Abstract

This research investigates the integration of the PageRank algorithm, a key metric for evaluating webpage hierarchy, into the domain of distributed computing. Originally developed by Google, PageRank measures the popularity of webpages through a stochastic process resembling the navigation of a random surfer. The study focuses on tackling the computational challenges associated with PageRank in distributed systems, leveraging the inherent Markovian properties inherent in random walks.

Our methodology centers on the application of map-reduce and Spark frameworks, facilitating scalable computations suitable for large-scale networks. We delve into the formal connection between Markov chains and the PageRank algorithm, elucidating the intrinsic significance of nodes in diverse network structures. Furthermore, we provide a comprehensive implementation of the algorithm using MR Job, offering insights into the dynamic evolution of PageRank scores.

The experimental design encompasses the manipulation of damping factors and iterations, allowing for a thorough analysis of convergence behavior under various computational scenarios. Through meticulous examination of the results, we discern patterns, anomalies, and the impact of parameter adjustments on PageRank scores. The findings contribute valuable insights into the fundamental dynamics of PageRank and its adaptability to distributed computing environments.

Contents

1. INTRODUCTION

PageRank stands as a cardinal metric in the hierarchical assessment of web pages. It can be construed as the frequency with which a random surfer visits a web page, thereby serving as a quantitative indicator of the webpage's popularity.

Over the preceding decade, PageRank has surfaced as a robust metric for gauging the relative significance of nodes within diverse networks, finding utility across the realm of computer science encompassing "distributed networks, data mining, web algorithms, and distributed computing. PageRank, derived from the steady-state distribution or top eigenvector of the Laplacian, signifies a modified random walk process."[1] Its efficacy lies in discerning pivotal nodes and illuminating node similarities.

Functioning as a probability distribution, PageRank embodies the likelihood that a user navigating through links randomly reaches a particular page. Despite recent efforts focusing on enhancing random walks in distributed networks, a noticeable gap persists in establishing verified results for the efficient distributed computation of PageRanks. This gap is attributed to the time complexity of finding eigenvalues, which is on the order of $O(n^3)$. Consequently, iterative approaches involving matrix-vector multiplications until approximate convergence are employed to address this challenge.

In addition to linear algebraic methods, such as the Power Iteration, an alternative prevalent approach for computing PageRanks is the Monte Carlo method. The Monte Carlo method, specifically, centers on the fundamental idea of approximating PageRanks by directly simulating the associated random walk and subsequently inferring the stationary distribution based on the observed distribution throughout the walk.

This investigation employs a straightforward Monte Carlo approach for simulating PageRank , but the primary focus of our research lies in the computation of PageRanks, leveraging the Markovian properties inherent in random walks within a distributed system. The methodology involves utilizing map reduce and Spark to achieve efficient and scalable computations in this context.

The PageRank algorithm, conceptualized as the actions of a stochastic web surfer, lends itself to a Markov chain framework for prognosticating system behavior as it transitions from one state to another, contingent solely upon the extant state. Nonetheless, the model grapples with the predicament of dangling nodes—nodes that defy inclusion in the Markov chain model. This investigation concentrates on the employment of Markov chain theory in the PageRank algorithm and

deliberates upon several methodologies devised to address the dangling node predicament.

2. MARKOV CHAINS AND PAGERANK ALGORITHM : A FORMAL VIEW

A. Markov Chains , Transition Matrix and PageRank Algorithm

Markov chains function as a predictive tool to analyze the dynamic evolution of a system as it moves through different states, emphasizing the current state in its modeling approach. This modeling technique utilizes a matrix and vector to encapsulate and project the anticipated trajectory of the system. Markov chains are particularly valuable in scenarios involving state transitions, representing a stochastic process for systems that occupy one of a finite set of states at any given time $t = 1, 2, 3 \ldots n$. At each time step t, the system transitions from state v to u with a probability denoted as p_{uv}—a parameter that remains constant over time. This transition probability stands as a crucial attribute of Markov chains, determining the subsequent state based solely on the current state, without consideration of preceding states.

The Transition Matrix, denoted as T, takes the "form of an $n \times n$ matrix, derived from the inherent transition probabilities in the Markov process, where n represents the number of distinct states. Each element in the transition matrix, represented as t_{uv}, corresponds to the probability of transitioning from state v to state u within a single time slot."[2] Hence, it is stipulated that $0 \leq t_{uv} \leq 1$ for all $u, v = 1, 2, \ldots, n$. Below is an exemplary transition matrix for an 11-state Markov chain:

$$t_{uv}^{T} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.33 & 0 & 0.33 & 0 & 0.33 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Several crucial properties define a valid Transition Matrix:

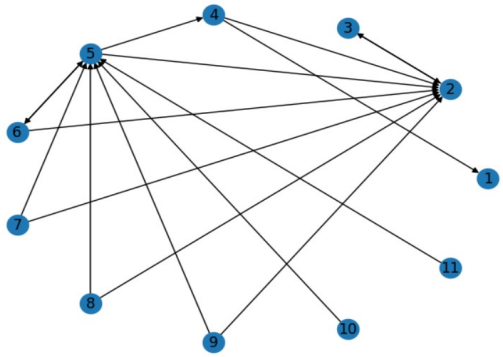

FIG. 1: Example of graph used for computation

1. The matrix must possess square dimensions and non-negativity, indicating equal numbers of rows and columns and all entries being non-negative. Each row and column serves as a representation of an individual state.

2. All entries in the matrix are required to denote probabilities, necessitating each entry to fall within the inclusive range of 0 to 1.

3. The summation of entries in a given row reflects the cumulative transition probabilities from one state to another. Consequently, the sum of entries in any column must equal one, establishing the matrix as a stochastic matrix.

"We formally delineate the PageRank of a graph $G = (V, E)$. Let ϵ be a fixed small constant (designated as the reset probability), wherein, with a probability of ϵ, the random walk commences from a node selected uniformly at random from the entire network. The PageRank vector of the graph is the stationary distribution vector π emerging from a specialized random walk. At each step of this random walk, with a probability of ϵ, the walk originates from a randomly chosen node; with the remaining probability of $1 - \epsilon$, the walk pursues a randomly selected outgoing edge (neighbor) from the current node, progressing to that neighbor. Consequently, the PageRank transition matrix on the state space, or vertex set V, can be succinctly expressed as:" [1]

$$P = \left(\frac{\epsilon}{n}\right) J + (1 - \epsilon)Q \tag{1}$$

" Here, J represents the matrix with all entries being 1, and Q signifies the transition matrix of a simple random walk on G. Specifically, Q_{ij} is defined as $1/k$ if j is one of the $k > 0$ outgoing links of i; otherwise, it is 0. " [1]

PageRank offers a more streamlined methodology for assessing the significance of a web page,

predicated on the enumeration of pages linking to it, commonly referred to as in-coming or back-links. Notably, the algorithm assigns differential weightings to these in-coming links, with those originating from reputable pages bearing greater influence than those from non-reputable sources. The computation of the PageRank (PR) for a given page (p) is expressed by the following formula:

$$PR(p) = d \sum_{q \in pa_p} \frac{PR_q}{O_q} + (1-d)/n \tag{2}$$

In this formula, the damping factor (d) assumes values within the range of $0 < d < 1$, and O_q signifies the count of out-going links from the page q. The equation underscores the iterative nature of PageRank computation, where the importance of a page is determined by the collective influence of pages pointing to it, factoring in the PageRank and out-degree of each such linking page. The damping factor introduces a measure of probability for the surfer to continue navigating the web graph, ensuring convergence in the PageRank computation process.

We elucidate the intricate " relationship between the PageRank algorithm and Markov chains, conceptualizing a random surfer navigating the Web. This surfer progresses from one page to another by randomly selecting an outgoing link from the current page, a stochastic process that may encounter dead ends or cycles within interconnected groups of pages. At times, the surfer randomly chooses any page on the Web, embodying a theoretical random walk recognized as a Markov chain or Markov process."[2] The PageRank of a page is defined by the limiting probability of a dedicated random surfer infinitely visiting that specific page.

The significance of a page is informed by the quantity and quality of both incoming and outgoing links. Pages accrue importance based on the number and quality of backlinks, with links from reputable pages carrying more weight. Moreover, if a prominent page directs to several others, its influence is distributed among them. The foundational PageRank formulation begins with the definition of a page p and its rank $PR(p)$:

$$PR(p) = \sum_{q \in pa_p} \frac{PR_q}{|O_q|}$$

Here, pa denotes the set of pages pointing to p, and O_q is the count of forward links from page q. This formulation is inherently recursive. The PageRank algorithm commences with an initial value of $PR_p^{(0)} = \frac{1}{n}$, where n represents the total number of pages on the Web. Subsequent iterations follow the recursive equation:

$$PR_p^{(k+1)} = \sum \frac{PR_q^k}{|O_q|} \text{ for } k = 0, 1, 2, \ldots \tag{3}$$

Matrix notation provides a concise representation of the iterative process:

$$q^{k+1} = Tq^k \tag{4}$$

The aforementioned predicament can be addressed through the derivation of the eigenvector; however, our resolution method involves an iterative approach until convergence is attained.

Here, q^k denotes the PageRank vector at the k^{th} iteration, and T represents the transition matrix for the Web. However, convergence challenges may arise, necessitating an irreducible, aperiodic Markov chain characterized by a primitive transition probability matrix.

To address convergence issues, the PageRank algorithm transforms the hyperlink structure of the Web into a primitive stochastic matrix. If there are n pages, matrix T of dimensions $n \times n$ has elements t_{pq} representing the probability of moving from page p to q in a single step. The fundamental model sets $t_{pq} = 1/|O_q|$. If a link from q to p exists, the transition probability is $\frac{1}{|O_q|}$; otherwise, it is 0.

This adaptation ensures the construction of a primitive stochastic matrix, fostering convergence and guaranteeing " the existence of a unique stationary distribution vector, denoted as q, which becomes the PageRank vector. The application of the power method with a primitive stochastic matrix"[2] ensures convergence to q, irrespective of the initial vector.

B. Considering Pagerank Mass

PageRank is conceptualized as a probabilistic distribution, mirroring the transfer of probability mass to neighboring nodes through outbound connections. To establish a closed loop, each node amalgamates all contributions to its PageRank, subsequently recalculating its PageRank score. This iterative process can be perceived as the diffusion of PageRank mass along outgoing edges.

In the transition from mappers to reducers, where emitted PageRank mass is consolidated, a meticulous approach is imperative. Presuming the summation of PageRank values across all nodes converges to unity, the cumulative sum of updated PageRank values within the reducer must consistently embody a valid probability distribution. The emission of each unit of PageRank mass is distinctly characterized as a value, complemented by the inclusion of node IDs pertaining to the respective neighbors.

In instances where the algorithm encounters graphs featuring dangling nodes, an inherent challenge arises in maintaining the conservation of total PageRank mass. This challenge stems from the omission of key-value pairs when mappers confront dangling nodes. To rectify this, an astute approach involves the equitable redistribution of the "lost" PageRank mass at dangling nodes across all nodes within the graph. Various methodologies exist to delineate the allocation of this absent PageRank mass.

3. IMPLEMENTATION

Algorithm 1 PageRank MapReduce Algorithm

```
1: class Mapper
2:     method Map(nid n, node N)
3:     p ← N.PageRank / |N.AdjacencyList|
4:     Emit(nid n, N)                          ▷ Pass along graph structure
5:     for all nodeid m ∈ N.AdjacencyList do
6:         Emit(nid m, p)                      ▷ Pass PageRank mass to neighbors
7:     end for
8: class Reducer
9:     method Reduce(nid m, [p1, p2, ...])
10:    M ← ∅
11:    for all p ∈ counts [p1, p2, ...] do
12:        if IsNode(p) then
13:            M ← p                           ▷ Recover graph structure
14:        else
15:            s ← s + p                       ▷ Sum incoming PageRank contributions
16:            M.PageRank ← s
17:            Emit(nid m, node M)
18:        end if
19:    end for
```

FIG. 2: PageRank Algorith implemented using MR Job

Algorithm 1 Page Rank Algorithm

```
1: Input: Links RDD, ITERATIONS
2: Output: Updated ranks RDD
3: links ← spark.textFile(...).map(...).persist()
4: ranks ← // RDD of (URL, rank) pairs
5: for i in 1 to ITERATIONS do
6:     // Build an RDD of (targetURL, float) pairs
7:     // with the contributions sent by each page
8:     contribs ← links.join(ranks).flatMap{(url, (links, rank)) ⇒
9:         links.map(dest ⇒ (dest, rank/links.size))}
10:    // Sum contributions by URL and get new ranks
11:    ranks ← contribs.reduceByKey((x, y) ⇒ x + y)
12: end for
13: return ranks
```

FIG. 3: PageRank Algorith implemented using spark

4. DATASET

Below is the data of actual page rank values of graph used for computation and graph represenation figure respectively.

1	0.009294800175269574	2	0.01817665742040969	3	0.0057674153787366745	4	0.026439943683319554
5	0.012148641115339096	6	0.0039011513263121815	7	0.011855242938075805	8	0.01656045283647962
9	0.00422871998576948	10	0.012724094283322952	11	0.025547518632071185	12	0.0038580871481454695
13	0.01080086603928246	14	0.006199740255298029	15	0.020464849307538696	16	0.007634314895204749
17	0.01191103607855998	18	0.008723121180626733	19	0.00969230598523152	20	0.007924568352754282
21	0.004400689145958627	22	0.009851861786912853	23	0.009834114345936906	24	0.004469389578262938
25	0.006654153124739182	26	0.004602580516372121	27	0.01691976513968672	28	0.011615897604790728
29	0.013593698550059654	30	0.012162422464080338	31	0.006450535779151547	32	0.022862245896540265
33	0.00951625617266443	34	0.004733472017786889	35	0.017858910327256942	36	0.009638195607805952
37	0.007096348088779035	38	0.007027748962807909	39	0.010238175569631473	40	0.00926902189782689
41	0.003971288486043121	42	0.009046221893745562	43	0.00979699831701972	44	0.0038530115654358318
45	0.00745136238444391	46	0.005554728755137127	47	0.011592913375255098	48	0.0037530989832055715
49	0.002480373034581318	50	0.0028115443154814864	51	0.00225010169037869	52	0.011055213958061064
53	0.017193662159040828	54	0.005439743159141144	55	0.0036271310294395467	56	0.023319656394608236
57	0.012506818244130522	58	0.004372200915639268	59	0.0015000000000000002	60	0.006531923724215526
61	0.005881379884213336	62	0.00441217931607407	63	0.010071530834092799	64	0.0077303429761124585
65	0.020351497381882036	66	0.007165155585854262	67	0.008753198361657279	68	0.011387690059778797
69	0.004214516124067494	70	0.016639789890356183	71	0.013704754501101729	72	0.0058655910173591175
73	0.015967880272520565	74	0.0026990589930586577	75	0.008463970783477782	76	0.0015000000000000002
77	0.012023108319517736	78	0.017399023107687803	79	0.011636677207925463	80	0.009729849234956277
81	0.010203182597719306	82	0.010612540785639406	83	0.010042125695777402	84	0.010869702320298998
85	0.007846498069982907	86	0.00234389239908853	87	0.004721633065288984	88	0.0034795815765720555
89	0.004719089554232328	90	0.01926550473543543	91	0.006025389221909258	92	0.008232666089813058
93	0.0038120901649104973	94	0.022103944263633237	95	0.019769983340580267	96	0.025110723707672998
97	0.014468057345994625	98	0.012203110381598423	99	0.01855546561325105	100	0.009260623239104588

TABLE I: Data used while studying the time difference between MR JOB and Spark

1	0.0070105174317468665	26	0.01025984895949025	51	0.00517204433754814	76	0.010556763779704388
2	0.006544078926434265	27	0.00380266364248512	52	0.010816441125056937	77	0.0037086783467675235
3	0.00544985705021603	28	0.017603648641295818	53	0.008605669166020215	78	0.009989402575614762
4	0.00258358381328304	29	0.02259998806526656	54	0.012155445621585133	79	0.003217272852496386
5	0.009315493586439908	30	0.008759634029949406	55	0.003803087043247294	80	0.006492448139681076
6	0.00258358381328304	31	0.014450270575298692	56	0.020743533901894737	81	0.010718486874938695
7	0.017446861612399375	32	0.0115461296762201	57	0.00258358381328304	82	0.008085271753590528
8	0.01644063865883611	33	0.006933081224292391	58	0.004422036504348783	83	0.01273788889453495
9	0.011245990459912427	34	0.014642173056998595	59	0.011105470095468868	84	0.021864898869480044
10	0.016775194681284175	35	0.008305476731651412	60	0.007090717967547384	85	0.019168719830029962
11	0.0092654192369833	36	0.008263736889644985	61	0.0114026088147884	86	0.011247889638999057
12	0.00503308575139082	37	0.008028342217225428	62	0.010407006207596626	87	0.00807786545403833
13	0.0037279310602492035	38	0.012549538484985518	63	0.005069081649100747	88	0.0062404507792679096
14	0.011315380831323016	39	0.005995746055224115	64	0.003619865256464753	89	0.010553217527074723
15	0.010416267921860872	40	0.005585484323746589	65	0.007163209160978382	90	0.009985671817730189
16	0.0037279310602492035	41	0.010260924413052106	66	0.008329950381752105	91	0.00959569338374858
17	0.005204665532236686	42	0.012148623753548712	67	0.00851609696618031	92	0.010728828842710158
18	0.015185434977860093	43	0.0030362188412925727	68	0.026404863976352277	93	0.009522921825553747
19	0.00258358381328304	44	0.007419220747825282	69	0.007658286561248744	94	0.02043440359625807
20	0.0031325726259694552	45	0.010848928888454918	70	0.00895309898756113	95	0.003816225571841872
21	0.0330270284886357	46	0.008489115660861897	71	0.011474671621379998	96	0.0179950677425297
22	0.007315507054486552	47	0.008532825373525173	72	0.01050312310294058	97	0.012020323778645129
23	0.005676543610291163	48	0.007179544197253395	73	0.013400780460429654	98	0.01071832119731175
24	0.014837708257794336	49	0.015901708767685787	74	0.009888396705231646	99	0.010355517548696883
25	0.00834930997956929	50	0.006574521264888836	75	0.018781621035071337	100	0.010185514197496743

TABLE II: Data used while implementing for varying iterations

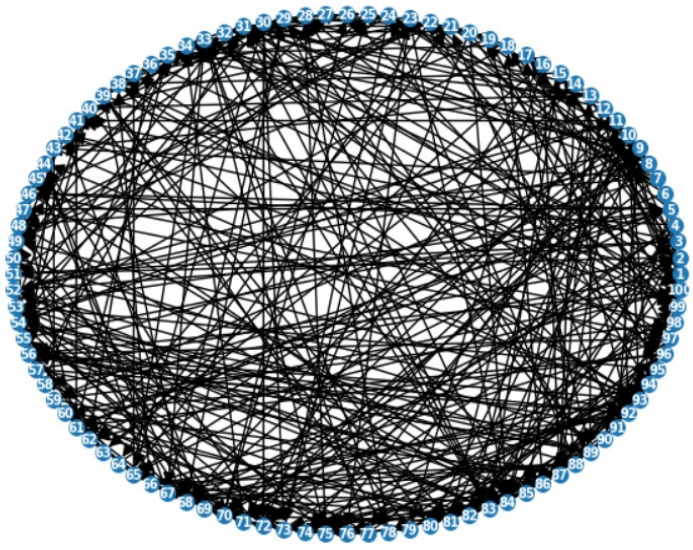

FIG. 4: Graph used while implementing for varying iterations

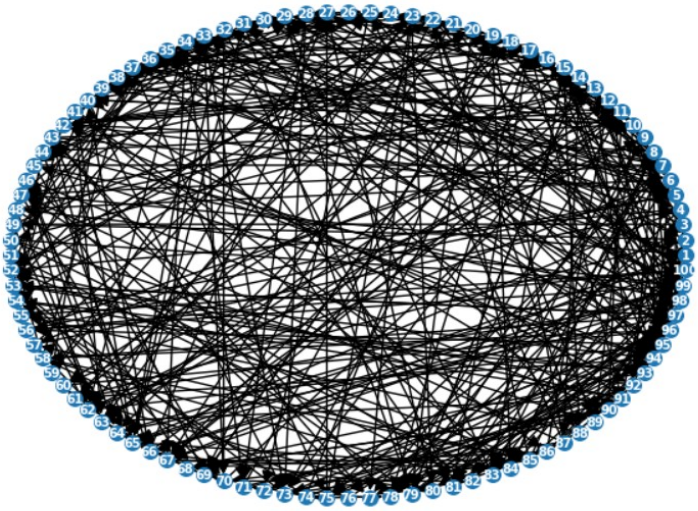

FIG. 5: Graph used while studying the time difference between MR JOB and Spark

5. RESULTS

A. Pagerank results with varying damping factors

As we can observe from the above figures that when the damping factor was 0, all nodes had equal pagerank scores, as $\alpha = 1$ it only consider the random jump factor.

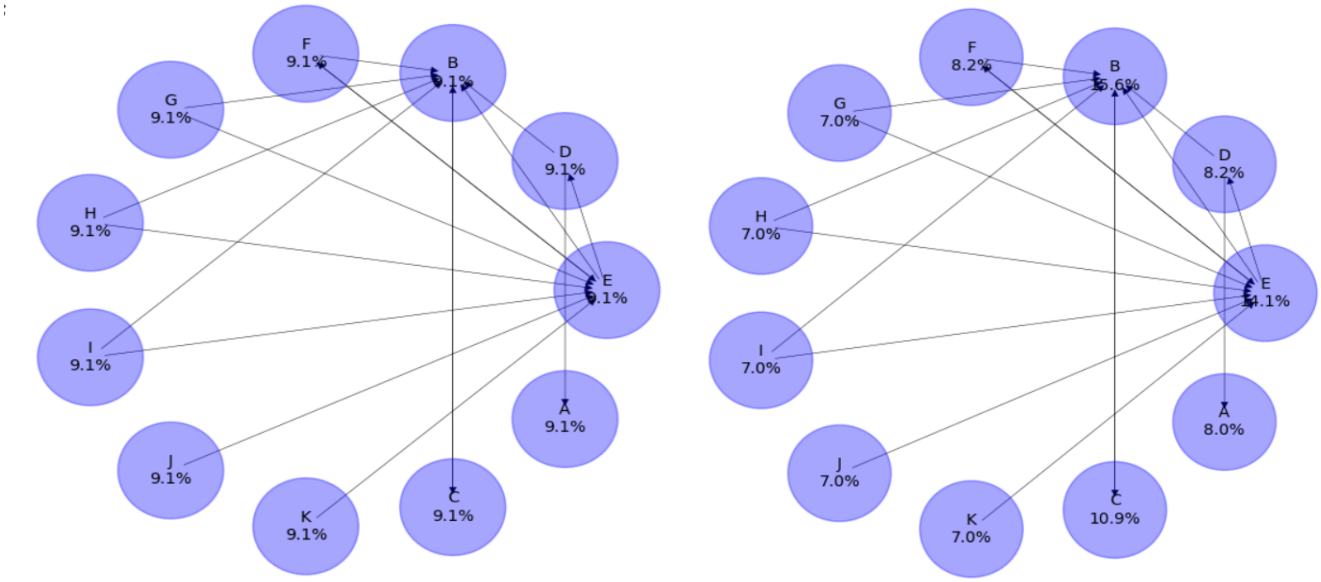

FIG. 6: Damping factor = 0 (left), Damping factor = 0.25 (right)

As the damping factor is increasing the weightage of the random jump factor is decreasing, whereas it is increasing for the random walk factor. Hence we can see an increase in the nodes with no outgoing edges (dangling nodes) and others converging to 0 as the damping factor is increasing.

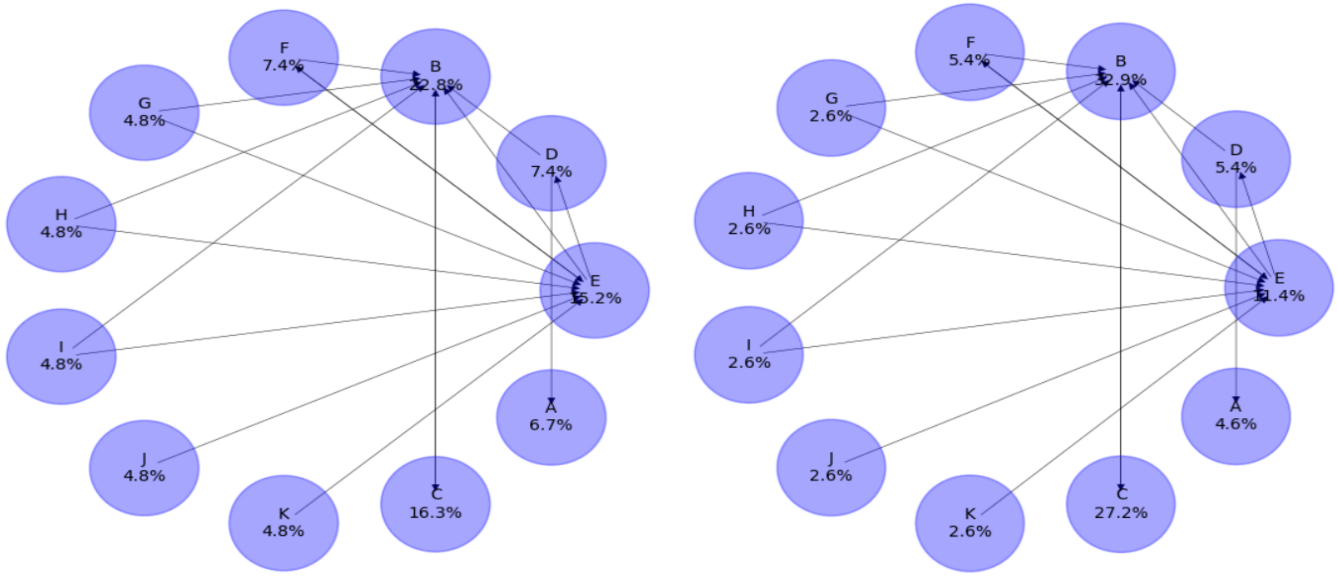

FIG. 7: Damping factor = 0.5 (left), Damping factor = 0.75 (right)

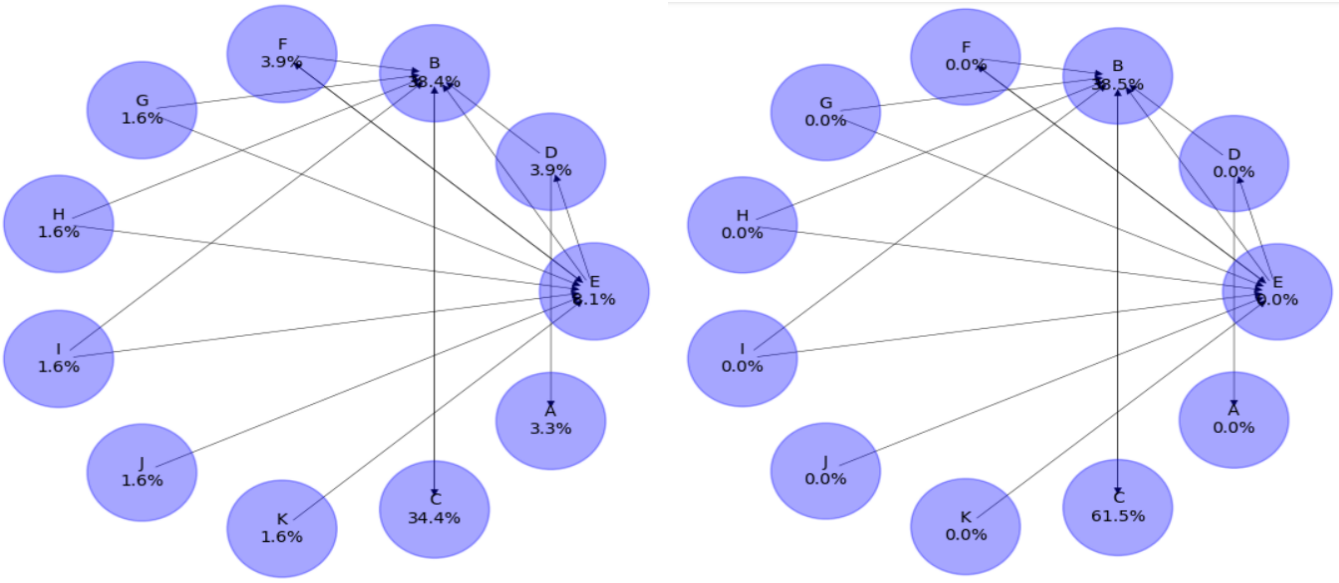

FIG. 8: Damping factor = 0.85 (left), Damping factor = 1 (right)

B. Pagerank results with varying iterations

When we implemented graph shown in Fig 1, we can see that the pagerank scores are converging towards the true pagerank scores (got from inbuilt pagerank function) as the iterations are increasing. We observed that after 5 iterations the values converged till 3 decimal places, but after 10 iterations we got an accuracy upto 6 decimal places which is highly accurate. As the iterations are increasing the convergence rate is falling. For 100 nodes graph we have got a highly accurate score at the 10th iteration , which is sufficient to rank the pages. Hence brute force is much better to calculate the pagerank scores when considering time complexity

C. Comparative study between Spark and MR Job

Here we have used the web graph in Fig 2 to implement pagerank algorithm in both spark and MR Job, we can observe that time taken in spark is much lesser than the time taken in MR Job. For MR Job the time taken to get the pagerank scores is 197.93 seconds whereas in Spark the time taken is 24.

6. CONCLUSIONS

This investigation delves into the application of the PageRank algorithm within the domain of distributed computing, utilizing the inherent Markovian properties of random walks. The implementation leverages map-reduce and Spark frameworks for scalable computations. The study explores the formal relationship between Markov chains and the PageRank algorithm, emphasizing the significance of nodes across diverse networks.

Experimental efforts involve the manipulation of damping factors and iterations to observe their impact on PageRank scores. The findings indicate that a higher damping factor accentuates the influence of the graph's structure, resulting in more informed rankings. Furthermore, the study tracks the evolution of PageRank scores with varying iteration numbers, shedding light on the convergence properties of the algorithm.The PageRank algorithm showcased a convergence trend with an escalation in the number of iterations. However, an asymptotic plateau was identified where additional iterations yielded marginal impact on PageRank scores.

The implementation also addresses the challenge of dangling nodes through the redistribution of PageRank mass. Various methodologies are discussed to ensure the conservation of total PageRank mass during computation.

Finally, the report compares the execution times of the PageRank algorithm implemented using MR Job and Spark. The results illustrate that Spark surpasses MR Job in terms of efficiency, showcasing the advantages of Spark's in-memory processing and optimized data structures.

In conclusion, this study offers insights into the dynamic evolution and convergence behavior of PageRank scores under different computational scenarios. The efficient distributed computation of PageRanks in large-scale networks is deemed crucial for applications in web algorithms, data mining, and distributed computing.

References

1. [1] 1208.3071.pdf (arxiv.org)

2. [2] (PDF) Application of Markov Chain in the PageRank Algorithm (researchgate.net)

Link to Google Colab Notebooks of code written:

https://drive.google.com/drive/folders/1ZMwkATHgflMU-3-hHIrOb_bpG1et_Pjh?usp=sharing

Github

https://github.com/Parv-Agarwal/Pagerank-on-MR-Job-and-Spark.git