Zhan *et al. J Big Data (2016) 3:16*
DOI 10.1186/s40537-016-0050-7

Journal of Big Data

RESEARCH Open Access

Identification of top-K influential communities in big networks

Justin Zhan*, Vivek Guidibande and Sai Phani Krishna Parsa

*Correspondence:
justin.zhan@unlv.edu
Department of Computer Science, Howard R. Hughes College of Engineering, University of Nevada, Las Vegas, USA

Abstract

Because communities are the fundamental component of big data/large data network graphs, community detection in large-scale graphs is an important area to study. Communities are a collection of a set of nodes with similar features. In a given graph there can be many features for clustering the nodes to form communities. Varying the features of interest can form several communities. Out of all communities that are formed, only a few communities are dominant and most influential for a given network graph. This might well contain influential nodes; i.e., for each possible feature of clustering, there will be only a few influential communities in the graph. Identification of such communities is a salient subject for research. This paper present a technique to identify the top-K communities, based on the average Katz centrality of all the communities in a network of communities and the distinctive nature of the communities. One can use these top-K communities to spread information efficiently into the network, as these communities are capable of influencing neighboring communities and thus spreading the information into the network efficiently.

Keywords: Graphs, Top-K communities, Katz centrality

Introduction

Community detection in large networks [1–3] is emerging in various kinds of applications. Network data contain a wealth of information, e.g., top-K nodes and top-K communities based on community strength/influence; when uncovered, this data can bolster predictive models and elucidate general network dynamics. Moreover, when solving complex problems, a diverse set of domains—such as optical character recognition (OCR) analysis; protein complex detection; and community discovery in social networks, neurology, genetics, transportation, social network analysis, structural analysis, and computation—can be represented in the form of graphs and networks for data representation. In particular, social networks are being represented in the form of graphs to address fundamental problems, such as discovering communities in the network [4, 5] or discovering the community that is most likely to contain the query node. Social-network graphs generally consist of nodes that represent users, and community detection on social-network graphs means identifying a set of similar nodes (users). In a network, the bond among the nodes inside a community would be denser than with those outside the community. Many existing clustering algorithms are available that converge the nodes in a graph with good bonding. Out of the entire network, by identifying the top users

of a community or of an entire network as well as top communities based on strength/influence, the focus of interest could be limited to a set of nodes/communities capable of spreading information into the entire network. For example, consider an e-commerce business offer: instead of sending rewards coupons to all the users, instead, identify the top users, who are loyal and frequent buyers, and most likely to use the rewards coupons. Sending these top users rewards coupons will benefit the business. Other strategies can be applied to new or infrequent buyers.

In all previous studies on these problems, a community has been defined as a densely connected sub-graph. According to Faisal et al. [6], this focus ignores another key aspect, namely, influence or importance; these authors presented interesting scenarios that highlighted importance and need to find the most influential communities in a network. Previously, Doo [7] focused on detecting the top-K influential communities in undirected graphs. They defined the influence of a community as the minimum weight of nodes in that community; the top influential community was the one with largest influence value. On the other hand, Du et al. [8] ranked communities according to the strength of each community, which varies with time. In this paper, we define the strength of communities in terms of their average Katz Centralities, taking into consideration each community's distinctive nature. All the communities in this study were to connect to a maximum number of communities. If a community is immediately connected to more number of communities than others are, then it can influence them all. By this means, a message can be propagated to the maximum number of communities present in the graph.

Related work

Ample of work has been done to find the most influential community in a network. One of the most significant methods used include the classic centrality measures, such as degree, betweenness, or similar kind of measures.

Xie et al. [4] proposed a method to extract the community structure, which appeared to be connected by means of a unique spectral property of the graph Laplacian of the adjacency matrix. This group used such structural parameters as algebraic connectivity and node degree distribution for community exploration. Similar to our work, they took into consideration the edge structure; in addition, they used the greedy algorithm for modularity optimization. Li et al. [5] used another approach, which was to study the flooding time, which is the time taken for the information to spread from one node/community to the other node/community. In this approach, processes were considered in which the topology of the graph at time t depended only on their topology at t-1. In their case, most of the dynamic graphs were Markovian and ergodic.

One emphatic approach for detecting the most influential community could be forecasted by detecting the number of nodes whose information radiates the most. One such model was proposed by Ma et al. [9], in which mining of social networks were done using heat-diffusion processes. Based on this, candidate was selected. The basic formula used for undirected social networks was:

$$\frac{\mathrm{f_i}(t+\Delta t)-\mathrm{f_i}(t)}{\Delta t}=\alpha\sum_{\mathrm{j}:(\mathrm{v_j},\mathrm{v_i})\varepsilon \mathrm{E}}\left(\mathrm{f_j}(t)-\mathrm{f_i}(t)\right)$$

where there is a social network graph, G = (V, E), where V is the vertex set and $V = \{v_1, v_2, \ldots v_n\}$. $E = \{(v_i, v_j) | \text{there is an edge from vi to vj}\}$. The value $f_i(t)$ describes the heat at node v_i at time t. $f(t)$ denotes the vector consisting of $f_i(t)$. The heat should be proportional to the time period Δt, and the heat difference $f_j(t) - f_i(t)$. α is the thermal conductivity, and E is the set of edges.

Faisal et al. [6] proposed an approach of identifying the boundary nodes in a community, as they played a vital role in communication for energy-efficient graph processing. They calculated the similarity of the nodes by using the Jaccard similarity, which is given as:

$$\text{Sim}(i, j) = \frac{|Adj(i) \cap Adj(j)|}{|Adj(i) \cup Adj(j)|}$$

where $Adj(i)$ and $Adj(j)$ are the adjacent list of nodes i and j, respectively.

Sweeney et al. [10] used a game theoretic model to detect communities in large networks. Modified Laplacian matrices along with neighborhood similarities were used, and a given network was segregated into dense networks. Kim [11] computed the popularity of a node in a community. Wu et al. [12] described a new method using distance centrality, and detected the communities without a present community number by considering the most central node and determining the similarities among all other nodes. Zhang and Wu [13] found the core nodes for the local community detection, and Mahmood and Small [14] found that each node could only be represented efficiently as a linear combination of nodes spanning the same subspace.

Preliminaries

Katz centrality measures the relative influence of each node in a given network by taking into account the node's immediate neighbors as well as non-immediate nodes that could be connected to the node by way of its immediate neighbors. Similar to Sub graph centrality and Total communicability, Katz centrality covers both local and global influence of a node on the entire network. The matrix resolving $(I - \alpha A)^{-1}$ first was used to rank nodes in a network in the early 1950s, when Katz used the column sums to calculate node importance [15]. The Katz centrality score of a node i was given by either $[(I - \alpha A)^{-1}.1]_i$ or $(I - \alpha A^T)^{-1}.1]_i$, depending on whether broadcast or receiving scores were required (a directed graph) [16]. In an undirected graph in which the Adjacency matrix obtained is a symmetric matrix ($A = A^T$), either of the formulae can be used to compute Katz centrality scores. The column matrix containing all number ones may be replaced by an arbitrary (positive) preference vector, **v** as required. Katz centrality of node i counts all walks beginning at node i, penalizing the contribution of walks of length k by α^k.

$$(I - \alpha A)^{-1} = I + \alpha A + \alpha^2 A^2 + \cdots + \alpha^k A^k + \cdots = \sum_{k=0}^{\infty} \alpha^k A^k \tag{1}$$

The bounds on α ($0 < \alpha < 1/\lambda_1$) ensure that the matrix $I - \alpha A$ is invertible and that the power series in (1) converges to its inverse. The bounds on α also force $(I - \alpha A)^{-1}$ to be nonnegative, as $I - \alpha A$ is a nonsingular M-matrix. Hence, both the diagonal entries

and the row/column sums of $(I - \alpha A)^{-1}$ are positive, and thus can be used for ranking purposes.

Given a graph, G = (V, E), a walk of length *m* denotes a set of *m* nodes $\{v_1, v_2, v_3, \ldots, v_m\}$, and $E = \{e_1, e_2, e_3, \ldots\}$ is the set of edges. Then, A is the adjacency matrix of the network G, denoting the immediate connectivity among the nodes. The Katz centrality of a node v_i is given by:

$$C_{\text{Katz}}(v_i) = \alpha \sum_{j=1}^{n} A_{j,I}\, C_{\text{Katz}}(v_j) + \beta \tag{2}$$

where α is a constant called damping factor, which is usually considered to be less than the largest eigenvalue, λ (i.e., $\alpha < 1/\lambda_1$) and β is a bias constant (also called the exogenous vector), which is used to avoid the zero centrality values. Hence, each node has a minimum, positive amount of centrality that it can transfer to other nodes by referring to them. In particular, when measuring the receiving capacity, the centrality of nodes that are never referred to is exactly this minimum positive amount. When measuring the broadcasting ability of a node, linked nodes have a higher centrality or the centrality of nodes that are never broadcasting to any other nodes. It follows that highly linked nodes have high centrality regardless of the centrality of the neighboring nodes. However, nodes that receive few links still may have high centrality if their neighboring nodes have a large centrality.

From (2), it is evident that Katz centrality is a parameter dependent index, i.e., it depends on α and β. Their values play a decisive role in obtaining Katz centrality values that fluctuate. Different choices of α and β lead to different centrality values, resulting in different node rankings. For instance, if $\alpha \to 0+$, then the Katz centrality reduces to a degree centrality [17]. If $\alpha \to (1/\lambda_1)$, then it reduces to an eigenvector centrality [18]; for example, if $\alpha = (1/\lambda_1)$ and $\beta = 0$, then the Katz centrality is the same as the eigenvector centrality. Hence, these parameters can be taken as a medium to tune between the rankings of nodes based either on a local influence (short walks) or a global influence (long walks).

In case of undirected graphs, both the receiving and broadcasting abilities are alike [16]. However, this is not the case for directed graphs. Table 1 provides the limiting behavior of various schemes.

The right eigenvector of a person would be high if he or she is able to influence someone who already influences a great amount of people. The left eigenvector of a person would be high if someone who gets a great many votes is voting him or her.

Furthermore, (2) can be generalized for the entire graph as:

$$C_{\text{Katz}} = \beta \left(I - \alpha A^{T}\right)^{-1}.\mathbf{1} \tag{3}$$

Table 1 Limiting behavior of various schemes

Method	Limiting ranking scheme			
	Out-degree	In-degree	Right eigenvector	Left eigenvector
$K^b(\alpha)$	$\alpha \to 0+$		$\alpha \to 1/\lambda_{1-}$	
$K^r(\alpha)$		$\alpha \to 0+$		$\alpha \to 1/\lambda_{1-}$

Consider the example in Fig. 1 to understand this concept in detail.

For the given graph in Fig. 1, the corresponding adjacency matrix is as follows:

$$\mathrm{A} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The eigenvalues of A are (2.64, −1.77, −1.00, 0.72, −0.58). With the given eigenvalues, the largest value of A is $\lambda = 2.64$. Assume that $\alpha = 0.2 < 1/\lambda$ and $\beta = 0.3$. Then, the Katz centralities are:

$$\mathrm{C_{Katz}} = \beta\,(\mathrm{I} - \alpha \mathrm{A})^{-1}.1 = \begin{bmatrix} 0.675 \\ 0.567 \\ 0.675 \\ 0.651 \\ 0.423 \end{bmatrix}$$

Therefore, nodes v_1 and v_3 have the highest Katz centrality, and would be the most influential nodes.

Definition: Let $G = (V, E)$ be a strongly connected, directed, unweighted network representing actors and their ties with the adjacency matrix A. $K_i^b(\alpha) = \left[(I - \alpha A)^{-1}.1\right]_i$ be the Katz broadcast centrality of a node i. Similarly, the Katz receive centrality of node I can be obtained as $K_i^r(\alpha) = \left[\left(I - \alpha A^T\right)^{-1}.1\right]_i$.

Now consider Fig. 2.

For the given directed graph in Fig. 2, corresponding adjacency matrix is as follows:

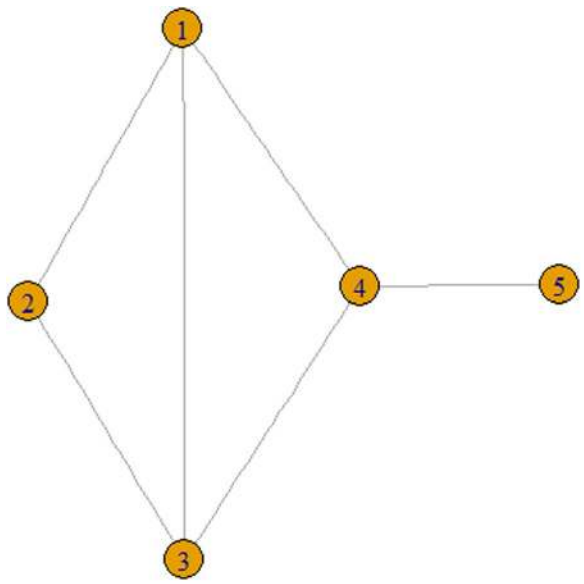

Fig. 1 An undirected graph with five actors

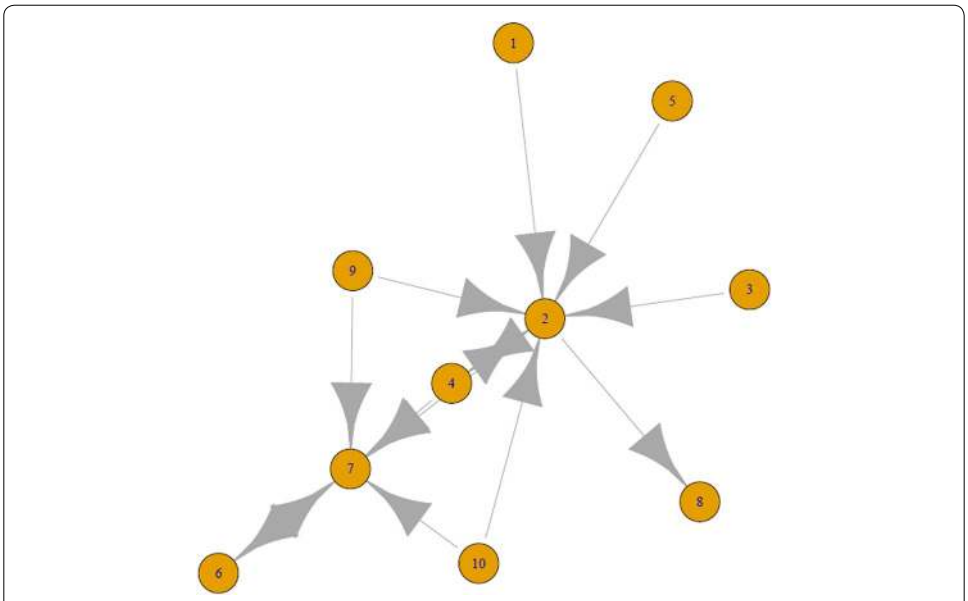

Fig. 2 A directed graph with 10 actors

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Λ_1 for the adjacency matrix A is 1, by choosing $\alpha = 0.85$ and $\beta = 1$:

$$C^{b}_{Katz} = \begin{bmatrix} 41.381381 \\ 47.507508 \\ 41.381381 \\ 47.048048 \\ 41.381381 \\ 6.666667 \\ 6.666667 \\ 1.000000 \\ 47.048048 \\ 47.048048 \end{bmatrix}$$

Matrix C^{b}_{Katz} gives the broadcasting ability of each node. Similarly, by using the same α and β values, the scores of each node for the Katz receive centrality scores are:

$$C^{r}_{Katz} = \begin{bmatrix} 1.00000 \\ 21.98198 \\ 1.00000 \\ 19.68468 \\ 1.00000 \\ 120.35736 \\ 140.42042 \\ 19.68468 \\ 1.00000 \\ 1.00000 \end{bmatrix}$$

Therefore, from these two broadcast and receive score matrices, it can be inferred that Node 2 is ideal for initiating a rumor spread into the network and Node 7 is ideal node for receiving the latest rumors in the network.

Map equation framework

The clustering of the networks resembles the cartography of the traffic infrastructure for navigation, hence the term 'map'. In this framework, the modules are identified that tend to stay in a cluster for a longer time. In a study by Rosvall and Bergstrom [19], this can be seen as a real scenario because a place could be called a 'city' if the traffic remains for a relatively long time. The map equation is constructed on a flow-based infrastructure. A random walker's lodge in a community can have either shorter or longer description lengths. For a good community structure, a shorter length is favored. The random walker sometimes stays for a longer time at certain regions in a community. In those cases, the description can be condensed. It is better to derive the code length from the stationary distribution of the random walker on the nodes and links rather than measuring the code length of a long walk and dividing by the number of steps. In general, given a network with n nodes and weighted directed links $\left(\mathrm{W}_{\alpha!\beta}\right)$ between nodes α, β, 2, 1, 2, . . . , n, the conditional probability that the random walker steps from node α to node β is given by the relative link weight $\mathrm{P}_{\alpha\geq\beta} = \mathrm{W}_{\alpha\geq\beta}/\sum\beta\mathrm{W}_{\alpha\geq\beta}$.

Infomap clustering [19] arrives at two descriptions by which a random walker spends more time within a cluster. It looks for a module partition M (i.e., a set of cluster assignments) of N nodes into m clusters that minimizes the following expected description length of a single step in a random walk on the graph:

$$\mathrm{L(M)} = \mathrm{q} * \mathrm{H(Q)} + \sum_{i=1}^{m} p_i H(P_i)$$

This equation has been efficiently demonstrated by Bae et al. [20]. and Shun et al. [21]. The Infomap algorithm shadows the Louvian method, to an extent. It initially augments compact communities, which is done by modularity in a local way. Summing over all node pairs gives the equation for modularity, Q:

$$\mathrm{Q} = \frac{1}{2m}\sum_{vw}\left[A_{vw} - \frac{k_v k_w}{2m}\right]\frac{s_v s_w + 1}{2}$$

where n and m are the number of nodes and the number of links, respectively; s is a membership variable such that if node v belongs to some community h, then $\mathrm{S}_h = 1$. A is the adjacency matrix and k_v is a node degree.

Discussion

Given a graph with a set of distinct communities $C = \{C_1, C_2, C_3 \ldots C_n\}$ with $|C| = n$. This set of communities might represent various types of data, such as:

- Facebook friends that belong to a community or group;
- A group of users in Twitter who follow a celebrity or famous person;
- A group of users who prefer a certain category of items on such e-Commerce websites as Amazon, eBay, or Flipkart; or

- A set of groups in an organization.

Given this data, we tried to identify the most central communities that are capable of broadcasting given information into multiple communities. Most of the studies focused on finding the top influential communities based on the criteria: How far will the influence of a group or community propagate in a network?

In our work, the focus was on identifying the central communities that are capable of sending a message to the entire network with high influence values as well as into how many distinct communities; this information could be broadcasted immediately. In general, the influence of a message sent by a person in friends' network decreases as the path of the message transfer increases; the approach using Katz centrality captures this property by penalizing a k step walk with α^k. For calculating the influence of a community over the entire network, we took into consideration the average Katz centrality score of individual nodes belonging to the community. At the community level, the interactions of the current community nodes with distinct neighboring community nodes were calculated to determine the number of distinct communities into which the current community could broadcast a message. By using the above two factors, a community could be ranked as an influential community. A 'community' was defined as an influential community if it had a high amount of average Katz centrality scores and if it was capable of broadcasting a given message into a maximum number of distinct communities out of all the communities in the network, apart from the current community under consideration. Using this approach, we tried to improve the rankings of communities by using fewer actors, yet still capable of sending the information to distinct communities.

To calculate the average influence value of each community, first, the influence values of each node that belongs to a community needs to be calculated. The Katz centrality for node i can be calculated as:

$$\mathrm{X_i} = \alpha \sum_j A_{\mathrm{ji}} \mathrm{x_j} + \beta \tag{4}$$

where A is the adjacency matrix of the graph G with eigenvalues $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$.

The Katz centrality computes the relative influence of a node within a network by measuring the number of the immediate neighbors (first-degree nodes) and also all other nodes in the network that connect to the node under consideration through these immediate neighbors. Extra weighting could be provided to the immediate neighbors by means of the parameter β. However, connections made with distant neighbors are penalized by an attenuation factor α, which should strictly be less than the inverse largest eigenvalue of the adjacency matrix in order for the Katz centrality to be computed correctly. The influence of a node on a distantly connected node will vary with this attenuation factor; this is analogous to real time, where the influence of a person on immediate neighbors would be relatively high compared to users who are connected to neighbors of the current user under consideration. The influence of the current user under consideration would be less, and reduces with as the length of the walk to another user increases.

In turn, the Katz centrality has two variations, Katz broadcast centrality and Katz receive centrality. Katz broadcast and receive centralities of node i can be calculated

using $C^{b}_{Katz} = \left[\beta(I - \alpha A)^{-1}.\mathbf{1}\right]_i$ and $C^{r}_{Katz} = \left[\beta\left(I - \alpha A^{T}\right)^{-1}.\mathbf{1}\right]_I$. In the case of an undirected graph, where $A = A^{T}$, either of the formulas could be used to calculate the broadcast and receive scores. In the case of a directed graph, $C^{b}_{Katz} = [\beta(I - \alpha A)^{-1}.\mathbf{1}]_i$ could be used to calculate the broadcast centrality score of node *i.* After calculating the Katz centralities of each user or node in the network, we take the average influence values of each community by taking into account the influence values of all the nodes that belong to a given community. This value accounts for the relative influence of a community on the entire network. CKatz_average(i) represents the community level influence on the network of communities. CKatz_average(i) for community C_i with k number of nodes $\{n_1, n_2, n_3, \ldots, n_k\}$ is computed in the following manner:

$$\text{CKatz\_average (i)} = (n_1 + n_2 + n_3 + cdots + n_k)/K \tag{5}$$

In general, there two types of communities: Hard and Soft communities. Within the scope of this paper, we only considered hard communities; that is, given a node *i*, it only belongs to one community *k* at a given time. After computing the Katz centralities of each community, out of all the communities present in the network apart from the current community under consideration, the distinct number were identified for immediate neighbors into whom each community was capable of broadcasting a given message. This can be calculated as:

$$(|N|/(|TC| - 1)) \tag{6}$$

where |N| is the number of distinct neighboring communities and |TC| is the total number of communities within the network. After calculating the strength and normalized distinct neighboring communities value, the strength of community can be obtained by using:

$$\text{Strength } (C_i) = \text{CKatz\_average (i)} * (|N_i|/(|TC| - 1)) \tag{7}$$

where $|N_i|$ represents the number of directly connected neighboring communities of community C_i and |TC| represents the total number of communities in the network. After calculating the strength values of each community, all the communities are sorted in the descending order of their strength values in order to retrieve the top-K communities later.

Implementation

The proposed model was implemented in R language, with RStudio, an integrated development environment (IDE); and by using IGraph packages extensively. An IGraph package was used for graph creation, both undirected and directed; community formation; and for capturing these details in a matrix. This matrix was parsed for community-wise boundary-node detection. Further, to rank the communities, the average value of the Katz broadcasting centrality for each community is computed (by using the Katz broadcast centralities of all nodes in each community). In turn this average Katz broadcast centrality of each community is multiplied by the normalized value of the number of distinct neighboring communities.

The pseudo-code for populating a community matrix is given in Table 2. In this part of code, the input file (edge list format) is read and the graph object is created by using the read.graph() method of the IGraph package. The default graph object is a directed-graph object; in case the data is that of an undirected graph, the graph object needs to be converted to an undirected graph object. This can be done by using the as.undirected() method of the IGraph package. The graph object, either directed or un-directed, is passed to the Infomap.community() method to identify the communities in the network. The membership of each node is captured in a list object of the R language, in this case, c_List. Later, this list object is parsed by using the indexes to construct the community matrix. If a node i belongs to community k, then Community_Matrix [i, k] consists of k (community number). The later stages of the algorithm uses this community Matrix & c_List to identify the boundary nodes of each community and to identify how many distinct one-step neighboring communities connect to the current community.

For the Katz broadcast centrality, the pseudo code for the computation of the scores of each node is given in Table 3. For the Katz broadcast centrality, the score for node i can be obtained by computing $\left[\beta(\mathrm{I}-\alpha\mathrm{A})^{-1}.\mathbf{v}\right]_{\mathrm{i}}$; for the Katz receive centrality, the score for node i can be obtained by computing $[\beta(\mathrm{I}-\alpha\mathrm{A}^{\mathrm{T}})^{-1}.\mathbf{v}]_{\mathrm{i}}$. The pseudo-code in Table 3 is valid for both undirected and directed graph data. In the case of an undirected graph, Katz broadcasting and receiving scores are the same because $\mathrm{A}=\mathrm{A}^{\mathrm{T}}$, where matrix A is an adjacency matrix and symmetric in nature.

The pseudo-code for identifying distinct communities that are in the neighborhood of each community is given in Table 4. In order to identify each community's distinct neighbors, community_Matrix and membership list(c_List) are used. The community_Matrix is parsed column-wise; whenever the algorithm finds a non-zero value, the node (row#) belongs to the community (column#). (In the community_Matrix, row numbers

Table 2 Construction of the community matrix

Algorithm 1
Community Matrix Construction

```
read the input file and construct the graph
apply infomap community detection algorithm to simulate
communities
  c_Object ← infomap.community(g_Object)
  c_List   ← membership(c_Object)
create and initialize community_Matrix, such that e_ik = 0 (where e_ik is an element in i^th row and k^th column)
temp_Var ← 1
while temp_Var <= vcount (graph) do
  set community_Matrix [temp_Var, c_List[temp_Var]] = k
end while
```

Table 3 Computation of the Katz broadcast centrality

Algorithm 2
Katz Broadcast Centrality Computation

```
nodes ← vcount (g_Object)
A ← as_adjacency_matrix(graph_Object)
A ← as.matrix(A)
I ← diag(x=1, nodes, nodes)
V ← matrix (data=1, nrow=nodes, ncol=nodes, byrow=TRUE)
find λ1 and compute the value of 1/ λ1
Choose α, such that 0 < α < 1/ λ1
β = 1
compute I- αA
solve(I- αA) //Inverse matrix
compute β (I- αA)^-1
Katz_matrix ← β (I- αA)^-1.v
```

represent the node labels, whereas column numbers represent the community numbers.) All the outbound-edge neighbors of the current node are retrieved, and their membership is tested to know whether they belong to a neighboring community or not. In this way, all the details of distinct neighboring communities are captured for every community. After obtaining the list of distinct communities, unique function of the R language is applied to retrieve the unique neighbors; later, the length method of the R language is used to determine the number of unique neighboring communities. After obtaining the number of unique neighboring communities, the normalized value is calculated, as per line 21 in Table 4), and the values are stored in nMatrix. This process is repeated for each column of the community_Matrix.

In addition, temp_Sum is a variable that captures the sum of the Katz centrality scores of all the nodes that belong to a community, and count_nodes keeps track of the total number of nodes in a community. By using these two features, average Katz centrality scores of each community can be calculated, and the values are stored in kMatrix. This information is used to calculate the strength values of each community.

The pseudo-code for the calculation of each community's strength value is given in Table 5, kMatrix consists of the average Katz broadcasting scores of each community, and nMatrix consists of the normalized neighbor count of each community. Using these two values, strength of each community is calculated and stored in result matrix. Result

Table 4 Detection of neighboring communities

```
Algorithm 3
Neighboring Community Detection
1:  temp_Columns ← 1
2:  count_nodes ←0
2:  while temp_Columns <= ncol(community_Matrix) do
3:    temp_Rows ←1
4:    temp_Sum  ← 0
5:    while temp_Rows <=  nrow (community_Matrix)  do
6:      if community_Matrix [temp_Rows, temp_Columns] ≠0 then
7:        count_nodes += 1
7:        node ← temp_Rows
8:        temp_Sum += Katz_matrix[temp_Rows]
9:        node_Index ←1
10:       node_neighbors ← neighbors(node)
11:       while node_Index <= length(node_neighbors) do
12:         if c_List(node) ≠ c_List(node_neighbors [node_Index]) then
13:           UPDATE(neighboring_Communities(temp_Columns))
14:         end if
15:       end while
16:   temp_Rows += 1
17:   end while
18:   unique_neighbors ←unique(neighboring_Communities(temp_Columns)
19:   count_neighbors ← length(unique_neighbors)
20:   kMatrix[temp_Columns,1] = temp_Sum/count_nodes
21:   nMatrix[temp_Columns,1] = count_neighbors/(|TC|-1)
22:   count_nodes ← 0
23:   temp_Rows   ←1
24:   temp_Sum    ← 0
25:   temp_Columns += 1
26: end while
```

Table 5 Identification of a top-K community

```
Algorithm 4
Top-K Community Identification
1:  temp_Var ←1
2:  while temp_Var <= ncol(community_Matrix) do
3:    result[temp_Var, 1] ←temp_Var
4:    result[temp_Var, 2] ← kMatrix[temp_Var,1]*nMatrix[temp_Var,1]
5:  end while
6:  arrange results in the decreasing order of strength values
7:  select top-K communities of out the sorted list
```

matrix is a two columned matrix, first column is used to store the community number and the second column is used to hold the strength values. Two columns facilitate the sorting operation, so as to keep track of community and community strength values after sorting. Later all the given communities are sorted in descending order of their strengths to allow us to pick the top-K communities.

The IGraph package

The IGraph package in R consists of routines for simple graphs and network analysis. It can handle large graphs very well, and provides functions for generating random and regular graphs, graph visualization, centrality methods and much more. Many community algorithms are available [22–26]; in our model, the infomap community-detection algorithm of IGraph library (in R language) was used to simulate communities from the given data file.

For generating a graph object using the IGraph package, node labels present in the input file (edge list file) when arranged in consecutive order and must start with the label 'zero'; the IGraph package expects the subsequent numbers to be in consecutive order. If the node labels in the input file are not present in this format—for example, if the edge list does not contain a node zero, then a node zero will be created by the IGraph package. This behavior might cause problems because an additional row will be created in the adjacency matrix; further, the inverse might not be computed as there is a chance for a determinant of matrix A to be zero. Even if the node labels are in the order as expected by the IGraph package—that is, the node labels start from zero and the rest of the labels are in consecutive order—then, the IGraph package increments each node by a value 'one'. In other words, a node with the label 'zero' is converted into a node with the label 'one', a node with the label 'one' is converted into a node with the label 'two', and so on. However, the type of conversion done by the IGraph package—incrementing the node labels by one—will not affect the results as long as all the node labels in the input file are in consecutive order from 0 to n nodes. After performing all the required operations on the converted node labels in the R environment, the results obtained can be converted back to an edge file format by subtracting 'one' from the node labels.

In order to make the effective use of the IGraph package's functionality, in our proposed model, we came up with a pre-processing algorithm (see Table 6), which reads the input file node label and aligns the node labels in the format as expected by the IGraph package. The preprocessing algorithm will make sure that the newly obtained file consists of node label 'zero'; in addition, it ensures that there are no missing nodes in the

Table 6 Processing of input files

Algorithm 5
Input File Processing

1: **for** each line $l_i \in$ input file **do**
2: read node1 and node2 values
3: **if** node1, node2 are not present in list **then**
4: add node1, node2 to the list
5: **end if**
6: **end for**
7: sort the list in ascending order
8: **for** each list element $e_i \in$ list **do**
9: add element and index of element to dictionary as key, value (k, v) pairs.
10: **end for**
11: **for** each line $l_i \in$ input file **do**
12: replace the node labels in input file node labels (key) with value pairs from dictionary
13: **end for**

input file, i.e., all the node labels are consecutively ordered. Please note that the preprocessing is required only if the input file is not present in the format as expected by the IGraph package.

After obtaining the pre-processed file, it was fed as input to construct the graph object (either undirected or directed). The infomap community-detection algorithm was run on top of the graph object to capture information obtained from communities in two ways. First, all the communities and nodes that are members of each community were captured. Second, the community to which each node belonged were captured using a list object in R language, with node labels as indexes and community numbers as elements for the corresponding node labels. Later, these two attributes were used to construct the community matrix, a matrix in which rows represent nodes, with the number of rows representing the total number of nodes in the graph and the columns representing the communities formed based in the graph topology, with number of columns as the total number of communities.

Initially, the community matrix was initialized to all zeroes; later, this matrix was populated with community numbers such that if a node i belonged to community k, then the element Community_Matrix [i, k] consists of k. The community matrix was used as the data structure, which was parsed column-wise (community-wise) to determine the nodes that fall into the community; then, the community matrix retrieves all the outbound neighbors and their community numbers to check if the outbound neighboring node belonged to a different community. If so, the neighboring community numbers was tracked, and the count of unique neighboring communities was retrieved for each community to be used in calculating the communities' strengths.

Conclusions

This paper presents a new way of calculating the top-K most influential communities in large networks. We designed a model that calculated the average Katz centrality of all the communities rather than following the standard approach using group centrality measures. After calculating the average, the communities connected to the most number of unique communities were considered, and defined as the community strength. We found that by contemplating the distinctive nature of the communities, the most influential one can be determined. From the experiments conducted on three sample datasets, the top 15 communities from all the datasets were ascertained, with the graphical representation. The approach followed in calculating the strengths gave importance to the communities capable of propagating information into distinct neighboring communities; the influence of the group was weighted over the network.

In the future, the model could be improved by considering the weights of edges. In addition, the internal nodal interaction could be improved by considering external community–community interactions.

Experiments

Experimental environment

All the test cases and output graphs were obtained when executed on the proposed algorithm in Intel® Xeon® CPU E5-1607 0 @ 3.00 GHz with 16.0 GB RAM, which uses a 64-bit Windows Operating System.

Results

Facebook dataset

The strength values for the Facebook dataset were dependent on every individual node influence on the network since individual node-centrality values were calculated as the average as well as at the community–community level; also included were the number of communities to which this community could directly contribute its Katz centrality value.

This dataset consisted of 4039 nodes and 88,234 edges. Varying the number of communities formed by using the infomap community detection algorithm tested the data; for each set of communities, also varied was the α value, used to compute the Katz centrality values of each node over the entire graph. Largest λ value obtained for the value for the network would not change with the number of communities formed, and the largest λ value was 0.006. The metrics used in distributing the data using IGraph's infomap method were the number of trails as 10 and modularity value as TRUE. Using the infomap community-detection method, 91 and 94 communities were formed in the first and the second attempt. Every time, the number of communities gets varied.

For each set of communities formed, the α value was varied as {0.003, 0.005}. Any value of α could be considered as far as it was less than the value of the highest λ value. After calculating the individual nodes for the Katz centrality values, community-wise average values were calculated. After that, the number of distinct neighboring communities was perceived for each community, and then strength values for each community were calculated. The next step after calculating the strength values of each community was to sort the obtained results in descending order in order to determine the top-K communities, which in these experiments were the top 15.

With $|\mathbf{C}| = 91$ and alpha value as 0.003, Table 7 shows the top 15 communities ranked in the descending order of their community numbers. Please note that the community numbers were arbitrary; they were not given numbers on the basis of the number of nodes contained in the community. Figure 3 gives the graphical representation of top 15 communities given in Table 10, with $\alpha = 0.003$.

With $|\mathbf{C}| = 91$ and when alpha value is increased from 0.003 to 0.005, Table 8 consists the top 15 communities ranked in the descending order of their strength values. Figure 4 gives the graphical representation of the top 15 communities given in Table 7 with $\alpha = 0.005$.

With the same graph object, and by using infomap clustering algorithm, another set of communities with $|\mathbf{C}| = 94$ was generated. With $|\mathbf{C}| = 94$ and the alpha value as 0.003, Table 9 consists the top 15 communities ranked in the descending order of their strength values. Figure 5 gives the graphical representation of the top 15 communities given in Table 9 with $\alpha = 0.003$.

With $|\mathbf{C}| = 94$ and alpha value as 0.005, Table 10 consists the top 15 communities ranked in the descending order of their strength values. Figure 6 gives the graphical representation of the top 15 communities given in Table 10 in descending order of their strength values, with $\alpha = 0.005$.

Autonomous systems dataset

This dataset consisted of 6474 nodes and 13,895 edges. The data was tested by varying the number of communities formed by using the infomap community-detection

Table 7 Top-K facebook communities (15) by community strength with |C| = 91 and α = 0.003

Top 15 communities	Community strength
2	0.34827931
20	0.32542475
3	0.27217018
1	0.25541676
8	0.25301664
5	0.23070149
7	0.21545911
19	0.20524559
12	0.19873984
13	0.19409956
25	0.16872643
23	0.16562121
44	0.13891725
41	0.13763725
9	0.13538653

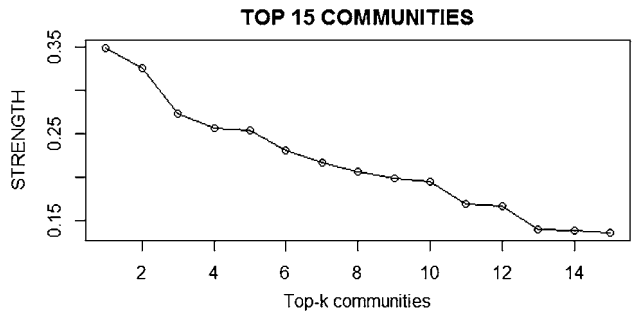

Fig. 3 The graph of top-15 communities when α = 0.003

Table 8 Top-K facebook communities (15) by community strength with |C| = 91 and α = 0.005

Top 15 communities	Community strength
2	0.55987447
3	0.40251962
20	0.37294907
1	0.28959429
8	0.28351022
4	0.27519171
5	0.24827503
7	0.22899202
13	0.22514774
19	0.22327368
12	0.22228522
25	0.18739987
23	0.17738614
9	0.1628733
44	0.14628848

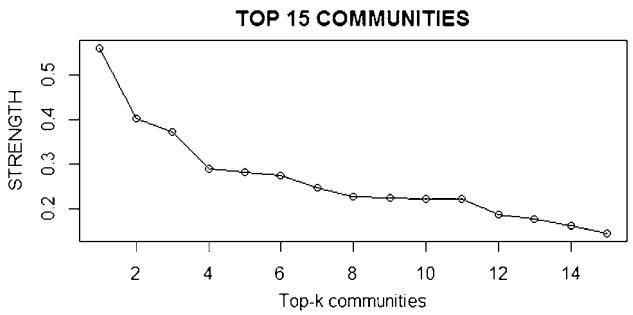

Fig. 4 The graph of top-15 communities when α = 0.005

Table 9 Top-K facebook communities (15) by community strength with |C| = 94 and α = 0.003

Top 15 communities	Community strength
2	0.35309423
21	0.34203471
3	0.26339049
1	0.25965741
8	0.25724355
7	0.22047152
5	0.21150900
19	0.21030857
12	0.20434944
13	0.2011278
23	0.18324383
25	0.17492941
44	0.14563905
9	0.14412115
41	0.13319734

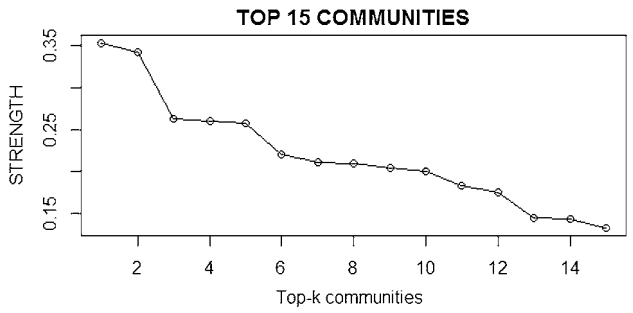

Fig. 5 The graph of top-15 communities when α = 0.003

algorithm on the data; for each set of communities, we tested by varying the α value. Largest λ value obtained for the value of the network would not change with the number of communities formed, and the largest λ value was 0.061. The metrics used in distributing the data using IGraph's infomap method were the number of trails as 10 and the modularity value as TRUE.

Table 10 Top-K facebook communities (15) by community strength with |C| = 94 and α = 0.005

Top 15 communities	Community strength
2	0.56761467
21	0.39507630
3	0.38953511
1	0.29450742
8	0.28837354
4	0.26631456
7	0.23462185
13	0.23396232
19	0.22878138
12	0.22855940
5	0.22762057
23	0.19593760
25	0.19428045
9	0.17338125
44	0.15336696

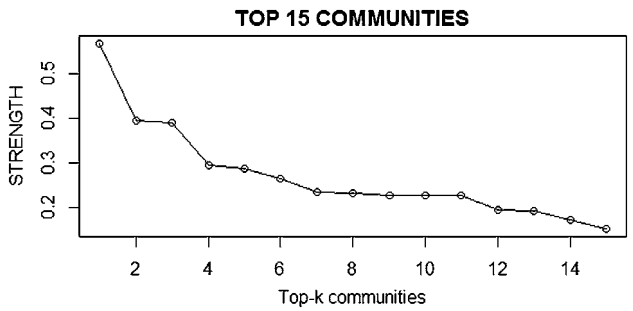

Fig. 6 The graph of top-15 communities when α = 0.005

Using the infomap community-detection method, 406 and 411 communities were formed in the first and the second attempts. For each set of communities formed, the α value was varied as {0.018, 0.020}. After calculating the Katz centrality values for individual nodes, community-wise average values were calculated. After that, the number of distinct neighboring communities was perceived for each community, and the strength values for each community were calculated. The next step after calculating the strength values of each community was to sort the results obtained in descending order to pick the top-K communities, which were the top 15 in these experiments. The graph outline was the same as the Facebook dataset, with the x axis representing the top 15 communities in the ascending order and the y axis representing the strength of the communities. Initially, the α value was taken as 0.018, and then incremented to 0.020.

With $|\mathbf{C}| = 406$ and the alpha value as 0.018, Table 11 shows the top 15 communities ranked in the descending order of their strength values. Figure 7 gives the graphical representation of the top 15 communities given in Table 11 with $\alpha = 0.018$.

Table 11 Top-K communities for autonomous systems (15) by community strength with $|\mathbf{C}| = 406$ and $\alpha = 0.018$

Top 15 communities	Community strength
1	2.382238856
2	1.420252055
5	0.965961599
3	0.774542642
9	0.723444622
27	0.722030104
36	0.53013892
7	0.508309547
19	0.489419269
10	0.477645542
25	0.378747968
8	0.374400865
31	0.340878789
55	0.32575921
13	0.304715633

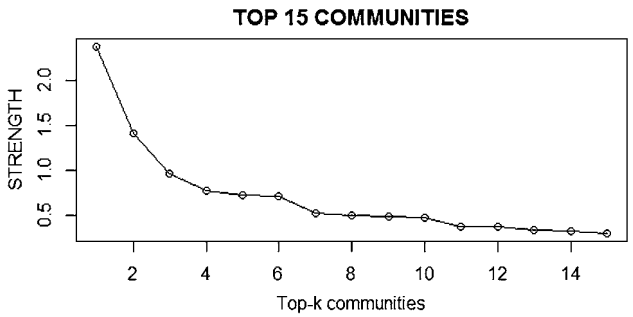

Fig. 7 The graph of top-15 communities when $\alpha = 0.018$

With $|\mathbf{C}| = 406$ and the alpha value as 0.020, Table 12 consists the top 15 communities ranked in the descending order of their strength values. Figure 8 gives the graphical representation of the top 15 communities given in Table 11 with $\alpha = 0.020$.

With the same graph object, and by using infomap clustering algorithm, another set of communities with $|\mathbf{C}| = 411$ was generated. With $|\mathbf{C}| = 411$ and the alpha value as 0.018, Table 13 consists the top 15 communities ranked in the descending order of their strength values. Figure 9 gives the graphical representation of the top 15 communities given in Table 13 with $\alpha = 0.018$.

With $|\mathbf{C}| = 411$ and alpha value as 0.020, Table 14 consists the top 15 communities ranked in the descending order of their strength values. Figure 10 gives the graphical representation of the top 15 communities given in Table 14 with $\alpha = 0.020$.

Wikipedia dataset

This dataset consists of 7115 nodes and 103,689 edges. The data was tested by varying the number of communities formed by using the infomap community-detection algorithm on the data; for each set of communities, the α value, used in computing the

Table 12 Top-K communities for autonomous systems (15) by community strength with $|C| = 406$ and $\alpha = 0.020$

Top 15 communities	Community strength
1	5.705042597
2	3.250283359
5	2.223298308
3	1.701964513
27	1.674371952
9	1.607148445
7	1.118534177
36	1.091291835
10	1.017736145
19	0.919028955
25	0.771858687
31	0.737656335
13	0.641765117
55	0.631717051
8	0.63080304

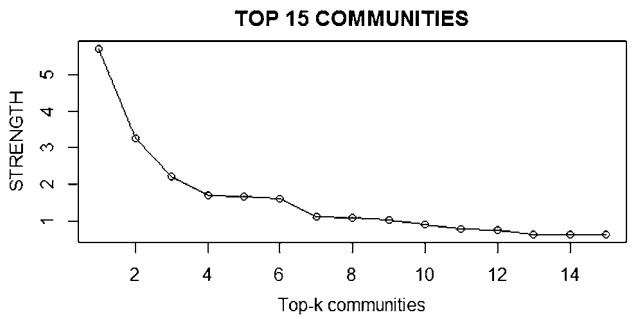

Fig. 8 The graph of top-15 communities $\alpha = 0.020$

Table 13 Top-K communities for autonomous systems (15) by community strength with $|C| = 411$ and $\alpha = 0.018$

Top 15 communities	Community strength
2	2.352501069
1	1.395429598
5	0.969208061
34	0.915517939
3	0.778288328
9	0.721428051
35	0.521542166
7	0.50211065
18	0.483450742
10	0.465771614
8	0.374186001
23	0.37412909
58	0.339157028
29	0.33672173
12	0.300999589

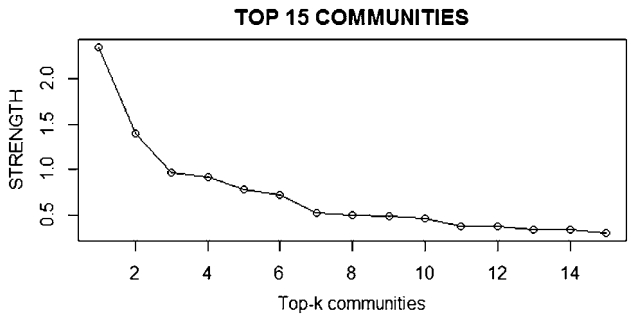

Fig. 9 The graph of top-15 communities when α = 0.018

Table 14 Top-K communities for autonomous systems (15) by community strength with |C| = 411 and α = 0.020

Top 15 communities	Community strength
1	5.633347026
2	3.193476528
5	2.230770503
34	2.22796836
3	1.710195207
9	1.602668589
7	1.104893516
35	1.058384531
10	0.99243595
18	0.907821284
23	0.762445776
29	0.728660526
58	0.665951755
12	0.633938713
8	0.630441029

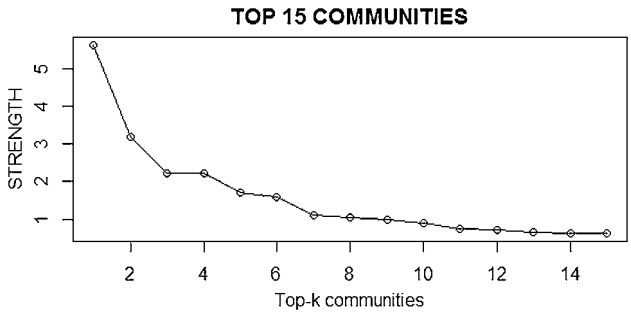

Fig. 10 The graph of top-15 communities when α = 0.020

Katz centrality values of each node over the entire graph, was varied. Largest λ value obtained for the value for the network would not change with the number of communities formed, and the largest λ value was 0.022. The metrics used in distributing the data using IGraph's infomap method were the number of trails as 10 and the modularity value

as TRUE. Infomap community detection method is used to identify communities in the network. We ran our proposed algorithm twice for each data set by varying the number of communities in the network. The number of communities formed in the first run is 499 and the number of communities formed in the second run is 508.

For each set of communities formed, the α value was varied as {0.018, 0.020}. Any value of α could be considered as long as it is less than the value of the highest λ value. After calculating the Katz centrality values for individual nodes, community-wise average values were calculated. After that, the number of distinct neighboring communities was perceived for each community, and then the strength values for each community were calculated. The next step after calculating the strength values of each community was to sort the obtained results in descending order in order to pick the top-K communities, which were the top 15 in these experiments.

The graph outline was same as for the Facebook dataset, with the x-axis representing the top 15 communities in the ascending order and the y-axis representing the strength of the communities. Initially, the α value was taken as 0.018, and then was incremented to 0.020.

With $|\mathbf{C}| = 499$ and alpha value as 0.018, Table 15 consists of the top 15 communities, ranked in the descending order of their strength values. Figure 11 gives the graphical representation of the top 15 communities given in Table 15 with $\alpha = 0.018$.

With $|\mathbf{C}| = 499$ and alpha value as 0.020, Table 16 consists of the top 15 communities, ranked in the descending order of their strength values. Figure 12 gives the graphical representation of the top 15 communities given in Table 16, with $\alpha = 0.020$.

With $|\mathbf{C}| = 508$ and alpha value as 0.018, Table 17 consists of the top 15 communities, ranked in the descending order of their strength values. Figure 13 gives the graphical representation of the top 15 communities given in Table 17, with $\alpha = 0.018$.

Table 15 Top-K wikipedia communities (15) by community strength with $|\mathbf{C}| = 499$ and $\alpha = 0.018$

Top 15 communities	Community strength
1	3.85492836
3	3.102804589
2	2.266908015
194	2.124327059
307	1.959665344
112	1.917481845
356	1.663644178
210	1.564172792
357	1.513951074
20	1.409692808
52	1.298656076
195	1.213671439
211	1.00457403
88	0.909502948
116	0.863233235

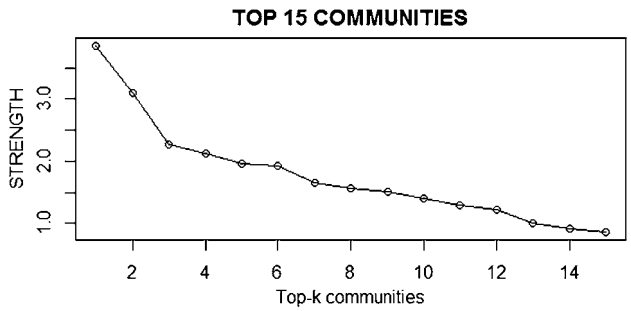

Fig. 11 The graph of top-15 communities when $\alpha = 0.018$

Table 16 Top-K wikipedia communities (15) by community strength with $|C| = 499$ and $\alpha = 0.020$

Top 15 communities	Community strength
1	7.461265545
3	5.766500756
194	4.062868307
307	3.831691768
2	3.744511405
112	3.700352434
356	3.231507336
210	3.167503039
357	3.136667658
20	2.69717456
52	2.49110016
195	2.355993439
211	1.913336252
88	1.628712712
116	1.559385089

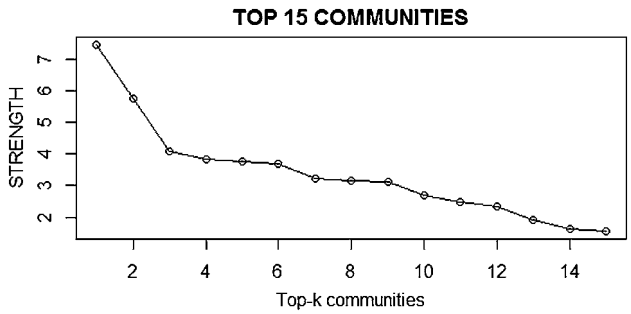

Fig. 12 The graph of top-15 communities when $\alpha = 0.020$

With $|C| = 508$ and alpha value as 0.020, Table 18 consists of the top 15 communities, ranked in the descending order of their strength values. Figure 14 gives the graphical representation of the top 15 communities given in Table 18, with $\alpha = 0.020$.

Table 17 Top-K wikipedia communities (15) by community strength with |C| = 508 and α = 0.018

Top 15 communities	Community strength
1	3.814052017
3	3.234831068
98	2.34702595
204	2.170719198
2	2.125448543
316	1.977256028
131	1.957304243
9	1.877103812
363	1.578121684
41	1.349556847
91	1.307287049
7	1.285289776
295	1.158538397
16	1.008435912
218	0.970438161

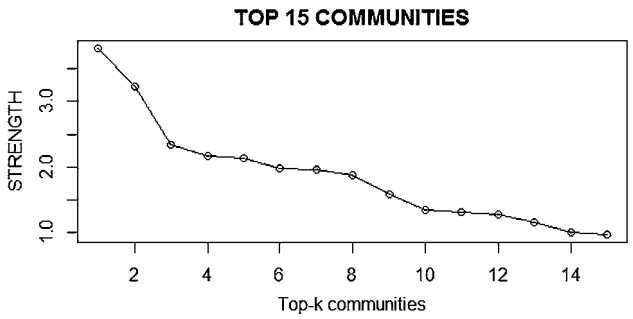

Fig. 13 The graph of top-15 communities when α = 0.018

Run time analysis

Table 18 gives the run time of each dataset. Please note that in these experiments, different sets of communities were generated for each dataset, and the alpha values were varied. However, varying the alpha value would not affect the run time for a set of communities since Algorithm 3 in Table 4 mainly was based on the community_Matrix, which is not dependent on Katz centralities of nodes. If the alpha values are varied for a set of communities, still we get the same run time values. Figure 15 gives the graphical representation of the run time analysis for Table 18.

Declarations

Availability of datasets

The datasets used as a part of the experiments were obtained from the Stanford Large Network Dataset Collection (SNAP). The proposed model was evaluated against three different datasets obtained from SNAP; two of them were undirected graphs and one was a directed graph. All the three datasets contained more than 1000 nodes and 10,000 edges shown in Table 19.

Table 18 Top-K wikipedia communities (15) by community strength with |C| = 508 and α = 0.018

Top 15 communities	Community strength
1	7.32804825
3	6.119591593
98	4.585996403
204	4.131136885
316	3.866086457
131	3.777201614
2	3.47038453
9	3.405845787
363	3.269619035
41	2.577470714
91	2.446963852
7	2.183529557
295	2.154473777
16	1.843539259
218	1.830656073

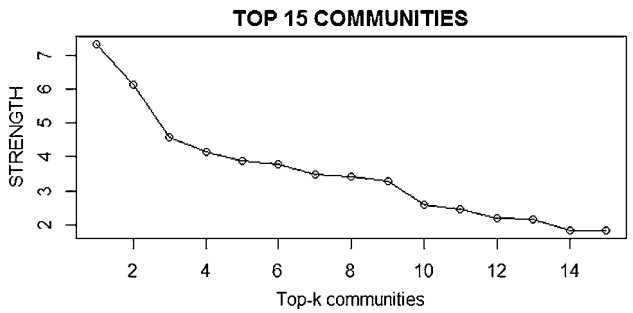

Fig. 14 The graph of top-15 communities when α = 0.020

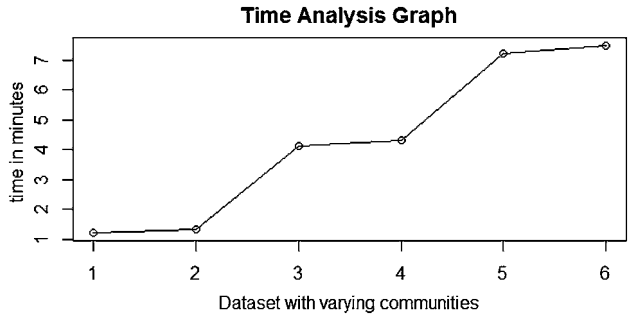

Fig. 15 Top 15 Wikipedia Communities

Facebook dataset

Social Circles, consisting of 'circles' (or 'friends' lists'), were obtained from Facebook. This data was collected by SNAP from survey participants using the Facebook app. The dataset included node features (profiles), circles, and ego networks. Replacing the

Table 19 Run time for each dataset

Dataset	No. of communities	Algorithm run time (min)
Autonomous systems dataset	406	1.221701
Autonomous systems dataset	411	1.324316
Wikipedia dataset	499	4.112323
Wikipedia dataset	508	4.296346
Facebook dataset	91	7.218095
Facebook dataset	94	7.491025

internal ids of Facebook for each user with a new value anonymized original data. While feature vectors from this dataset were provided, the interpretations of those features were obscured by SNAP. For instance, such data where the original dataset may have contained a feature "political = Democratic Party", the new data simply would contain "political = anonymized feature 1". Thus, it was possible to determine whether two users had the same political affiliations by using the anonymized data, but not what their individual political affiliations represented.

Dataset statistics Nodes 4039 and edges 88,234.
We used this information (i.e., edge file) as the test dataset for evaluating the algorithm, and constructed communities using the infomap community-detection algorithm of the IGraph package. We tested our algorithm against this undirected dataset multiple times by using the infomap algorithm to generate a different number of communities each time; every time, the nodes that were close to each other were grouped together to represent a community. This dataset [27] supports the conclusion of this article.

Autonomous systems dataset

According to *Wikipedia* [28], an autonomous system "is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators on behalf of a single administrative entity or domain that presents a common, clearly defined routing policy to the Internet". The graph of routers comprising the Internet can be organized into sub-graphs, called autonomous systems (AS). Each AS exchange traffic flows with some neighbors (peers).

We can construct a communication network of who-talks-to-whom from the border gateway protocol (BGP) logs. In contrast to citation networks, where nodes and edges only get added (and not deleted) over time, the AS dataset exhibits both the addition and deletion of nodes and edges over time. This notion helped us to evaluate our algorithm again with multiple communities generated by the infomap community-detection algorithm.

Dataset statistics Nodes 6474 and edges 13,895. The dataset [29] supports the conclusion of this article.

Wikipedia dataset

Wikipedia is a free encyclopedia, written collaboratively by volunteers around the world. A small part of *Wikipedia* contributors are administrators, who are users with access to additional technical features that aid in maintenance. In order for a user to become an

administrator, a Request for Adminship (RfA) is issued, and the *Wikipedia* community decides who to promote to be an administrator by means of a public discussion or a vote. Using the latest complete dump of *Wikipedia's* page edit history, all the data for administrator elections and vote history are contained in this dataset. The network contains all the *Wikipedia* voting data from the inception of *Wikipedia* up until January 2008. Nodes in the network represent *Wikipedia* users, and a directed edge from node *i* to node *j* represent that user *i* voted on user *j.*

Dataset statistics Nodes 7115 and edges 103,689.

This dataset was used as criteria to evaluate our algorithm against directed graphs. As mentioned in the above paragraph, a directed edge between nodes *i* and *j* represents that user *i* voted for user *j.* The infomap algorithm was used for community detection, available in the IGraph package of the R language; this dataset also was used to simulate various communities each time when the infomap community-detection algorithm was run on this. (Because the structural properties of a graph were studied, group behavior and the behavior of an individual were not within the scope of this study.) By running this dataset, we analyzed and identified the top-K communities capable of influencing the maximum number of communities in their neighborhood. This dataset [30] supports the conclusion of this article.

Abbreviations

AS: autonomous system; BGP: border gateway protocol; IP: internet protocol; OCR: optical character recognition; SNAP: stanford large network dataset collection.

Authors' contributions

All authors contributed equally. All authors read and approved the final manuscript.

Authors' information

Dr. Justin Zhan is the director of ILAB. He is a faculty member in the Department of Computer Science, College of Engineering, University of Nevada, Las Vegas. Previously, he was a faculty member at North Carolina A&T State University, Carnegie Mellon University, and the National Center for the Protection of Financial Infrastructure at South Dakota State University. His research interests include Big Data, Information Assurance, Social Computing, and Health Science. He is a steering chair of the ASE/IEEE International Conference on Social Computing (SocialCom); the ASE/IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT); and the ASE/IEEE International Conference on BioMedical Computing (BioMedCom). He has served as a conference general chair, a program chair, a publicity chair, a workshop chair, and a program committee member for 160 international conferences; he has been as an editor-in-chief, an editor, an associate editor, a guest editor, an editorial advisory board member, and an editorial board member for 30 journals. He has published 180 articles in peer-reviewed journals and conferences, and delivered more than 30 keynote speeches and invited talks. He has been involved in a number of projects as a Principal Investigator (PI) or a Co-PI that were funded by such agencies as the National Science Foundation, Department of Defense, and the National Institutes of Health.

Mr. Vivek Gudibande is a graduate student in the Department of Computer Science, College of Engineering in University of Nevada, Las Vegas. He received his Bachelor of Technology degree in Information Science and Engineering from Visvesvaraya Technological University, Bangalore, India. He is one of the research members of Big Data Hub ILab—UNLV. His research work includes big data analytics.

Mr. Sai Phani Parsa is a Master's student in the Department of Computer Science at the College of Engineering, University of Nevada, Las Vegas (UNLV). He has received the Bachelor of Technology in Computer Science from Sree Nidhi Institute of Science and Technology, India, in May 2014. Currently, he is working under the guidance of Dr. Zhan in Big Data Hub- ILAB at the University of Nevada, Las Vegas.

Acknowledgements

We gratefully acknowledge the United States Department of Defense (W911NF-14-1-0119, W911NF-13-1-0130, W911NF-16-1-0416), the National Science Foundation (1560625, 1137443, 1247663, 1238767), United Healthcare Foundation (#1592), Oak Ridge National Laboratory (#4000144962), and National Consortium for Data S cience for their support. Through these financial support, the following articles have been published [31–39].

Competing interests

The authors declare that they have no competing interests.

Received: 28 July 2016 Accepted: 24 August 2016
Published online: 08 September 2016

References

1. Girvan M, Newman ME. Community structure in social and biological networks. Proc Natl Acad Sci. 2002;99(12):7821–6.
2. Ahn YY, Bagrow JP, Lehmann S. Link communities reveal multiscale complexity in networks. Nature. 2010;466:761–4. doi:10.1038/nature09182.
3. Fortunato S. Community detection in graphs. Phys Rep. 2010;486(3):75–174.
4. Xie J, Kelley S, Szymanski B. Overlapping Community detection in networks: The state-of-the-art and comparative study. ACM Comput Surv (CSUR). 2013;45(4). doi:10.1145/2501654.2501657.
5. Li RH, Qin L, Yu JX, Mao R. Influential community search in large networks. Proc VLDB Endow. 2015;8(5):509–20.
6. Faisal SM, Tziantzioulis G, Gok AM, Hardavellas N, Ogrenci-Memik S, Parthasarathy S. Edge importance identification for energy efficient graph processing. In: 2015 IEEE International Conference on big data. Santa Clara, CA; 2015. p. 347–54. doi:10.1109/BigData.2015.7363775.
7. Doo M, Liu L. Extracting top-k most influential nodes by activity analysis. In: 2014 IEEE 15th International conference on information reuse and integration (IRI). Redwood City, CA; 2014. p. 227–36. doi:10.1109/IRI.2014.7051894.
8. Du N, Jia X, Gao J, Gopalakrishnan V, Zhang A. Tracking temporal community strength in dynamic networks. IEEE Trans Knowl Data Eng. 2015;27(11):3125–37.
9. Ma H, Yang H, Lyu MR, King I. Mining social networks using heat diffusion processes for marketing candidates selection. In: Proceedings of the 17th ACM conference on Information and knowledge management (CIKM '08). New York: ACM; 2008. p. 233–42. doi:10.1145/1458082.1458115.
10. McSweeney PJ, Mehrotra K, Oh JC. A game theoretic framework for community detection. In: 2012 IEEE/ACM International Conference on advances in social networks analysis and mining (ASONAM). Istanbul; 2012. p. 227–34. doi:10.1109/ASONAM.2012.47.
11. Kim SR. Efficient sequential and parallel algorithms for popularity computation on the World Wide Web with applications against spamming. Computational science and its applications—ICCSA 2004, Volume 3045 of the series lecture notes in computer science. Berlin: Springer; 2004. p. 367–75. doi:10.1007/978-3-540-24767-8_38.
12. Wu L, Bai T, Wang Z, Wang L, Hu Y, Ji J. A new community detection algorithm. 10th International conference on fuzzy systems and knowledge discovery (FSKD). 2013. p. 898–902.
13. Zhang T, Wu B. A method for local community detection by finding core nodes. 2012 IEEE/ACM International conference on advances in social networks analysis and mining (ASONAM). Istanbul; 2012. p. 1171–1176. doi:10.1109/ASONAM.2012.202.
14. Mahmood A, Small M. Subspace based network community detection using sparse linear coding. IEEE Trans Knowl Data Eng. 2016;28(3):801–12.
15. Katz L. A new status index derived from sociometric analysis. Psychometrika. 1953;18(1):39–43. doi:10.1007/BF02289026.
16. Benzi M, Klymko C. On the limiting behavior of parameter-dependent network centrality measures. SIAM J Matrix Anal Appl. 2015;36(2):686–702. doi:10.1137/130950550.
17. Srinivas A, Veluswamy R. Identification of influential nodes from social networks based on enhanced degree centrality measure. 2015 IEEE International advance computing conference (IACC). Banglore; 2015. p. 1179–84. doi:10.1109/IADCC.2015.7154889.
18. Bihari A, Pandia M. Eigenvector centrality and its application in research professionals' relationship network. 2015 International conference on futuristic trends on computational analysis and knowledge management (ABLAZE). Noida; 2015. p. 510–14. doi:10.1109/ABLAZE.2015.7154915.
19. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. PNAS. 2008;105(4):1118–23. doi:10.1073/pnas.0706851105.
20. Bae SH, Halperin D, West J, Rosvall M, Howe B. Scalable flow-based community detection for large-scale network analysis. 2013 IEEE 13th International conference on data mining workshops. Dallas, TX; 2013. p. 303–10. doi:10.1109/ICDMW.2013.138.
21. Shum SH, Campbell WM, Reynolds DA. Large-scale community detection on speaker content graphs. 2013 IEEE International conference on acoustics, speech and signal processing. Vancouver, BC; 2013. p. 7716–20. doi:10.1109/ICASSP.2013.6639165.
22. de Sousa FB, Zhao L. Evaluating and comparing the IGraph community detection algorithms. 2014 Brazilian Conference on intelligent systems (BRACIS). Sao Paul; 2014. p. 408–13. doi:10.1109/BRACIS.2014.79.
23. Lancichinetti A, Fortunato S. Community detection algorithms: a comparative analysis. Phys Rev E. 2009;80(5):056117. doi:10.1103/PhysRevE.80.056117.
24. Orman G, Labatut V. A comparison of community detection algorithms on artificial networks. Discovery science (DS). Lecture notes in artificial intelligence, vol 5808. Portugal: Springer; 2009. p. 242–56. doi:10.1007/978-3-642-04747-3_20.
25. Leskovec J, Lang KJ, Mahoney M. Empirical comparison of algorithms for network community detection. In: Proceedings of the 19th International conference on World wide web (WWW '10). New York: ACM; 2010. p. 631–40. doi:10.1145/1772690.1772755.
26. Orman G, Labatut V, Cherifi H. Qualitative comparison of community detection algorithms. Commun Comput Inf Sci. 2011;167:265–79.
27. Social circles: Facebook. https://www.snap.stanford.edu/data/egonets-Facebook.html. Accessed 12 Mar 2016.
28. Autonomous system. https://www.en.Wikipedia.org/wiki/Autonomous_system. Accessed 15 Mar 2016.
29. Autonomous systems AS-733. https://www.snap.stanford.edu/data/as.html. Accessed 15 Mar 2016.
30. Wikipedia vote network. https://www.snap.stanford.edu/data/wiki-Vote.html. Accessed 21 Mar 2016.
31. Lin JCW, Yang L, Fournier-Viger P, Wu JMT, Hong TP, Wang LSL, Zhan J. Mining high-utility itemsets based on particle swarm optimization. Eng Appl Artif Intell. 2016;55:320-30.
32. Lin JCW, Wu TY, Fournier-Viger P, Lin G, Zhan J, Voznak M. Fast algorithms for hiding sensitive high-utility itemsets in privacy preserving utility mining. Eng Appl Artif Intell. 2016;55:269–84.

33. Lin JCW, Liu Q, Fournier-Viger P, Hong TP, Voznak M, Zhan J. A sanitization approach for hiding sensitive itemsets based on particle swarm optimization. Eng Appl Artif Intell. 2016;53:1–18.
34. Pirouz M, Zhan J. Optimized relativity search: node reduction in personalized page rank estimation for large graphs, J Big Data. 2016;3(1):12.
35. Selim H, Zhan J. Towards shortest path identification on large networks. J Big Data. 2016;3(1):10.
36. Lin JCW, Li T, Fournier-Viger P, Hong TP, Zhan J, Voznak M. An efficient algorithm to mine high average-utility itemsets. Adv Eng Inform. 2016;30(2):233-243.
37. Chopade P, Zhan J. Structural and functional analytics for community detection in large-scale complex networks. J Big Data. 2016;2(1):1.
38. Fang X, Zhan J. Sentiment analysis using product review data. J Big Data. 2016;2(1):1.
39. Zhan J, Fang X. A computational framework for detecting malicious actors in communities. Int J Inform Priv Secur Int. 2014;2(1):1-20.