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 [21] 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 [10]. PageRank also finds applications in networks outside the internet. For example, identify new possible drug targets in proteins [12] in biochemistry; predicting how many pedestrians/vehicles visit the individual places or streets [13] etc.

The PageRank was initially proposed in [21]. It is the classical algorithm for ranking web pages. Over the years, many variations of PageRank [5, 16, 17, 25, 28, 32] had been evolved. Randomized algorithms [2, 5, 25, 28, 32] 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 [6, 8, 11, 16, 19, 31] as well. In [6, 8, 16], a subset of graph is chosen and PageRank is computed on that subset using static methods for any updates. In [11, 19, 31] random walk model is used, where random walk segments are adjusted in case of updates. However, these algorithms could not provide satisfying accuracy in terms of relative ranking of nodes or running time for updates, especially for Big Data networks. Hence, a simple dynamic PageRank algorithm that runs fast 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 [22] where only node addition was considered. The proposed algorithm uses the expected value of random surfers to re-calculate the score for changes in the topology. Here by the expected value of random surfers, we mean the estimated number of visits at a node by the random surfers, considering the network is static at that point. For example, when a new node (or link) is added, the score is calculated by adding the estimated scores contributed by in-links to that new node (or target node) and estimating the visits by random surfers through the link chain it is being added to. The same method is used to update existing nodes through the out-links of the newly added node. These are the links that the random surfer may use to visit (or go out from) that node if the static PageRank algorithm was used at that point. 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. Further 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. V_s is the updated set of nodes, and 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 is used in the worst case for node addition. For node deletion, additional E space is required. Thus FPPR takes a total $\mathcal{O}(|V + E|)$ 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 to test FPPR for link addition and deletion. The experimental results show that the ranking of the proposed method is highly correlated with the ranking of the benchmark Power Iteration-based recalculation and better than the state-of-the-art methods like Fast Incremental PageRank (FIPR) [31] and Offset Score Propagation (OSP) [29]. The FPPR is also tested on the simulation of a real-world graph of growth and decay together. The proposed method has better performance than FIPR [31]. In summary, the major contributions of the paper are

1. We propose FPPR algorithm for the dynamic network, which takes less computation and space with respect to other comparing methods, including classical and state-of-the-art algorithms. The proposed algorithm can estimate the PageRank for both node and link addition and deletion.
2. We experimentally show that the updated page ranks are highly correlated with those of the Power Iteration (PI) method. Spearman's rank correlation coefficient is used to compare the ranking of the proposed FPPR with that of the comparing methods.
3. We showed with experiments that the proposed algorithm works with different network changes for evolving networks. Both growing and decaying networks are simulated.

The paper is organized as: Section 1.1 describes the graph model, Section 2 provides a brief literature review, the proposed FPPR method and its rationale are presented in Section 3, experiments performed and corresponding results are reported in Section 4. Finally, Section 5 describes the conclusions of the research work.

1.1 The Model

In the paper, a directed network is represented with graph $G(V, E)$ where V is a 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 are 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
$\Gamma_{in}(u)$	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	$1 - \rho$
m	upper bound of number of edges which could be visited in each iteration
q_{offset}	(n × 1) offset seed vector
$\|\|q_{offset}\|\|$	$L1$ length of q_{offset}
c	restart probability
ϵ	error tolerance
$d_{out}(u)$	outdegree of u i.e. $\|\Gamma_{out}(u)\|$
d^k_{avg}	average outdegree of k nodes
$AV(u)$	approximate visits of u
$AV(u, v)$	contribution 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

2 Related Work

Since the inception of WWW in the 1970s, the size of the internet has been increasing on a rapid scale. There is a need for ranking pages to find relevant information from it. This led to the invention of search engines to make web searches possible for any user query. With the increase in size rank of the pages need updates. The early search engines use algorithms like HITS [14] and PageRank [21]. Over the years, there have 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.

2.1 PageRank

PageRank [21] 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 transition matrix (P^*) is as follows.

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

where I is the unit square matrix, n is the number of nodes in the network. 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 pseudo-code for calculating the 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.

2.2 Static Monte Carlo based PageRank Algorithms

The Monte-Carlo methods are used to approximate classical PageRank [21]. In [25], 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 that is interpreted [32] 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, 5, 25, 28, 32] 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 method, 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 the total number of random walks 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 random walks initiated at every node u in the graph with an 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 an 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 dangling node is a node with no out-links. In this method, R number of random walks is 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.

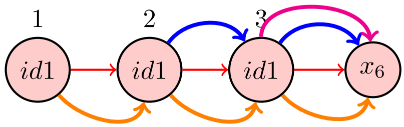
It is shown in [28] that out of all the mentioned methods, the last strategy shows better performance in terms of execution time.

2.3 PageRank for Dynamic Networks

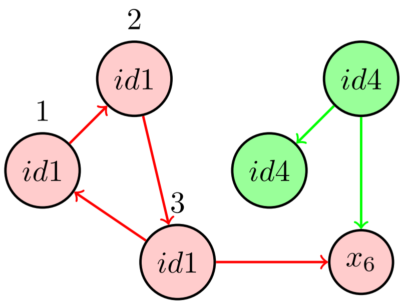
Dynamic networks are those networks where nodes and links get added or deleted dynamically in real time. This is more practical in today's web 2.0 applications. Many algorithms for the dynamic network were proposed in the literature. These are broadly classified into two categories:

1. **Aggregation algorithms** [6, 8, 16]: 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 accuracy, approximation error, and slower execution time. Accuracy in this type of algorithm purely depends on the selected subset [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** [4, 19, 20, 29, 31]: On the other hand, Monte Carlo based approaches use the theory of Markov Chains [11]. 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 [19, 31]. On the other hand, the Offset Score Propagation (OSP) [29] algorithm first calculates offset scores around the modified edges and then propagates the offset scores across the updated graph. Finally, it merges these scores with the current Random Walk Restart (RWR) [27] scores to get the updated RWR scores. In [20], whenever an update happens, the residual is calculated, and the PageRank vector is updated by adding the residual. Most recent algorithm proposed in [4] uses Chebyshev polynomials to approximate PageRank. 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 is to be removed, and the simulated new random walk segment follows the same distribution but is actually a different segment. Hence it brings errors.



(a)



(b)

Fig. 1 Link chains examples (a) without loop and (b) with loop

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

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.

3.1 FPPR for Node addition and Link addition

When a new node is added, FPPR [22] calculates the expected score of random surfers considering the static graph. That is it tries to calculate the expected score generated by the random walks if the Monte Carlo method is executed on the graph after the changes in the network. 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 out-link 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. This linear assumption makes it easier to check the visits carried by random walks initiated at any node in the chain (all the connected nodes with the same ID), especially when there exists a loop in the chain. For example, in Fig. 1(a), R random walks starting from the node (1) holding chain ID 'id1' would have reached 'x6' in 3 hops as shown in orange color, i.e., the expected number of visits is $R * (0.8)^3$. Similarly, the random walk initiated at node (2) must have gone 'x6' in 2 hops (the expected visit is $R * (0.8)^2$) as shown in blue color, and the random walk initiated at node (3) must have gone 'x6' in 1 hop (the expected visit is $R * (0.8)^1$) as shown in pink color. All these contribute to the score of the new node. This linearity assumption will provide the correct expectation value for chain 'id1' in Fig. 1(b) even though it is a loop in the network. Accordingly, we defined Link Sensitivity Index in Definition 2. This linearity assumption is taken in a pessimistic way. The literal meaning of 'pessimistic' is to think about the worst case happening often. While estimating the random walker visits, we believe that the existence of a loop is the worst-case scenario. Because it will be hard to estimate how many times the random walkers had taken the loop and how many visits it contributed to any node. As a 'pessimistic' view, we consider there will always be some existence of a loop, and a linearity assumption is required to calculate the baseline values of the update. Note that the same formulation may not be valid for chain 'id4' in Fig. 1(b). However, the algorithm thinks pessimistic that either 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' doesn't have a loop, but still, the algorithms assume the chain is linear/has a loop and accordingly calculate the PageRank scores of 'x6'. However, we expect this error will be very small. A similar process is adopted in addressing the out-link 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, 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 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 iff u, v are *neighbors and it is 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 amount of scores a node gets from the whole link-chain it is joining. In other words, it is the sum of scores received by the random surfers initiated from the nodes of the chain which are added before this node. We assume the link chain is linear, and the LSI is mathematically defined by:*

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

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

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: $finalResult \div totalVisits$

FPPR does not keep track of all the random walk segments or aggregations. FPPR takes only $4 \times |V|$ space for node/link addition as the proposed algorithm uses 4 vectors of size V namely $linkID$, $approxVisits$, $LinkIDlength$, $outdegree$. 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. Each node in the network will have a $linkID$ and all the nodes in a chain will have the same $linkID$. 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 distinct chains in the graph. More than one chain may pass through one node. In that case, our proposed method uses the lower $linkID$ for that node. One may choose it randomly because this convention does not have any effect on the scores. As we are keeping track of every link(chain) in the graph, we store the length of each $linkID$.

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 ← linkID[v].length + 1
temporary_Var ← R × LSI(n = linkID[u].length)
    comment: LSI calculated by Eqn. 5
approxVisits ← AV(u) comment: Using Eqn. 3
edgeWT(u, v) ← AV(u, v) comment: Using Eqn. 4
finalResult[v] ← approxVisits + linkSensitivityIndex
totalVisits ← totalViists + approxVisits + linkSensitivityIndex
Output: finalResult ÷ totalVisits
```

3.1.1 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 parameters 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. In other words, whatever score is contributed from the source node to the target node in the addition phase is now removed/subtracted from the target node. From the next level, the edge weight ($AV(u, v)$) and PageRank score of the nodes are updated accordingly, i.e., all the neighbors of the nodes that updated (scores ($AV(u, v)$) being subtracted) are pushed into the queue and further Breadth First Traversal continues until the queue is empty. Over time during the addition phase, it might happen that a source node's contribution ($AV(u, v)$) is greater than the target node's score itself. In that case, the target node's score is already undervalued. So, no need for further subtraction/removal of its score. This is the reason that the algorithm SBFU puts only selected nodes, for which incoming edge weights and incoming node's score have been updated, into the queue.

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 ∈ Γ_out(m) do
        if neighbour is not in visited then
            linkID[neighbour] = linkID[m]
            update the linkID[neighbour].length and linkID[delNode].length
            previousEdgeWt ← edgeWt(m, neighbour)
            currentEdgeWt ← AV(m, neighbour)
            changedEdgeWt ← previousEdgeWt − currentEdgeWt
            if changedEdgeWt > 0 then
                totalVisits ← totalVisists − changedEdgeWt
                finalResult[neighbour] ← finalResult[neighbour] − changedEdgeWt
                edgeWt(m, neighbour) ← 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 ← edgeWt(source, target)
totalVisits ← totalVisits − weight
finalResul[target] ← finalResult[target] − weight
update the linkID of the target as discribed in the text
Output: totalVisits, finalResult
```

3.1.2 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.

4 Experiments and Results

Table 2 Dataset used for different experiments

Experiment for Node Addition & Deletion						
Name	$\|V\|$	$\|E\|$	Min $\|\Gamma_{in}\|$	Max $\|\Gamma_{in}\|$	Min $\|\Gamma_{out}\|$	Max $\|\Gamma_{out}\|$
Weibo1 [7]	40	66	0	12	0	6
Weibo2 [7]	206	206	0	45	0	2
Weibo3 [7]	759	780	0	201	0	3
Weibo4 [7]	817	901	0	28	0	297
Random351	351	497	0	7	0	7
Random527	527	783	0	8	0	10
Random751	751	1094	0	8	0	9
Random801	801	1212	0	7	0	11
BA1 [1]	55	154	0	36	0	3
BA2 [1]	105	304	0	48	0	3
BA3 [1]	255	754	0	105	0	3
BA4 [1]	305	904	0	102	0	3
L1 [23]	75k	500k	0	3046	0	1803
L2 [18]	77k	900k	1	2540	0	2508
L3 [24]	14k	9M	0	6173	1	435
L4 [24]	43k	14.5M	0	3539	0	359
Experiment for Link Addition & Link Deletion						
ER1 [9]	100	1977	9	30	10	35
ER2 [9]	200	7949	24	55	24	54
ER3 [9]	350	7750	11	34	0	90
ER4 [9]	500	11327	9	35	0	126
GNM1 [15]	100	120	0	4	0	5
GNM2 [15]	100	1000	2	19	1	18
GNM3 [15]	500	250	0	4	0	4
GNM4 [15]	500	1700	0	10	0	10
Experiment for Real World Simulation						
ReW1 [24]	445	1357	0	21	0	17
ReW2 [24]	1218	3697	0	25	0	24
ReW3 [24]	1480	4057	0	23	0	25
ReWSIM1	53	167	0	8	0	13
ReWSIM2	89	250	0	8	0	11
ReWSIM3	98	2233	0	89	0	50

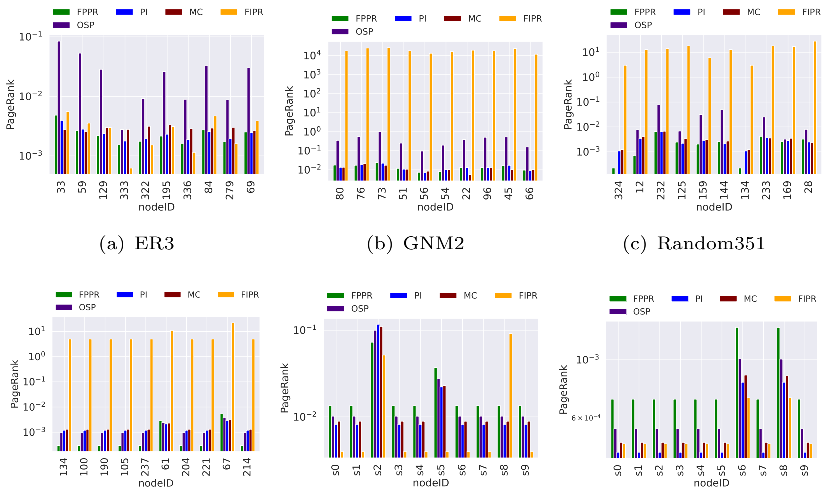
4.1 Dataset

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 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 [9], GNM graphs [15] are used for experimenting FPPR for link addition and deletion. For link deletion, up to 10 percent of the randomly chosen links are deleted. The real-world networks are Wiebo reshare network of [7] and reptilia-tortoise network, aves-weaver network, mammalia-voles network, bio-mouse-gene network, bio-human-gene2 network from the NetworkRepository [24] and Slashdot social network [18], Epinions social network [23] from Stanford Large Network Dataset Collection. 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.

Algorithm 5 Random graph generator

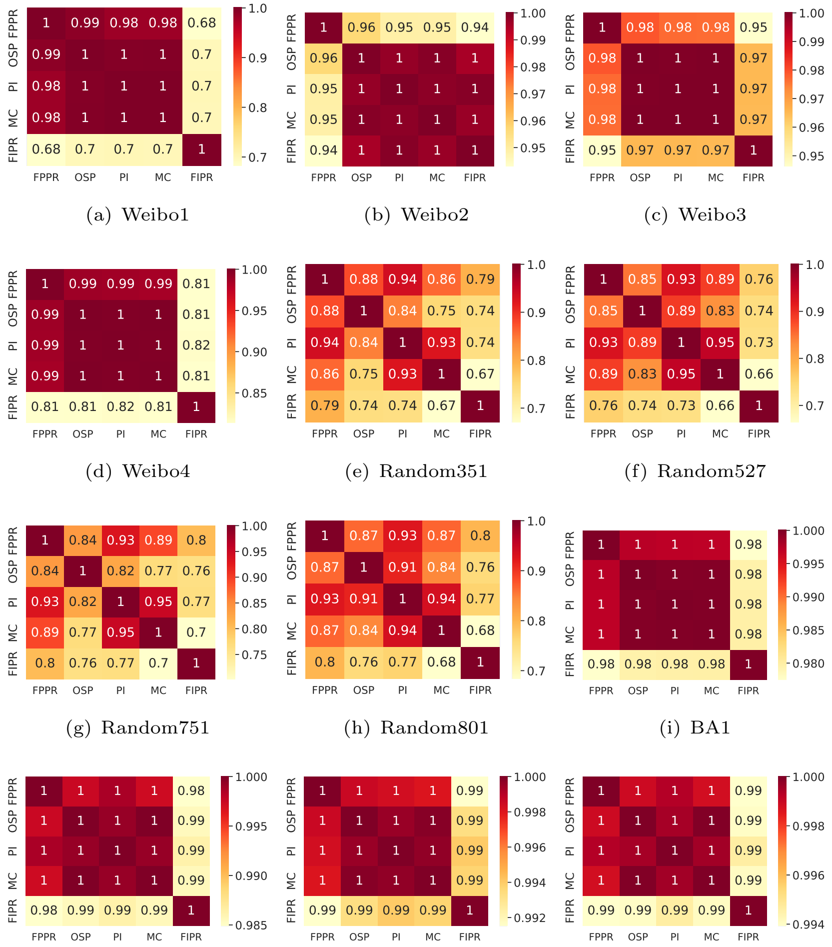
Input: number of nodes N comment: single node '0' exist in the graph
for all $x \in \Gamma_{in}(1, N)$ **do**
  add x to Graph G
  $indegree \leftarrow random[0, 1]$
  if $indegree$ **then**
    $G.addEdge(random[0, x-1], x)$
  end if
  $outdegree \leftarrow random[0, x-1]$
  $shuffle.list[0, ..x-1]$
  while $outdegree$ **do**
    $G.addEdge(x, list[outdegree])$
    $outdegree--$
  end while
end for



(a) ER3 (b) GNM2 (c) Random351

(d) BA3 (e) Weibo1 (f) Weibo4

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



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

(d) Weibo4 (e) Random351 (f) Random527

(g) Random751 (h) Random801 (i) BA1

(j) BA2 (k) BA3 (l) BA4

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

4.2 Comparing Methods

The proposed methods have been compared with the following methods.

- **PowerIteration (PI)** [21]: 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)** [28]: 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 walks considered in the experiment is 1000.

- **Fast Incremental PageRank on Dynamic Networks (FIPR)** [31]: The method is designed for dynamic networks and proposed in 2019. As our algorithm is designed for dynamic networks, we included this method as related research. Parameter R is set to 16 in the experiments.
- **Offset Score Propagation (OSP)** [29]: This method was created for dynamic networks and proposed in 2018. As our algorithm is designed for dynamic networks, we included this method as related research.

4.3 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 fair. Therefore Spearman's rank correlation coefficient [26] 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 between 0.8 to 1 is considered to be strongly correlated [30]. Apart from the Spearman Rank correlation coefficient, the comparing parameters include (i) Change in Spearman Rank correlation coefficient over time, (ii) Execution time.

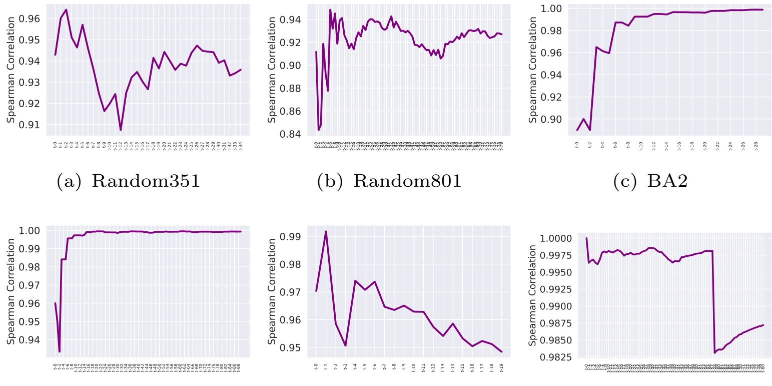
4.4 Results

4.4.1 FPPR for node addition

Accuracy: The results of Spearman's ranking coefficient for all nodes in the network for node addition are shown in 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 0.98 and 0.97, respectively. Further, the proposed method is equal to or better than FIPR for all the networks except Weibo2. The proposed method is also performing better or close to the OSP algorithm. 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.

Spearman's rank correlation with changes in network: As part of the experiment, we would like to see how Spearman's correlation coefficient changes with the 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.

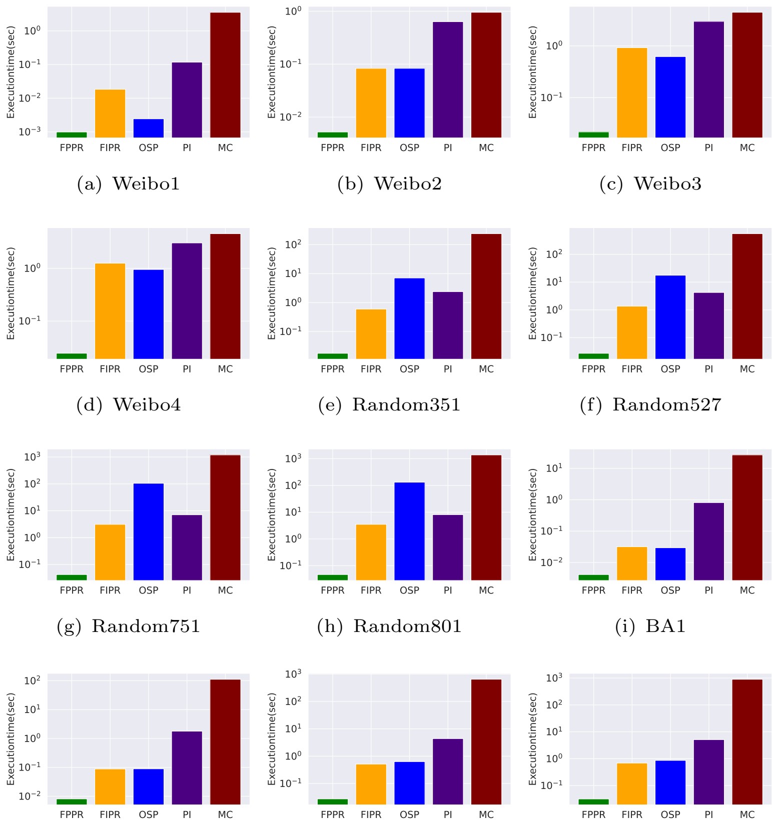
Execution Time: We checked the overall execution time of different comparing algorithms and found that the proposed algorithm is much faster than all the methods we experimented with for all data sets with respect to node addition. The comparison is presented in Fig. 5.



(a) Random351 (b) Random801 (c) BA2

(d) BA4 (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.



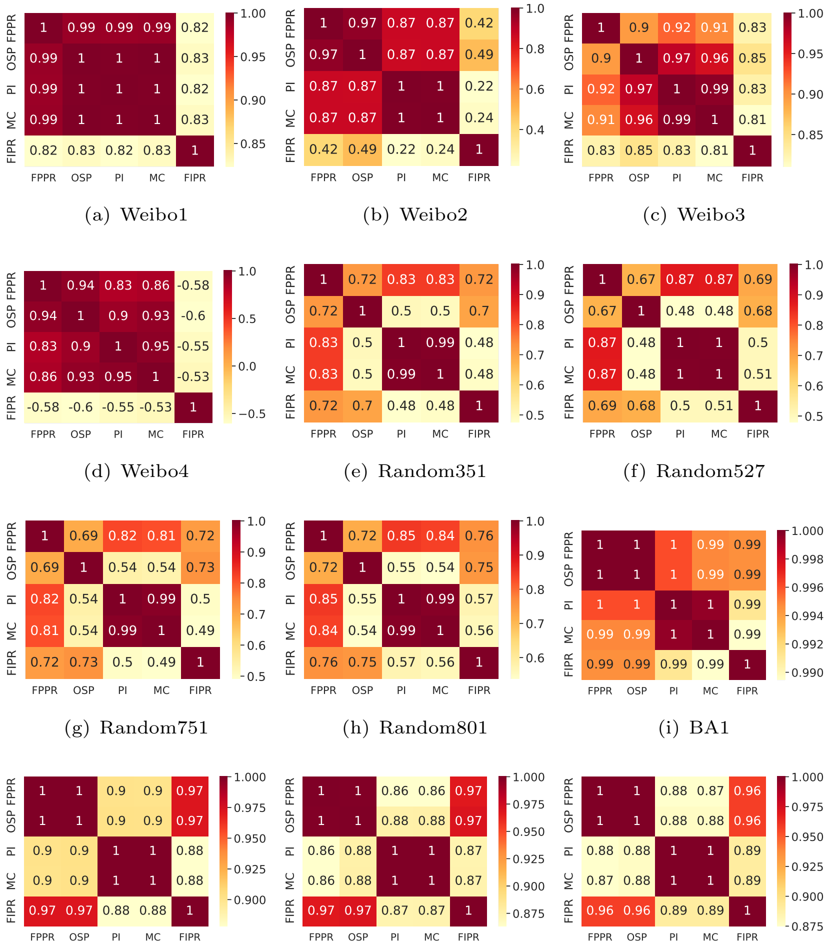
(a) Weibo1 (b) Weibo2 (c) Weibo3

(d) Weibo4 (e) Random351 (f) Random527

(g) Random751 (h) Random801 (i) BA1

(j) BA2 (k) BA3 (l) BA4

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



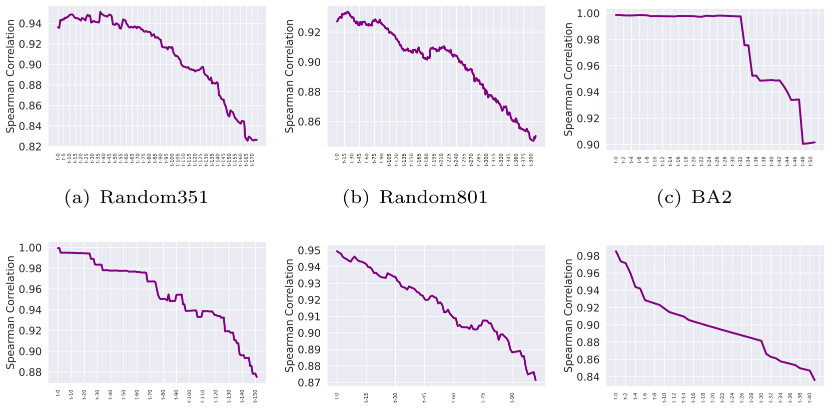
(a) Weibo1 (b) Weibo2 (c) Weibo3

(d) Weibo4 (e) Random351 (f) Random527

(g) Random751 (h) Random801 (i) BA1

(j) BA2 (k) BA3 (l) BA4

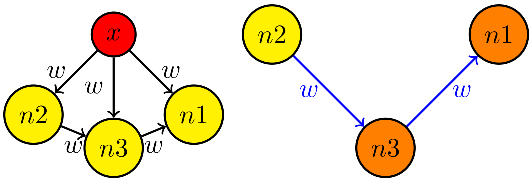
Fig. 6 Spearman Correlation between all four approaches over the different data sets after 50% node deletion



(a) Random351 (b) Random801 (c) BA2

(d) BA4 (e) Weibo2 (f) Weibo4

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



(a) (b)

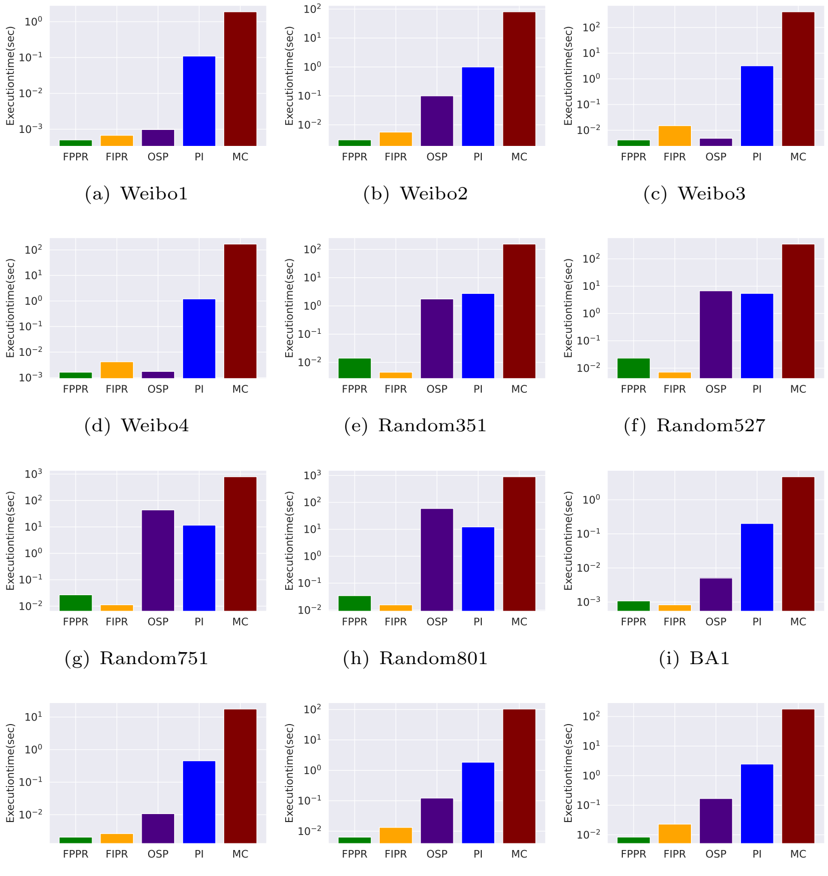
Fig. 8 Examples

4.4.2 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 the benchmark PI method. Median and mean Spearman's correlation with PI method for all the experiments performed for node deletion are 0.89 and 0.90, respectively. FPPR is also observed to be equal to or better than FIPR and OSP.

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 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.

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.



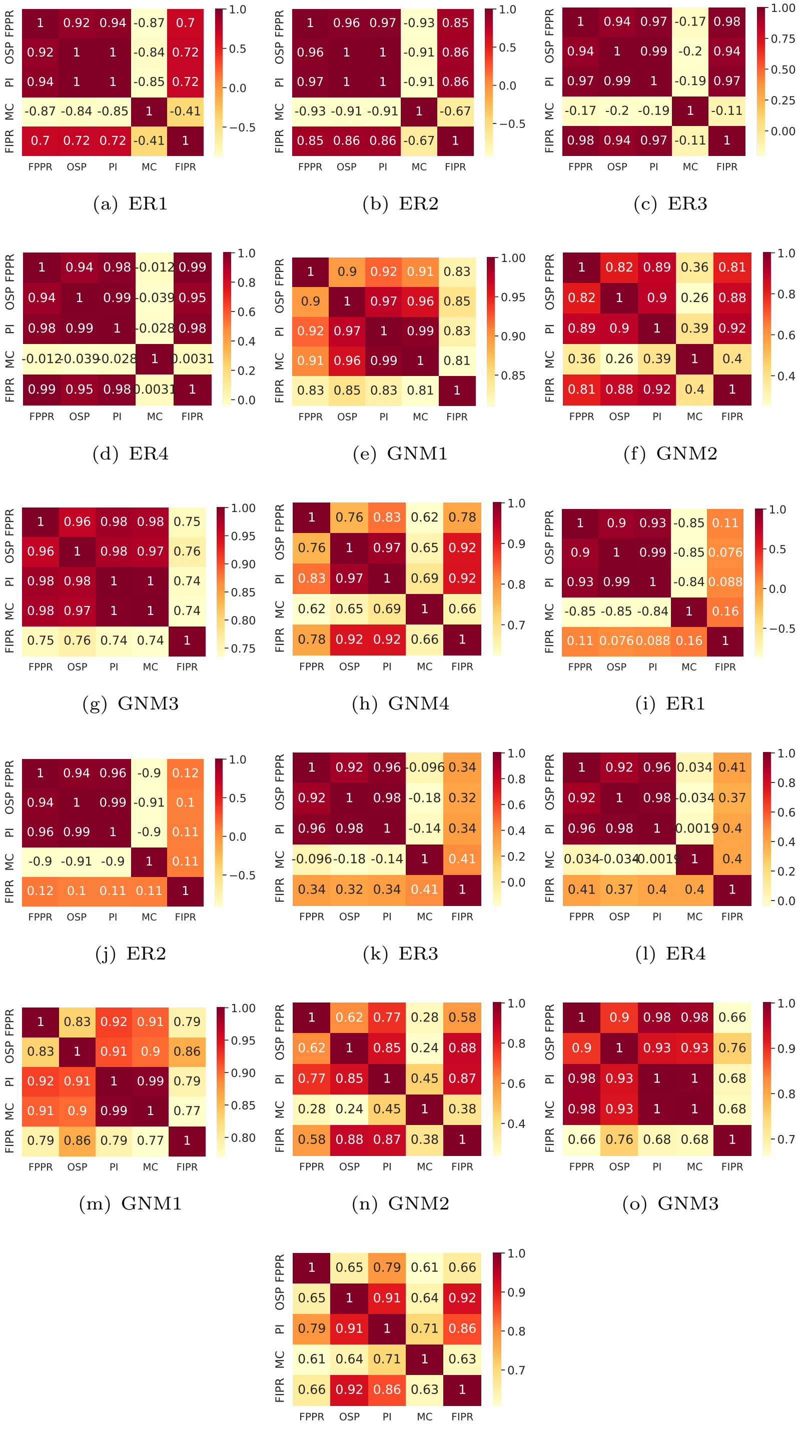
(a) Weibo1 (b) Weibo2 (c) Weibo3

(d) Weibo4 (e) Random351 (f) Random527

(g) Random751 (h) Random801 (i) BA1

(j) BA2 (k) BA3 (l) BA4

Fig. 9 Comparing plots of the execution time of different algorithms for node 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. 10 Spearman Correlation between all four approaches over the different data sets for Link addition and deletion

4.4.3 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 the PI method for all the experiments performed are 0.95 and 0.93 for link addition, and 0.94 and 0.90 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 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 a 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 addition 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.

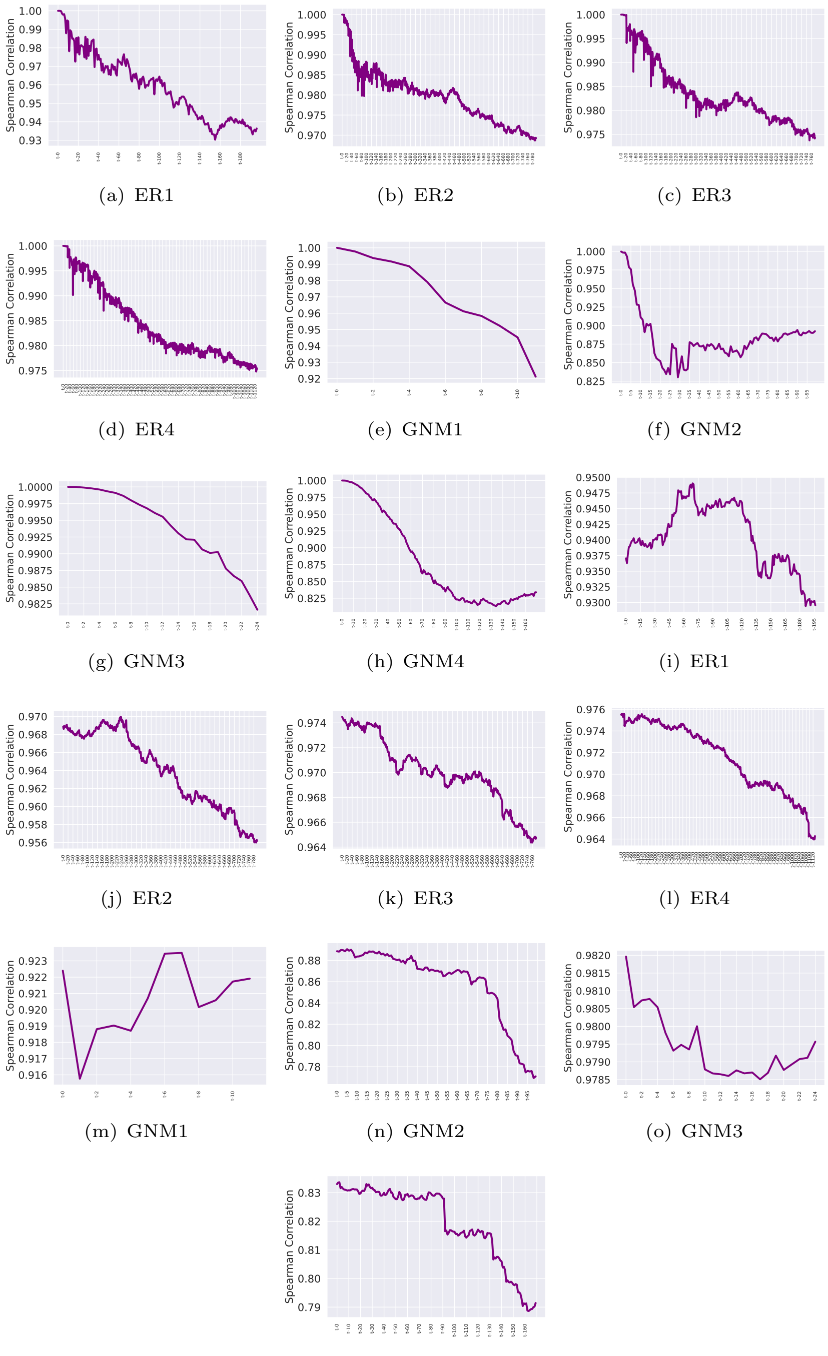
Execution Time: The execution time of the proposed FPPR is much faster than all the comparing methods for link addition and deletion as evident from Fig. 12.

4.5 Accuracy of FPPR with the change of parameter ρ

As part of the experiment, the behavior of FPPR is noted with respect to the change of the parameter ρ (the probability of a random surfer restarting its walk). The parameter ρ is set to values $0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$ and checked the accuracy of FPPR. The results are captured in the Fig. 13. It is evident from the results that the changes in Spearman's Correlation Coefficient are very small (between 0.02 to 0.04) in general except Weibo network where it is about 0.12.

4.6 Performance on Large Scale Data

Four different large-scale temporal datasets have been used to testify the performance of the proposed FPPR algorithms. These networks have nodes ranging from 14K to 77K while the number of links are ranging from 500K to 14.5M. As the datasets are temporal we added the nodes and edges on the network based on their times and calculated the Spearman's Correlation at the end of the generation of the full network. The results are shown in Fig. 14. The results of Spearman's Correlation coefficient are very close to 1 for 3 out of 4 datasets while the results of L2 are also more than 0.8. These results show high accuracy of the proposed method. One of the main objectives of the experiment was to see the performance in terms of execution time. The proposed algorithm is able to generate results in less than 10 seconds for all the data sets.We also captured the accuracy of the FPPR VS Power method along with the FPPR VS Matrix multiplication method using Pseudocode. 0 for few graphs in Fig. 14(c).



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

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

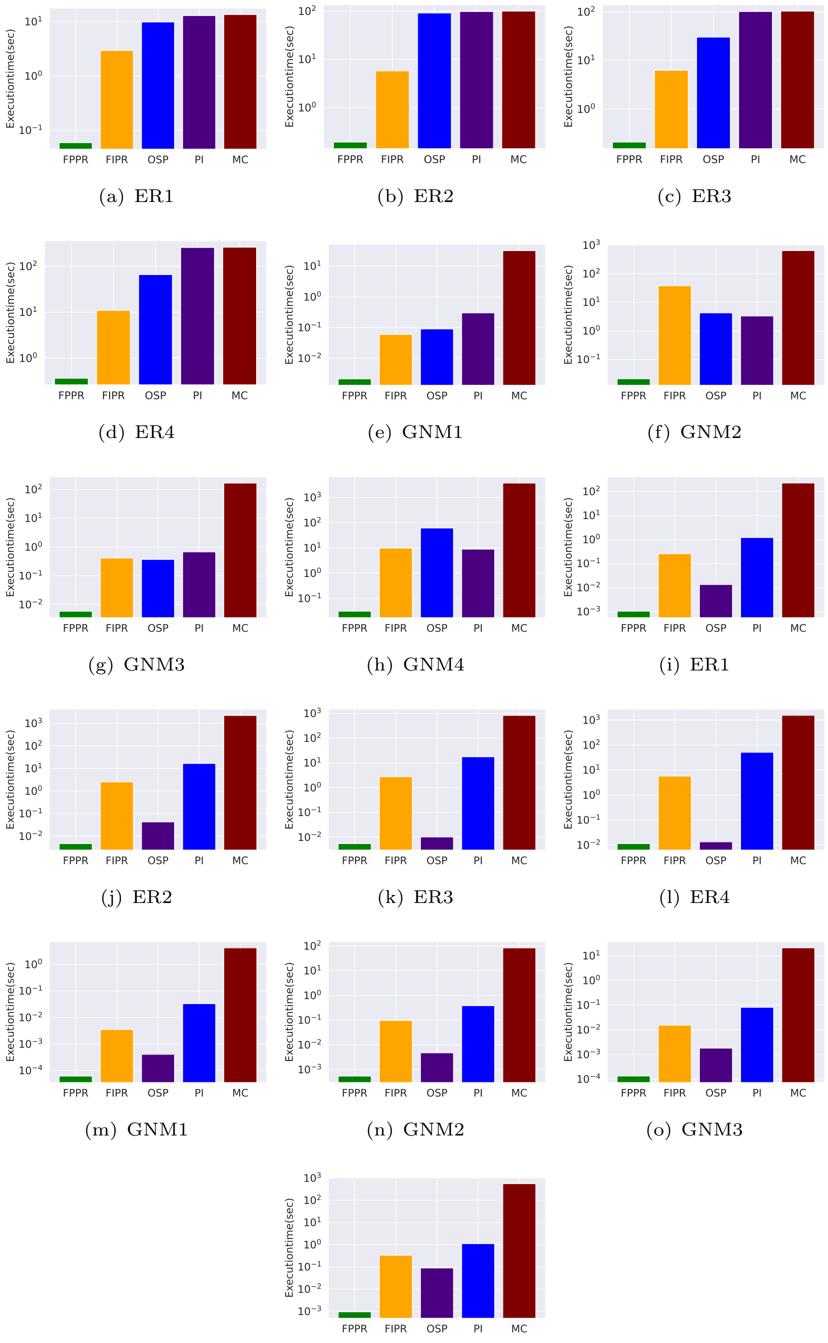
(g) GNM3 (h) GNM4 (i) ER1

(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 the addition/deletion of 10 links in the network.



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

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

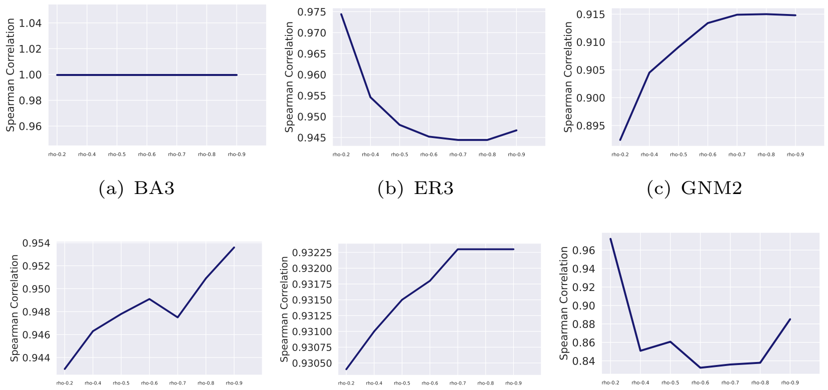
(g) GNM3 (h) GNM4 (i) ER1

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

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

(p) GNM4

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



(a) BA3 (b) ER3 (c) GNM2

(d) Random351 (e) ReW3 (f) Weibo3

Fig. 13 PageRank of FPPR with respect to PI considering the change in the parameter ρ

4.7 Approximation error with the exact PageRank

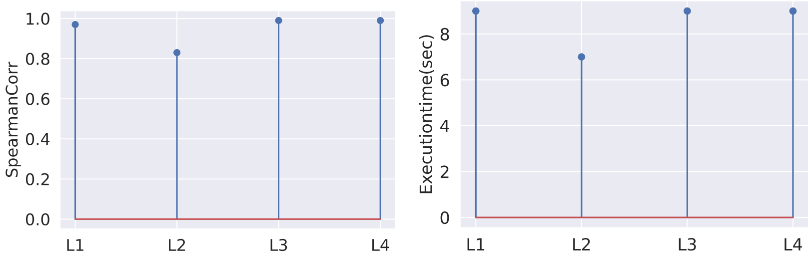
Although Power Iteration is a well-known method for practically performing the PageRank calculation, we wish to see how this is different for matrix multiplication methods. We experimented with small-size datasets and calculate Spearman's correlation coefficient for the proposed FPPR against both the Power Iteration method and the Matrix Multiplication method. The result is plotted in Fig. 15. As expected the correlation coefficient is lower with the matrix multiplication method, however, it is not too far from it. In fact, in both cases, the results show a high correlation with the ranking as the results are over 0.8 for all the data sets.

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	$O\frac{(knR)}{(\lvert E\rvert)}$	$O\frac{(knR)}{(\lvert E\rvert)}$	$\mathcal{O}(nR)$
OSP	$O(mlog_{(1-c)}\frac{\epsilon}{(\lvert\lvert q_{offset}\rvert\rvert)})$	$O(mlog_{(1-c)}\frac{\epsilon}{(\lvert\lvert q_{offset}\rvert\rvert)})$	$\mathcal{O}(V^2)$
Our Method	$\mathcal{O}(k \times d_{avg}^k)$	$\mathcal{O}(\lvert V_s\rvert + \lvert E_s\rvert)$	$\mathcal{O}(\lvert V\rvert + \lvert E\rvert)$

4.8 Computation Complexity

Computation complexities for all the operations, node addition, node deletion, link addition, and link deletion of different algorithms along with the space complexity are shown in Table 3. The worst-case complexity of the proposed algorithm for node addition is $\mathcal{O}(k \times d_{avg}^k)$, where k is the number of nodes/links added to the network



(a) Spearman's Correlation with PI Method (b) Total Execution Time

Fig. 14 Results on large scale networks

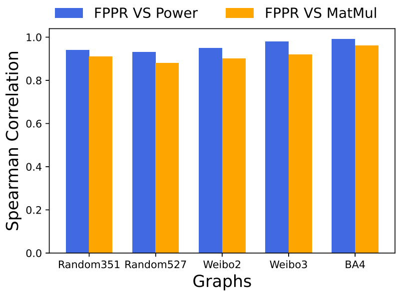


Fig. 15 Spearman Correlation of FPPR with Power Iteration and Matrix Multiplication method

as in each update the Algorithm 1 updates for all the outgoing links (line 13) and calculates for all incoming edges (line 2). Each calculation can be done in linear time with the formula presented in Equations 3, 4 and 5. 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. Space required for the proposed algorithm is $4 \times |V|$ for node and link addition to keep 4 vectors corresponding to $linkID$, $approxVisits$, $LinkIDlength$, and $outdegree$. In order to have better deletion techniques intermediate $AV(u, v)$ are stored for each links in the network. Hence, it is taking $O(|E|)$ space. That follows the cumulative space requirement is $O(|V| + |E|)$.

4.9 FPPR for real-world graph simulation (modeling growth and decay together)

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 have been incorporated: node addition, node deletion, link addition, and link deletion. 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

Algorithm 6 Real World graph simulation

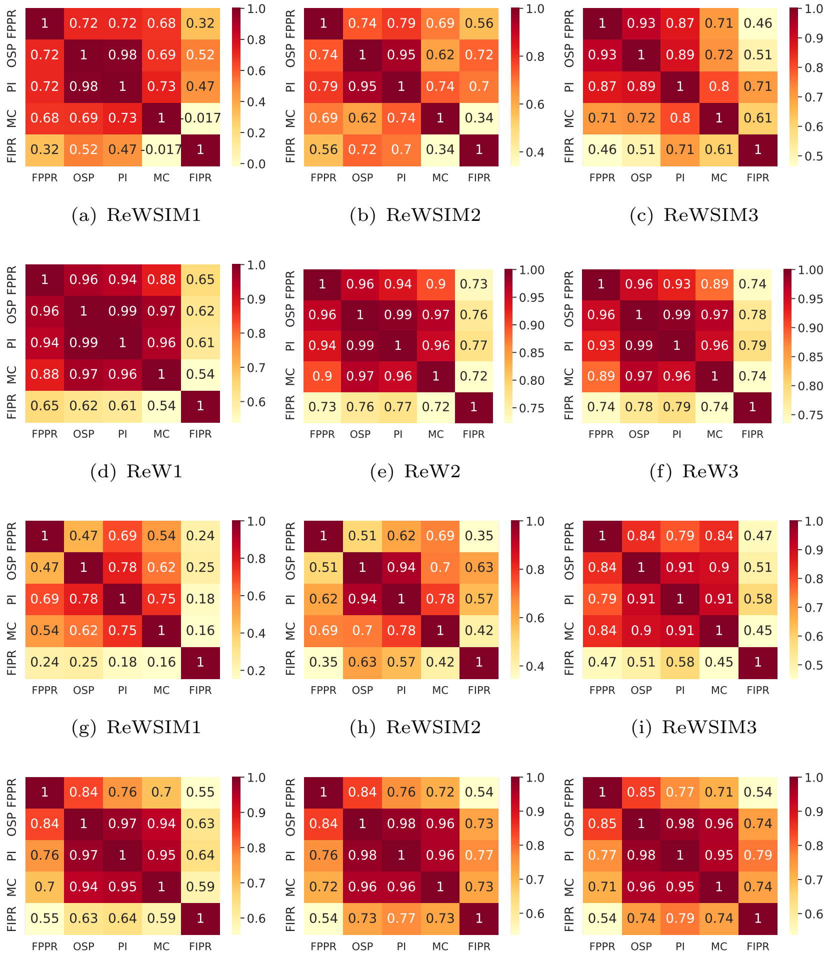
```
Input: number of operations K
while K do
    addnodeProb ← random.uniform(0, 1)
    addlinkProb ← random.uniform(0, 1)
    delnodeProb ← random.uniform(0, 1)
    dellinkProb ← random.uniform(0, 1)
    if addnodeProb < ρn+ then
        algo.(1)
        K−−
    end if
    if addlinkProb < ρl+ then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

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 decaying 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 the 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 50, 100, and 200. For growth, FPPR is in good correlation with respect to the 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 0.86. For decay, FPPR is reasonably correlated with the PI method, and median and mean values of 0.76 and 0.76 were achieved. In all the dataset graphs, FPPR performed better than the FIPR. The results are shown in the Fig. 16 ((a) to (f)) for graph growth, Fig. 16 ((g) to (l)) for graph decay.

Spearman's rank correlation with changes in network: The changes in the spearman rank correlation are captured in regular intervals (after every ten operations) for both graph growth and decay. The results are shown in the Fig. 17 ((a) to (f)) for node growth, Fig. 17 ((g) to (l)) for link growth and Fig. 18 ((a) to (f)) for node decay, Fig. 18 ((g) to (l)) for link decay. The graph represents time on the x-axis, the change in Spearman rank coefficient of all the comparing methods with respect to the benchmark PI method on the y-axis, and a bar plot showing the number of nodes/links on the z-axis. In the graph growth phase, it is evident that the proposed FPPR is performing better than or equal to FIPR. FPPR is even performing equally with respect to the static MC method for the graphs RWS1, RWS2, ReW1, and ReW3. FPPR had a gradual dip in Spearman rank correlation in the graph decay phase but performed better than or equal to FIPR.

Execution Time: The proposed FPPR has faster execution times compared to all the comparing methods in both graph growth and decay phases. The results are presented in Fig. 19 ((a) to (f)) for graph growth and in Fig. 19 ((g) to (l)) for graph decay.



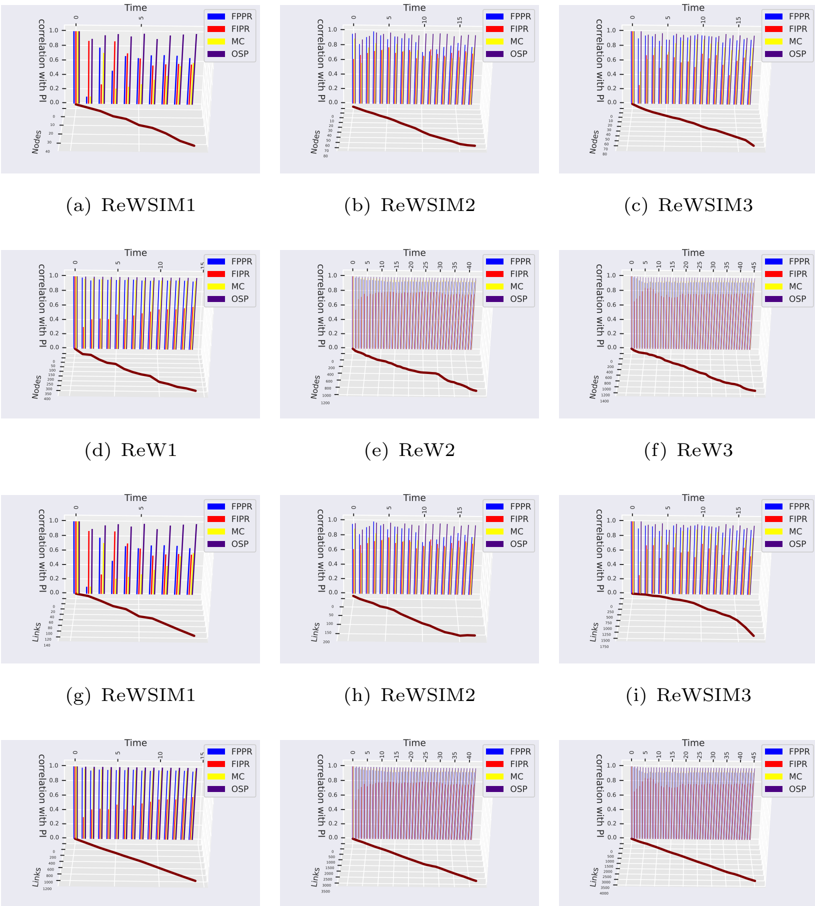
(a) ReWSIM1 (b) ReWSIM2 (c) ReWSIM3

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

(g) ReWSIM1 (h) ReWSIM2 (i) ReWSIM3

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

Fig. 16 Spearman Correlation between all four approaches over the different data sets for RW growth and decay



(a) ReWSIM1 (b) ReWSIM2 (c) ReWSIM3

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

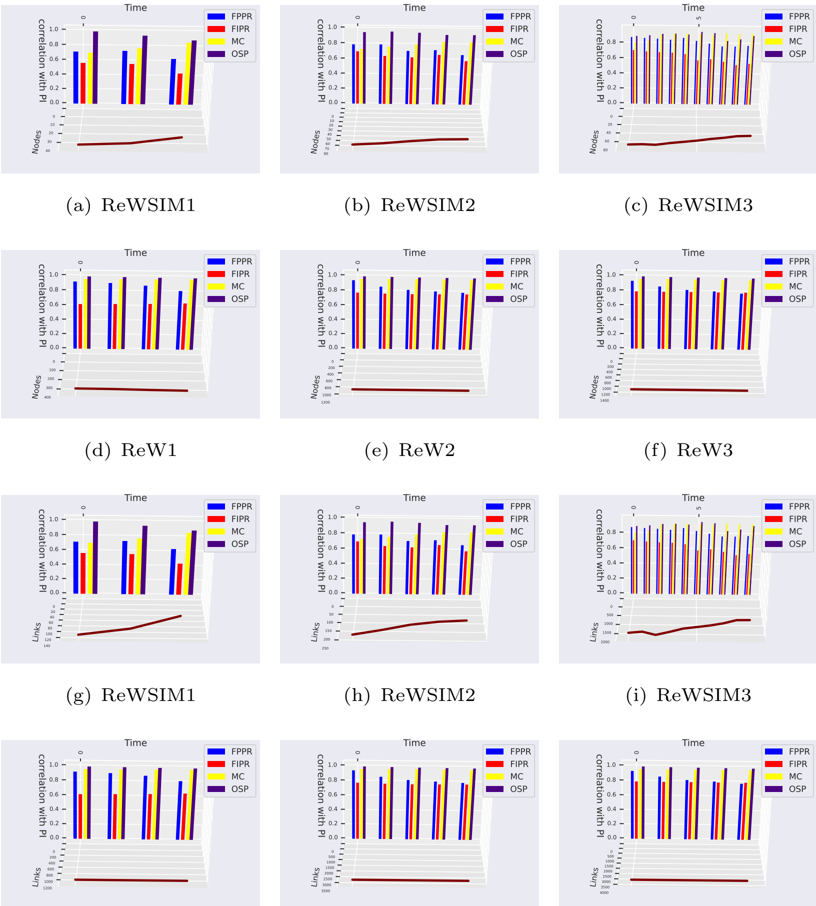
(g) ReWSIM1 (h) ReWSIM2 (i) ReWSIM3

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

Fig. 17 Spearman Correlation of FPPR VS PI over time along with node and link growth

5 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, and link deletion. We showed through experimental results that the results of the proposed algorithm are highly correlated with that of the benchmark Power Iteration method. In particular, the minimum correlation in ranking found for the addition of node, deletion of node, the addition of link, and deletion of the link are 0.95, 0.82, 0.77, and 0.83 respectively. The proposed FPPR is also shown to perform better in the ranking than the FIPR algorithm and better than or equal to the state-of-the-art OSP algorithm for all different topological changes in the network. The execution time is significantly faster while providing more acceptable results.



(a) ReWSIM1 (b) ReWSIM2 (c) ReWSIM3

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

(g) ReWSIM1 (h) ReWSIM2 (i) ReWSIM3

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

Fig. 18 Spearman Correlation of FPPR VS PI over time along with node and link decay

While the PageRanks of growing and sinking networks with the proposed FPPR method show 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.

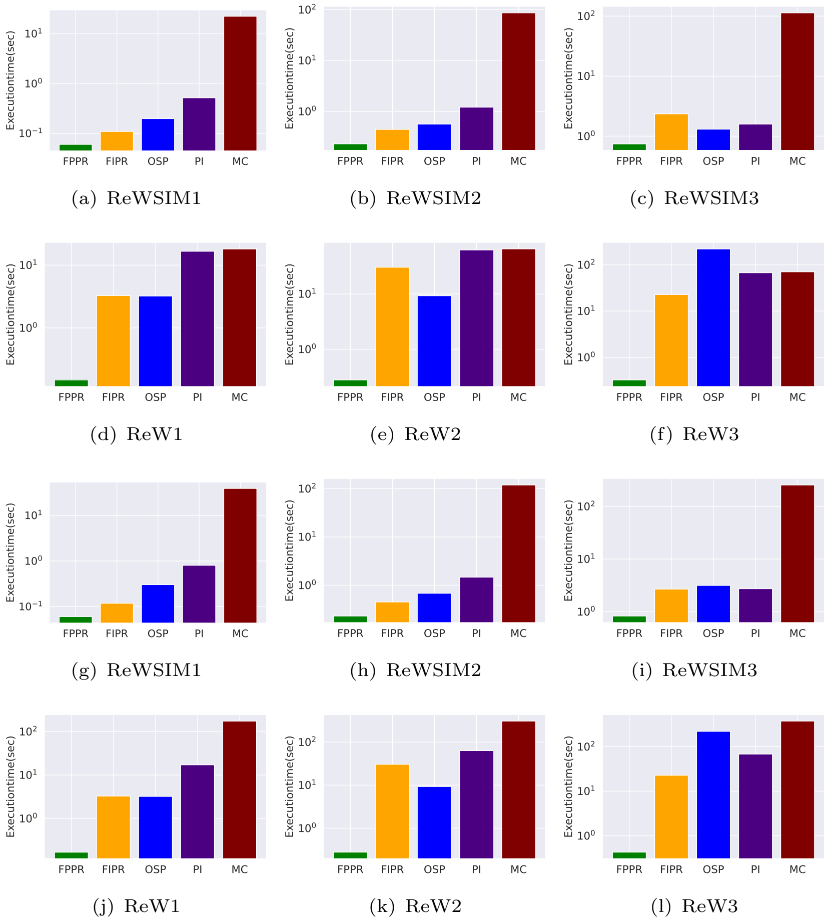


Fig. 19 Comparing plots of the execution time of different algorithms for RW growth and decay

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

[4] Esteban Bautista and Matthieu Latapy. A local updating algorithm for personalized pagerank via chebyshev polynomials. *Social Network Analysis and Mining*, 12(1):1–11, 2022.

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

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

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

[8] 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.

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

[10] 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.

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

[12] 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.

[13] Bin Jiang. Ranking spaces for predicting human movement in an urban environment. *International Journal of Geographical Information Science*, 23(7):823–837, 2009.

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

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

[16] 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.

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

[18] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[19] 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.

[20] Naoto Ohsaka, Takanori Maehara, and Ken-ichi Kawarabayashi. Efficient pagerank tracking in evolving networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 875–884, 2015.

[21] 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.

[22] 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.

[23] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In *International semantic Web conference*, pages 351–368. Springer, 2003.

[24] 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.

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

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

[27] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*, pages 613–622. IEEE, 2006.

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

[29] Minji Yoon, Woojeong Jin, and U Kang. Fast and accurate random walk with restart on dynamic graphs with guarantees. In *Proceedings of the 2018 World Wide Web Conference*, pages 409–418, 2018.

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

[31] 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.

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