) RWS1 (h) RWS2 (i) RWS3

(j) ReW1 (k) ReW2 (l) ReW3

Fig. 16 Comparing plots of execution time of different algorithms for RW growth and decay