Attack Tolerance and Resiliency of Large Complex Networks

Xiaoying Pang
xypang@stanford.edu

Shanwei Yan
syan@stanford.edu

David Zucker
zuckerd@stanford.edu

ABSTRACT

1. INTRODUCTION

In this paper, we consider the problem of measuring the impact and tolerance to complex network attacks in response to manipulation of a small number of identified influential nodes. We incorporate literature and techniques from computer science, physics, sociological and financial domains to analyze this problem from a cross-disciplinary perspective.

Our cross-disciplinary literature review has shown that there are many approaches to identify the most influential nodes in a complex network with varying levels of accuracy and performance. However, we are not aware of studies that apply cross-disciplinary influential node discovery approaches to the same network and test their accuracy and performance using a standardized set of criteria.

To explore this problem further, we measure influential node discovery techniques in the social, communication, and financial domains using datasets from the Facebook and epinions social media networks, a modified version of the CAIDA autonomous system (AS) network, an S&P 500 stock correlation network and Stanford web graph. Each dataset is described further in section 5 below.

To perform our analysis, we leverage multiple cross-disciplinary deterministic and heuristic influence algorithms as identified in section 4 below. We measure each algorithms influential node detection capabilities and performance within each dataset as well as across datasets. We then execute several cascade processes to measure the spread of influence within the dataset. Additional information on our experimental model is found in section 3 below.

Our intent is to determine the extent to which algorithms and relevant graph analysis techniques can be successfully leveraged across multiple domains to potentially limit the impact of an attack, quantify vulnerabilities, and potentially inform network design. Scenarios where our analysis may be useful include forecasting marketing or advertising expenditures on a social network to maximize impact, identifying vulnerabilities in the Internet back-bone or other physical networks, and measuring the impact of organizational financial performance within highly correlated financial networks.

2. LITERATURE REVIEW

Out literature review spans cross-disciplinary papers addressing influential node identification algorithms design, logical network resiliency analysis, and financial portfolio construction.

Kempe et al.[5] are the first to propose a hill-climbing greedy algorithm to find the most influential seed nodes in a complex network. Using an analysis framework based on the submodular functions, they prove that any solution obtained by the greed strategy should be within 63% of the optimal for both independent cascade and linear threshold models. Leskovec et al. [7] further improve the hill-climbing greedy algorithm by exploiting submodularity and propose the Cost Effective Lazy Forward (CELF) algorithm, which achieves as much as 700 times speedup. W. Chen et al. [15] notice that the general greedy approach takes $O(knRm)$ time to complete, where n is the number of nodes, m is the number of edges, R is the number of random cascade simulations needed to estimate the marginal gain for every node. They propose that instead of simulating the random cascade R times for every additional candidate node, one should generate R subgraphs of the original graph by randomly removing edges with probability $1 - p$, where p is the propagation probability of the random cascade. This approach is much more computationally efficient because one can get the reward function for the existing seed set and for every other node in the network all at once. They further propose to combine their algorithm with the CELF into the *MixGreedyIC* which uses their new approach to select the first seed and use the CELF in later rounds. C. Borgs et al. [3] based their algorithm on the intuition that if a node appears often as an influencer to other nodes, then it is likely a good candidate for the most influential node. Their algorithm returns a $(1 - 1/e - 1/\epsilon)$-approximation to the influence maximation problem, with success probability 0.6, in time $O((m + n)\epsilon^{-3} \log(n))$. K. Jung et al. [8] proposed a IRIE framework to solve the influence maximization problem by combining a simple iterative influence ranking algorithm and an influence estimation method. The IMRank algorithm proposed by S. Cheng et al. [13] finds a self-consistent ranking by reordering nodes iteratively according to their ranking based marginal influence spread. The collective influence (CI) defined by F. Morone et al. [6] is an effective quantity to use in order to find the minimum set of nodes that keep the whole network connected, which is also defined in [6] as the most influential nodes in a network. Lastly, [10] has shown that k-Shell score is also very effective in selecting the most central node in a network.

Domain specific papers reviewed include financial portfolio construction and network resiliensy analysis literature.

C.K. Tse et al.[4] propose several approaches to the construction of US stock financial portfolios based on price, volume and return correlations. This notion is expanded in N. Koochakzadeh et al. [11] through the exploration of clustering stock by the Louvain method and constructing portfolios for comparisons against S&P 500 returns.
In [9], Faloutsos et al. analyze autonomous system (AS) graphs and identify three power law metrics and related graph properties that accurately describe AS graph topology. The identified metrics and properties are useful in the creation of accurate artificial models for future AS graph study. Although, this study uses a small dataset from 1997 and 1998, the power laws metrics and graph properties are valid components in AS graph investigation. C.K. Tse et al.[4] expand on [9] to discuss the behavior of networks under attack including diameter, giant component and isolated cluster size changes to understand the survivability of networks. They found that exponential and scale-free networks are robut against random failures, but vulnerable to intentional attacks.
We implemented many of the algorithms identified in these readings. Our goal is to use these tools to identify the most influential nodes in our autonomous system, stock network, and social network data and to see how their influence propagate through the network, and in the end, find the most effective way to select the most influential nodes for different networks.

3. EXPERIMENTAL METHOD

Our experimental method consists of three primary phases influential node detection, cascade execution, and strongly connected component (SCC) attack with findings and analysis presented at the end of each phase.
Phase I: Influential Node Detection To determine the $k = 3$ most influential nodes within the given graph, we execute CELF, NLT, IRIE, IMRank, CI, and k-Shell algorithms and capture k influential nodes and algorithm runtime as output.
Phase II: Cascade Execution We leverage the Independent Cascade (IC) algorithm [5] to execute the cascade process across graphs with the k most influential nodes identified in Phase I as input and capture the total number of infected nodes, the number of cascade iterations, and the set of impacted nodes during each iteration.
Phase III: SCC Attack We measure the resiliency of each graph as the strength of the SCC when an attacker has knowledge of a very large k number of the most influential nodes. We remove influential nodes as identified by the algorithms in influence descending order and measure the size of the SCC in each iteration. Once the SCC dissolves the process is completed and we capture the iterations required to disconnect the SCC.

4. ALGORITHMS

We have implemented the following algorithms in Python.

4.1 Cost Effective Lazy Forward (CELF)

CELF is an improved hill-climbing greedy algorithm for influence maximization. Leskovec et al. [7] first proposed this algorithm and show that comparing to the general greedy strategy adopted by Kempe et al. [5], CELF can reduce the computing time by a factor of 700. W. Chen et al. [15] further improve the efficiency of the algorithm by replacing expensive random cascade simulations for every node in the graph with randomly removing edges from graphs. Their proposed *MixGreedyIC* uses the new approach to select the first seed node and return to the CELF to select the rest of the seed nodes. Our implementation of the CELF is based on the *MixGreedyIC*.

Algorithm 1 Cost Effective Lazy Forward

function CELF(G, k)
  $S = \emptyset, R = 10000$
  $Dict(v) = \emptyset, \forall v \in V$
  for i = 1 to R **do**
    Get subgraph G'_i by randomly removing edges from G with probability $1 - p$
    $Dict(v) = Dict(v) \cup R_{G'_i}(v), \forall v \in V$
  end for
  while $|S| < k$ **do**
    $cur_v = False, \forall v \in V$
    while True **do**
      $v = \arg\max_w |Dict(w)|$
      if $cur_v = True$ **then**
        $S = S \cup v$
        $Reward(S) = Reward(S) \cup Dict(v)$
      else:
        $Dict(v) = Dict(v) \cap \overline{Reward(S)}$
        $cur_v = True$
      end if
    end while
  end while
end function

4.2 Near Linear Time (NLT)

C. Borgs et al. [3] proposed an influence maximization algorithm that builds a hypergraph in time $O((m+n)\epsilon^{-3}\log(n))$ by simulating influence spread in the transpose graph of the original one and then repeatedly choose the node with the highest degree as the most influential nodes. They theoretically proved that this approach can achieve a $(1-1/e-1/\epsilon)$-approximation to the optimal solution, with success probability at least 3/5. Here n is the number of nodes, m is the number of edges, ϵ is the precision which takes value in the range of $(0, 1)$.

4.3 Influence Rank Influence Estimation (IRIE)

The IRIE algorithm [8] combines a simple influence rank (IR) method and an influence estimation method. The influence rank algorithm is derived from a belief propagation approach, which uses a simple iterative scheme to compute the influence rank of every node. However, the influence ranking is only useful in selecting the first most influential node. Due to the large overlaps between the influence sets of the highly ranked nodes, an influence estimation method, e.g. Monte-Carlo simulation or Maximum Influence OutAborescence (MIOA) [14], is needed to help select the rest of the influential nodes. For any node $u \in V$, the IRIE ranking for v is

$$r(v) = (1 - AP_S(v))(1 + \alpha \sum_{u \in N_{out}(v)} P_{vu} r(u))$$

where $1 - AP_S(v)$ is the probability of node v not being influenced by S which can be calculated by either Monte-

Algorithm 2 Near Linear Time

function NLT(G, k, ϵ)
  $R = 144(m+n)\epsilon^{-3}\log(n)$
  H = BuildHyperGraph(R)
  return BuildSeedSet(H, k)
end function
function BuildHyperGraph(R)
  $step = 0$
  while $step < R$ **do**
    Choose a node u from G randomly
    Simulate influential spread starting from u, in G^T, and get $Z = I(u)$
    $H = H \cup Z$
  end while
  return H
end function
function BuildSeedSet(H, k)
  $S = \emptyset$
  for $i = 1$ to k **do**
    $v_i = \arg\max_v deg_H(v)$
    $S = S \cup v_i$
    Remove v_i from H
  end for
  return S
end function

Carlo or MIOA. Nodes with high $r(v)$ are chosen as the most influential nodes.

4.4 IMRank

The IMRank algorithm [13] finds a self-consistent ranking by reordering nodes iteratively in terms of their ranking-based marginal influence spread. It also includes a Last-to-First Allocating (LFA) strategy which states that 1) each node can only be activated by nodes ranked higher than it; 2) when a node can be activated by multiple nodes, the highest-ranked node activates it first. The initial ranking can be obtained by various criteria, e.g. degree, out-degree, PageRank, etc.

4.5 Collective Influence (CI)

F. Morone et al. [6] defines the most influential nodes as the ones forming a minimum set that guarantees the connectivity of the network. They use the size of the giant component as a measure of the influence and formulate the influence maximization problem into finding the minimum fraction of influencers removed to fragment the whole network. They propose to use the collective information (CI) which is defined as,

$$CI_l i = (k_i - 1) \sum_{j \in \partial Ball(i,l)} (k_j - 1)$$

to identify the most influential nodes, where $\partial Ball(i,l)$ is the set of nodes whose shortest distance from node i is l.

4.6 K-Shell Decomposition (k-Shell)

The k-Shell scores [10] of nodes in a graph are computed by conducting the k-Shell decomposition. Starting from order 0, nodes with degrees less than the current order s are removed repeatedly until no node with degree less than s is left. The removed nodes become the sth core of the graph. The process continues until no node is left in the graph.

Algorithm 3 IRIE

function IRIE(G, k, p)
  $S = \emptyset$
  $r(v) = 1$, $ap(v) = 0$, $\forall v \in V$
  while $|S| < k$ **do**
    r = IR(G, r, ap, p)
    $S = S \cup \arg\max_v r(v)$
    ap = GetCascadeProbability(G, S, p)
  end while
  return S
end function
function IR(G, r, ap, p)
  $MaxIter = 20, \alpha = 0.7, iter = 0$
  while $iter < MaxIter$ **do**
    for $v \in V$ **do**
      $r_{new}(v) = (1 - ap(v))(1 + p\alpha \sum_{u \in N_{out}(v)} r(v))$
    end for
    if $\max(r_{new} - r) < 0.0001$ **then**
      break
    end if
    $r = r_{new}$
  end while
  return r_{new}
end function
function GetCascadeProbability(G, S, p)
  $ap(v) = 0, \forall v \in V$
  for $v \in V$ **do**
    if $v \in S$ **then**
      $ap(v) = 1$
    else if $N_{in}(v)$ is not $\emptyset$ **then**
      $ap(v) = 1 - \prod_{w \in N_{in}(v)} (1 - ap(w)p)$
    end if
  end for
  return ap
end function

Algorithm 4 IMRank

function IMRank(G, k, p)
  Obtain the initial ranking $r^{(0)}$ using degrees (undirected) or out-degrees (directed) in descending order
  $t = 0$
  repeat
    t = t+1
    $M_{r^{(t)}}$ = GetM($G, r^{(t-1)}, p$)
    Obtain a new ranking $r^{(t)}$ by sorting $M_{r^{(t)}}$ in descending order
  until $r^{(t)} = r^{(t-1)}$
  return $r^{(t)}[1:k]$
end function
function GetM(G, r, p)
  for $v \in V$ **do**
    $M_r(v) = 1$
  end for
  for $i = n$ to 2 **do**
    for $j = 1$ to i **do**
      $M_r(v_{r_j}) = M_r(v_{r_j}) + pM_r(v_{r_i})$
      $M_r(v_{r_i}) = (1 - p)M_r(v_{r_i})$
    end for
  end for
  return M_r
end function

	AS	S&P 500 Bear	S&P 500 Bull	facebook-Stanford	facebook-UCSD	epinions	web-stanford
Graph Type	Directed	Undirected	Undirected	Undirected	Undirected	Directed	Directed
Nodes	26,475	474	474	11,621	14,948	75,879	281,903
Edges	106,312	12,839	14,833	568,330	443,221	508,837	2,312,497
SCC Size	99.6%	81.85%	77.22%	99.7%	99.9%	42.47%	53.4%
Approx. diameter	16	8	6	8	8	13	316
90% eff. diameter	4.44	3.53	2.75	3.27	3.53	4.95	10.06

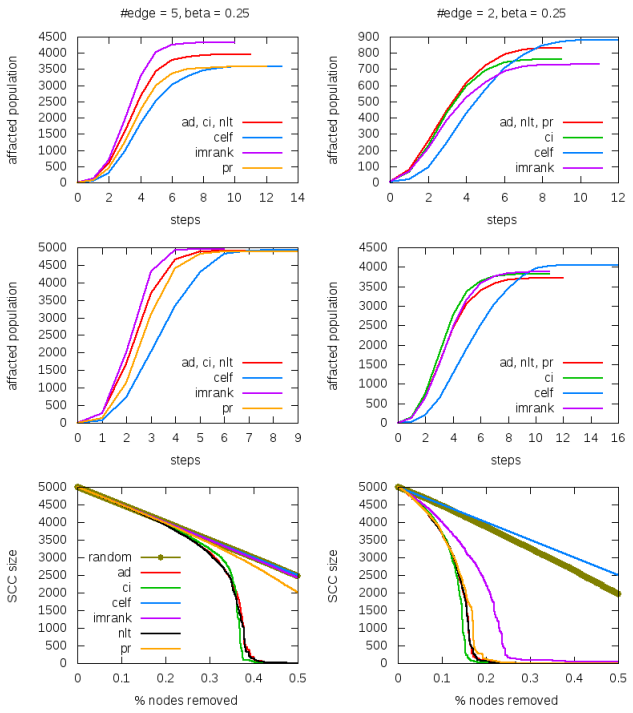

Figure 1: Independent cascades and giant component attacks for a geometric perferential attachment graph. **top:** Independent cascades with propagation probabilities drawing from a beta distribution with mean 0.5. **middle:** Independent cascades with propagation probailities set to 1.0. **bottom:** Giant component size vs. the fraction of nodes removed. **left:** $\beta = 0.25$, number of edges to attach to each node is 5. **right:** $\beta = 0.25$, number of edges to attach to each node is 2.

Nodes with the highest k-Shell scores are selected as the most influential nodes.
We also apply more traditional methods, i.e. the PageRank and the adaptive degree (AD). AD ranks nodes based on their out-degree (directed) or degree (undirected), and removes the selected nodes from the graph as it proceeds.

5. DATA

We measure influential node discovery techniques in the communication, social and financial domains using four separate datasets

1. Autonomous system (AS) graph: directed autonomous system graph of relationships captured by CAIDA on Nov. 05 2007.

2. Web-stanford graph: webpages from stanford.edu and the hyperlinks between them. This directed graph was collected in 2002.

3. Facebook graphs: Stanford and UCSD facebook intra-school data collected in 2005 [2].

4. Epinions: social network of a general consumer review site.

5. S&P 500 stock correlation graph: two undirected weighted graph of S&P 500 companies connected based on adjusted stock price correlation generated using Bear Market data (from Oct 2007 to Jan. 2009) and Bull Market data (from Feb. 2009 to Aug. 2015).

A summary of the data used in this paper can be found in Table 1 above.

5.1 Data Collection

5.1.1 Autonomous System (AS)

The AS graph is a modified version of the original CAIDA AS graph captured on 05 Nov 2007. The graph includes an edge attribute to represent the relationship between nodes (peer, provider, customer, sibling). We pruned approximately 400 sibling edges as they do not indicate data flow and were increasing the size of the SCC arbitrarily.

5.1.2 S&P 500

We generate the S&P 500 stock graphs in the similar way adopted by Chi K. Tse et al. [2]. In establishing edges of the network, we use a winner-take-all approach, which makes binary decisions to connect two stocks according to their cross correlation being larger than a threshold value. Specifically, first, we obtained the historical stock daily adjusted closed price and trading volume for the stocks listed in S&P500, and calculated the price return. Let $p_i(t), v_i(t)$ and $r_i(t) = \ln p_i(t)/p_{i-1}(t)$ be the adjusted closed price, trading volume and price return of stock i on day t. Next, we screened out those stocks which were not traded every day over the periods of time being considered. Two time periods were considered in our analysis: the *Bear Period* from October 2007 to January 2009, and the *Bull Period* from February 2009 to August 2015. Then, we calculated the cross correlation of the time series of the stocks daily adjusted closed price, trading volume and price return. If the absolute value of the correlation coefficient ρ is larger than a threshold value, e.g. 0.8, we consider this pair of stocks to be connected, and add an edge between them. We built two graphs for further analysis, $\rho = 0.9$ for Bear Period, and $\rho = 0.95$ for Bull Period, based on daily adjusted closed price.

6. RESULTS

6.1 Artificially Generated Network

We applied our influential node identification algorithms to an artificially generated geometric preferential attachment graph [1] to provide an understanding of how our algorithms perform in terms of the total affected population size under various cascade scenarios and the ability to reduce the SCC size during giant component (GC) attacks. We generated two baseline graphs with $\beta = 0.25$ and 5,000 nodes each. The first and second baseline graphs have 5 and 2 edges to be attached to each node, respectively. The β parameter ($0 \leq \beta \leq 0.5$) controls the attachment radius from within which a node selects another node to connect to.

Figure 1 illustrates the Independent Cascade (IC) and giant component attack results using the baseline graphs identified above. The ICs are executed using the top $k = 3$ influential nodes identified by our algorithms with propagation probabilities drawn from a beta distribution with $\mu = 0.5$ in the top two subplots and propagation probability of 1 in the middle two subplots. To conduct the GC attacks, nodes are removed from the graph in order of descending influence and the size of the SCC is measured after each iteration. GC attack results are illustrated in the bottom two subplots of figure 1.

Here are some interesting observations from figure 1

- AD (red) and CI (green) perform very well during IC and SCC attacks. CI is the most effective method when attacking the GC as the fewest number of nodes need to be removed to cause the largest amount of impact to the SCC.
- IMRank (magenta) and CELF influential nodes affect large portions of the graph during IC, but perform poorly when breaking down SCC. One possible explanation for CELF GC attack performance is the algorithm seeks nodes with the maximum marginal gain in terms of total affected population. Once a few nodes within the SCC are selected, it may stop selecting nodes from the SCC due to the diminishing marginal return.
- CELF is the best algorithm to affect the most nodes when IC propagation probability is 1.0. The CELF cascade spread more slowly, but surpassed other methods within a few iterations and affected more nodes. This may also be due to the CELF algorithms tendency to not repeatedly pick nodes within a dense region of the graph.
- AD, NLT, and CI generated similar results as explained by the algorithms approach or similarities under certain conditions. NLT is a more advanced implementation of AD. AD ranks nodes based on degree, while NLT performs similar ranking within a hypergraph generated by sampling and traversing the transpose of the original graph. AD and CI generate similar results depending on the shortest path parameter. When l = 0, CI is reduced to AD. Although we used l = 1, CI and AD results can still be similar.
- Pagerank (orange) is an effective algorithm to identify most influential nodes for IC. However, depending on graph density, pagrank may not be a very effective GC or SCC attack method. In a dense graph, the high pagerank nodes may not be within the SCC. The plots at the bottom of figure 1 illustrate the difference in pagerank performance as graph density varies.

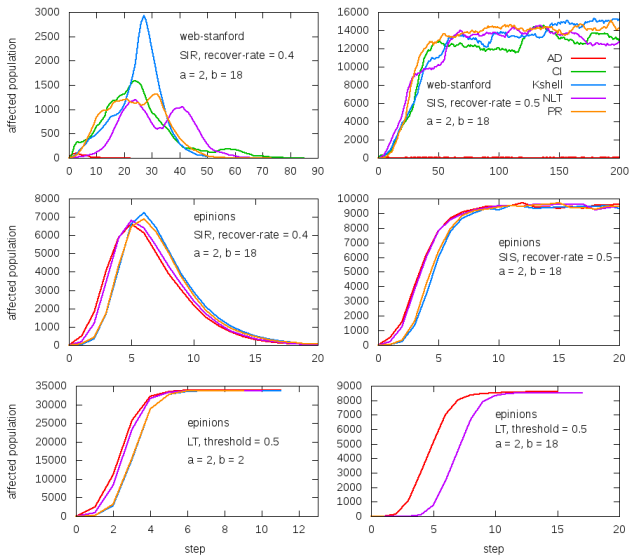

Figure 2: SIR, SIS and LT simulations for different graphs. Propagation probabilities follow beta distributions.

6.2 Real World Complex Networks

We obtained the most influential nodes using the algorithms proposed in section 4 and real world complex datasets identified in section 5. We performed the GC attacks against the AS, epinions and web-stanford graphs as illustrated in figure 3.

We observed AD (red), CI (green) and NLT (magenta) excel in destroying the GC in the AS graph while AD (red) and NLT (magenta) excel in destroying the epinions GC. NLT (magenta) and PR (orange) excel in destroying the GC in the web-stanford graph. Note, the poor performance of IMRank (gray) CELF (brown) overall. These outcomes are similar to our experiments on the geometric preference attachment graphs.

Independent Cascade (IC) (see figure 4), susceptible-infectious-recovered (SIR), susceptible-infectious-susceptible (SIS) and Linear Threshold (LT) cascade (see figure 2) algorithms were executed on several real world complex datasets using $k = 3$ top most influential nodes identified by our influential node algorithms.

Key cascade findings include:

- Cascade performance of our algorithms varied as a function of the beta-distribution. When using large propagation probabilities ($\mu = 0.5$), the cascades produced by the k=3 top influential nodes were very similar (measured by the number of nodes reached) across algorithms. However, the cascade speed varied by algorithm. As the propagation probabilities decreased to $\mu = 0.1$ and $\mu = 0.01$, the effectiveness of the various algorithms varied and cascade differences appear especially within large graphs. For smaller highly connected graphs, the results remained similar. These findings are illustrated in figure 4.
- AD performed very well across several graphs with

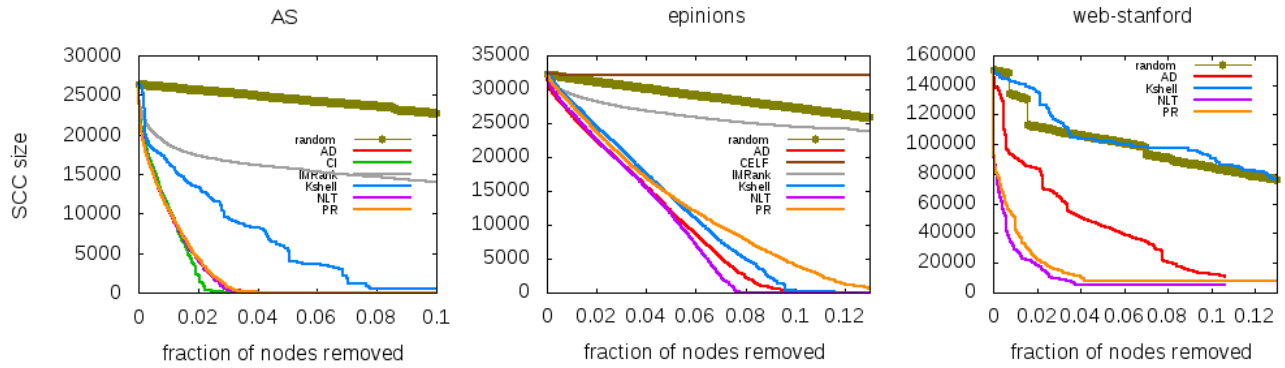

Figure 3: GC attacks on real-world complex networks.

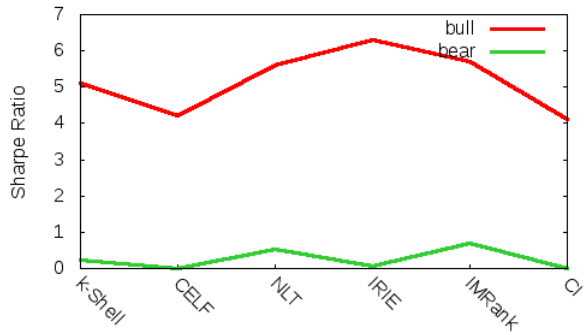

Figure 6: The Shape Ratio of the portfolios constructed by different methods. The S&P 500 baseline is at zero.

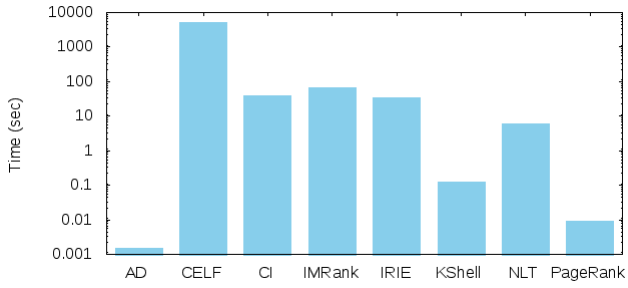

Figure 7: Computing time to select the top 50 most influential nodes from a geometric preferential attachment graph with 5000 nodes and 5 edges attach to each node and $\beta = 0.25$.

the exception of web-stanford. The size of the web-Stanford IC, SIR and SIS cascades using AD most influential nodes were very small as compared with size of simulations using nodes from other algorithms, as evidenced in the top row of figure 4 and 2. A closer examination of the data revealed the $k = 3$ influential nodes did not reside in the SCC and are therefore not effective in spreading influence. This finding may imply ADs effectiveness varies by the graphs level of connectivity. For most relatively small and highly connected networks, AD is a very effective influential node identification method due to accuracy and speed. However, for large and less connected graphs, AD may be a poor choice.

- CI and NLT performed consistently well across all graphs tested. They were among the top performing SCC attack algorithms as indicated in figures 3 and 4..

6.3 Stock Market Network

We applied k-Shell, CELF, NLT, IRIE, IMRank and CI algorithms to find the most influential nodes in the S&P 500 networks. We then built the portfolios containing the top 3 influential stocks returned from each algorithm, calculated their yearly return, standard deviation, and compared these portfolios against the benchmark, S&P500 during the same period.

Figure 6 shows the Sharpe Ratios of the 3-stock portfolios constructed by different algorithms. The Sharpe Ratio is the average return earned in excess of the benchmark portfolio return rate per unit of volatility or risk. In the original definition, it is risk-free rate. Here, in order to make the comparison clear, we replace the risk-free rate with the S&P yearly return rate. One can see that the portfolio based on IMRank is the best during the bear period, and that based on IRIE is the best during the bull market, and during both bear Period and bull Period, all the portfolios created by these algorithms beat S&P 500 returns. We are encouraged to see the portfolio consists of most influential stocks perform better than the S&P 500.

6.4 Algorithm Performance & Computational Complexity

To construct an algorithm performance comparison, we measured the computing time required to identify the top 50 most influential nodes from a geometric preferential attachment graph (5,000 nodes, 5 edges per node, $\beta = 0.25$) as used in Section 6.1. As is evident in figure 7, AD and PageRank are the first and second fastest influential node algorithms, respectively, while CELF is several orders of magnitude slower than all other methods.

During our experiments, we encountered several computational bottlenecks requiring algorithm implementation modifications to ensure viability when analyzing large datasets.

CELF Our initial CELF Python implementation was sufficient to analyze small graphs, however, could not scale to

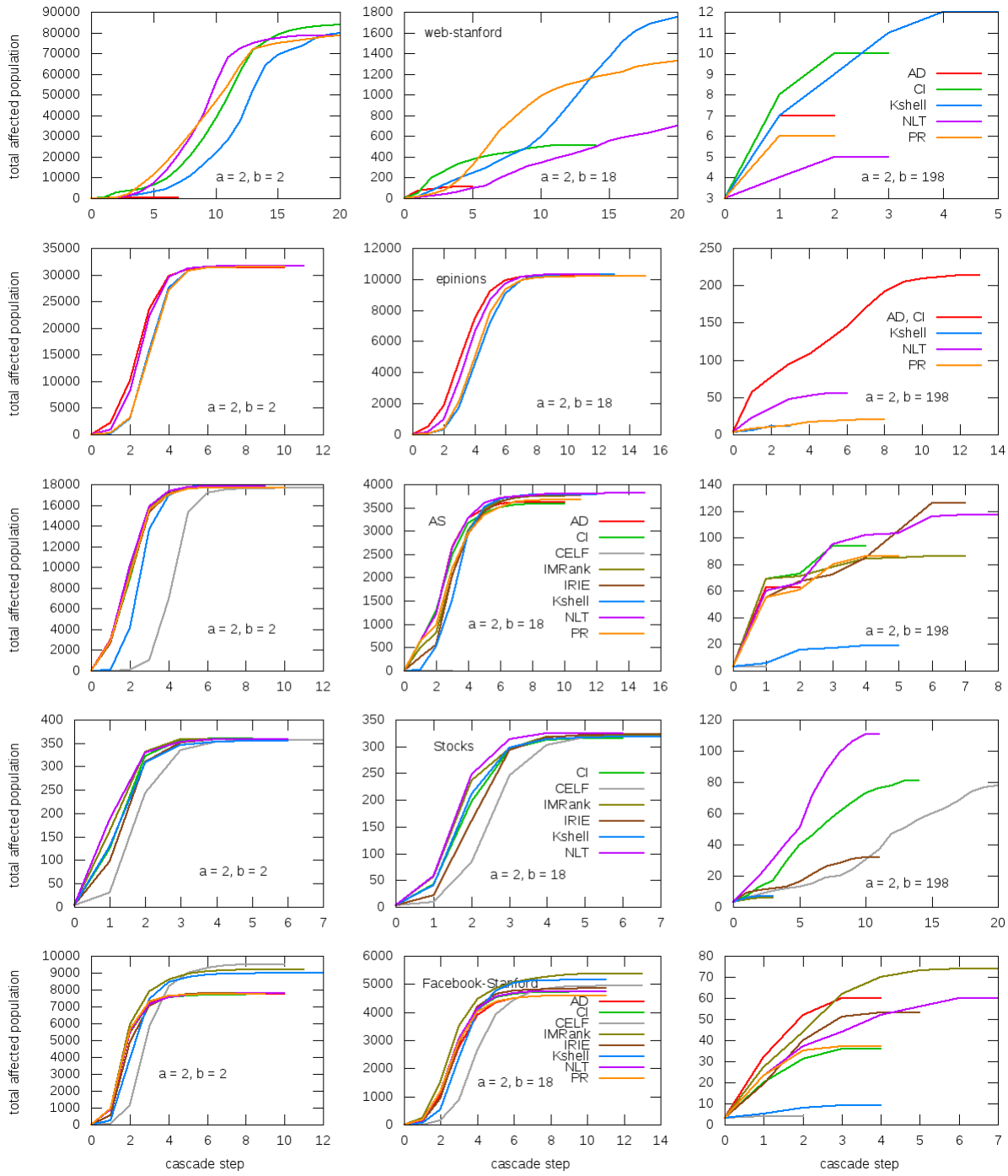

Figure 4: Independent Cascades for different networks. The propagation probabilities follow a beta distribution with $\alpha = 2, \beta = 2$ (1st column), $\alpha = 2, \beta = 18$ (2nd column), $\alpha = 2, \beta = 198$ (3rd column).

analyze the AS graph with 28k nodes using 16GB of main memory (as it retained a set of all reachable nodes for all graph nodes during computation). To resolve this issue, we improved the implementation to write the reachable nodes to disk and read back as needed. We also developed CELF in C++ using SNAP C++ 2.4. Together with OpenMP pragmas, we were able to improve CELF performance significantly. Figure 7 states performance using the improved Python implementation.

IRIE Our initial IRIE implementation leveraged Monte Carlo simulations to estimate the node visit probabilities from each seed node. In order to improve performance, we replaced the Monte Carlo simulations with MIOA estimation as mentioned in previous sections.

Our initial or modified algorithm implementation scaled to the size of our graphs successfully, however, several algo-

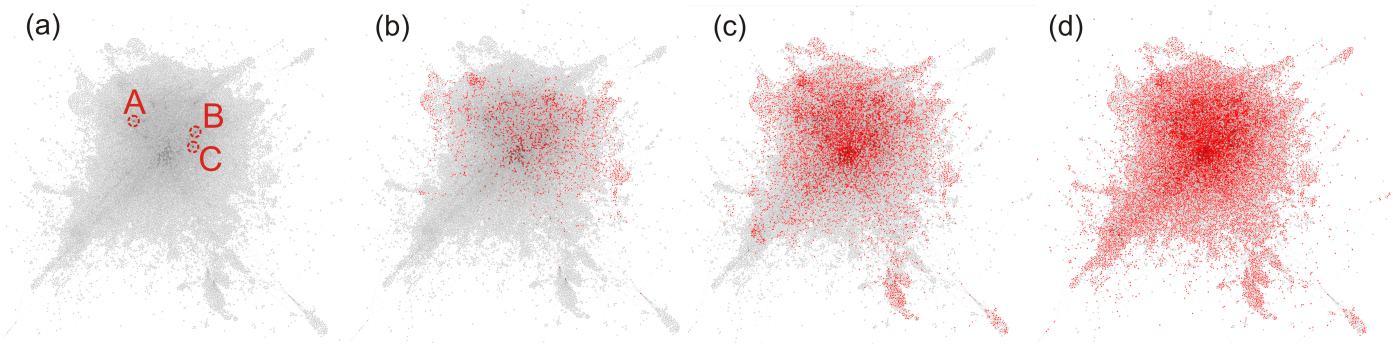

Figure 5: Step 0, 1, 2 and 8 (final) of an independent Cascade on the autonomous system network starting from three most influential nodes identified by the NLT algorithm.

rithms took many hours to run and many benefit from different computing paradigms or larger hardware.

7. CONCLUSION

In this paper, we successfully implemented and compared six influential node identification methods (CELF, NLT, IRIE, IMRank, CI, k-Shell) and simulated the corresponding influence cascade using a mix of four cascade approaches (IC, SIR, SIS, LT) across seven real world complex datasets. We augmented our datasets with an artificially generated geometric preferential attachment graph to provide a baseline for our analysis.

We also studied the extent to which influential nodes identified by various methods can be leveraged to disrupt the connectivity of the graph by observing how the size of the SCC changes as nodes are sequentially removed in order of descending influence.

Additionally, we incorporated a high level performance analysis to compare the performance of influential node identification methods with the computational complexity and runtime.

Our experiments illustrate a division of capabilities across influential node identification methods. CI and NLT are well suited for both identification of influential nodes and disrupting graph connectivity while IMRANK and CELF are well suited for one task but not the other. AD is the most computationally efficient method and is performs well in highly connected networks, but may not be the best method for large or disconnected networks.

8. REFERENCES

[1] A. Flexman, A. Frieze, J. Vera, A geometric preferential attachment model of networks, WAW 2004.

[2] A. L. Traud, P. J. Mucha, M. A. Porter, Social Structure of Facebook Networks, arXiv:1102.2166, 2011.

[3] C. Borgs, M. Brautbar, J. Chayes, B. Lucier, Maximizing Social Influence in Nearly Optimal Time, arXiv:1212.00884v4, 2013.

[4] C.K.Tse, J. Liu, F.C.M. Lau, A Network Perspective of the Stock Market, Journal of Empirical Finance, 2010.

[5] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 137146. ACM, 2003.

[6] F. Morone, H. A. Makse, Influence maximization in complex networks through optimal percolation, Nature, 524, 65, 2015.

[7] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 420429. ACM, 2007.

[8] K. Jung, W. Heo, W. Chen, IRIE: Scalable and Robust Influence Maximization in Social Networks, ICDM12, 2012.

[9] M. Faloutsos, P. Faloutsos, C. Faloutsos, (1999). On Power-Law Relationships of the Internet Topology, Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication, 1999.

[10] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanle, H. A. Makse, Identification of Influential spreaders in Complex networks, Nature Physics, 6, 888893, 2010.

[11] N. Koochakzadeh, F. Keshavarz, A. Sarraf, A. Rahmani, K. Kianmehr, M. Rifaie, Reda Alhajj, J. Rokne, Stock Investment Decision Making: A Social Network Approach, Studies in Computational Intelligence pp 47-57, 2011.

[12] R. Albert, H. Jeong, A. Barabasi, Error and attack tolerance of complex networks, Nature, 406, 378, 2000.

[13] S. Cheng, H. Shen, J. Huang, W. Chen, X. Cheng, IMRank: Influence Maximization via Finding Self Consistent Ranking, arXiv:1402.3939v1, 2014

[14] W. Chen, C. Wang, and Y. Wang, Scalable influence maximization for prevalent viral marketing in large-scale social networks, KDD, 2010.

[15] W. Chen, Y. Wang, S. Yang, Efficient Influence Maximization in Social Networks, KDD, 2009.