Link Prediction using Random Walks

By

Rishav Saigal

Shashank Shekhar

Faizaan Ahmed Khan

Swagarika Jaharlal Giri

UNDER THE GUIDANCE OF

Mr. Koushik Mallick

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND
ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Session 2015-2016

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

[Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal]

CANAL SOUTH ROAD, BELIAGHATA, KOLKATA-700015

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

TO WHOM IT MAY CONCERN

I hereby recommend that the Project entitled **Link Prediction using Random Walks** Prepared under my supervision by **Rishav Saigal** (Reg. No. 131170110059, Class Roll No.CSE/2013/078), **Shashank shekhar** (Reg. No. 131170110070, Class Roll No.CSE/2013/079), **Faizaan Ahmed Khan** (Reg. No. 131170110031, Class Roll No.CSE/2013/089), **Swagarika Jaharlal Giri** (Reg. No. 131170110116, Class Roll No.CSE/2014/B03) of B.Tech 7th Semester, may be accepted in partial fulfillment for the degree of **Bachelor of Technology in Computer Science & Engg.** under Maulana Abul Kalam Azad University of Technology (MAKAUT).

Project Supervisor

Department of Computer Science and Engineering

RCC Institute of Information Technology

Countersigned:

……

Head

Department of Computer Sc. & Engg,

RCC Institute of Information Technology

Kolkata – 700015.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

CERTIFICATE OF APPROVAL

The foregoing Project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

FINAL EXAMINATION FOR 1 _____

EVALUATION OF PROJECT 2 _____

(Signature of Examiners)

ACKNOWLEDGEMENT

We are using this opportunity to express our gratitude to everyone who supported us throughout the course of this project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project. We are highly indebted to our project guide, Mr. Koushik Mallick for his guidance and constant supervision as well as for providing necessary information regarding the project.

Rishav Saigal (CSE/2013/078)

Shashank Shekhar (CSE/2013/079)

Faizaan Ahmed Khan (CSE/2013/089)

Swagarika Jaharlal Giri (CSE/2014/B03)

Table of Contents

1. Introduction

Link Prediction as we say is to predict the formation of link between two nodes at a time t_k in the future. It attempts to uncover missing links and predict the emergence of future links in complex networks based on the available information. Through Link Prediction we can predict the likelihood of a future association between two nodes, knowing that there is no association between the nodes in the current state of the graph.

Link Prediction has applications in variety of fields such as Social Networks, Bioinformatics, E-commerce and Security Domain. For example, in Bioinformatics, Link Prediction can be used to find interactions between proteins, in E-commerce it can help build recommendation system feature on E-commerce websites such as Flipkart, Amazon and in the Security Domain it can assist in identifying hidden groups of terrorists or criminals.

Because of its broad applications in various domains, the study of link prediction has become a research hotspot. The key challenges of the link prediction problem are owing to the sparsity and huge size of the networks.
In the DBLP dataset, in the year 2000, the ratio of actual and possible link is as low as 2×10^{-5}. So, in a uniformly sampled dataset with one million training instances, we can expect only 20 positive instances. The ratio between the number of positive links and the number of possible links also slowly decreases over time, since the negative links grow quadratically whereas positive links grow only linearly with a new node.

Recently, a large number of approaches have been proposed to address this issue. The methods can be classified into two categories: Supervised methods and unsupervised methods. The supervised method for link prediction is identified as the state-of-the-art method, which predict the unobserved links by a binary classifier. However, the supervised methods, typically suffer from the so-called class imbalance and feature selection problem. Moreover, most classifiers are based on the class distribution of the training data, thus they could perform poorly in some datasets that do not meet the prior assumptions.

Instead, the unsupervised methods work in an agnostic way, thus they can naturally avoid this problem. In addition, unsupervised methods do not need to decide which node features and edge features to use for link prediction, thus they also avoid the feature selection problem. We focus on unsupervised methods for link prediction. The key point of the unsupervised methods for link pre-diction is to find an appropriate similarity measure between nodes of a graph. The widely applied methods to measure similarities between nodes of a graph are based on random walks on graph.

We are working on Random Walks and its variants to solve the link prediction problem and to a comparative study of them in the course of which we aim to develop an algorithm using Random Walks where we can optimize the problem and reach high level of precision in our prediction.

2. Review of Literature

2.1 Local random walk

We consider an undirected simple network G (V, E), where V is the set of nodes and E is the set of edges. Multiple edges and self-connections are not allowed. For each pair of nodes$(x, y) \in V$, There is a score, S_{xy}, which shows the score of similarity between two nodes. The higher the score, the greater the probability of new edges create. Random walk is a Markov chain that describes the sequence of nodes visited by a random walker. The key fraction of the algorithm is the transition probability matrix P.
$P_{xy} = a_{xy}/k_x$ present the probability that a random walker starting at node x will walk to y in the next trap,
Where a_{xy} equals 1 if node x and node y are connected, 0 otherwise, and k_x denotes the degree of node x. When set a random walker starting from node x, with the probability Π_{xy} (t) this walker locates at node y after t traps, Here is the formula

$$\vec{\pi}_x(t) = P^T \vec{\pi}_x(t-1) \quad \text{----------(1)}$$

Where $\vec{\pi}_x$ (0) is an N ×1 vector with the x^{th} element equal to 1 and other all equal to 0, and T is the matrix transpose. The initial resource is usually assigned according to the importance of the nodes. The initial resource of node x is set proportional to its degree k_x. Then, after normalization the similarity between the node x and node y is

$$S_{xy}^{LRW}(t) = \frac{k_x}{2|E|} . \pi_{xy}(t) + \frac{k_y}{2|E|} . \pi_{yx}(t) \quad \text{----------(2)}$$

Where |E| is the number of edges in the network. It is obvious that $S_{xy} = S_{yx}$.

2.2 Random walk with restart

This algorithm is also consider about a simple network G(V,E), There is also a transition probability matrix P, with $P_{xy} = a_{xy}/k_x$ present the probability that a random walker starting at node x will walk to y in the next trap, where a_{xy} equals 1 if node x and node y are connected, 0 otherwise, and k_x denotes the degree of node x. Differently, there is an additional restart probability, when a random walker starting from node x, who will iteratively move to a random neighbour with probability α and return to node x with probability 1 - α, and denoting by q_{xy} the probability the walker locates at node y when it come to a steady state, then the formula is

$$\vec{q}_x = \alpha P^T \vec{q}_x + (1-\alpha)\, \vec{\varepsilon}_x \quad \text{----------(3)}$$

Where $\vec{\varepsilon}_x$ is an N ×1 vector with the x^{th} element equal to 1 and other all equal to 0. We can get the solution straightforward, as

$$\vec{q}_x = (1-\alpha)(1-\alpha P^T)^{-1}\, \vec{\varepsilon}_x \quad \text{----------(4)}$$

Accordingly, the similarity between the node *x* and node *y* can be defined as

$$S_{xy}^{RWR} = q_{xy} + q_{yx} \quad \text{----------(5)}$$

2.3 Random walk with Resistance

We describe a random walk algorithm, named random walk with resistance (RWS), based on two key ideas:

1) Bias the random walker towards staying close to the starting point by adding a small amount of resistance on each edge of the network,

2) Introduce a condition to discourage the random walker from roaming into new territories via hub nodes, by adding additional resistance for a random walker to overcome when meeting a new node that she has never visited before

The algorithm can be best described as a modification to the simple random walk algorithm:

$$f_{v,ij}^{(k+1)} = \begin{cases} \max(0, q_{v,i}^{(k)} P_{i,j} - \epsilon), if\ q_{v,i}^{(k)} > 0; \\ \max\left(0, q_{v,i}^{(k)} P_{i,j} - \epsilon\right), if\ q_{v,i}^{(k)} = 0\ , max_t\left(q_{v,i}^{(k)} P_{i,j}\right) \geq \beta; \\ 0, otherwise. \end{cases} \text{----------(6)}$$

The probability of reaching node j at time point k + 1 is then calculated by adding up the probabilities to enter j from all paths, and re-normalized so that the probability vector sums to 1:

$$\psi_{v,j}^{(k+1)} = \sum_i f_{v,ij}^{(k+1)} / \sum_{ij} f_{v,ij}^{(k+1)} \text{----------(7)}$$

The algorithm first checks whether a destination node, j, is new to the random walker. If not, i.e., $q_{v,j}^{(k)} > 0$, the random walker will visit the node with probability $q_{v,i}^{(k)} P_{i,j} - \epsilon$. The probability will be set to 0 if it is smaller than 0.

If the node j is new to the random walker, then an additional parameter, β, is introduced to discourage the random walker from visiting the new node, unless there is at least one path for v to enter j with sufficiently large probability.

2.4 Common Neighbors

The common-neighbor's predictor captures the notion that two strangers who have a common friend may be introduced by that friend. This introduction has the effect of "closing a triangle" in the graph and feels like a common mechanism in real life. Newman has computed this quantity in the context of collaboration networks, verifying a positive correlation between the number of common neighbors of x and y at time t, and the probability that x and y will collaborate at some time after t.

It is represented as:

$$S_{xy}^{CN} = |\Gamma(x) \cap \Gamma(y)| \text{----------(8)}$$

2.5 Adamic Adar Score

It can be defined as a frequency weighted common neighbours score. This measure refines the simple counting of common features by weighting rarer features more heavily. The Adamic/Adar predictor formalizes the intuitive notion that rare features are more telling; documents that share the phrase "for example" are probably less similar than documents that share the phrase "clustering coefficient."

If "triangle closing" is a frequent mechanism by which new edges form in a social network, then for x and y to be introduced by a common friend z, person z will have to choose to introduce the pair ⟨x,y⟩ from (choose |Γ(z)| with 2) pairs of his friends; thus an unpopular person (someone with not a lot of friends) may be more likely to introduce a particular pair of his friends to each other.

$$Score(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\, r(z)|} \quad \text{----------(9)}$$

2.6 Algorithm Performance Evaluation Metric

To evaluate the algorithm's performance precision metric has been used. We focus on the top-L links and compare the precision value of different score matrices. Precision is defined as the no of relevant items to the number of selected items. To calculate the precision, all the non-existent links according to their similarity score metrices which are calculated by the algorithms are sorted in descending order. Then top-L links are taken. If there are l links successfully predicted then, Precision= l/L. Higher value of precision means higher prediction. The links at the top are most likely to exist.

3. Objective

We are working on Random Walks and its variants to solve the link prediction problem in the course of which we aim to develop an algorithm using Random Walks where we can optimize the problem and reach high level of precision in our prediction. In our work, we take a snapshot of a network at time t, and our aim is to predict the probability edges that will emerge in the network between t and the future time t`. We would like to predict what new relationship will create between t and t` for each node in the network. So, the problem can also be viewed as a link prediction or problem, where our goal is to suggest each user a list of people that the user is most likely to create new relationship with. We analyze the characteristics of several networks such as no of nodes, no of edges, degree distribution, etc. We divide the network into two sets one is training set and other one is testing set. We do some experiments with some score similarity metrics such as the Common Neighbors and Adamic Adar, and other random walk algorithms like Local Random Walk (LRW), Random Walk with Resistance (RWS), Random Walk with Restart (RWR). We focus on our proposed method AA-RWR and AA-RWS algorithm. We calculate correlation between predicted score and topological similarity score. We evaluate above algorithms in terms of prediction performance by using Precision metric. To compare the results of different algorithms we calculate mean average precision value.

4. System Design

We use Matlab software to implement this project. Matlab (matrix laboratory) is a multi-paradigm numerical computing environment. The datasets are taken from different online sources listed in the references.

5. Methodology for implementation (Proposed Algorithm)

5.1 Adamic Adar Random Walk with Restart

Adamic Adar is an unsupervised learning technique based on topological similarity where the nodes with rarer features are given more weight. It provides the score on the basis of the number of connections that a node is having with other nodes in a network, a node with less connections is given more value and the node with higher number of connection is given larger values. What happens here is, if two essays have a common string like "for example" then we call it less similar than two essays with an uncommon string like "atomic energy".

Suppose, in a given network of our college, if student x and y needs to be introduced by a common friend, where students {a, b, c} are the common friends in between x and y, such that student c is less popular having less no. of friends, the probability of x being introduced to y by c is higher than that of a and b.

Mathematically, it is represented as

$$Score(x,y) = \sum_{z\in\Gamma(x)\cap\Gamma(y)} \frac{1}{\log|r(z)|} \quad \text{---------(10)}$$

On the basis of Adamic Adar score, we formulate some improvements on Random Walk with Restart which needs a transition probability matrix to calculate the RWR score, which is simply defined as $P_{xy} = a_{xy} / k_x$, and related only to the degree of a node. Here we try to manipulate the probability matrix such that the Adamic Adar score is taken into consideration.

$$P_{xy} = 1 + \left(\frac{ScoreAA}{Max(ScoreAA)}\right) \times \frac{a_{xy}}{k_x} \quad \text{----- (11)}$$

Using this as the new Probability Score Matrix for RWR, we formulate a new algorithm AA-RWR

Algorithm: AA-RWR Computes the similarity matrix S

---

1. INPUT: adj (Adjacency Matrix), α;
2. OUTPUT: The similarity matrix S;
3. Initialize: The transition probability matrix: $P \leftarrow 0_{n\times n}$
4. Calculate the weight of each node in adj1 as $1/log_k$
5. ScoreAA = $adj \times adj1$
6. Calculate sum(adj), which is sum of each row of the adjacency matrix
7. Replicate all n columns by $1/sum(adj)$ and store in a matrix d
8. Evaluate a_{xy} / k_x as $Pvar = d \times adj$
9. $ScoreMatrix = 1 + \left(\frac{ScoreAA}{Max(ScoreAA)}\right) \times Pvar$;
10. $P \leftarrow ScoreMatrix$;
11. Normalize P to get NP;
12. $S = \alpha \times inverse\ (I_{n\times n} - (1-\alpha) \times NP)$;
13. return S;

In the above algorithm, we obtain the Adamic Adar implemented Transition matrix from line 4th- 10th and then Normalize it before running the random walk with restart algorithm on line 12th, to finally obtain the similarity matrix S.

5.2 Adamic Adar Random Walk with Resistance

Random walk with Resistance takes the degree as similarity score of nodes and performs random walk such that it is resists visiting any hub node which can lead it to another cluster in the network. We used Adamic Adar score as obtained in equation (11), as the probability transition matrix and perform random walk with resistance on it, applying this on real dataset gave us higher accuracy in our similarity matrix.

Algorithm: AA-RWS Computes the similarity matrix S

---

1. INPUT: adj (Adjacency Matrix), α;
2. OUTPUT: The similarity matrix S;
3. Initialize: The transition probability matrix: $P \leftarrow 0_{n\times n}$
4. Calculate the weight of each node in adj1 as $1/log_k$
5. ScoreAA = $adj \times adj1$
6. Calculate sum(adj), which is sum of each row of the adjacency matrix
7. Replicate all n columns by $1/sum(adj)$ and store in a matrix d
8. Evaluate $a_{xy} \,/\, k_x$ as $Pvar = d \times adj$
9. $ScoreMatrix = 1 + \left(\frac{ScoreAA}{Max(ScoreAA)}\right) \times Pvar$;
10. $P \leftarrow ScoreMatrix$;
11. Normalize P to get NP;
12. Now we calculate the RWS score S as

$$S = \begin{cases} \max(0, q_{v,i}^{(k)} P_{i,j} - \epsilon), if\ q_{v,i}^{(k)} > 0; \\ \max\left(0, q_{v,i}^{(k)} P_{i,j} - \epsilon\right), if\ q_{v,i}^{(k)} = 0\ , max_t\left(q_{v,i}^{(k)} P_{i,j}\right) \geq \beta; \\ 0, otherwise. \end{cases}$$

13. return S;

In the above algorithm, we obtain the Adamic Adar Score implemented transition matrix from the 4th to 10th line and then we Normalize it, we run the random walk with resistance in the 12th line of the algorithm which is discussed in our literature survey, and finally we obtain the similarity matrix S.

We also considered Common Neighborhood Score instead of Adamic Adar score in the above algorithms and the formulations to get the algorithms CN-RWR, and CN-RWS for our comparative study.

6. Implementation Details

6.1 Datasets

In the following, we report on the main features of our data sets and we also analyze the networks later. In this project, we collect four social networks and the weight and the direction of the edges are ignored:

(i) The Dolphin network: A network representing the dolphin's social interactions.
(ii) The Football network: A network of football teams in an American college.
(iii) The Facebook network: a snapshot of the Facebook social network where nodes represent people and edges represent friendship or links.
(iv) The Homo-sapiens network, which is a snapshot of Protein to Protein interaction network in homo sapiens.

We collect some basic information about the networks as summarized below, where E represents the no. of edges, N represents the no. of nodes, Max D represents the Maximum Degree of the network and Avg D represents the Average degree of the network, and ρ represents the density of the network.

Dataset	**N**	**E**	**Max D**	**Avg D**	**P**
Dolphin	62	159	12	5.1290	0.0841
Football	115	613	12	10.6609	0.0935
Facebook	115	613	12	10.6609	0.0935
Homo sapiens	3890	38739	594	19.6874	0.0051

From the above table, we can infer that the Homo Sapiens PPI network is large network with very less density, and have high lower degree nodes and are more sparse than the others.

Following figures show the degree distribution (no of degree) of these networks where x axis represents the no of degree of nodes and the y axis represents the no of nodes of the network.

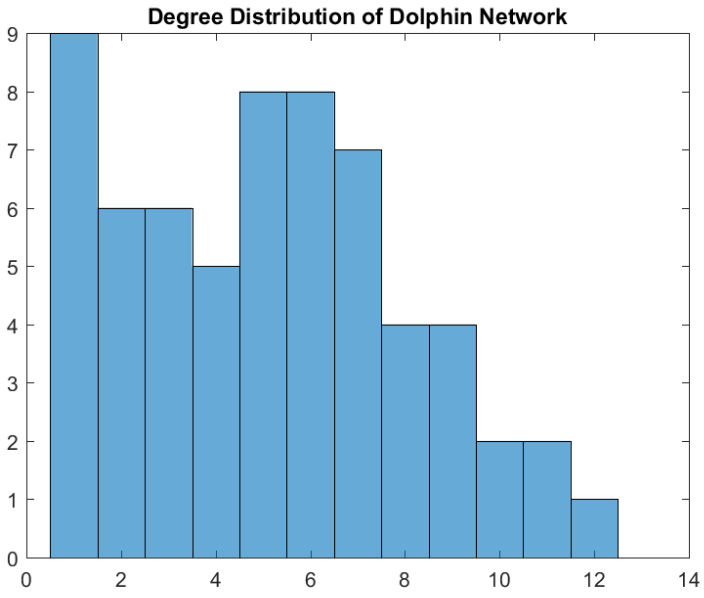

Fig 1: Degree Distribution of Dolphin Network

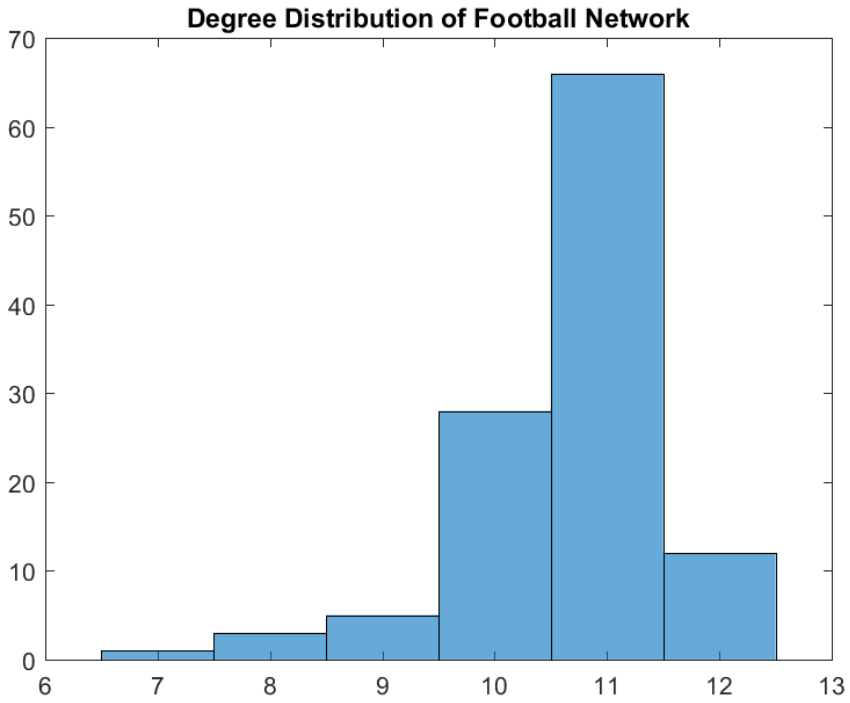

Fig 2: Degree Distribution of Football Network

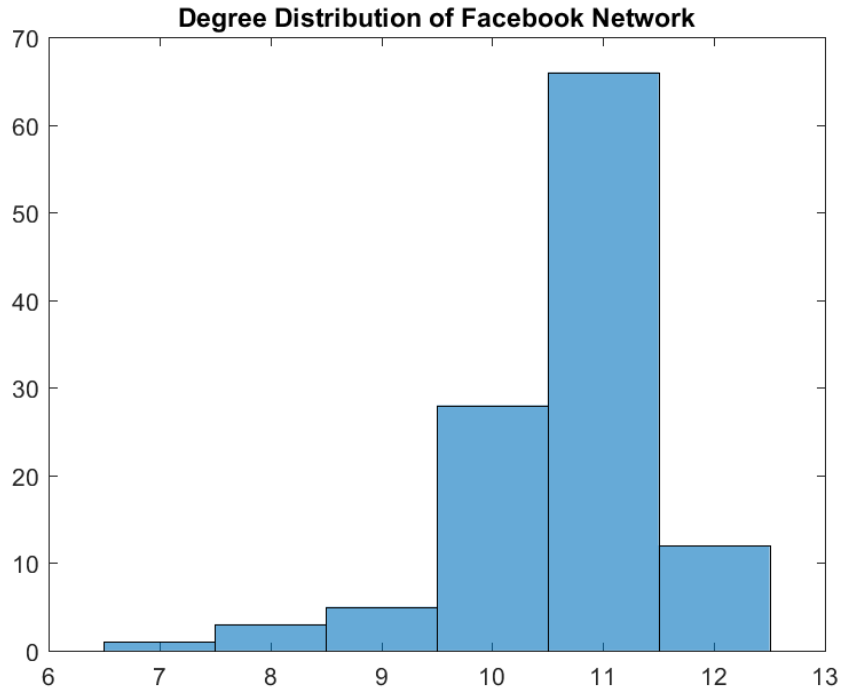

Fig 3: Degree Distribution of Facebook Network

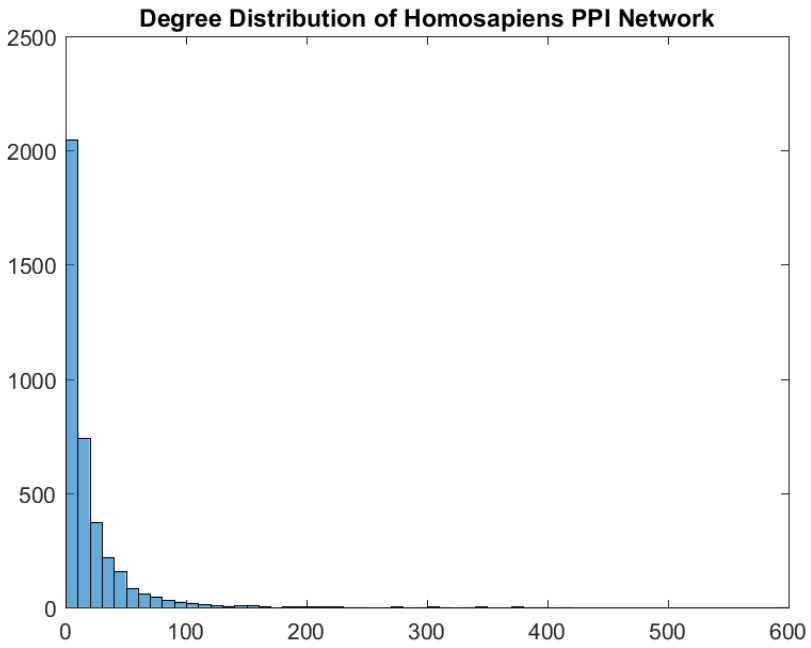

Fig 4: Degree Distribution of Homo sapiens network

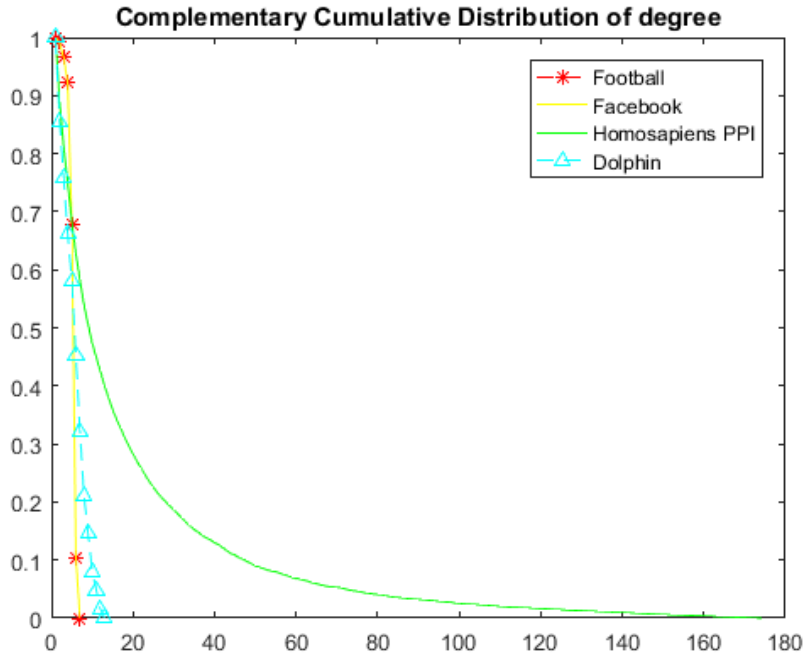

Fig 5: Complementary cumulative conditional distribution of the degree

6.2 Implementation

We take dolphin, Facebook, football and homo-sapiens networks and create adjacency matrix of these networks. From the adjacency matrix remove 25% of the links and create adjacency train matrix. We take football network as an example that, in football network we have total 613 links in the original adjacency matrix. Then we divide the original dataset into training and test datasets. We do 4-fold cross validation of the connected links from each row of the original adjacency matrix and store the indices of these connected links in a vector. Then we get test set contains only 1-fold and training set contains remaining folds.

Now the train data is passed to our algorithm as an adjacency matrix, and similarity score matrix is the output from there. The output is first made similar and then we check for the no. of common values in the top K list only to calculate the precision finally. If K value is taken as a nearby value with the average degree of the network it will provide better results.

7. Result Analysis

We summarize the results of all algorithm in all datasets we have used, we present the values as well as the graphs for the same here for proper understanding and easy of analyzing.

7.1 Dolphin Network

DOLPHIN NETWORK (TOP 5 AVERAGE PRECISION VALUE)							
LRW (μ=0.9)	**CN-LRW (μ=0.9)**	**RWR (α=0.9)**	**CN-RWR (α=0.9)**	**AA-RWR (α=0.9)**	**RWS (β=0.002, ε=0.002)**	**CN-RWS**	**AA-RWS**
0.59032	0.07097	0.62194	0.62258	**0.64194**	0.6016	0.6387	0.6387

DOLPHIN NETWORK (TOP 10 AVERAGE PRECISION VALUE)							
LRW (μ=0.9)	**CN-LRW (μ=0.9)**	**RWR (α=0.9)**	**CN-RWR (α=0.9)**	**AA-RWR (α=0.9)**	**RWS (β=0.002, ε=0.002)**	**CN-RWS**	**AA-RWS**
0.44516	0.08065	0.47226	0.46774	**0.47581**	0.4155	0.4677	0.4694

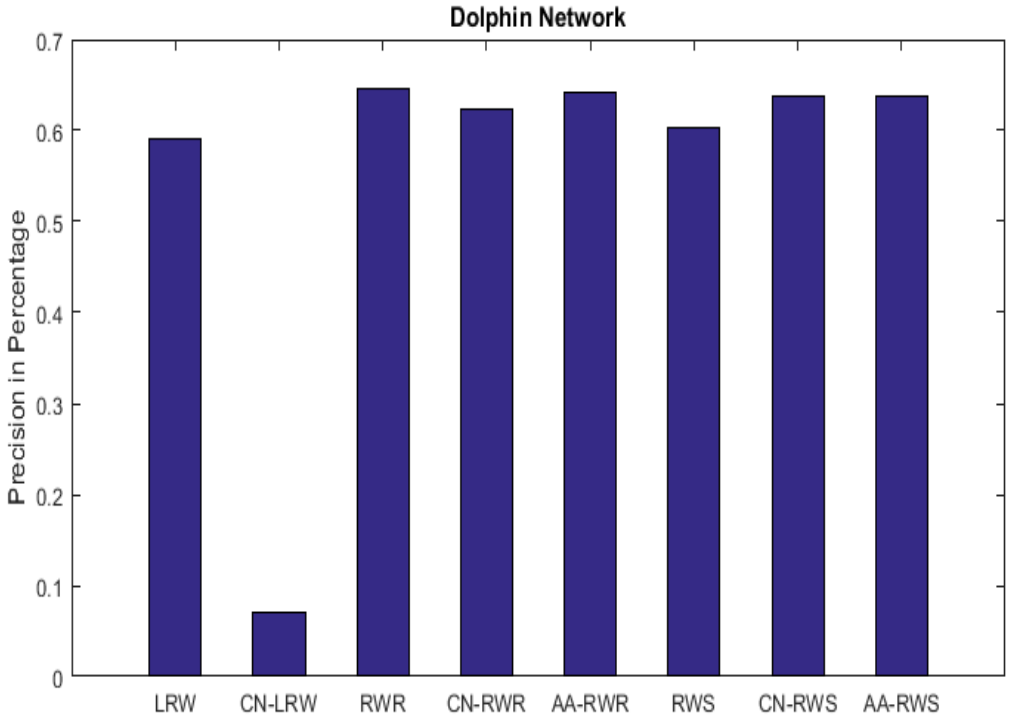

Fig 6: Top 5 Average Precision Value

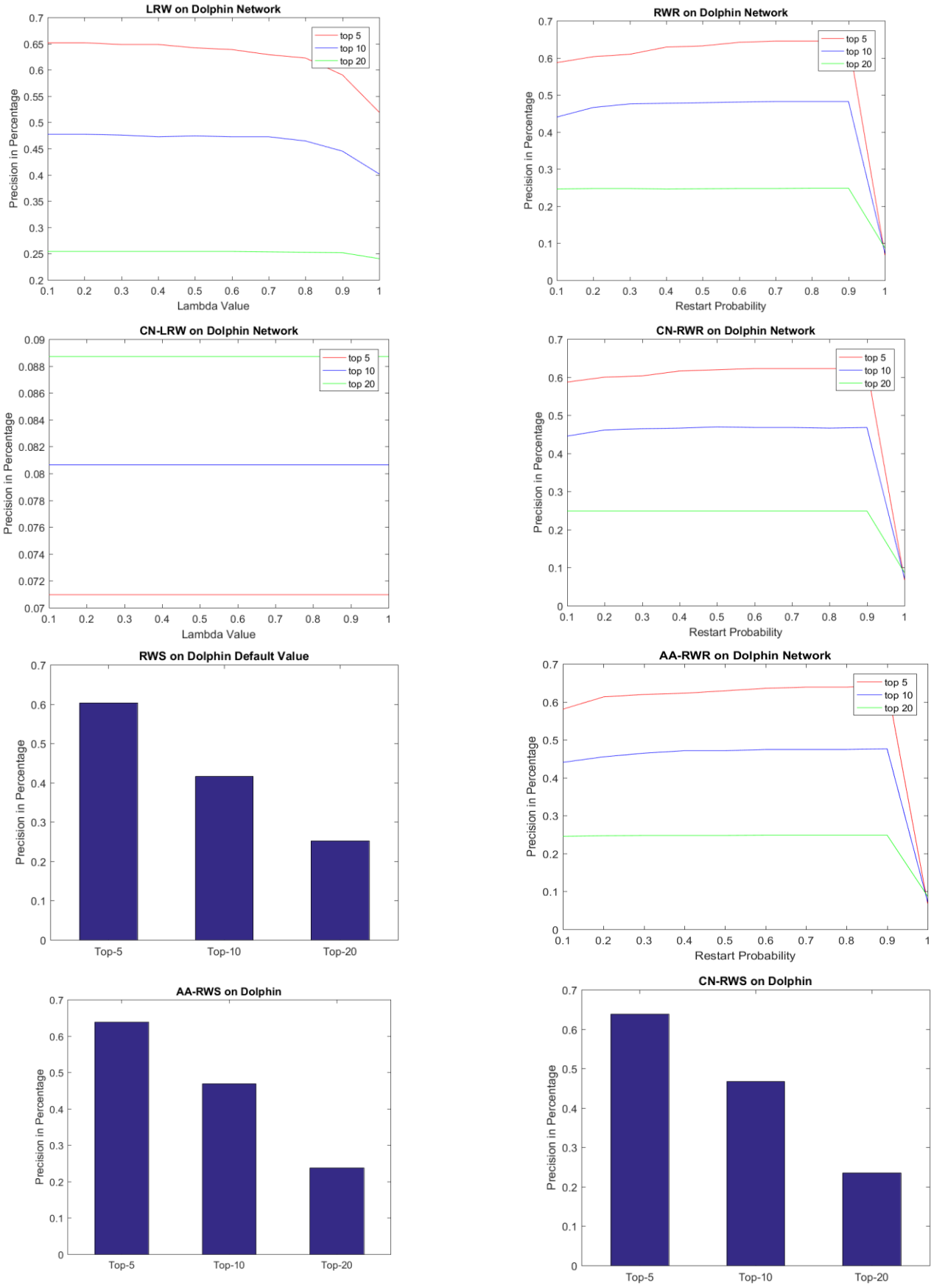

Fig: 7-14 Shows the change of Precision as per the change of different parameters in dolphin network

7.2 Football Network

FOOTBALL NETWORK (TOP 5 AVERAGE PRECISION VALUE)							
LRW (μ=0.9)	**CN-LRW (μ=0.9)**	**RWR (α=0.9)**	**CN-RWR (α=0.9)**	**AA-RWR (α=0.9)**	**RWS (β=0.002, ε=0.002)**	**CN-RWS**	**AA-RWS**
0.79652	0.78609	0.8	0.8	0.8	0.69	0.8	0.8
FOOTBALL NETWORK (TOP 10 AVERAGE PRECISION VALUE)							
LRW (μ=0.9)	**CN-LRW (μ=0.9)**	**RWR (α=0.9)**	**CN-RWR (α=0.9)**	**AA-RWR (α=0.9)**	**RWS (β=0.002, ε=0.002)**	**CN-RWS**	**AA-RWS**
0.8313	0.89217	0.88957	**0.89217**	0.88957	0.72	0.887	0.8887

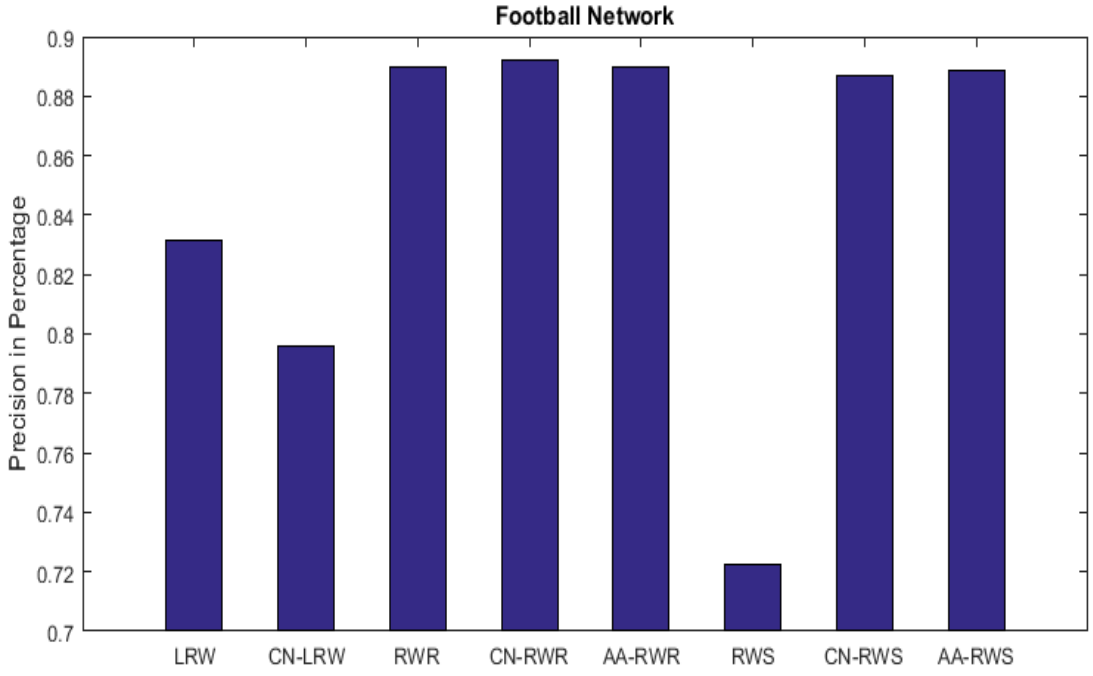

Fig15: Top 10 average precision value of Football Network.

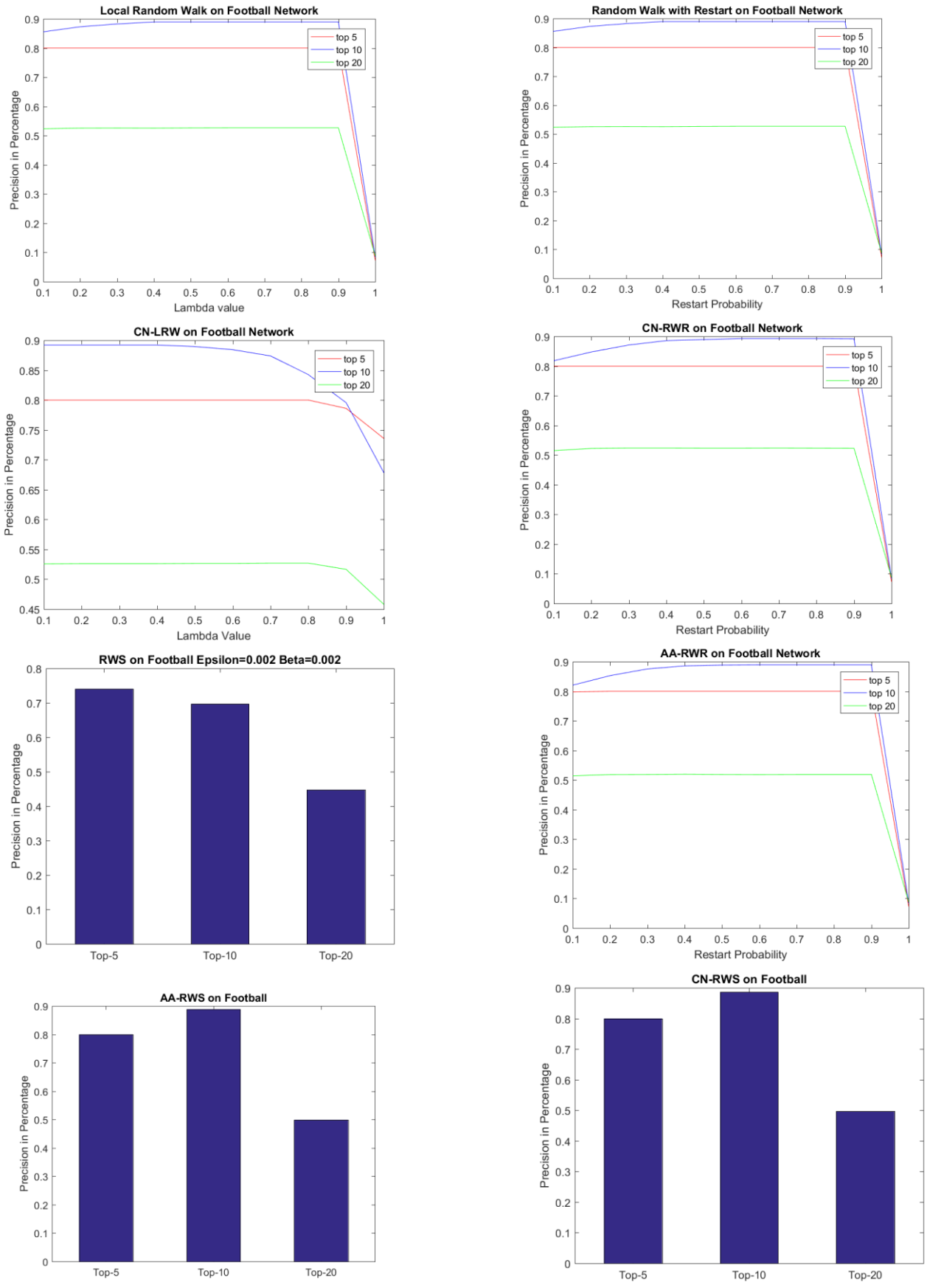

Fig 16-23**:** Shows the change of Precision as per the change of different parameters in football network

7.3 Facebook Network

FACEBOOK NETWORK (TOP 5 AVERAGE PRECISION VALUE)							
LRW (μ=0.9)	**CN-LRW (μ=0.9)**	**RWR (α=0.9)**	**CN-RWR (α=0.9)**	**AA-RWR (α=0.9)**	**RWS (β=0.002, ε=0.002)**	**CN-RWS**	**AA-RWS**
0.8	0.79826	0.8	0.8	**0.88957**	0.7565	0.8	**0.82**
FACEBOOK NETWORK (TOP 10 AVERAGE PRECISION VALUE)							
LRW (μ=0.9)	**CN-LRW (μ=0.9)**	**RWR (α=0.9)**	**CN-RWR (α=0.9)**	**AA-RWR (α=0.9)**	**RWS (β=0.002, ε=0.002)**	**CN-RWS**	**AA-RWS**
0.8365	0.79739	0.88	0.89304	**0.9061**	0.7122	0.8904	**0.9061**

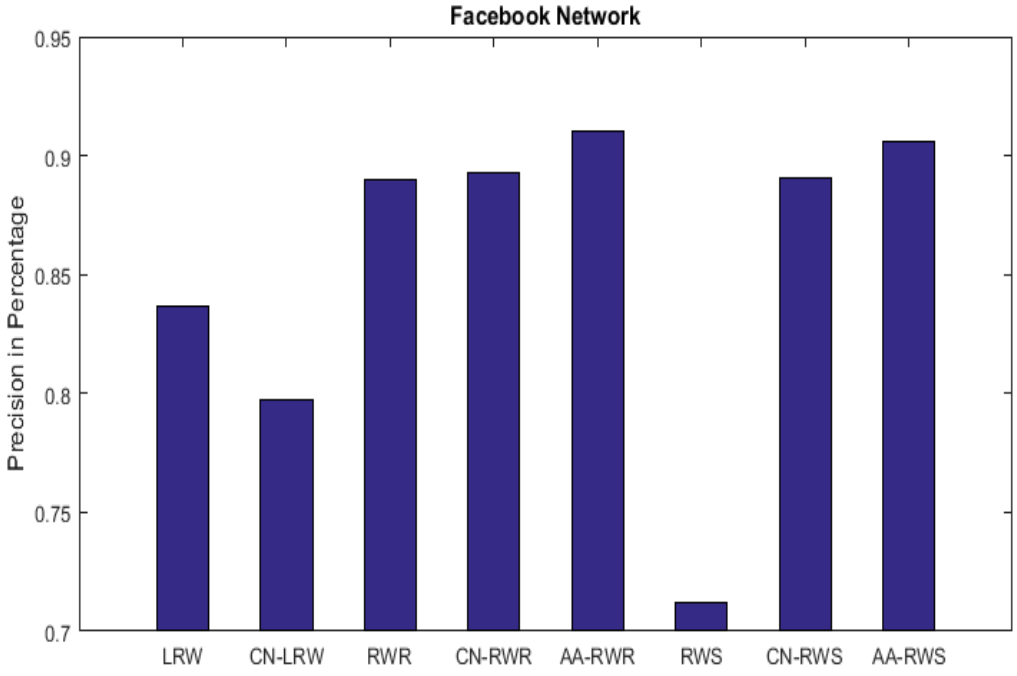

Fig 24: Top 10 average precision value of Facebook Network.

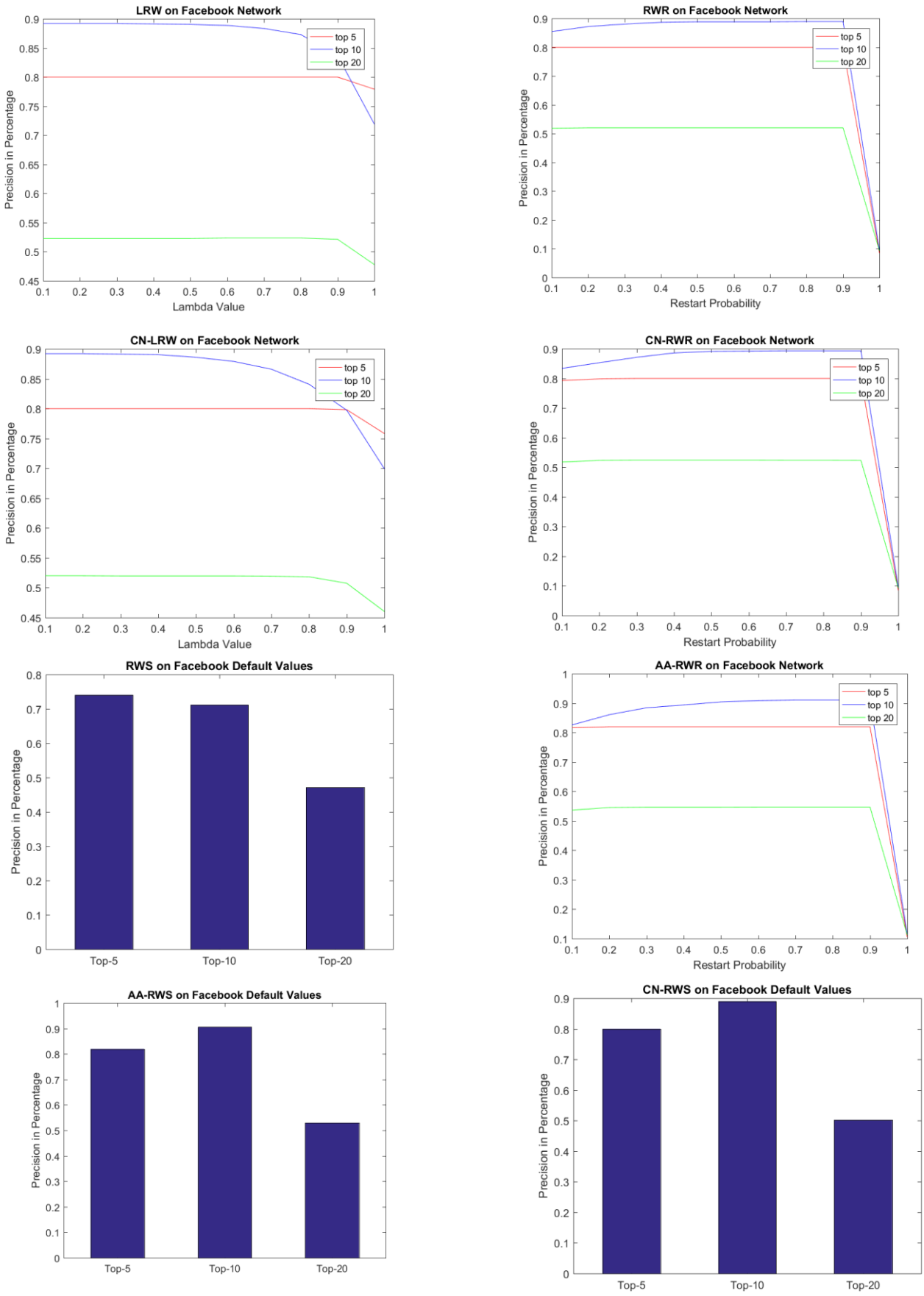

Fig 24-32**:** Shows the change of Precision as per the change of different parameters in Facebook network

7.4 Homo-sapiens PPI Network

PPI NETWORK (TOP 5 AVERAGE PRECISION VALUE)							
LRW (μ=0.9)	**CN-LRW (μ=0.9)**	**RWR (α=0.9)**	**CN-RWR (α=0.9)**	**AA-RWR (α=0.9)**	**RWS (β=0.002, ε=0.002)**	**CN-RWS**	**AA-RWS**
0.392134	0.00396	0.699332	0.711825	**0.72982**	0.1328	0.7159	**0.7351**
PPI NETWORK (TOP 10 AVERAGE PRECISION VALUE)							
LRW (μ=0.9)	**CN-LRW (μ=0.9)**	**RWR (α=0.9)**	**CN-RWR (α=0.9)**	**AA-RWR (α=0.9)**	**RWS (β=0.002, ε=0.002)**	**CN-RWS**	**AA-RWS**
0.291902	0.0022	0.62079	0.6349	**0.6531**	0.09	0.6385	**0.6354**

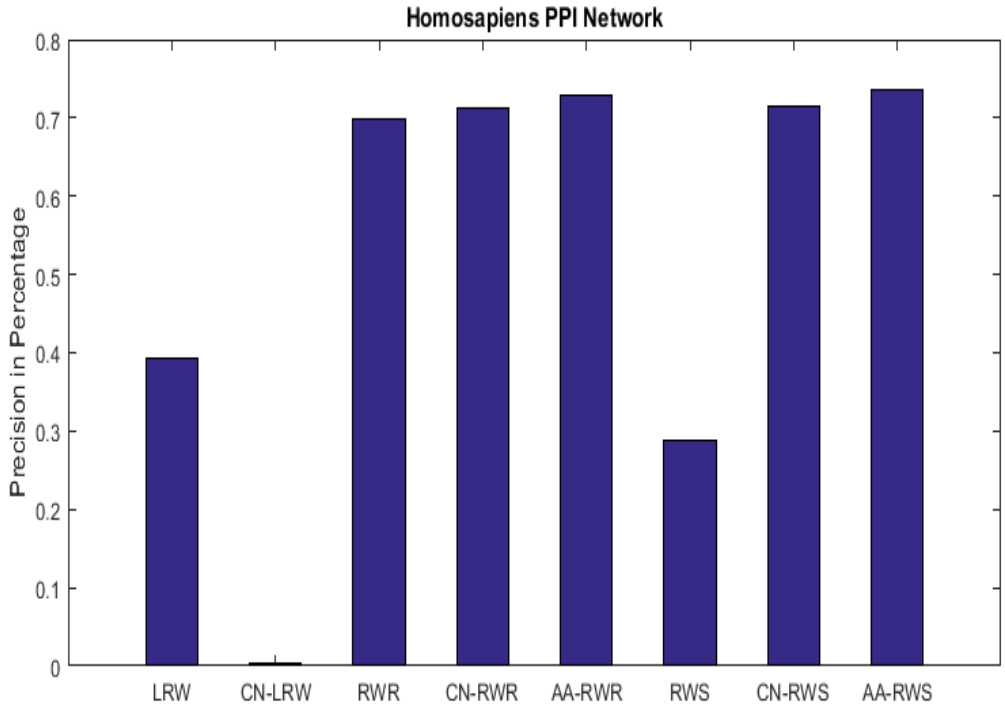

Fig 33: Top 5 average precision value for all algorithms in the PPI network

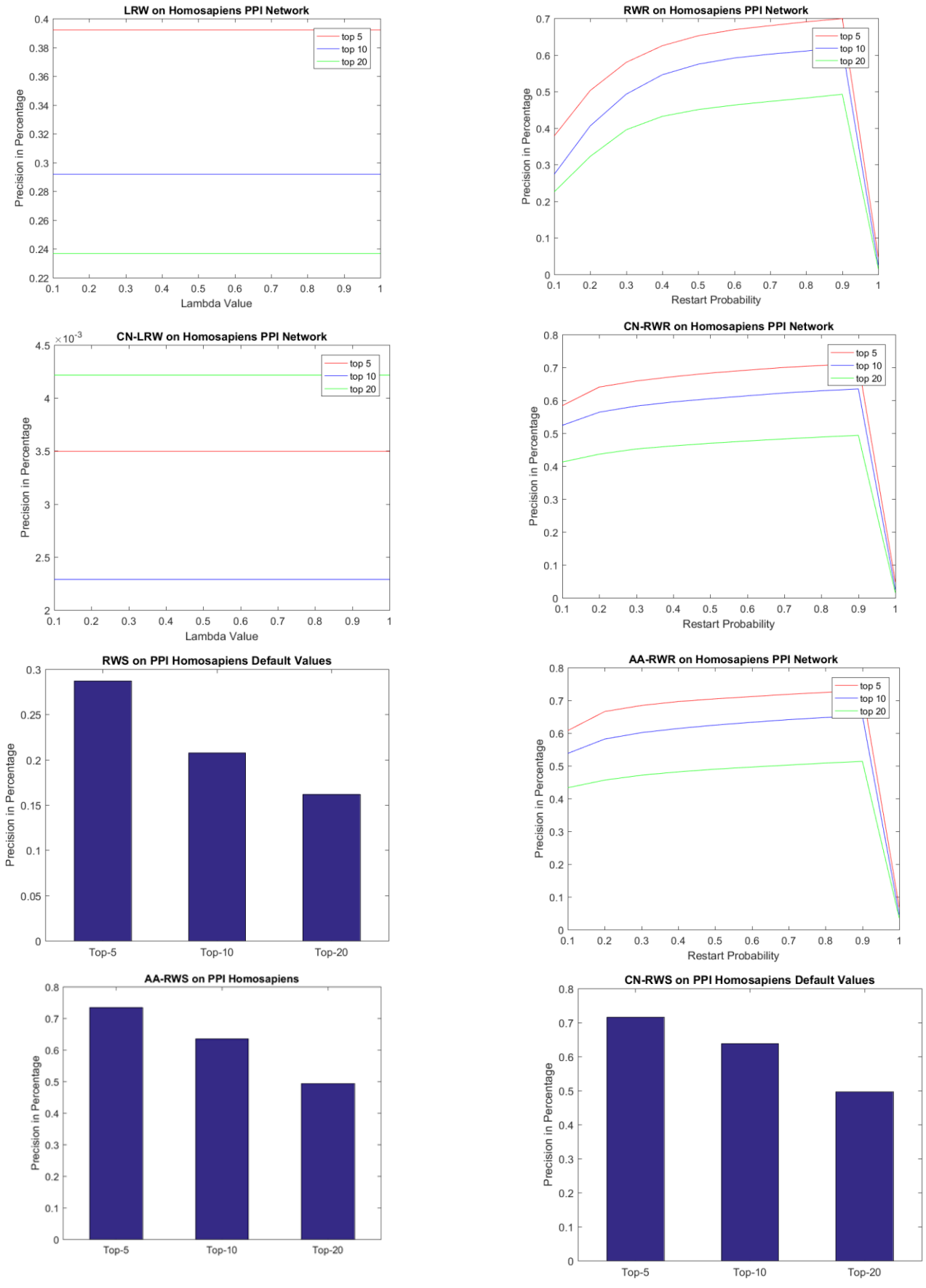

Fig 34-41**:** Shows the change of Precision as per the change of different parameters in Facebook network

8. Conclusion

In our project, we have presented two algorithms for link prediction using Random Walks, AA-RWR and AA-RWS. The experimental results show that when we are using these two the results are better than their parent algorithms, hence we can say that our algorithms can perform better in link prediction tasks. We analyzed the structure of the networks through the distribution of the degree of the nodes, and discussed how they influence the performance of the algorithms. We also did a comparative study amongst different kind of random walk algorithms namely LRW, CN-LRW, RWR, CN-RWR, AA-RWR, RWS, CN-RWS, AA-RWS.

In the future work, we plan to apply algorithms in larger network with more node attributes using big data.

Appendix 1: Source Code

1. Function to formulate train and test datasets

```
function [adjtst,adjtrn] = rowwisse(p)
[r,c]=size(p);
adjtrn=sparse(zeros(r,c));
adjtst=sparse(zeros(r,c));
d=sum(p,2);

  for i =1:length(d)
   if (d(i)<=4)
      adjtrn(i,:)=p(i,:);
      %adjtst(i,:)=p(i,:);
      continue;
   end
   ind=crossvalind('Kfold',nnz(p(i,:)),4);
   [x,y,z]=find(p(i,:));
   adjtrn(i,y(ind~=4))=1;
   adjtst(i,y(ind==4))=1;
  end
```

2. Function to find the LRW Similarity Score

```
function [lrwScore ] = LRW( train, steps, lambda)
%% Calculate the LRW
    deg = repmat(sum(train,2),[1,size(train,2)]);
    train = train ./ deg; clear deg;
    % Seeking a transfer matrix
    I = sparse(eye(size(train,1)));
    % Generate the unit matrix
    sim = I;
    stepi = 0;
    while(stepi < steps)
    % Random walk of the iteration
        sim = (1-lambda)*I + lambda * train' * sim;
        stepi = stepi + 1;
        stepi
    end
    lrwScore=sim;
end
```

3. Function to find the RWR Similarity Score

```
function P = convertToProbabilityMatrix(G)
    denom=sum(G);
    denom(denom==0) = 1;
    P=G*diag(1./denom);
End

function R = convertToRandomWalkWithRestart(G,c)

    P = convertToProbabilityMatrix(G);

    R = c*inv(eye(size(P)) - (1-c)*P);
end
```

4. Function to find the RWS Similarity Score

```
function current = RWS(a,epsilon,beta) %a is the network.
nodes=size(a,1);

if (nargin < 4)
    stop_value = 1/mean(sum(a));
end
if (nargin < 3)
    beta = 1/mean(sum(a))/nodes;
end
if (nargin < 2)
    epsilon = 1/sum(sum(a))/mean(sum(a));
end

[m,n]=size(a);
aa=max(a,eye(size(a)));%%%%%%%%%%%%%%%%%%%%%%%%%pass to self
bb=sum(aa);
bb=1./bb;
cc=ones(m,1)*bb.*aa;
cc=cc';

cc(find(isnan(cc))) = 0;

current=eye(m,n);
newcurrent=current;

active=ones(m,1);

step=0;
improve_count=0;
active_nodes=0;

     improve=ones(n,1);
    while sum(active),
        [step sum(active) improve_count]

         active_nodes=sum(active);
        current=newcurrent;

        curr_act=active*ones(1,m).*current;
        curr_stop=(active*ones(1,m)-1).*current;

temp=curr_act*cc-((((current*cc>0)-(current>0))==1).*( (curr_act>0)*aa
).*beta);
        temp=temp.*(temp>0);
        newcurrent=temp;

        newcurrent=newcurrent.*(newcurrent>(beta*cc));


        newcurrent=newcurrent-(current>0)*aa.*epsilon;
        newcurrent=newcurrent.*(newcurrent>0);


        newcurrent=newcurrent-curr_stop;
```

```
newcurrent=(1./sum(newcurrent'))'*ones(1,nodes).*newcurrent;%normalize


        improve=sum(abs(current-newcurrent)')';
        active=(improve>stop_value);

        if active_nodes==sum(active)
            improve_count=improve_count+1;
        else improve_count=0;
        end
    step=step+1;



        if improve_count>500
            break
        end

    end
```

5. Function to find the CN Score Transition Matrix

```
function scorecn=cn(adj)
[row,col]=size(adj);
scorecn = sparse(row,col);
degree = sum(adj,2);

scoreCN=adj*adj;

maxCN = max(scoreCN(:));

degree=sum(adj);
degree=1./degree;
degree=repmat(degree',1,col);
degree=degree.*adj;
scorecn=(1+scoreCN/maxCN).*degree;
```

6. Function to find the AA Score Transition Matrix

```
function scoreAA= AANew(train)

[row,col]=size(train);
train1 = train ./ repmat(log(sum(train,2)),[1,size(train,1)]);
% Calculate the weight of each node, 1 / log (k_i)
train1(isnan(train1)) = 0;
train1(isinf(train1)) = 0;
% Set the exception value obtained by dividing the divisor by 0 to 0

sim = train * train1;   clear train1;
% To achieve the calculation of similarity matrix

maxCN = max(sim(:));
```

```
degree=sum(train);
degree=1./degree;
degree=repmat(degree',1,col);
degree=degree.*train;
scoreAA=(1+sim/maxCN).*degree;
 scoreAA(isnan(scoreAA)) = 0;
 scoreAA(isinf(scoreAA)) = 0;
```

7. Function to make the matrix symmetric

```
function score1=dosym(adj,score)
%adj is adjmatrix
% score is calculated
[ro,col]=size(adj);
edge=nnz(adj);
degree=sum(adj,2);

coef=degree/edge;
coef=repmat(coef,1,col);
score1=score.*coef+(score.*coef)';
```

8. Function to compare for common values

```
function commn1 = commnval1(adj,lap,top)
[r,c]=size(adj);
indsadjtst=cell(0);
index=1:c;
for i = 1:length(adj)
   indsadjtst(i,:)={[find(adj(i,:))]};
   scr=lap(i,:);
   [x,d]=sort(scr,'descend');
   sind=index(d);
   indpos=find(adj(i,:)>0);
   commn1(i,:) = nnz(intersect(sind(1:top),indpos));
end
```

9. Function to calculate Precision

```
function precisn=findpre(value,top,n)
%for i=1:115
for i=1:n
    preval(i,:)=value(i,:)/top;

end
sumval=sum(preval);
precisn=sumval/length(value)
```

10. Driver script to run Local Random Walk

```
load('dolphn-netwrk.mat');
adj=full(adj);
[a,b]=rowwisse(adj);
A =full(a);
B =full(b);
Score=LRW(B,5,0.9);
ScoreS=dosym(B,Score);
CValue=commnval1(adj,ScoreS,10);
Precision = findpre(CValue,10,62);
```

11. Driver script to run Random Walk with Restart

```
load('dolphn-netwrk.mat');
adj=full(adj);
[a,b]=rowwisse(adj);
A =full(a);
B =full(b);
Score=convertToRandomWalkWithRestart(B,0.9);
ScoreS=dosym(B,Score);
CValue=commnval1(adj,ScoreS,10);
Precision = findpre(CValue,10,62);
```

12. Driver script to run Random Walk with Resistance

```
load('dolphn-netwrk.mat');
adj=full(adj);
[a,b]=rowwisse(adj);
A =full(a);
B =full(b);
Score=RWS(B,0.002,0.002);
ScoreS=dosym(B,Score);
CValue=commnval1(adj,ScoreS,10);
Precision = findpre(CValue,10,62);
```

13. Driver script to run CN- LRW

```
load('dolphn-netwrk.mat');
adj=full(adj);
[a,b]=rowwisse(adj);
A =full(a);
B =full(b);

CnScore=cn(B);
CnScore = full(CnScore);
NScore = spdiags (1./sum (CnScore,2), 0, size(CnScore,1), size(CnScore,1))
* CnScore ;
Score=LRW(B,5,0.9);

ScoreS=dosym(B,Score);
CValue=commnval1(adj,ScoreS,10);
Precision = findpre(CValue,10,62);
```

14. Driver script to run CN- RWR

```
load('dolphn-netwrk.mat');
adj=full(adj);
[a,b]=rowwisse(adj);
A =full(a);
B =full(b);

CnScore=cn(B);
CnScore = full(CnScore);
NScore = spdiags (1./sum (CnScore,2), 0, size(CnScore,1), size(CnScore,1))
* CnScore ;
Score=convertToRandomWalkWithRestart(NScore,0.9);
ScoreS=dosym(B,Score);
CValue=commnval1(adj,ScoreS,10);
Precision = findpre(CValue,10,62);
```

15. Driver script to run CN-RWS

```
load('dolphn-netwrk.mat');
adj=full(adj);
[a,b]=rowwisse(adj);
A =full(a);
B =full(b);

CnScore=cn(B);
CnScore = full(CnScore);
NScore = spdiags (1./sum (CnScore,2), 0, size(CnScore,1), size(CnScore,1))
* CnScore ;

Score=RWS(NScore,0.002,0.002);
ScoreS=dosym(B,Score);
CValue=commnval1(adj,ScoreS,10);
Precision = findpre(CValue,10,62);
```

16. Driver script to run AA-RWR

```
load('dolphn-netwrk.mat');
adj=full(adj);
[a,b]=rowwisse(adj);
A =full(a);
B =full(b);
scoreAA = AANew(B);
NScore = spdiags (1./sum (scoreAA,2), 0, size(scoreAA,1), size(scoreAA,1))
* scoreAA ;
Score=convertToRandomWalkWithRestart(NScore,0.9);
ScoreS=dosym(B,Score);
CValue=commnval1(adj,ScoreS,10);
Precision = findpre(CValue,10,62);
```

17. Driver script to run AA-RWS

```
load('dolphn-netwrk.mat');
adj=full(adj);
[a,b]=rowwisse(adj);
A =full(a);
B =full(b);
scoreAA = AANew(B);
NScore = spdiags (1./sum (scoreAA,2), 0, size(scoreAA,1), size(scoreAA,1))
* scoreAA ;
Score=RWS(scoreAA,0.002,0.002);
ScoreS=dosym(B,Score);
CValue=commnval1(adj,ScoreS,10);
Precision = findpre(CValue,10,62);
```

References

[1] Link Prediction Based on Random Walks⋆

Li LI∗, Weisi FENG, Chenyang JING, Feng TAN, Ping HE, Jing WANG Faculty of Computer and Information Science, Southwest University, Chongqing 400715, China

[2] A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity

Chengwei Lei 1 and Jianhua Ruan 1∗ 1Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249, USA

[3] Link prediction in complex network,

L. Linyuan, University of Electronic Science and Technology 39 (5) (2010) 652.

[4] Link prediction based on local random walk,

W. Liu, L. L¨u, EPL (Europhysics Letters) 89 (5)(2010) 58007.

[5] The Link Prediction Problem for Social Networks∗

David Liben-Nowell† Laboratory for Computer Science Massachusetts Institute of Technology Cambridge, MA, Jon Kleinberg Department of Computer Science Cornell University

[6] Discovering collective viewpoints on micro-blogging events based on community and temporal aspects, in: Advanced Data Mining and Applications

B. Zhao, Z. Zhang, Y. Gu, X. Gong, W. Qian, A. Zhou,

[7] Link Prediction in Social Networks: the State-of-the-Art ,

WANG Peng1,3∗ , XU BaoWen1,2,3† , WU YuRong1& ZHOU XiaoYu1

[8] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. Social Networks, 25(3):211–230, July 2003.

[9] Adamic, L.A., Adar, E., 2000. Frequency of Friendship Predictors.