AffinityRank: Fulfilling the Promise of TrustRank

Rohit Kaul
(rkaul@alumni.stanford.edu)

1. Introduction

The significance of network analysis to rank web pages was demonstrated by the PageRank algorithm. Two main weaknesses is that model were exploited by web spammers to manipulate their ranking through link farms. These are, a fixed allotted rank per page, and the ability of rank sink topology to accrue disproportionate amount of PageRank.

TrustRank [1] extended the concept of random surfer model to develop an approach for combating web spam. At its core, it is a variant of the random surfer model with jumps to a select set of pages. While its an improvement over the PageRank model, it is not very resilient to link manipulation itself and is affected by some of the same limitations as PageRank, such as, rank sink problem and determining node ranks based on incoming link topology only.

TrustRank remains a significant algorithm, with applications to calculating node proximity and uncovering network structure [8]. In this report, the AffinityRank algorithm [4], which was also designed to suppress link spam, is studied and its characteristics compared with TrustRank. AffinityRank is based on a fundamentally different diode model that assigns rank to a node based on not only the incoming network topology, but the outgoing link structure as well.

The main contributions of this report are,

1. Generalizing the diode model of Affinity-Rank to extend its applicability to undirected networks and for detecting sub-graphs/clusters.
2. Comparative analysis of TrustRank and AffinityRank, including host-level link analysis of 2012 webdata-commons web crawl which comprises of over 100 million hosts with two billion edges.
3. Tackle the main weakness of AffinityRank, that is, computational aspects of non-linear solution of equations. A fast and approximate solution is derived, and evaluated by verifying a base case example.

2. Prior Work

TrustRank paper [1] describes the approach of using a trusted set of nodes for spam resilient link analysis. Using a seed set is also fundamental to the algorithms studied in this report. The difference with respect to the seed set is that while in TrustRank the final ranks of trusted nodes depends on the graph, in AffinityRank, it can be set to a specific value.

An alternate ranking approach with a diode network analogy is described in the AffinityRank paper [4]. An endorsement mechanism from a set of trusted seed nodes is used to produce spam resilient ranking. In this report, that voting scheme is altered to remove the limitation posed by diode model, and generalized for applications to undirected graphs, as well as node clustering, which is not possible with the original approach.

Faloutsos, et al. [5] describe an algorithm, which is also based on electricity analogy to discover a connection sub-graph between two nodes. It is based on a goodness function of subgraphs, defined as the total delivered current from source to sink on that sub-graph. It involves iteratively adding paths between the two nodes. In this report, the generalized AffinityRank model allows for link weight changes and is directly used to partition the graph between two sets of nodes, based on the ranking from the updated voting scheme.

3. Generalized AffinityRank Model

3.1 Formulation

In this report the generalized model of Affinity-Rank is developed for community detection. Affinity rank for a node *i* in a network depends on the nodes which edges pointing to node *i* (Set *I*), and the nodes that node *i* directly points out to (Set *O*). In case of reciprocal links, same node conceptually exists in both the sets, with different edge weights. The model described in [4] is for directed graphs with binary edge weight (0,1), hence the diode analogy. The generalized model described in this section is applicable for directed and undirected graphs. This model is derived based on following basic conditions:

1. An edge from node *i* to node *j* is carries an endorsement value (v_{ij}) by node *i*.
2. Sum of endorsements received by a node is equal to the sum of all endorsements provided by that node.
3. Endorsement transferred over an edge $i \rightarrow j$ is equal to rank difference of the end nodes times the edge weight w_{ij}.
4. A set of source nodes (*S*) are assigned a pre-determined fixed rank. Ranks of other nodes are relative to the source set.
5. Each node has a virtual edge of weight w_a to a sink node *a*, that has fixed zero rank.

Applications of ranking with respect to a source set include spam resilient web page ranks, topic specific ranking. The sink node has similar function as rank dampening, it reduces the rank of nodes (even in a single chain), as the distance of a node from the source increases. For spam resilient ranking of web pages, one additional condition is required:

6. Edge weight w_{ij}, is a function of rank difference between the two nodes of an edge.

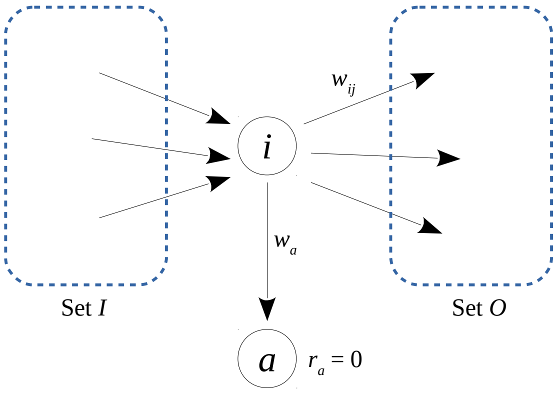

Figure 1: Affinity rank model

The analogy for Affinity rank can be though of as a physical system, such as an electric circuit network, where current is the endorsement and potential is the rank, or as a network of mechanical springs, where force is endorsement and displacement is the rank. For a linearized step, the edge weights are fixed. In the spring analogy, the source nodes are pulled to a certain distance and the rank of other nodes is their displacement, which depends on how the entire network deforms to achieve equilibrium at each node. As depicted in Figure 1, for any node $i \notin S$ in the graph, from rule 2,

$$\sum_{j \in I} v_{ij} = \sum_{k \in O} v_{ik} + v_a$$

From rule 3,

$$\sum_{j \in I} w_{ij}(r_j - r_i) = \sum_{k \in O} w_{ik}(r_i - r_k) + w_a r_i$$

Therefore,

$$\begin{aligned} \left(\sum_{j \in I} w_{ij} + \sum_{k \in O} w_{ik} + w_a\right) r_i & \\ - \sum_{j \in I} w_{ij} r_j & \\ - \sum_{k \in O} w_{ik} r_k &= 0, \quad i \notin S \\ r_i &= v_{0,} \quad i \in S \end{aligned} \tag{1}$$

Equation (1) give the matrix formulation to compute Affinity rank, $W\vec{r} = \vec{v}$. Matrix W is diagonally dominant along rows, due to extra w_a term (rule 5), this ensures convergence by linear solvers. Note that the source nodes need not have the same fixed rank, that property is utilized for clustering, in section 3.4. For spam resiliency, $w_{ij} = w_0 \times func(r_i - r_j)$, that is, endorsements from a higher rank node to a lower rank node carry more weight. Therefore, $W(r)\vec{r} = \vec{v}$. An iterative solution for computing ranks within a linearized step is shown in equation (2).

$$r_i = \frac{\sum_{j \in I} w_{ij} r_j + \sum_{k \in O} w_{ik} r_k}{\sum_{j \in I} w_{ij} + \sum_{k \in O} w_{ik} + w_a} \tag{2}$$

Equation (2) can be reduced to the diode model (linearized step) as described in [4]. For undirected graphs, since edge weights do not change, equation (3) is the final iterative solution to the simplified model.

$$r_i = \frac{\sum_{j \in I} r_j + \sum_{k \in O} r_k}{n_I + n_O + \frac{w_a}{w_0}} \tag{3}$$

3.2 Relation to PageRank Formulation

From the previous section, it appears that there cannot be a relation with flow type models. However, AffinityRank also includes the outgoing edge component, and in this section, PageRank matrix is derived as a function of that matrix component.

PageRank scores, r, can be computed iteratively by solving the system of equations in (4).

$$\left[I - \beta G^T\right]\vec{r} = \frac{(1-\beta)}{N}\vec{I} \tag{4}$$

G^T is the web transition matrix, $g_{ij}=1/n_i$ if link from node i to j exists, 0 otherwise. n_i is the number of outgoing links from node i. The equivalent PageRank matrix for $\left[I - \beta G^T\right]\vec{r} = \vec{V}$ is,

$$P = [I - \beta G^T] \tag{5}$$

Convergence is assured, since P is diagonally dominant along columns (for $\beta < 1$). The AffinityRank matrix, W can be separated into three components – for incoming edges (W_I), outgoing edges (W_O), and edge to sink node Λ, which is a diagonal matrix of λ, where $\lambda = w_a/w_0$

$$W = W_I + W_O + \Lambda \tag{6}$$

$$W_O = N_O[I - G] \tag{7}$$

N_O is diagonal matrix of number of outgoing edges per node. From equation (5), $G=(1/\beta)[I-P^T]$, substitute in (7) to get relation between P and W_O.

$$P = W_O^T(\beta N_O^{-1}) + (1-\beta)I \tag{8}$$

3.3 Affinity versus Proximity

In TrustRank, equation (4) is modified such that $v_i = (1-\beta)/|S|$ if $i \in S$, 0 otherwise.
Instead of random jump, teleporting is limited to the source set, S. Having formulated the algorithm, in this section, the difference between TrustRank and Affinity-Rank is illustrated by an example described in class notes.

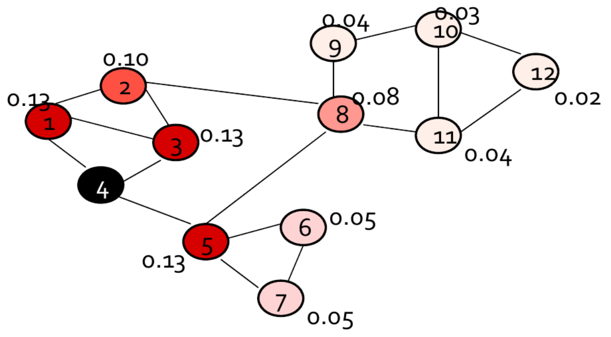

Figure 2: Random Walk with Restarts Example

Table1: Normalized Ranks (restarts) (r_i/r_4)

id	1	2	3	4	5	6	7	8	9	10	11	12
TR	.61	.44	.61	1.0	.59	.24	.24	.37	.13	.15	.11	.09
AR	.61	.44	.61	1.0	.44	.32	.32	.28	.17	.13	.16	.12

Value of λ, is chosen such that the second highest normalized score is same between TrustRank (TR) and AffinityRank (AR). Consider nodes 7 and 8. TrustRank ranks node 8 higher than node 7, this is because all the rank emanating from source node 4 (minus the decay part) passes through node 8, whereas node 7 has receives decay-factor times half the rank of node 5. AffinityRank, however, ranks node 8 lower than node 7, because node 8 has outgoing links to subgraph 9 through 12.

Since AffinityRank considers the incoming sub-graph from S, as well as its outgoing network structure, its application for clustering is explored in the next section.

3.4 Application of Affinity to Clustering

Equation (1) poses no restriction on the initial values of the source set, and guarantees their final rank to be the same as initially applied. The strategy for clustering/community-detection is to select two sets of source nodes and set one of them to a positive value, and the nodes in other set to an equal and opposite value.

This is particularly applicable for graphs that follow a power law degree distribution, with top nodes having densely connected groups, and link across groups through weaker ties. Since at least two nodes are required for generating two different subgraphs, one strategy would be to first run PageRank, and choose the two nodes that are high ranking, with shortest path between them exceeding a minimum threshold.

3.4.1 Results of AffinityRank based Clustering

The strategy for affinity based clusering is applied to Zachary's Karate club network. The source set consists of nodes {1, 34}. Node 1 is set to +1.0 initial rank, and node 34 is set to -1.0 initial rank.

The results are shown in Figure 3. Nodes with rank >= 0 are colored red, and nodes with rank < 0 are colored blue. Darker colors show higher rank. The two partitions match exactly with the color coding of the true solution presented in class notes.

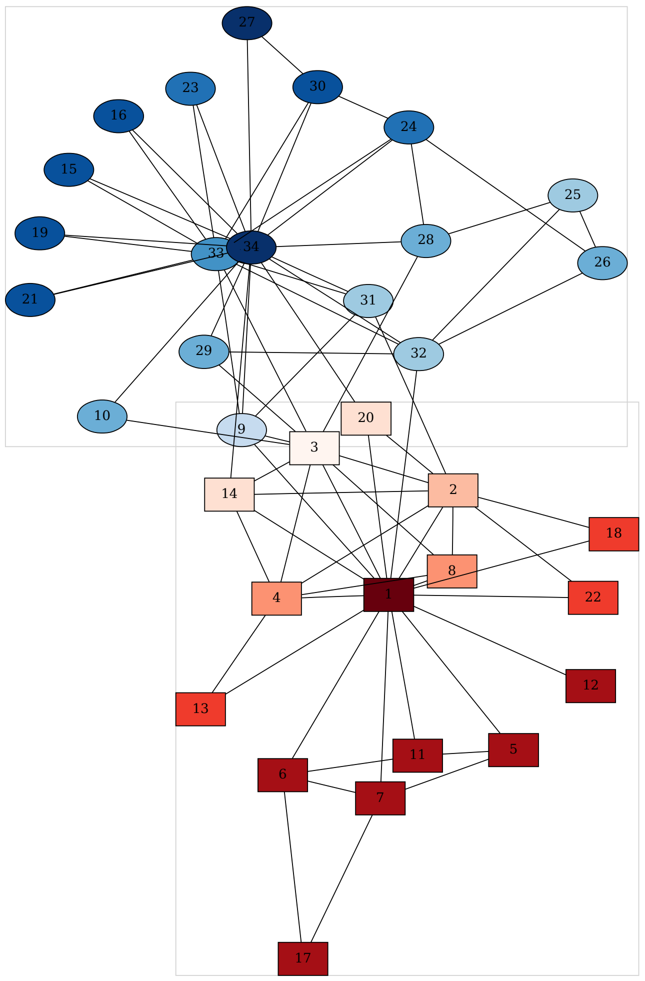

Figure 3: AffinityRank based clustering

4. Spam Resilient Link Analysis

TrustRank and AffinityRank diode model, employ different properties to combat link spam. As discussed in the previous section, for AffinityRank, it is the consideration of both the incoming and the outgoing link topology. In case of TrustRank, it is (a) rank splitting and (b) rank attenuation. However, TrustRank has the following weaknesses:

1. Rank attenuation works if the chosen seed nodes are several hops away from spam nodes, which, given the "Small World Phenomena" put bulk of non-spam pages at about the same distance as the spam-networks.
2. Due to 1., seed nodes have to be very carefully chosen and be limited to a small set, which does not scale.
3. Although the random jump component and all "rank leaks" are shared between the seed nodes, *TrustRank* does not address "rank sink" problem. Links can still be manipulated to rank a spam page high. To address this shortcoming, [2] proposes using the ratio of *TrustRank* to *PageRank* in order to estimate spam.
4. Rank of a node depends on the incoming network topology. Consider two pages, both with similar incoming link structure, but one points to good pages, and the other links to a spam farm. *TrustRank* "trusts" both the pages equally, because the outgoing link structure is not taken into account while computing node rank.

4.1 Illustrative Example

In this section, Figure 2 from [1] is chosen to demonstrate that is easily possible to take a bad nodes and tweak its outgoing link structure such that it ranks higher than a good node.

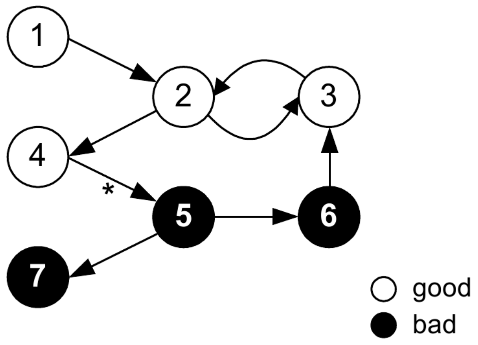

Figure 4: Original example graph [1], page 578

In the original graph, node 3 ranks higher than node 6, as expected, with source set {2,4}.

Table 2: Normalized ranks, example graph

id	1	2	**3**	4	5	**6**	7
TR	0.00	1.00	**0.68**	0.84	0.72	**0.30**	0.30
AR	0.00	1.00	**0.67**	1.00	0.46	**0.31**	0.31

Now, the out-links of node 6 are modified to include reciprocal links to indicate link spam.

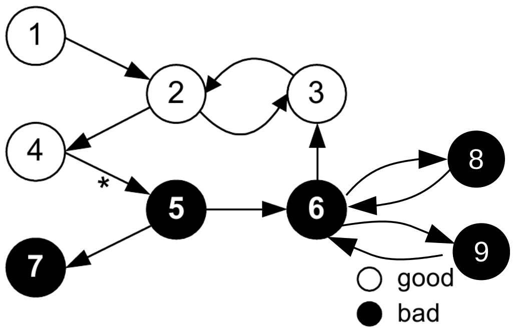

Figure 5: Manipulated example graph

Table 3: Normalized ranks, manipulated graph

id	1	2	**3**	4	5	**6**	7	8	9
TR	0.0	1.0	**.60**	.91	.77	**.63**	.33	.18	.18
AR	0.0	1.0	**.67**	1.0	.42	**.19**	.28	.13	.13

By manipulating its outgoing network, node 6 is able to outrank a “good” node (3). However, in AffinityRank, it reduces its affinity to source set even further, thereby reducing its rank-score.

4.2 Model for Incremental Updates

In the previous section, spam resilient ranking of AffinityRank was demonstrated via a simple example. One of the main weakness of AffinityRank is the computationally expensive nonlinear analysis, that is, as the ranks of nodes change, the link strengths have to be altered, which in turn changes the ranks. For example, in Figure 5, first run indicates rank of node 2 is greater than nodes 1, 3 and rank of node 6 is greater than nodes 8, 9, which, in the diode model trims out edges $9\rightarrow6$, $8\rightarrow6$, $3\rightarrow2$ and $1\rightarrow2$. In the generalized model, these links get attenuated.

In this section, a model for incremental update is developed, which can be applied to changing graph, once AffinityRank is computed in the original graph. It is based on the observation that for nodes within few hops from source, as the depth of graph increases (e.g., continued crawl), change in the outgoing link structure has diminishing effect. This can be demonstrated by a simple link chain ($1\rightarrow2\rightarrow i\rightarrow i+1..\rightarrow n$). Rank of node i, $r_i=(r_{i-1}+r_{i+1})/(2+\lambda)$. If the rank is to be dependent on the incoming links only, let $r_i = \beta_i r_{i-1}$, then, $\beta_i=1/((2+\lambda)-\beta_{i+1})$ which can be solved recursively, with boundary condition, for the last node, $\beta=1/(1+\lambda)$. As the length of chain increases, $\beta_i\approx\beta_{(i+1)}$, which for a link chain gives a constant value of β. For λ=0.25, β=0.6096, and a simple analysis confirms it.

$$\beta_i=1+\frac{\lambda}{2}\left(1-\sqrt{1+\frac{4}{\lambda}}\right) \tag{9}$$

Similarly, for a tree structure (branching factor, *b*), it can be shown that the rank of nodes closer to root approaches a fixed factor of their parent.

$$\beta_i=\frac{1}{2b}\left(1+b+\lambda-\sqrt{(1+b+\lambda)^2-4b}\right) \tag{10}$$

In general, there can be a change in the incoming network structure for a node, or in its outgoing structure, as web crawler downloads new pages. Links from newly discovered pages, several hops away from any source node will have little to no effect on AffinityRanks. However, if the new page is closer to a source page, or if a new source page itself is added, that can have notable effect on rankings.

In order to incrementally update ranks, the model shown in Figure 1 is modified to represent the outgoing edges by a single equivalent edge to a new sink node.

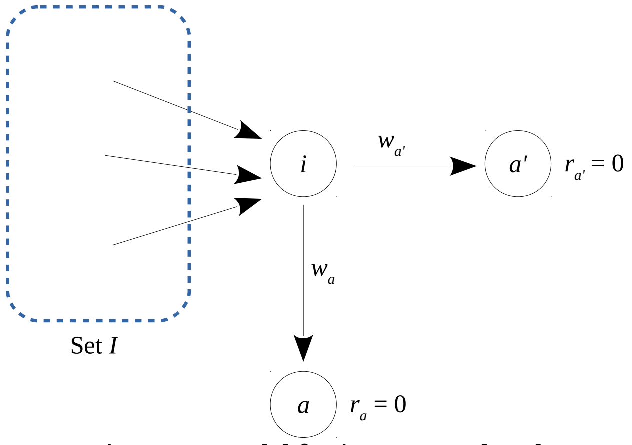

Figure 6: Model for incremental update

Since the matrix can be separated into components for the sink node, inlinks and outlinks, the incremental update algorithm substitutes the outbound endorsements by the product of W_O and r, at time = t. Derivation for incremental updates follows.

$$W^t r^t = v^t$$
$$[\Lambda + W_I^t + W_O^t]r^t = v^t$$
$$[\Lambda + W_I^t]r^t + [W_O^t]r^t = v^t$$

$$[\Lambda + W_I^t]r^t + B_O^t = v^t \tag{11}$$

Let,

$$\Lambda_O=[I\,v^t]^{-1}[I\,B_O^t]$$

From equation 11,

$$[\Lambda+\Lambda_O+W_I^t]r^t = v^t$$

which leads to the incremental model,

$$[\Lambda^t + W_I^{t+1}]\{\Delta r^{t+1}\}=\{\Delta v^{t+1}\} \tag{12}$$

Ranks at configuration t+1 can be computed as,

$$r^{t+1}=r^t+\Delta r^{t+1} \tag{13}$$

In practice, this calculation may be split into an offline (full compute) that runs occasionally and online (incremental update) process that runs continuously. The offline component serves two purposes (a) It provides the true solution to the current graph, eliminating accrued errors and (b) It precomputes Λ_O, which is used for subsequent online rank calculations, until the next offline full compute.

4.3 Evaluation of Incremental Updates

For evaluation, we take an example of a balanced binary tree of depth 10, compute its ranks (t=1). The root node, then adds two additional links to its direct children (t=2). The full offline calculation is compared with incremental online calculation. This is a base case, since every node (except root) now has a change in the incoming link structure, but no change in the outgoing links, therefore the actual computed scores should exactly match the incremental scores. The results are highlighted in the table below.

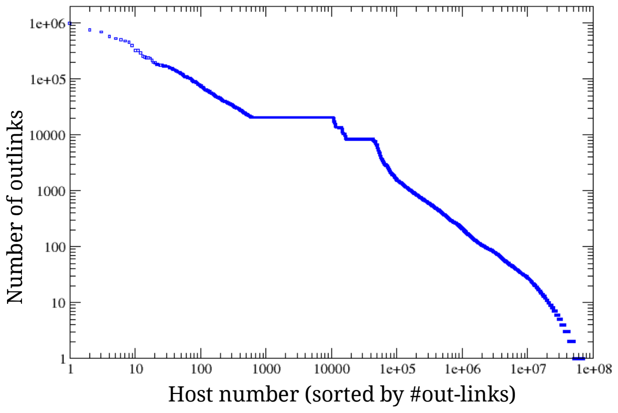

Figure 7: Host-host out-degree distribution

Spam List
The distribution of the number of subdomains per domain is shown below.

Table 4: Evauation of Incremental Updates versus Full Compute

						<- Offline	Online ->	
Depth	Node	Score1 (computed)	Score2 (computed)	Delta (%)	Bo(i)	Λo(i,i)	Δ(i)	Score2 = (Score1+Δ)
0	1	**1.0000**	1.00000	0.0000	1.175326	1.17533	**1.000000**	1.00000
1	3	0.4123	0.67794	39.1778	0.484568	1.17517	**0.265602**	0.67794
2	7	0.1701	0.27959	39.1782	0.199772	1.17476	0.109537	0.27959
3	15	0.0702	0.11537	39.1799	0.082330	1.17335	0.045201	0.11537
4	31	0.0290	0.04769	39.1819	0.033915	1.16940	0.018683	0.04768
5	63	0.0120	0.01981	39.1876	0.013933	1.15683	0.007762	0.01981
6	127	0.0051	0.00835	39.1956	0.005697	1.12199	0.003273	0.00835
7	255	0.0022	0.00367	39.2171	0.002281	1.02312	0.001440	0.00367
8	511	0.0011	0.00179	39.2397	0.000868	0.79734	0.000703	0.00179
9	1023	0.0007	0.00108	39.2790	0.000262	0.40000	0.000426	0.00108
10	2047	0.0005	0.00086	39.2790	0.000000	0.00000	0.000341	0.00086

4.4 Results of Web Link Analysis

Crawl Data
The data for analyzing AffinityRank versus TrustRank over large web crawl was obtained from webdatacommons.org. This data is based on 2012 web crawl and is available at a page level and host/subdomain level. Page level crawl consists of more than 3 billion pages and 128 billion links, with 94% of all pages connected in the graph. The largest strongly connected component consists of over 50% of all pages. Details of graph statistics, at various levels of granularity are available here [9]. For this project, host level graph was used for analysis since it is sufficiently large size, with 101 million nodes and over 2 billion edges.

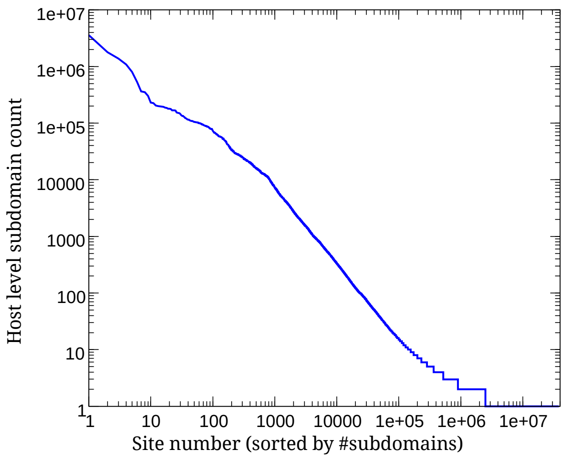

Figure 8: Host sub-domain distribution

The number of host level domains can go up to millions. Most of the site will more than a million sub-domains are well known sites e.g., tublr.com, blogspot.com. Sites with explicit keywords in sitename dominate in the range of few thousand to few tens of thousand sub-domains. That list of keywords was extracted into a separate spam-words list.

Source List
Generating a good set of source nodes was a three step process. In the first step, a set of cross category queries was selected based on categorizations available in various online sources (dmoz, amazon, google trends). Examples include "digital cameras", "fashion", "news", "universities". In the second step, these queries were performed on a web search engine requesting 100 results, and the host level domains were added to candidate pool. The assumption here is that search engines are sophisticated enough to not show spam pages on popular queries. In the third step, the candidate set of source Ids was cleaned up to ensure that they do not contain any spam-words, and the site is within a threshold of number of sub-domains. This produced a set of 93,408 seed sites, with about 44% of them being .edu domains.

Implementation Notes
For web graph analysis, diode model is used due to its faster run time. The program for running link analysis was developed on a custom C++ codebase optimized for memory consumption. The node information, e.g., number of outlinks, current and previous rank is encapsulated in a separate class. Before starting iterations, pointers to node objects, corresponding to outlink ids are written out to disk. This way, the graph is stored on disk and read once per iteration, and only the node objects need to be kept in memory.

Evaluation
For evaluating the efficacy of TrustRank and AffinityRank in suppressing spam, ranking of spam pages between the two algorithms is compared. Definition of spam site is one that has contains the word from spam list, described above. Figure 9 shows the cumulative number of spam nodes upto top 10 million ranked hosts, which is roughly 10% of the graph. Since the top ranking nodes (TrustRank or Topic-sensitive rank) are more likely to be recommended, very low faction of spam is critical. Figure 10 shows the cumulative percent of known spam nodes versus top percentile ranking.

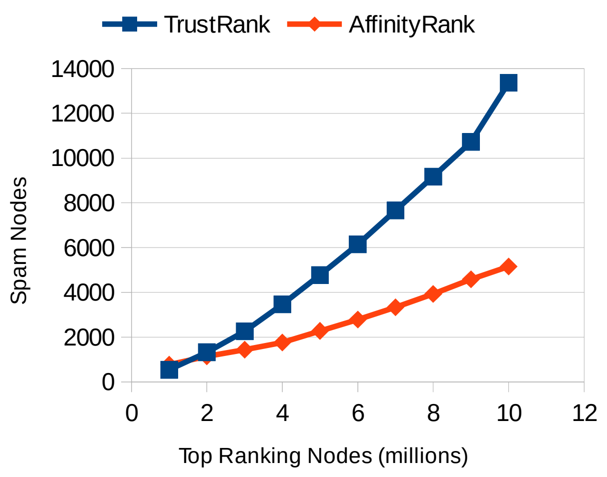

Figure 9: AffinityRank versus TrustRank for top 10% highest ranked nodes in web graph

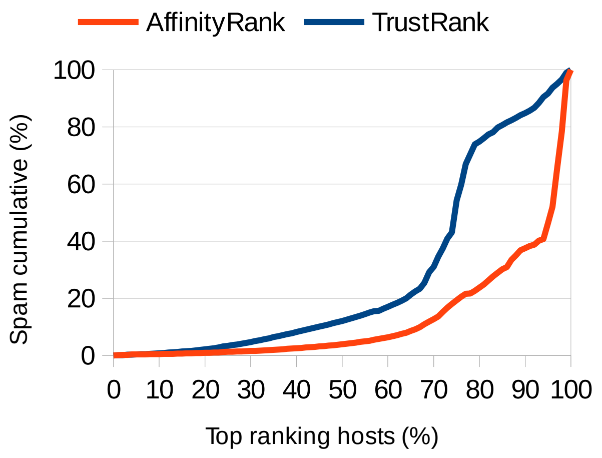

Figure 10: Cumulative spam versus percentile rank

From the figures, it is clear that AffinityRank consistently ranks spam hosts lower than TrustRank. From Figure 10, it can be observed that about 60% of known spam hosts occupy the last 5 percentile ranks.

5. Conclusions

In this report, two link analysis algorithms designed for spam resilient ranking are compared with theoretical and experimental analysis. Capability of AffinityRank in suppressing spam pages over TrustRank is clearly demonstrated by a sample example, from TrustRank paper, as well as host-level link analysis of web crawl comprising of over 100 million nodes.

Application of TrustRank and AffinityRank to study link structure is also presented. TrustRank can be used to find proximity between nodes, whereas AffinityRank can be used to find clusters with respect to a set of nodes. In order to compute clusters, a generalized model is developed which removes diode constraints and makes clustering method possible.

The main shortcoming of AffinityRank is its nonlinear solution. It is addressed in this report by developing an approximate incremental update approach. The results of incremental approach are evaluated over a base case and confirmed to provide identical results.

References

[1] Z. Gyongyi, H. Garcia-Molina, J. Pedersen. Combating Web Spam with TrustRank. In Proc. of VLDB, 2004

[2] Z. Gyongyi, P. Berkhin, H. Garcia-Molina, J. Pedersen. Link Spam Detection Based on Mass Estimation. In Proc. of VLDB, 2006.

[3] J. Caverlee, L. Liu, and S. Webb, “SocialTrust: Tamper-resilient trust establishment in online communities,” Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries, ACM New York, NY, USA, 2008

[4] R. Kaul, Y. Yun, S. Kim. Ranking Billions of Web Pages Using Diodes. In Communications of the ACM, Vol. 52, No.8, August 2009

[5] C. Faloutsos, K. Mccurley, and A. Tomkins. KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining, page 118--127. New York, NY, USA, ACM Press

[6] B. Bahmani, A. Chowdhury, A. Goel. Fast Incremental and Personalized PageRank. In Proc. of VLDB, 2010

[7] K. Biscuitwala, V. Ramesh, K. Tezlaf. Analyzing Twitter Spam. CS224w Project Report, Stanford University, 2011

[8] X. Falco, J. Fan, P. Kellyl. Adding Dimensions to Ranking in Social Graphs, CS224w Project Report, Stanford University, 2013

[9] Web Data Commons, Common Crawl http://webdatacommons.org/hyperlinkgraph/index.html, August 2012