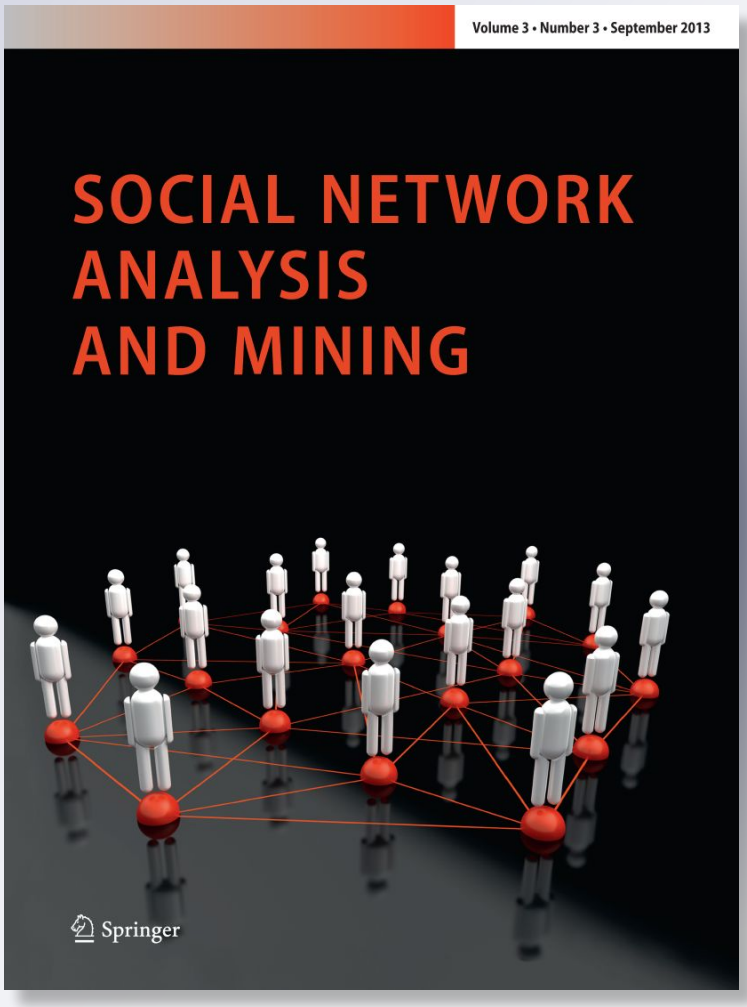
Performance of procedures for identifying influentials in a social network: prediction of time and memory usage as a function of network properties

P. M. Krishnaraj, Ankith Mohan & K. G. Srinivasa

Social Network Analysis and Mining

ISSN 1869-5450
Volume 7
Number 1

Soc. Netw. Anal. Min. (2017) 7:1-11
DOI 10.1007/s13278-017-0454-1



Springer

Soc. Netw. Anal. Min. (2017)7:34
DOI 10.1007/s13278-017-0454-1

ORIGINAL ARTICLE

Performance of procedures for identifying influentials in a social network: prediction of time and memory usage as a function of network properties

P. M. Krishnaraj[1] · Ankith Mohan[1] · K. G. Srinivasa[2]

Received: 31 January 2017 / Revised: 2 June 2017 / Accepted: 19 July 2017

Abstract Identification of influential nodes in a social network is an interesting problem since these nodes assist in faster information propagation in the network. In the current work, the procedures based on Cole and Weiss are used to identify influentials in networks of various sizes and their performances are analyzed. Further it is established that preprocessing techniques like clustering and sampling improve the performance of Cole and Weiss procedures. Finally, the computational resources required for these procedures are estimated based on network properties.

1 Introduction

1.1 Purpose of study

Statement of the problem For a given social network—(1) apply the procedures given by Cole and Weiss to find influentials, (2) apply preprocessing techniques to check the changes in the performances of procedures, and (3) predict the time and memory required to find the influentials based on its network properties.

✉ K. G. Srinivasa
kgsrinivasa@gmail.com

P. M. Krishnaraj
krishnarajpm@gmail.com

Ankith Mohan
ankith.mohan@gmail.com

[1] Department of Information Science and Engineering, Ramaiah Institute of Technology, Bengaluru, India

[2] Department of Information Technology, CBP Government Engineering College, Jaffarpur, New Delhi, India

2 Literature review

A social network is a collection of individuals who use a common medium to communicate with each other and share information. Social network can be formally defined as a graph $G(V, E)$ where the individuals comprising the network are vertices, V, and the communication among these individuals are edges, E.

Social influence is said to occur when an individual adapts his own behavior, attitude or belief to that of others in the network (Leenders 2002). The influentials in a network can be "co-creators" or "co-adapters" for the propagation of a program or reform (Kempe et al. 2005). These influentials can play a significant role in the introduction and longevity of program implementation (Riggan and Supovitz 2008; Valente and Pumpuang 2007). The idea of identifying influentials in a network has not been examined sufficiently although comprehensive research to investigate influence in networks using social network analysis exists (Cole and Weiss 2009). Given a social network S, having certain network properties (average path length, density, etc) and semantics (age, gender, carrier of pathogen, etc), Kang et al. (2012, 2016) computed the importance of vertex v in S using diffusion centrality. Diffusion centrality of vertex v, $dc(v)$, in S with respect to a property p following a diffusion model D is calculated as the difference in the amount the diffusion that would result if v has the property p and the amount of diffusion that would occur if v did not possess the property p. Moores et al. (2014) used combinatorial local centrality to find optimal set of individuals in a complex network that follows susceptible-infected-recovered (SIR) model to initiate an epidemic. Given a social network $G(V, E)$ having V vertices and E edges and topic distribution $\theta_v \in \mathbb{R}^T \ \forall \ v \in V$, Tang et al. (2009) found topic-level influence network

Published online: 29 July 2017

$G_z = (V_z, E_z) \forall\ z \in [1, T]$. V_z is subset of nodes related to topic z, $E_z = (v_s, v_t, \mu_{st}^z)$ is the set of pairwise weighted influence relations over V_z, where each edge is from node v_s to v_t with weight μ_{st}^z. From this topic-level influence network, nodes having the highest probability to influence other nodes on the topics along with their corresponding edges are identified. Given V vertices and T topics, for every vertex $v \in V$, topic distribution θ_v is defined as $\theta_v = \begin{bmatrix} \theta_{v1} \\ \theta_{v2} \\ \vdots \\ \theta_{vT} \end{bmatrix} \in \mathbb{R}^T$ where each element θ_{vz} is the importance of the node on topic z and $\Sigma_z \theta_{vz} = 1$. In Aral and Walker (2012), instead of taking an individual's importance in the diffusion of behavior as only dependent on his individual attributes or personal network characteristics, this paper shows that the joint distribution of influence, susceptibility and likelihood of spontaneous adoption in local network around individuals together determine their importance to the propagation of behavior. Influentials are identified in Ma et al. (2016) using the k-shell decomposition analysis, and the distance between multiple influentials plays a pivotal role in determining the extent of diffusion.

This paper, however, does not concern itself with the diffusion model of the property to be propagated by the nodes and topicwise influentiality. Instead, the focus is on the identification of those nodes that distinguish themselves from others solely by virtue of their network properties.

2.1 Social network analysis

Social network analysis presumes that individuals are interconnected and that communication among individuals defines this interconnectedness (Wasserman et al. 1994). The interconnectedness among individuals can be represented as a sociogram. Birk (2005) had used sociograms to identify influentials in a network. Centrally situated individuals are considered to be sources of influence in a sociogram. However, different researchers might perceive sociograms differently and come to different conclusions about influentials. To overcome this subjective interpretation of sociograms, statistical techniques have been developed to identify influentials in a network. While methods of contagion (Marsden and Friedkin 1993) or selection (Frank 1998) explain the influence in a network, their intent is not to identify influentials.

Social network data represent communication patterns among individuals in a network. Using Freeman's (1979) in-degree centrality (also known as in-ties), the total communication directed at each individual can be measured. This distribution of in-ties for all individuals is sorted, and a scree plot is obtained. This plot gives the distribution of in-degree for all individuals in the network. From this scree plot, a defensible threshold for identifying influential individuals can be established. Cattell (1966) identified a point of inflection and categorized those individuals with scores situated above this point as influential. However, such a subjective method of analyzing the scree plot may be subject to potential research bias.

As an alternative, Cole and Weiss (2009) have proposed a set of four statistical methods that use the concept of in-degree distribution to identify influentials in a network. In the current work, Cole and Weiss's methods have been adopted to identify influentials. Two new preprocessing methods have been proposed to improve the time and memory utilized in identifying the influentials by the Cole–Weiss procedure. It is shown by extensive analysis that when a network is not of small size, the efficiency of the Cole–Weiss procedure exhibits a marked improvement when these preprocessing methods are incorporated. Additionally, prediction equations have been derived to predict the time and memory needed to identify the individuals when these procedures are adopted.

3 Methods

In this section, Cole and Weiss's (2009) procedures are provided followed by a brief introduction to the preprocessing methods.

3.1 Cole–Weiss procedures

3.1.1 Method 1: Absolute cut score

This method is based on a predetermined absolute score above which individuals are categorized influential and below which they are not. This is achieved by drawing a horizontal line at the vertical height of the chosen score over the scree plot. Those individuals whose in-degrees are above this line are deemed influential.

This method is based on a single point (score) and does not focus on the variation in the in-tie distribution. Thus, it is possible that every individual or no individual in the network may be classified as influential.

3.1.2 Method 2: Fixed percentage of population

Another method is to select a fixed percentage of the population as influential. Graphically, a vertical line is drawn on the scree plot. If the top 10% of individuals in an organization are to be categorized as influentials, this is done by selecting the leftmost (10%) individuals in the graph.

This method also does not account for variation in the in-degree distribution, but ensures that a given percentage of the network population are deemed influential based upon their in-ties relative to that of others.

3.1.3 Method 3: Standard deviation

In this method, the mean and standard deviation of the in-degrees are initially calculated. A horizontal line which is x (such as 2 or 3) standard deviations above the mean is drawn over the scree plot. Those individuals whose in-degrees fall above this line are classified as influentials.

The method greatly depends on the in-ties distribution since the in-degree scores of individuals are compared against the average level of in-degrees in standard deviation units.

3.1.4 Method 4: Random permutation

Through the use of random permutations, it is possible to identify those individuals who received a significantly greater number of in-ties than would have been possible by chance alone. However, sampling distribution of networks that would have occurred, conditional on the fixed marginals, needs to be generated for this purpose. In order to obtain the sampling distribution of influence for the network, individuals' out-degrees are randomly reassigned to other individuals in the network. Influence scores are now recalculated. Both the actual and random influence scores are sorted, and each individual score in the actual dataset is compared with that of the individual of the same rank in the random dataset. A large number (say, 1000 or 5000) of permutations of random out-degree allocations are drawn in order to create a sampling distribution of influence subject to conditional independence (Buja and Eyuboglu 1992). Since the row marginals (that of out-degrees) are fixed, the ties are not completely independent as their new destinations are restricted to only emanate from their original sources in the actual data. This restriction helps in creating a sampling distribution of influence that is comparable to the actual data. The result is a distribution that would arise by random chance and can be used to identify those influentials whose influence is statistically greater than random chance. Given a certain level α (say $\alpha = 0.05$), if an individual's actual influence score is higher than his ranked counterpart for 95% of random permutations, then that individual is labeled as a 'significant influential.'

As discussed above, the 'Random Permutation' method requires random assignment of in-degrees when the out-degree distribution is fixed. To do this, one needs to simulate from such a distribution. A possible method for this is to model the given network (graph) with the exponential random graph model (ERGM) and then to simulate from the fitted ERGM. If Y denotes the adjacency matrix of the network, then an ERGM for it is given by the probability density

$$P(Y = y|\theta) = \frac{\exp(\theta' s(y))}{c(\theta)},$$

where $s(Y)$ is the statistic which is supposed to explain the network behavior, θ is the associated parameter and $c(\theta)$ is the normalizing constant of the probability density. θ is normally estimated by the maximum likelihood estimation (MLE) method. Simulation is then performed from the fitted model subject to the constraint that the out-degrees are fixed. This is the procedure implemented in this manuscript.

3.1.5 Stringency of cutpoint

Each of these methods requires a stringency level for a cut point. The number of in-ties required for absolute cut score, percentage of individuals to identify for fixed percentage of population, number of standard deviations above the mean for the standard deviation method and the α level for random permutation method.

3.2 Preprocessing methods

3.2.1 Method 1: Cluster analysis

All of the methods in Sect. 3.1 are observed to perform well for small networks. However, as the size of the networks increases, the performance shows a deteriorating trend (see Sect. 5). To combat this deterioration in performance, the networks are subjected to cluster analysis. This ensures that closely interconnected individuals are grouped into clusters such that each individual is in greater communication with individuals in the same cluster than with those in other clusters. Although there are many techniques for cluster analysis, the fast greedy clustering algorithm (Clauset et al. 2004) is used in this study to identify clusters in a network. Once all the clusters in the network are identified, each of the obtained clusters is then subjected to the four methods to identify influentials. In this study, only those clusters which have more than 10 edges are considered since the smaller clusters are highly unlikely to contain an influential element of the large network being studied. Thus, the idea is to divide the large network into homogeneous sub-clusters before applying methods of influential analysis to them.

This preprocessing technique allows the differentiation of those links that exist within clusters from those that lie between clusters. By disregarding the links that exist between clusters, the number of links to be considered for

identifying influentials can be greatly reduced, thus improving performance while finding influentials.

3.2.2 Method 2: Sampling

The cluster analysis preprocessing method is observed to improve the performance of influence identification. However, as the size of the networks increases further, from moderate to large, cluster analysis-based influence identification also begins to suffer from inefficiency (see Sect. 5). To counter this drop in performance, yet another preprocessing technique using random sampling methods is proposed. The networks will be first subjected to sampling and then to cluster analysis, and finally, the methods to identify influentials will be applied.

In this preprocessing technique, first random subnetworks of increasing size from the given network are sampled until a 'stable subgraph' and the clusters therein are identified. A 'stable subgraph' is a subgraph that satisfies the conditions (a stopping rule) provided in the *STABLE GRAPH* procedure of Algorithm 4. These conditions ensure that the number of 'major clusters' within the random subnetworks stabilizes at this size of the random subnetwork. A 'major cluster' is defined to be a cluster which contains greater than 1% of the number of vertices of the network. The basic idea behind this algorithm is as follows. Assume that the number of clusters increases as a smooth function, say $g(n)$ (such as a cubic) of the sample size until a certain point, and then stabilizes. This point of inflection is expected to be the root of the derivative of a certain order of $g(n)$. Then, to identify this point one could use the method of finite differences for approximating the derivatives of $g(n)$ and check where the higher-order differences change sign.

After this step, all the vertices that are not present in any of the major clusters identified in the 'stable subgraph' are collected and placed in a list called 'not yet.' A vertex is randomly selected from 'not yet,' and the pagerank (Brin and Page 1998) of this selected vertex is calculated in each of the previously identified clusters. Once the pagerank in all the clusters are found, the chosen vertex is allocated to the cluster where it has the highest pagerank. If the selected vertex cannot be placed in any of the existing clusters, it is allocated to a new cluster called 'introverts.' This process of randomly selecting a vertex, computing its pagerank and allocating it to an existing cluster or to the 'introverts' cluster is performed for each and every vertex in 'not yet.' Each of the existing clusters is now subjected to the influential identification methods. However, the 'introverts' cluster is first subjected to cluster analysis before applying the influential identification methods.

The idea behind using random sampling is that probabilistic results then naturally apply. The important result that will help is the 'law of large numbers.' According to this, as the sample size grows, sample mean will converge to the population mean and sample proportion will converge to the corresponding population proportion. Therefore, under suitable regularity conditions (which are generally assumed) influentials determined by a sufficiently large random sample from any given network (however large) will be most likely the influentials of the entire network itself.

4 Implementation

Details on the implementation of the three procedures in Sect. 3 are as follows. As mentioned, the basic procedure is that of Cole and Weiss, comprising four distinct methods. Pseudocode for all the involved algorithms is sketched as follows.

4.1 Cole–Weiss methods

Pseudocode for a procedure called *COLE–WEISS* for invoking the four influence identification methods, Sect. 3.1, of Cole and Weiss is presented here. This procedure takes two arguments, a graph object $G(V, E)$ where $E = (u, v)\ \forall\ u, v \in V$, and four lists namely abs_s, fix_s, sd_s and $rand\_s$ each corresponding to the stringencies described in Sect. 3.1.5. A function *simplify*(G) removes self-loops and multiple edges in G. Each of the influential identification procedures returns an object comprising each stringency value and the respective identified influentials when the procedures are invoked with G and the corresponding stringency list. Thus, the *COLE–WEISS* procedure returns each of the four objects returned to it by the influence identification methods.

Algorithm 1 Cole Weiss

```
1: procedure COLE-WEISS(G, abs_s, fix_s, sd_s, rand_s)
2:     simplify(G)
3:     abs_r ← ABSCUT(G, abs_s)
4:     fix_r ← FIXCUT(G, fix_s)
5:     sd_r ← SDCUT(G, sd_s)
6:     rand_r ← RANDCUT(G, rand_s)
7:     return (abs_r, fix_r, sd_r, rand_r)
8: end procedure
```

The pseudocode for each of the influential identification methods is presented in 'Appendix.' Each of the procedures takes two parameters, a graph object $G(V, E)$ where $E = \{(u, v)\ \forall\ u, v\ \in\ V\}$, and a list of stringencies (refer Sect. 3.1.5). Each procedure returns an object comprising each stringency and the corresponding identified influentials.

4.1.1 Absolute cut score method

The pseudocode for this method is provided in Algorithm 7 as a procedure called *ABSCUT*.

4.1.2 Fixed percentage of population method

For this method, the pseudocode is described in Algorithm 8 as a procedure named *FIXCUT*. The procedure uses two functions:

- *sort_in_descending*(*mylist*): Returns a list of the elements in *mylist* sorted in descending order.
- *head*(*mylist*, *num*): Returns a list of the first *num* elements of *mylist*.

4.1.3 Standard deviation method

The pseudocode for this method is provided in Algorithm 7 as the procedure, *SDCUT*.

4.1.4 Random permutation method

For the method, the pseudocode is described in Algorithm 8 as a procedure, *RANDCUT*. This procedure takes an additional parameter, *numsims*, which denotes the number of simulations to be performed. The procedure uses the following functions:

- *ergm*($matrix \sim edges$): Returns an ERGM model fitted on *matrix* based on edges. If the coefficient of the fitted model is ∞, then the model is saturated.
- *sim*($model$, $\sim out, numsims$): Returns as matrices, *numsims* simulations of *model* constraining on out-degrees.
- *quantile*($list, prob = p$): Returns the value corresponding to the $(1-p)$th quantile of *list*.

4.2 Cluster analysis

The pseudocode for the method is presented as the procedure, *INF-CLUSTER*. The arguments are the same as that of *COLE–WEISS*, a graph object *G*(*V*, *E*), and four lists, *abs_s*,*fix_s*, *sd_s* and *rand_s*. The procedure initially removes self-loops and multiple edges in *G* using *simplify*(*G*). Then, *G* is clustered using the function *cluster_fast_greedy*(G) which uses the fast greedy algorithm to identify clusters in *G* and returns them as graph objects. For each of the clusters having number of edges greater than 10, *COLE–WEISS* procedure is invoked with that cluster and the list of stringencies. *INF-CLUSTER* returns four objects, with each object containing the clusters, the corresponding stringency values and the influentials identified.

Algorithm 2 Influence Cluster

```
1: procedure INF-CLUSTER(G, abs_s, fix_s, sd_s, rand_s)
2:     simplify(G)
3:     clusters ← cluster_fast_greedy(G)
4:     abs_res, fix_res, sd_res, rand_res ← φ
5:     for each cluster ∈ clusters do
6:         if length(E(cluster)) > 10 then
7:             (abs_r, fix_r, sd_r, rand_r) ← COLE − WEISS(cluster, abs_s, fix_s, sd_s, rand_s)
8:             abs_res[cluster] ← abs_r
9:             fix_res[cluster] ← fix_r
10:            sd_res[cluster] ← sd_r
11:            rand_res[cluster] ← rand_r
12:        end if
13:    end for
14:    return (abs_res, fix_res, sd_res, rand_res)
15: end procedure
```

4.3 Sampling

The pseudocode for the method is presented as the procedure, *INF-SAMPLE*. The arguments are the same as that of *COLE–WEISS* procedure, a graph object *G*(*V*, *E*), and four lists, *abs_s*,*fix_s*, *sd_s* and *rand_s*. The procedure takes four additional arguments, *abs_i_s*,*fix_i_s*, *sd_i_s* and *rand_i_s* which are the stringencies for the *introverts* graph. *simplify*(*G*) is used to remove the self-loops and multiple edges from *G*. The procedure *STABLE GRAPH*(G) is called to obtain the stabilized graph and the clusters therein. Only those clusters having number of vertices greater than 1% of the total number of vertices in *G* are used for further computations. *NOT CLUSTERED*($G, clusters$) is invoked to find all those vertices in *G*(*V*, *E*) that have not been placed in any of the clusters in *clusters*. For each vertex randomly selected from the list of vertices that have not been clustered, its pagerank is calculated in each of the previously obtained clusters using *PR CLUSTERS*. The randomly selected vertex is allocated to the cluster where it has the highest pagerank using *add_vertex_edges*. If, however, the vertex cannot be allocated to any of the existing clusters, then it is appended to a list called *introverts_v*. Once this process is completed, each of the clusters having number of edges greater than 10 is passed as argument to *COLE–WEISS* along with the stringency list. The graph corresponding to the vertices in *introverts_v* is extracted from *G* using *randomly_extract_sample*. This graph is then passed as parameter to *INF-CLUSTER*. Thus, the clusters and the influentials identified for the various stringency values are finally obtained.

The following functions are used by the procedure:

- *random*(*mylist*, 1): Returns a randomly selected element from *mylist*.
- $mygraph[myindex] \cdot add\_vertex\_edges(myvertex, G)$: Adds the vertex *myvertex* and the edges connecting

myvertex and all the other vertices in *mygraph*[*myindex*] that exist in G to *mygraph*[*myindex*].

- *subset_data_frame*$(DF(from, to), introverts\_vertices)$: Returns a data frame $DF_{sub}(from_{sub}, to_{sub})$ where either $from_{sub}$ or $to_{sub} \in introverts\_vertices$.
- *randomly_extract_sample*$(G, mynum)$: Returns a graph, $G_{sub}(V_{sub}, E_{sub})$ where V_{sub} is a list of *mynum* number of vertices randomly extracted from V, E_{sub} is the list of edges such that $E_{sub} = (u, v)\ \forall\ u, v\ \in\ V_{sub}$.
- G^C: Returns the complement of G.
- *extract_sample*$(G, myvertices)$: Extracts from G a graph object, $G_{sub} = (V_{sub}, E_{sub})$ where V_{sub} is the list of vertices in *myvertices* and E_{sub} is the list of edges such that $E_{sub} = (u, v)\ \forall\ u, v \in V_{sub}$

Algorithm 3 Influence Sample

procedure INF-SAMPLE($G, abs\_s, fix\_s, sd\_s, rand\_s, abs\_i\_s, fix\_i\_s, sd\_i\_s, rand\_i\_s$)
  $simplify(G)$
  $(clusts, sub) \leftarrow STABLEGRAPH(G)$
  $final \leftarrow clusts[length(V(clusts)) > 0.01 * length(V(sub))]$
  $introverts, introverts\_v \leftarrow \phi$
  $not\_yet \leftarrow NOTCLUSTERED(G, final)$
  while $length(not\_yet) > 0$ **do**
    $select \leftarrow random(not\_yet, 1)$
    $not\_yet.remove(select)$
    $prs \leftarrow PRCLUSTERS(select, final, sub)$
    if $max(prs) == NA$ **then**
      $introverts\_v.append(select)$
    else
      $chosen \leftarrow index(max(prs))$
      $final[chosen].add\_vertex\_edges(select, G)$
    end if
  end while
  $abs\_res, fix\_res, sd\_res, rand\_res \leftarrow \phi$
  for each $fin \in final$ **do**
    if $length(edges(fin)) > 10$ **then**
      $(abs\_r, fix\_r, sd\_r, rand\_r) \leftarrow COLE - WEISS(fin, abs\_s, fix\_s, sd\_s, rand\_s)$
      $abs\_res[cluster] \leftarrow abs\_r$
      $fix\_res[cluster] \leftarrow fix\_r$
      $sd\_res[cluster] \leftarrow sd\_r$
      $rand\_res[cluster] \leftarrow rand\_r$
    end if
  end for
  $introverts \leftarrow subset\_data\_frame(G, introverts\_v)$
  $(abs\_i\_r, fix\_i\_r, sd\_i\_r, rand\_i\_r) \leftarrow INF - CLUSTER(introverts)$
  return $(abs\_res, fix\_res, sd\_res, rand\_res, abs\_i\_r, fix\_i\_r, sd\_i\_r, rand\_i\_r)$
end procedure

Algorithm 4 Influence Sample (continued)

procedure STABLE GRAPH(G)
  $major\_clusters\_count, temp, diff \leftarrow \phi$
  $res \leftarrow 0$
  $index \leftarrow 1$
  $percentage \leftarrow 0.1$
  while $percentage < 1$ **do**
    $len \leftarrow length(V) * percentage$
    $subgraph \leftarrow randomly\_extract\_sample(G, len)$
    $clusters \leftarrow cluster\_fast\_greedy(subgraph)$
    for each $cluster \in clusters$ **do**
      if $length(V(cluster)) > 0.01 * len$ **then**
        $major\_clusters\_count[index]++$
      end if
    end for
    $temp[index] \leftarrow major\_clusters\_count[index]$
    if $index > 1$ **then**
      $diff[index - 1] \leftarrow abs(temp[index] - temp[index - 1])$
    end if
    if $index > 2$ **then**
      if $abs(diff[index-1] - diff[index-2]) <= res$ **then**
        break
      end if
      $res \leftarrow abs(diff[index-1] - diff[index-2])$
    end if
    $index++$
    $percentage += 0.1$
  end while
  return $(clusters, subgraph)$
end procedure

Algorithm 5 Influence Sample (continued)

procedure NOT CLUSTERED($G, clusters$)
  $all \leftarrow V$
  $done \leftarrow \phi$
  for each $cluster \in clusters$ **do**
    $done.append(V(cluster))$
  end for
  $not\_clustered \leftarrow done^C$
  return $not\_clustered$
end procedure

Algorithm 6 Influence Sample (continued)

```
procedure PR CLUSTERS(reqd_vertex)
    pageranks, cluster_vertices ← ϕ
    for each cluster ∈ clusters do
        cluster_vertices.append(V(cluster), reqd_vertex)
        reqd_graph ← extract_sample(G, cluster_vertices)
        prs ← page_rank(reqd_graph)
        pageranks.append(prs[reqd_vertex])
    end for
    return pageranks
end procedure
```

5 Results

In this section, first a brief outline of the datasets used is provided. This is followed by a description of the performance of the three major procedures based on Cole–Weiss and preprocessing techniques.

5.1 Datasets

In this paper, three datasets of moderately large size were analyzed to investigate the performance of the procedures under consideration. Table 1 displays the features of these datasets.

5.2 Performance of the procedures

The algorithms were programmed using the R programming language (R Core Team 2016) on a 128 GB RAM, 64-bit Linux system running R version 3.3.1. To study the performance of different methods relative to network size, for each of the datasets, 10–100% of the network was randomly extracted and the influentials were identified in these networks using these methods.

As described previously, the objective is to compare the performance of three procedures that have been proposed for identification of influentials. These are

1. Cole–Weiss methods without any preprocessing,
2. Cole–Weiss methods after clustering, and
3. Cole–Weiss methods after sampling and clustering.

Displayed in Figs. 1 and 2 are, respectively, the time and memory utilized by these three procedures as a function of the most important network properties (as justified later in this section), namely the number of edges and the average degree. It can be clearly seen that preprocessing increases the efficiency of Cole–Weiss very drastically. In particular, clustering leads to significant improvement and a further preprocessing by sampling adds a very substantial benefit.

Now the contribution of network properties in the determination of resource usage has been analyzed. For each of the procedures, regression models have been fitted for response versus predictors. Here the response is either time or memory used by the procedure to identify influentials using all four influence identification methods, and the predictors are network properties: number of vertices, number of edges, density, average degree, assortativity and average path length. As shown in Tables 2 and 3, the

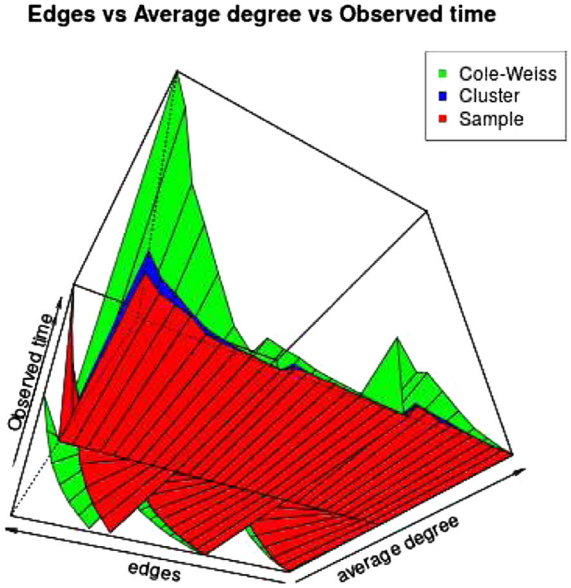


Fig. 1 Comparison of observed time for the three procedures

Table 1 Datasets

	Route view	Pretty good privacy	CAIDA
Vertices	6474	10,680	26,475
Edges	13,895	24,316	53,381
Vertex type	AS	User	AS
Edge type	Comm.	Interaction	Comm.
Directedness	Undirected	Undirected	Undirected
Weightedness	Unweighted	Unweighted	Unweighted
Data source	KONECT (2000b)	KONECT (2000a)	KONECT (2007)

AS autonomous systems, *Comm.* communication

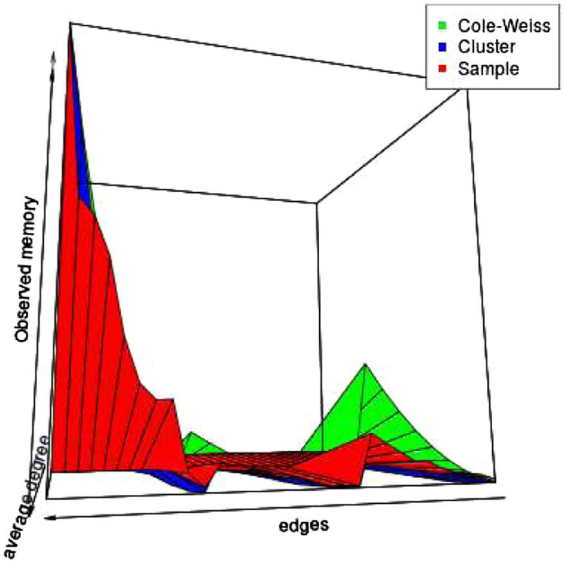


Fig. 2 Comparison of observed memory for the three procedures

network properties which have significant prediction ability are the first four, whereas assortativity and average path length have not been found useful. In some cases, even density or number of vertices is not useful. In these tables, the symbols for the significance levels of p values are given by Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.5 '.' 0.1 ' ' 1.

5.3 Prediction based on network properties

The main objective is to predict the computational resources needed by the procedures to completely identify the influentials in a given network. Toward this end, prediction equations are derived based only on certain network properties. To derive the prediction equation for the response (time or memory) in terms of significant predictors, a fresh regression analysis is conducted by keeping only the significant predictors as covariates. The obtained prediction equations for the three procedures are as follows.

Prediction equations Notation: t = time, m = memory, v = number of vertices, ed = number of edges, den = density, avd = average degree

- *Time*
  - *Cole–Weiss*
    - $t = -8.327 \times 10^3 + 2.974 \times v + 1.186 \times ed + 1.816 \times 10^7 \times den - 8.251 \times 10^3 \times avd$
    - $R^2 = 0.9818$
  - *Cluster*
    - $t = 620.78140 - 0.11501 \times v + 0.24856 \times ed - 425.90168 \times avd$
    - $R^2 = 0.9778$

Table 2 Regression estimates for time versus network properties

	Cole–Weiss	Cluster	Sample
Intercept	−5.343e+03	2.924e+02	1.265e+02
Vertices	2.957e+00 (***)	−9.947e−02 (.)	−7.052e−02 (.)
Edges	1.173e+00 (***)	2.934e−01 (***)	1.588e−01 (***)
Density	1.733e+07 (***)	3.453e+05	1.645e+05
Assort.	2.592e+03	−4.722e+02	−4.526e+02
Avg. deg.	−8.034e+03 (***)	−4.176e+02 (***)	−1.945e+02 (*)
Avg. path.	−5.527e+02	1.270e+01	2.815e+01
R^2	0.9824	0.9802	0.9769

Assort. assortativity, *Avg. deg.* average degree, *Avg. path.* average path length

Table 3 Regression estimates for memory versus network properties

	Cole–Weiss	Cluster	Sample
Intercept	−1.649e+08	2.811e+07	2.986e+06
Vertices	1.156e+05 (***)	−2.878e+03	2.487e+03 (*)
Edges	6.025e+04 (***)	1.234e+04 (***)	2.502e+03 (***)
Density	6.610e+11 (***)	5.152e+10	1.278e+10
Assort.	7.906e+07	−1.549e+07	−4.793e+06
Avg. deg.	−3.208e+08 (***)	−3.953e+07 (***)	−1.245e+07 (***)
Avg. path.	−2.894e+07	−5.042e+06	−1.459e+06
R^2	0.988	0.9715	0.9691

Assort. assortativity, *Avg. deg.* average degree, *Avg. path.* average path length

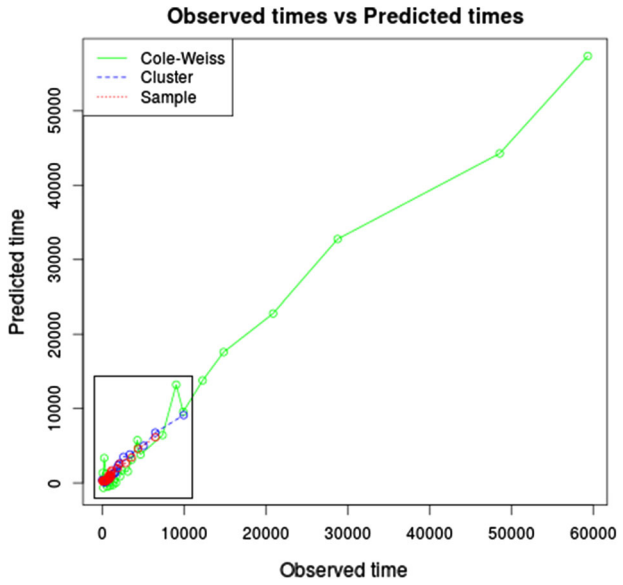


Fig. 3 Comparison of observed time versus predicted time for all of the methods

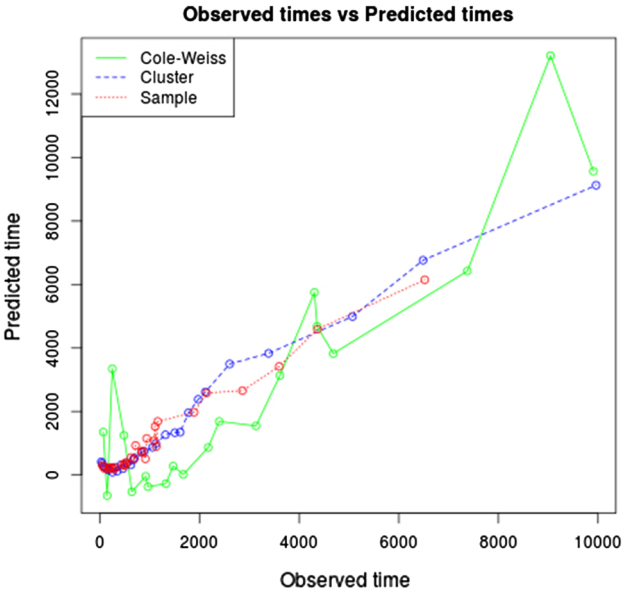


Fig. 4 Enlarged view of the selected portion in Fig. 3

- *Sample*
  - $t = 405.2404 - 0.07602 \times v + 0.16040 \times ed - 199.99952 \times avd$
  - $R^2 = 0.9747$

Now the efficiency of the prediction equations are checked by comparing the predicted time with those observed in the actual computations. Given in Fig. 3 is the observed versus predicted time for the three procedures (Fig. 4).

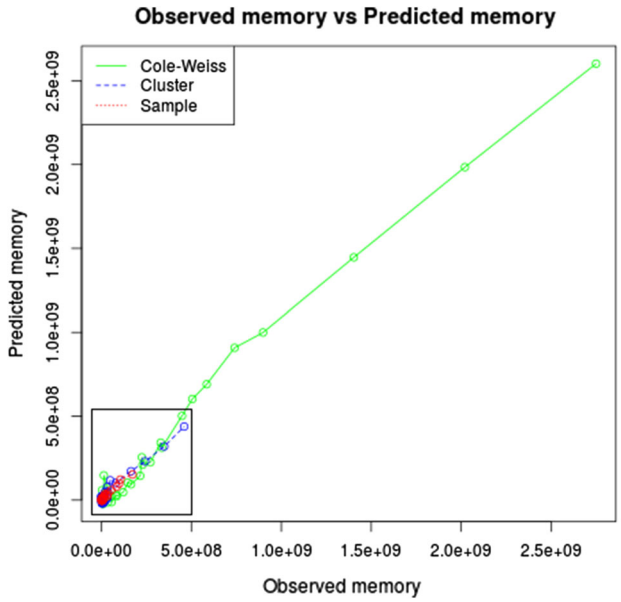


Fig. 5 Comparison of observed memory versus predicted memory for all of the methods

- *Memory*
  - *Cole–Weiss*
    - $m = -3.418 \times 10^8 + 1.205 \times 10^5 \times v + 5.929 \times 10^4 \times ed + 7.519 \times 10^{11} \times den - 3.404 \times 10^8 \times avd$
    - $R^2 = 0.9866$
  - *Cluster*
    - $m = 41646274 - 5646 \times v + 13370 \times ed - 41489372 \times avd$
    - $R^2 = 0.9534$
  - *Sample*
    - $m = 9.741 \times 10^6 + 1.810 \times 10^3 \times v + 2.767 \times 10^3 \times ed - 1.315 \times 10^7 \times avd$
    - $R^2 = 0.9587$.

Efficiency of the prediction equation for the memory consumed can be checked from Fig. 5, which displays the observed versus predicted time for the three procedures (Fig. 6).

5.4 Conclusion and discussion

It may be seen from Figs. 3 and 5 that the time and memory utilized by the three procedures can be predicted very efficiently. This is also reflected by the high R^2 values provided by the regression analysis. This shows that before attempting influence analysis, one can derive the network

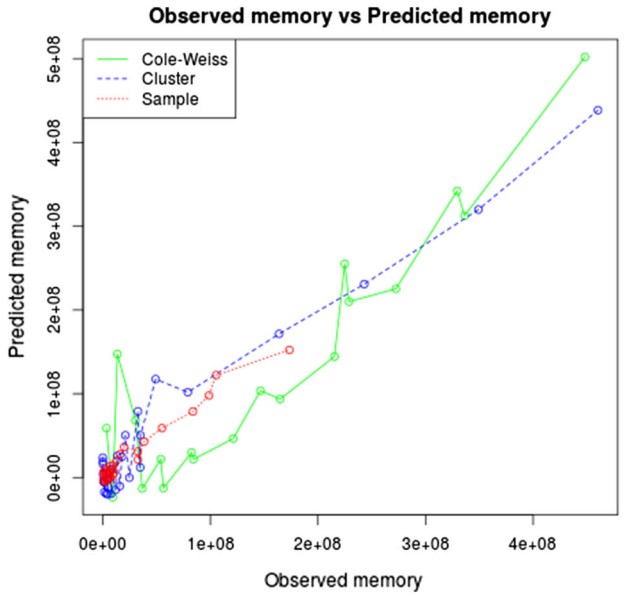


Fig. 6 Enlarged view of the selected portion in Fig. 5

properties and use the prediction equations to obtain an idea regarding how much time and memory will be needed for the computations. It has also established that when the network size is moderate or big, the three procedures can be ranked in decreasing order of efficiency as: Cole–Weiss methods after sampling and clustering; Cole–Weiss methods after clustering; and finally Cole–Weiss methods without any preprocessing. Since the differences between the procedures are so marked, the procedure of Cole–Weiss methods after sampling and clustering is recommended for analyzing any large network.

This paper does not consider the model of information diffusion (Gaussier 2011). The reason for this is that diffusion models are appropriate for certain special situations, whereas the focus is on influential identification. It is possible, however, to extend the paper to include such considerations.

In Sect. 3.2.2, nodes are sampled at random from the network. This is the recommended approach for sampling from a network without any particular characteristics. Alternate approaches well suited to tackle the idiosyncrasies of networks may be applied. For example, given a network that has a higher concentration of nodes in the center than in the periphery, one approach would be to ensure that the probability of sampling nodes from the interior is greater than that of choosing random nodes from the exterior.

Appendix

Algorithm 7 Absolute Cut Score Method

```
1: procedure ABSCUT(DF, stringencies)
2:     G(V, E) ← DF(from, to)
3:     simplify(G)
4:     indeg ← indegree(G)
5:     abscut ← φ
6:     for each str ∈ stringencies do
7:         cutoff ← str * max(indeg)
8:         influentials ← φ
9:         for each v ∈ V do
10:            if indeg[v] > cutoff then
11:                influentials.append(v)
12:            end if
13:        end for
14:        abscut[str] ← influentials
15:    end for
16:    return abscut
17: end procedure
```

Algorithm 8 Fixed Percentage of Population Method

```
1: procedure FIXCUT(DF, stringencies)
2:     G(V, E) ← DF(from, to)
3:     simplify(G)
4:     desc_indeg ← sort_in_descending(indegree(G))
5:     fixcut ← φ
6:     for each str ∈ stringencies do
7:         cutoff ← str * length(V)
8:         influentials ← φ
9:         influentials ← head(desc_indegree, cutoff)
10:    end for
11:    fixcut[str] ← influentials
12:    return fixcut
13: end procedure
```

Algorithm 9 Standard Deviation Method

```
procedure SDCUT(DF, stringencies)
    G(V, E) ← DF(from, to)
    simplify(G)
    indeg ← indegree(G)
    sdcut ← ϕ
    for each str ∈ stringencies do
        cutoff ← mean(indeg) + str ∗ sd(indeg)
        influentials ← ϕ
        for each v ∈ V do
            if indeg[v] > cutoff then
                influentials.append(v)
            end if
        end for
        sdcut[str] ← influentials
    end for
    return sdcut
end procedure
```

Algorithm 10 Random Permutation Method

```
procedure RANDCUT(DF, stringencies, numsims)
    G(V, E) ← DF(from, to)
    simplify(G)
    adj_mat ← get_adjacency_matrix(G)
    indeg ← indegree(G)
    randcut ← ϕ
    for each str ∈ stringencies do
        model ← ergm(adj_mat ∼ edges)
        if model$coef! = Inf then
            results ← sim(model, ∼ out, numsims)
            in_degs ← ϕ
            for each index ∈ numsims do
                mat ← results[index]
                sim_indegree ← indegree(mat)
                in_degs.append(sim_indegree)
            end for
            cutoff ← quantile(in_degs, prob = str)
            influentials ← ϕ
            for each v ∈ in_deg do
                if indeg[v] >= cutoff then
                    influentials.append(v)
                end if
            end for
            randcut[str] ← influentials
        end if
    end for
    return randcut
end procedure
```

References

Aral S, Walker D (2012) Identifying influential and susceptible members of social networks. Science 337(6092):337–341

Birk SM (2005) Application of network analysis in evaluating knowledge capacity. New Dir Eval 107(107):69–79

Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. In: Proceedings of the 7th World-Wide Web conference. Brisbane

Buja A, Eyuboglu N (1992) Remarks on parallel analysis. Multivar Behav Res 27(4):509–540

Cattell RB (1966) The scree test for the number of factors. Multivar Behav Res 1(2):245–276

Clauset A, Newman ME, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70(6):066111

Cole R, Weiss M (2009) Identifying organizational influentials: methods and application using social network data. Connections 29(2):45–61

Frank KA (1998) Quantitative methods for studying social context in multilevels and through interpersonal relations. Rev Res Educ 23:171–216

Freeman LC et al (1979) Centrality in social networks: conceptual clarification. Soc Netw 1(3):215–239

Gaussier E (2011) Models of information diffusion in social networks. In: Proceedings of the second symposium on information and communication technology. ACM, p 2

Kang C, Kraus S, Molinaro C, Spezzano F, Subrahmanian V (2016) Diffusion centrality: a paradigm to maximize spread in social networks. Artif Intell 239:70–96

Kang C, Molinaro C, Kraus S, Shavitt Y, Subrahmanian V (2012) Diffusion centrality in social networks. In: Proceedings of the 2012 international conference on advances in social networks analysis and mining (ASONAM 2012). IEEE Computer Society, pp 558–564

Kempe D, Kleinberg J, Tardos E (2005) Influential nodes in a diffusion model for social networks. In: International colloquium on automata, languages, and programming. Springer, Berlin, pp 1127–1138

KONECT (2000a) Pretty good privacy—network analysis of pretty good privacy—konect. http://konect.uni-koblenz.de/networks/arenas-pgp

KONECT (2000b). Route views—network analysis of route views—konect. http://konect.uni-koblenz.de/networks/as20000102

KONECT (2007) Caida—network analysis of caida—konect. http://konect.uni-koblenz.de/networks/as-caida20071105

Leenders RTA (2002) Modeling social influence through network autocorrelation: constructing the weight matrix. Soc Netw 24(1):21–47

Ma L-L, Ma C, Zhang H-F, Wang B-H (2016) Identifying influential spreaders in complex networks based on gravity formula. Phys A Stat Mech Appl 451:205–212

Marsden PV, Friedkin NE (1993) Network studies of social influence. Sociol Methods Res 22(1):127–151

Moores G, Shakarian P, Macdonald B, Howard N (2014) Finding near-optimal groups of epidemic spreaders in a complex network. PLoS ONE 9(4):e90303

R Core Team (2016) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna

Riggan M, Supovitz JA (2008) Interpreting, supporting, and resisting change: the geography of leadership in reform settings. In: Understanding reform in high schools, The implementation gap, pp 103–125

Tang J, Sun J, Wang C, Yang Z (2009) Social influence analysis in large-scale networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 807–816

Valente TW, Pumpuang P (2007) Identifying opinion leaders to promote behavior change. Health Educ Behav 34(6):881–896

Wasserman S, Faust K (1994) Social network analysis: methods and applications, vol 8. Cambridge university press