information

MDPI

Article

Nearest Neighbor Search in the Metric Space of a Complex Network for Community Detection

Suman Saha * and Satya P. Ghrera

Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, Solan 173215, India; sp.ghrera@juit.ac.in

* Correspondence: suman.saha@juit.ac.in; Tel.: +91-7-602728183

Academic Editors: Ana Paula Couto da Silva and Pedro O. S. Vaz de Melo
Received: 22 December 2015; Accepted: 11 March 2016; Published: 16 March 2016

Abstract: The objective of this article is to bridge the gap between two important research directions: (1) nearest neighbor search, which is a fundamental computational tool for large data analysis; and (2) complex network analysis, which deals with large real graphs but is generally studied via graph theoretic analysis or spectral analysis. In this article, we have studied the nearest neighbor search problem in a complex network by the development of a suitable notion of nearness. The computation of efficient nearest neighbor search among the nodes of a complex network using the metric tree and locality sensitive hashing (LSH) are also studied and experimented. For evaluation of the proposed nearest neighbor search in a complex network, we applied it to a network community detection problem. Experiments are performed to verify the usefulness of nearness measures for the complex networks, the role of metric tree and LSH to compute fast and approximate node nearness and the the efficiency of community detection using nearest neighbor search. We observed that nearest neighbor between network nodes is a very efficient tool to explore better the community structure of the real networks. Several efficient approximation schemes are very useful for large networks, which hardly made any degradation of results, whereas they save lot of computational times, and nearest neighbor based community detection approach is very competitive in terms of efficiency and time.

Keywords: complex network; nearest neighbor; metric tree; locality sensitive hashing; community detection

1. Introduction

The nearest neighbor (NN) search is an important computational primitive for structural analysis of data and other query retrieval purposes. NN search is very useful for dealing with massive data sets, but it suffers from the curse of dimension [1,2]. Though nearest neighbor search is a extensively studied research problem for low dimensional data, a recent surge of results shows that it is the most useful tool for analyzing very large quantities of data, provided a suitable space partitioning the data structure is used, like, kd-tree, quad-tree, R-tree, metric-tree and locality sensitive hashing [3–6]. One more advantage of using nearest neighbor search for large data analysis is the availability of efficient approximation scheme, which provides almost same results in very less time [7,8].

Though nearest neighbor search is very successful and extensively used across the research domains of computer science, it is not studied rigorously in complex network analysis. Complex networks are generally studied with a graph theoretic framework or spectral analysis framework. One basic reason for this limitation may be the nodes of the complex networks do not naturally lie on a metric space, thus restricting the use of nearest neighbor analysis which is done using metric or nearness measures.

Other than graphs, the complex networks are characterized by small "average path length" and a high "clustering coefficient". A network community (also known as a module or cluster) is typically a group of nodes with more interconnections among its members than the remaining part of the network [9–11]. To extract such group of nodes from a network one generally selects an objective function that captures the possible communities as a set of nodes with better internal connectivity than external [12,13]. However, very little research has been done for network community detection, which tries to develop nearness between the nodes of a complex network and use the nearest neighbor search for partitioning the network [14–20]. The way metric is defined among the nodes should be able to capture the crucial properties of complex networks. Therefore, we need to create the metric very carefully so that it can explore the underlying community structure of the real life networks [21].

Extracting network communities in large real graphs such as social networks, web, collaboration networks and bio-networks is an important research direction of recent interest [11,22–25]. In this work, we have developed the notion of nearness among the nodes of the network using some new matrices derived from the modified adjacency matrix of the graph which is flexible over the networks and can be tuned to enhance the structural properties of the network required for community detection.

The main contributions of this work are, (1) development of the concept of nearness between the nodes of a complex network; (2) comparing the proposed nearness with other notions of similarities; (3) study and experiment on approximate nearest neighbor search for complex network using M-tree and LSH; (4) design of efficient community detection algorithm using nearest neighbor search. We observed that nearest neighbor between network nodes is a very efficient tool to explore better community structure of the real networks. Further several efficient approximation scheme are very useful for large networks, which hardly made any degradation of results, whereas saves lot of computational times.

The rest of this paper is organized as follows. Section 2 describes the notion of nearness in complex network and proposed method to compute distance between the nodes of a complex network. Sections 3 and 4 describe the algorithm of the nearest neighbor search over complex network using of metric tree and locality sensitive hashing methods respectively. In Section 5, the proposed algorithm for network community detection using nearest neighbor search is discussed. The results of the comparison between community detection algorithms are illustrated in Section 6.

2. Proposed Notion of Nearness in Complex Network

The notion of nearness between the nodes of a graph is used in several purposes in the history of literature of graph theory. Most of the time the shortest path and edge connectivity are popular choices to describe nearness of nodes. However, the edge count does not give the true measure of network connectivity. A true measure of nearness in a complex network should able to determine how much one node can affect the other node to provide a better measure of connectivity between nodes of a real life complex network. Research in this direction need special attention in the domain of complex network analysis, one such is proposed in this article and described in the following subsections.

2.1. Definitions

Definition 1 (Metric space of network)**.** *Given, a graph $G = (V, E)$ the metric is defined over the vertex set V and d, a function to compute the distance between two vertices of V. Pair (V, d) distinguished metric space if d satisfies reflexivity, non-negativity, symmetry and triangle inequality.*

Definition 2 (Nearest neighbor search on network)**.** *The nearest-neighbor searching problem in complex network is to find the nearest node in a graph $G = (V, E)$ between a query vertex v_q and any other vertex of the graph V $\{v_q\}$, with respect to a metric space $M(V, d)$ associated with the graph $G = (V, E)$.*

Definition 3 (Approximate nearest neighbor search on network)**.** *For any $v_q \in V$, An ϵ approximate NN of $v_q \in V$ is to find a point $v_p \in V \{v_q\}$ s.t. $d(v_p, v_q) \leq (1+\epsilon)d(v, v_q)\ \forall\ v \in V \{v_q\}$.*

2.2. Nearness in Complex Network

Methods based on node neighborhoods. For a node x, let $N(x)$ denote the set of neighbors of x in a graph $G(V, E)$. A number of approaches are based on the idea that two nodes x and y are more likely to be affected by one another if their sets of neighbors $N(x)$ and $N(y)$ have large overlap.
Common neighbors: The most direct implementation of this idea for nearness computation is to define $d(x, y) := |N(x) \cap N(y)|$, the number of neighbors that x and y have in common.
Jaccard coefficient: The Jaccard coefficient, a commonly used similarity metric, measures the probability that both x and y have a feature f, for a randomly selected feature f that either x or y has. If we take features here to be neighbors in $G(V, E)$, this leads to the measure $d(x, y) := |N(x) \cap N(y)|/|N(x) \cup N(y)|$.
Preferential attachment: The probability that a new edge involves node x is proportional to $|N(x)|$, the current number of neighbors of x. The probability of co-authorship of x and y is correlated with the product of the number of collaborators of x and y. This corresponds to the measure $d(x, y) := |N(x)| \times |N(y)|$.
Katz measure: This measure directly sums over the collection of paths, exponentially damped by length to count short paths more heavily. This leads to the measure $d(x, y) := \beta \times |paths(x, y)|$ where, $paths(x, y)$ is the set of all length paths from x to y. (β determines the path size, since paths of length three or more contribute very little to the summation.)
Commute time: A random walk on G starts at a node x, and iteratively moves to a neighbor of x chosen uniformly at random. The hitting time $H(x, y)$ from x to y is the expected number of steps required for a random walk starting at x to reach y. Since the hitting time is not in general symmetric, it is also natural to consider a commute time $C(x, y) := H(x, y) + H(y, x)$.
PageRank: Random resets form the basis of the PageRank measure for Web pages, and we can adapt it for link prediction as follows: Define $d(x, y)$ to be the stationary probability of y in a random walk that returns to x with probability α each step, moving to a random neighbor with probability $1 - \alpha$.

Most of the methods are developed for different types of problems like information retrieval, ranking, prediction e.t.c. and developed for general graphs. In the article [21], the authors studied a measure specially designed for complex network.

2.3. Proposed Nearness in Complex Network

In this subsection, we developed the notion of nearness among the nodes of the network using some linear combination of adjacency matrix A and identity matrix of same dimension for the network $G = (V, E)$. The similarities between the nodes are defined on matrix $L = \lambda I + A$ as spherical similarity among the rows and determine by applying a concave function ϕ over the standard notions of similarities like, Pearson coefficient(σ_{PC}), Spacerman coefficient(σ_{SC}) or Cosine similarity(σ_{CS}). $\phi(\sigma)()$ must be chosen using the chord condition, *i.e.*, metric-preserving ($\phi(d(x_i, x_j) = d_\phi(x_i, x_j)$), concave and monotonically-increasing, to obtain a metric. It works by picking a pair of rows from L and computing the distance defined in the $\phi(\sigma)()$. The function ϕ converts a similarity function (Pearson coefficient (σ_{PC}), Spacerman coefficient (σ_{SC}) or cosine similarity (σ_{CS})) into a distance matrix. In general, the similarity function satisfies the positivity and similarity condition of the metric, but not the triangle inequality. ϕ is a metric-preserving ($\phi(d(x_i, x_j) = d_\phi(x_i, x_j)$), concave and monotonically-increasing function. The three conditions above are referred to as the chord condition. The ϕ function is chosen to have minimum internal area with the chord.

The choice of λ and $\phi(\sigma)()$ in the above sub-modules play a crucial role in the graph to metric transformation algorithm to be used for community detection. The complex network is characterized by a small average diameter and a high clustering coefficient. Several studies on network structure analysis reveal that there are hub nodes and local nodes characterizing the interesting structure of

the complex network. Suppose we have taken $\phi = arccos$, σ_{CS} and constant $\lambda \geq 0$. $\lambda = 0$ penalizes the effect of the direct edge in the metric and is suitable to extract communities from a highly dense graph. $\lambda = 1$ places a similar weight of the direct edge, and the common neighbor reduces the effect of the direct edge in the metric and is suitable to extract communities from a moderately dense graph. $\lambda = 2$ sets more importance for the direct edge than the common neighbor (this is the common case of available real networks). $\lambda \geq 2$ penalizes the effect of the common neighbor in the metric and is suitable for extracting communities from a very sparse graph.

3. Nearest Neighbor Search on Complex Network Using Metric Tree

There are numerous methods developed to compute the nearest neighbor search for points of a metric space. However, finding the nearest neighbor search on some data where dimension is high suffer from curse of dimension. Some recent research in this direction revealed that dimension constrained can be tackled by using efficient data structures like metric tree and locality sensitive hashing. In this section we have explored metric tree to perform the nearest neighbor search on complex network with the help of metric mapping of complex network described in the previous section.

3.1. Metric-Tree

A metric tree is a data structure specially designed to perform the nearest neighbor query for the points residing on a metric space and perform well on high dimension particularly when some approximation is permitted. A metric tree organizes a set of points in a spatial hierarchical manner. It is a binary tree whose nodes represent a set of points. The root node represents all points, and the points represented by an internal node v is partitioned into two subsets, represented by its two children. Formally, if we use $N(v)$ to denote the set of points represented by node v, and use $v.lc$ and $v.rc$ to denote the left child and the right child of node v, then we have $N(v) = N(v.lc) \cup N(v.rc)$ $\phi = N(v.lc) \cap N(v.rc)$ for all the non-leaf nodes. At the lowest level, each leaf node contains very few points.

An M-Tree [26] consists of leaf node, internal node and routing object. Leaf nodes are set of objects N_v with pointer to parent object v_p. Internal nodes are set of routing objects N_{RO} with pointer to its parent object v_p. Routing object v_r store covering radius $r(v_r)$ and pointer to covering tree $T(v_r)$, Distance of v_r from its parent object $d(v_r, P(v_r))$. Feature values stored in the object v_j are object identifier oid (v_j) and distance of v_j from its parent object $d(v_j, P(v_j))$

The key to building a metric-tree is how to partition a node v. A typical way is as follows: We first choose two pivot points from $N(v)$, denoted as $v.lpv$ and $v.rpv$. Ideally, $v.lpv$ and $v.rpv$ are chosen so that the distance between them is the largest of all distances within $N(v)$. More specifically, $||v.lpv - v.rpv|| = max_{p_1,p_2 \in N(v)}||p_1 - p_2||$. A search on a metric-tree is performed using a stack. The current radius r is used to decide which child node to search first. If the query q is on the left of current point, then $v.lc$ is searched first, otherwise, $v.rc$ is searched first. At all times, the algorithm maintains a candidate NN and there distance determines the current radius, which is the nearest neighbor it finds so far while traversing the tree. The algorithm for nearest neighbor search using metric tree is (Algorithm 1) given below.

3.2. Nearest Neighbor Search Algorithm Using M-Tree

The theoretical advantage of using metric tree as a data structure for nearest neighbor search is: Let $M = (V, d)$, be a bounded metric space. Then for any fixed data $V \in R^n$ of size n, and for constant $c \geq 1$, $\exists \epsilon$ such that we may compute $d(q, V)|_\epsilon$ with at most $c \cdot \lceil \log(n) + 1 \rceil$ expected metric evaluations [27].

Algorithm 1 NN search in M-Tree

Require: $M = (V, d)$ & q
Ensure: $d(q, v_q)$
1: Insert root object v_r in stack
2: Set current radius as $d(v_r, q)$
3: Successively traverse the tree in search of q
4: PUSH all the objects of traversal path into stack
5: Update the current radius
6: If leaf object reached
7: POP objects from stack
8: For all points lying inside the ball of current radius centering q, verify for possible nearest neighbor and update the current radius.
9: **return** $d(q, v_q)$

4. Nearest Neighbor Search on Complex Network Using Locality Sensitive Hashing

Metric trees, so far represent the practical state of the art for achieving efficiency in the largest dimension possible. However, many real-world problems consist of very large dimension and beyond the capability of such search structures to achieve sub-linear efficiency. Thus, the high-dimensional case is the long-standing frontier of the nearest-neighbor problem. The approximate nearest neighbor can be computed very efficiently using Locality sensitive hashing.

4.1. Approximate Nearest Neighbor

Given a metric space (S, d) and some finite subset S_D of data points $S_D \subset S$ on which the nearest neighbor queries are to be made, our aim to organize S_D s.t. NN queries can be answered more efficiently. For any $q \in S$, NN problem consists of finding single minimal located point $p \in S_D$ s.t. $d(p, q)$ is minimum over all $p \in S_D$. We denote this by $p = NN(q, S_D)$.

An ϵ approximate NN of $q \in S$ is to find a point $p \in S_D$ s.t. $d(p, q) \leq (1 + \epsilon)d(x, d)\ \forall\ x \in S_D$.

4.2. Locality Sensitive Hashing (LSH)

Several methods to compute first nearest neighbor query exists in the literature and locality-sensitive hashing (LSH) is most popular because of its dimension independent run time [28,29]. In a locality sensitive hashing, the hash function has the property that close points are hash into same bucket with high probability and distance points are hash into same bucket with low probability. Mathematically, a family $H = \{h : S \to U\}$ is called (r_1, r_2, p_1, p_2)-sensitive if for any $p, q \in S$

- if $p \in B(q, r_1)$ then $Pr_H[h(q) = h(p)] \geq p_1$
- if $p \notin B(q, r_2)$ then $Pr_H[h(q) = h(p)] \leq p_2$

where $B(q, r)$ denotes a hyper sphere of radius r centered at q. In order for a locality-sensitive family to be useful, it has to satisfy inequalities $p_1 > p_2$ and $r_1 < r_2$ when D is a distance, or $p_1 > p_2$ and $r_1 > r_2$ when D is a similarity measure [4,5]. The value of $\delta = log(1/P_1)/log(1/P_2)$ determines search performance of LSH. Defining a LSH as $a(r, r(1 + \epsilon), p1, p2)$, the $(1 + \epsilon)$ NN problem can be solved via series of hashing and searching within the buckets [5,30,31].

4.3. Locality Sensitive Hash Function for Complex Network

In this sub-section, we discuss the existence of locality sensitive hash function families for the proposed metric for complex network. The LSH data structure stores all nodes in hash tables and searches for nearest neighbor via retrieval. The hash table is contain many buckets and identified by bucket id. Unlike conventional hashing, the LSH approach tries to maximize the probability of collision of near items and put them into same bucket. For any given the query q the bucket $h(q)$

considered to search the nearest node. In general k hash functions are chosen independently and uniformly at random from hash family H. The output of the nearest neighbor query is provided from the union ok k buckets. The consensus of k functions reduces the error of approximation. For metric defined in the previous Section 2 we considered k random points from the metric space. Each random point r_i define a hash function $h_i(x) = sign(d(x, r_i))$, where d is the metric and $i \in [1, k]$. These randomized hash functions are locality sensitive [32,33].

Algorithm 2 NN search in LSH

Require: $M = (V, d)$ & q
Ensure: $d(q, V)$
1: Identify buckets of query point q corresponding to different hash functions.
2: Compute nearest neighbor of q only for the points inside the selected buckets.
3: **return** $d(q, V)$

The theoretical advantage of using locality sensitive hashing as a data structure for nearest neighbor search is: Let $M = (V, d)$, be a bounded metric space. Then for any fixed data $V \in R^n$ of size n, and for constant $c \geq 1$, $\exists \epsilon$ such that we may compute $d(q, V)|_\epsilon$ with at most $mn^{O(1/\epsilon)}$ expected metric evaluations, where m is the number of dimension of the metric space. In case of complex network $m = n$ so expected time is $n^{O(2/\epsilon)}$ [27,34].

5. Proposed Community Detection Based on Nearest Neighbor

In this section we have described the algorithm proposed for network community detection using nearest neighbor search. Our approach differs from the existing methods of community detection. The broad categorization of the available algorithms is generally based on graph traversal, semidefinite programming and spectral analysis. The basic approach and the complexity of very popular algorithms are listed in the Table 1. There are more algorithms developed to solve network community detection problem a complete list can be obtained in several survey articles [11,35,36]. Theoretical limitations and evaluation strategies of community detection algorithms are provided in the articles [37–41]. Content based node similarity (discussed in [42,43]) methods uses additional information of the network node and not available in general complex networks. A partial list of algorithms developed for network community detection purpose is tabulated in Table 1. The algorithms are categorized into three main group as spectral (SP), graph traversal based (GT) and semi-definite programming based (SDP). The categories and complexities are also given in Table 1.

Table 1. Algorithms for network community detection and their complexities.

Author	Reference	Category	Order
Van Dongen	(Graph clustering, 2000 [44])	GT	$O(nk^2)$, $k < n$ parameter
Eckmann & Moses	(Curvature, 2002 [45])	GT	$O(mk^2)$
Girvan & Newman	(Modularity, 2002 [46])	SDP	$O(n^2m)$
Zhou & Lipowsky	(Vertex Proximity, 2004 [47])	GT	$O(n^3)$
Reichardt & Bornholdt	(spinglass, 2004 [48])	SDP	parameter dependent
Clauset *et al.*	(fast greedy, 2004 [49])	SDP	$O(nlog_2n)$
Newman & Girvan	(eigenvector, 2004 [12])	SP	$O(nm^2)$
Wu & Huberman	(linear time, 2004 [50])	GT	$O(n + m)$
Fortunato *et al.*	(infocentrality, 2004 [51])	SDP	$O(m^3n)$
Radicchi *et al.*	(Radicchi *et al.* 2004 [25])	SP	$O(m^4/n^2)$

Table 1. *Cont.*

Author	Reference	Category	Order
Donetti & Munoz	(Donetti and Munoz, 2004 [52])	SDP	$O(n^3)$
Guimera *et al*.	(Simulated Annealing, 2004 [53])	SDP	parameter dependent
Capocci *et al*.	(Capocci *et al*. 2004 [54])	SP	$O(n^2)$
Latapy & Pons	(walktrap, 2004 [14])	SP	$O(n^3)$
Duch & Arenas	(Extremal Optimization, 2005 [15])	GT	$O(n^2 logn)$
Bagrow & Bollt	(Local method, 2005 [55])	SDP	$O(n^3)$
Palla *et al*.	(overlapping community, 2005 [56])	GT	$O(exp(n))$
Raghavan *et al*.	(label propagation, 2007 [57])	GT	$O(n+m)$
Rosvall & Bergstrom	(Infomap, 2008 [58])	SP	$O(m)$
Ronhovde & Nussinov	(Multiresolution community, 2009 [59])	GT	$O(m\beta logn)$, $\beta \approx 1.3$
De Meo *et al*.	(Mixing information, 2014 [41])	SDP	$O(n^3)$
Jin *et al*.	(Geometric Brownian motion, 2014 [60])	SDP	$O(n^3)$

5.1. Distance Based Community Detection

There exist no algorithms in the literature of network community detection which compute direct nearest neighbor between nodes to the best of our knowledge; however, concepts of nearness used in some of the algorithms and they are described below.

Walktrap Algorithm (WT): This algorithm by Pons and Latapy [14] uses a hierarchical agglomerative method. Here, the distance between two nodes is defined in terms of random walk process. The basic idea is that if two nodes are in the same community, the probability to get to a third node located in the same community through a random walk should not be very different. The distance is constructed by summing these differences over all nodes, with a correction for degree. The complexity of the algorithm is $O(n^3)$ as reported in Latapy & Pons (walktrap, 2004 [14]).

Label Propagation Algorithm (LP): This algorithm by Raghavan *et al*. [57] uses the concept of node neighborhood and the diffusion of information in the network to identify communities. Initially, each node is labeled with a unique value. Then an iterative process takes place, where each node takes the label which is the most spread in its neighborhood. This process goes on until the conditions, no label change, is met. The resulting communities are defined by the last label values with the complexity $O(n+m)$ for each iteration as reported in Raghavan *et al*. (label propagation, 2007 [57]).

Geometric Brownian motion (GBM): This concept was borrowed from statistical physics by Zhou *et al*. [47] and extended by Jin *et al*. [60] with the inclusion of the concept of bispace. This method develops the notion of Brownian motion on networks to compute the influences between the nodes, which used to discover communities of social networks. The complexity of the algorithm is $O(n^3)$ as reported in Jin *et al*. (GBM, 2014 [60]).

5.2. Proposed Algorithm for Network Community Detection Using Nearest Neighbor Search

In this subsection we have described k-central algorithm for the purpose of network community detection by using the nearest neighbor search in a complex network. The community detection methods based on partitioning of graph is possible using nearest neighbor search, because the nodes of the graph are converted into the points of a metric space. This algorithm for network community detection converges automatically and does not compute the value of objective function in iterations therefore reduce the computation compared to standard methods. The k-central algorithm for community detection is (Algorithm 3) given below.

Algorithm 3 k-central algorithm

Require: $M = (V, d)$
Ensure: $T = \{C_1, C_2, \ldots, C_k\}$ with minimum $cost(T)$
1: Initialize centers $z_1, \ldots, z_k \in R^n$ and clusters $T = \{C_1, C_2, \ldots, C_k\}$
2: **repeat**
3: **for** $i = 1$ to k **do**
4: **for** $j = 1$ to k **do**
5: $C_i \leftarrow \{x \in V \; s.t. \; |z_i - x| \leq |z_j - x|\}$
6: **end for**
7: **end for**
8: **for** $j = 1$ to k **do**
9: $z_i \leftarrow Central(C_i)$; where $Central(C_i) \in C_i$
10: **end for**
11: **until** $|cost(T_t) - cost(T_{t+1})| = 0$
12: **return** $T = \{C_1, C_2, \ldots, C_k\}$

5.3. Complexity And Convergence

Complexity of the network community detection algorithms are the least studied research topic in network science. However, the rate of convergence is one of the important issues of algorithmic complexity and low rate of convergence is the major pitfall of the most of the existing algorithms. Due to the transformation into the metric space, our algorithm is equipped with the quick convergence facility of the k-partitioning on metric space by providing a good set of initial points. Another crucial pitfall suffer by majority of the existing algorithms is the validation of the objective function in each iteration during convergence. Our algorithm converges automatically to the optimal partition thus reduces the cost of validation during convergence.

Theorem 4. *During the course of the k center partitioning algorithm, the cost (community-wise total distance from the corresponding centers) monotonically decreases.*

Proof. Let $Z^t = \{z_1^t, \ldots, z_k^t\}$, $T^t = \{C_1^t, \ldots, C_k^t\}$ denote the centers and clusters at the start of the t^{th} iteration of k partitioning algorithm. The first step of the iteration assigns each data point to its closest center; therefore $cost(T^{t+1}, Z^t) \leq cost(T^t, Z^t)$

On the second step, each cluster is re-centered at its mean; therefore $cost(T^{t+1}, Z^{t+1}) \leq cost(T^{t+1}, Z^t)$. □

The main achievement of our algorithm is to use the rich literature of clustering using nearest neighbor. Clustering is easy NP-Hard in metric space, whereas graph partitioning is NP-Hard. Our algorithm converges automatically to optimal clustering. It does not require verifying the value of objective function guide next iteration, like popular approaches, thus saving the time of computation.

6. Experiments and Results

In this section we described in details several experiments to asses the, proposed nearness measure for the nodes of the network, efficiency of several approximation scheme to compute node nearness and performance of proposed algorithm for community detection. Several experiments conducted in this regard are detailed below along with their parameter settings, results and conclusions.

6.1. Experimental Designs

We performed three different experiments to asses the performance of the proposed network nearest neighbor search for community detection. The first experiment is designed to evaluate the

nearness measure, the second experiment is designed to explore the effectiveness of approximate nearest neighbor search for network community detection and the third experiment is designed to verify behavior of the algorithm and the time required to compute the algorithm. One of the major goals of the last experiment is to verify the behavior of the algorithm with respect to the performance of other popular methods exists in the literature in terms of standard modularity measures. Experiments are conducted over several real networks Table 2 to compare the results (Tables 5 and 6) of our algorithm with the state-of-the-art algorithms (Table 1) available in the literature in terms of modularity most preferred by the researchers of the domain of network community detection. The details of the several experiments and the analysis of the results are given in the following subsections.

6.2. Performance Indicator

Modularity: The notion of modularity is the most popular for the network community detection purpose. The modularity index assigns high scores to communities whose internal edges are more than that expected in a random-network model which preserves the degree distribution of the given network.

6.3. Datasets

A list of real networks taken from several real life interactions are considered for our experiments and they are shown in Table 2 below. We have also listed the number of nodes, number of edges, average diameter, and the k value used in Subsection 5.2. The values of the last column can be used to assess the quality of detected communities.

Table 2. Complex network datasets and values of their parameters.

Name	Type	# Nodes	# Edges	Diameter	k
DBLP	U	317,080	1,049,866	8	268
Arxiv-AstroPh	U	18,772	396,160	5	23
web-Stanford	D	281,903	2,312,497	9.7	69
Facebook	U	4039	88,234	4.7	164
Gplus	D	107,614	13,673,453	3	457
Twitter	D	81,306	1,768,149	4.5	213
Epinions1	D	75,879	508,837	5	128
LiveJournal1	D	4,847,571	68,993,773	6.5	117
Orkut	U	3,072,441	117,185,083	4.8	756
Youtube	U	1,134,890	2,987,624	6.5	811
Pokec	D	1,632,803	30,622,564	5.2	246
Slashdot0811	D	77,360	905,468	4.7	81
Slashdot0922	D	82,168	948,464	4.7	87
Friendster	U	65,608,366	1,806,067,135	5.8	833
Amazon0601	D	403,394	3,387,388	7.6	92
P2P-Gnutella31	D	62,586	147,892	6.5	35
RoadNet-CA	U	1,965,206	5,533,214	500	322
Wiki-Vote	D	7115	103,689	3.8	21

6.4. Experiment 1: Experiment with Nearness Measure

In this experiment we tried to asses the usefulness of proposed nearness measure between the nodes of complex network. For this purpose we have equiped our algorithm with different measures of nearness along with our measure. Experimental steps are as follows:
Nearness measures: Six different measures are taken for construction the distance based community detection. They are jaccard coefficient (JA), preferential attachment (PA), Katz measure (KM),

commute time (CT), page rank (PR) and proposed metric (PM). details of the measures are already discussed in Subsection 2.2 and proposed metric is detailed in Subsection 2.3.
Algorithm: The community detection algorithm proposed in Section 5 is used and exact nearest neighbor between nodes are considered and computed communities besed on those different nearness measures.
Network data: Different types of real network data is taken, small, large, very sparse and relatively dense and they are discussed in Table 2.
Results: Compared the community structure obtained by algorithms, equipped with different measures of node nearness, in terms of modularity and shown in Table 3.
Observation: It can be observed from Table 3 that algorithm based on proposed metric (shown in column PA) provides better modularity than other for community detection.

Table 3. Experiment 1: Experiment with nearness measure.

Name	JC	PA	KM	CT	PR	PM
Facebook	0.4806	0.4937	0.5037	0.4973	0.5206	0.5434
Gplus	0.3061	0.3253	0.3411	0.3309	0.3671	0.3998
Twitter	0.3404	0.3465	0.3508	0.3481	0.3582	0.3691
Epinions1	0.0667	0.0816	0.0943	0.0861	0.1150	0.1401
LiveJournal1	0.1010	0.1097	0.1167	0.1122	0.1284	0.1432
Pokec	0.0183	0.0205	0.0222	0.0211	0.0251	0.0288
Slashdot0811	0.0066	0.0080	0.0087	0.0082	0.0101	0.0127
Slashdot0922	0.0086	0.0105	0.0116	0.0109	0.0137	0.0171
Friendster	0.0360	0.0395	0.0422	0.0405	0.0467	0.0526
Orkut	0.0424	0.0476	0.0518	0.0491	0.0587	0.0675
Youtube	0.0375	0.0483	0.0574	0.0515	0.0724	0.0903
DBLP	0.4072	0.4103	0.4118	0.4110	0.4148	0.4207
Arxiv-AstroPh	0.4469	0.4590	0.4682	0.4624	0.4837	0.5045
web-Stanford	0.3693	0.3738	0.3765	0.3749	0.3815	0.3896
Amazon0601	0.2057	0.2174	0.2266	0.2207	0.2419	0.2615
P2P-Gnutella31	0.0180	0.0246	0.0302	0.0266	0.0394	0.0503
RoadNet-CA	0.0701	0.0893	0.1051	0.0949	0.1312	0.1633
Wiki-Vote	0.0874	0.1109	0.1308	0.1179	0.1633	0.2023

6.5. Experiment 2: Experiment on Approximation

In this experiment we explore the effectiveness of several approximation techniques of nearest neighbor search on complex network designed via metric tree and locality sensitive hashing. For this purpose we have equiped our algorithm with different data structures (metric tree and LSH) with varying approximation ratio. Experimental steps are as follows:
Metric and algorithm: The algorithms considered in this experiment used proposed measures of node nearness detailed in Subsection 2.3. The community detection algorithm proposed in Section 5 is used in this experiment.
Approximation: Computed communities using approximate nearest neighbor via metric tree and locality sensitive hashing. Different precision of approximation is considered ranges from 0–0.5 and computed five times each over both the scheme of approximation.
Network data: Different types of real network data is taken to verify the acceptablity of degradation over the networks and is shown in Table 4.
Results: Compared the community structure obtained by algorithms, equipped with approximate nearest neighbor instead of exact measures of node nearness, in terms of modularity and shown in Table 4.
Observations: Observed that both the approximation schemes are very good for community detection and slightly degrade the results under ranges of Approximations Table 4.

Table 4. Experiment 2: Experiment on approximation.

Name	Exact	0.1 Mtree	0.1 Lsh	0.2 Mtree	0.2 Lsh	0.3 Mtree	0.3 Lsh	0.4 Mtree	0.4 Lsh	0.5 Mtree	0.5 Lsh
Facebook	0.5472	0.5468	0.5462	0.5463	0.5452	0.5459	0.5441	0.5454	0.5431	0.5450	0.5421
Gplus	0.4056	0.4053	0.4049	0.4050	0.4042	0.4047	0.4035	0.4044	0.4028	0.4041	0.4021
Twitter	0.3709	0.3706	0.3701	0.3702	0.3693	0.3699	0.3685	0.3695	0.3677	0.3692	0.3669
Epinions1	0.1447	0.1446	0.1445	0.1445	0.1443	0.1445	0.1441	0.1444	0.1439	0.1443	0.1437
LiveJournal1	0.1458	0.1456	0.1454	0.1455	0.1450	0.1453	0.1447	0.1452	0.1443	0.1450	0.1439
Pokec	0.0295	0.0294	0.0293	0.0294	0.0292	0.0293	0.0290	0.0293	0.0289	0.0292	0.0287
Slashdot0811	0.0125	0.0124	0.0123	0.0123	0.0122	0.0121	0.0120	0.0120	0.0119	0.0119	0.0117
Slashdot0922	0.0168	0.0167	0.0167	0.0167	0.0166	0.0166	0.0164	0.0166	0.0163	0.0165	0.0162
Friendster	0.0536	0.0535	0.0534	0.0534	0.0532	0.0533	0.0529	0.0532	0.0527	0.0531	0.0525
Orkut	0.0690	0.0689	0.0688	0.0688	0.0685	0.0687	0.0683	0.0686	0.0680	0.0685	0.0678
Youtube	0.0936	0.0936	0.0935	0.0935	0.0934	0.0935	0.0932	0.0934	0.0931	0.0934	0.0930
DBLP	0.4215	0.4211	0.4206	0.4207	0.4197	0.4204	0.4189	0.4200	0.4180	0.4196	0.4171
Arxiv-AstroPh	0.5081	0.5077	0.5072	0.5073	0.5063	0.5069	0.5053	0.5065	0.5044	0.5061	0.5035
web-Stanford	0.3908	0.3904	0.3900	0.3901	0.3891	0.3897	0.3883	0.3894	0.3874	0.3890	0.3866
Amazon0601	0.2650	0.2647	0.2644	0.2645	0.2638	0.2642	0.2633	0.2640	0.2627	0.2637	0.2621
P2P-Gnutella31	0.0523	0.0523	0.0523	0.0523	0.0523	0.0523	0.0523	0.0523	0.0523	0.0523	0.0523
RoadNet-CA	0.1692	0.1690	0.1686	0.1687	0.1681	0.1685	0.1675	0.1682	0.1670	0.1680	0.1664
Wiki-Vote	0.2095	0.2094	0.2093	0.2093	0.2090	0.2092	0.2088	0.2091	0.2085	0.2090	0.2083

6.6. Experiment 3: Experiment to Evaluate Proposed Algorithm

In this experiment we have compared several algorithms for network community detection with our proposed algorithm developed using the nearest neighbor search in complex network, which is discussed in Section 5. The experiment is performed on a large list of network data sets Table 2. Two versions of the experiment are developed for comparison purposea based on modularity and time taken. The results are shown in the Tables 5 and 6 respectively.

Table 5. Comparison of our approaches with other best methods in terms of modularity.

Name	Spectral	SDP	GT	WT	LP	GBM	NN-Search	M-Tree	LSH
Facebook	0.4487	0.5464	0.5434	0.5117	0.5042	0.4742	0.5472	0.5450	0.5421
Gplus	0.2573	0.4047	0.3998	0.3528	0.3412	0.2963	0.4056	0.4041	0.4021
Twitter	0.3261	0.3706	0.3691	0.3545	0.3513	0.3375	0.3709	0.3692	0.3669
e Epinions1	0.0280	0.1440	0.1401	0.1034	0.0940	0.0589	0.1447	0.1443	0.1437
LiveJournal1	0.0791	0.1455	0.1432	0.1220	0.1169	0.0966	0.1458	0.1450	0.1439
Pokec	0.0129	0.0294	0.0288	0.0235	0.0223	0.0172	0.0295	0.0292	0.0287
Slashdot0811	0.0038	0.0130	0.0127	0.0095	0.0090	0.0060	0.0125	0.0119	0.0117
Slashdot0922	0.0045	0.0176	0.0171	0.0127	0.0119	0.0078	0.0168	0.0165	0.0162
Friendster	0.0275	0.0536	0.0526	0.0443	0.0423	0.0343	0.0536	0.0531	0.0525
Orkut	0.0294	0.0689	0.0675	0.0549	0.0519	0.0398	0.0690	0.0685	0.0678
Youtube	0.0096	0.0934	0.0903	0.0640	0.0573	0.0319	0.0936	0.0934	0.0930
DBLP	0.4011	0.4214	0.4207	0.4136	0.4125	0.4060	0.4215	0.4196	0.4171
Arxiv-AstroPh	0.4174	0.5079	0.5045	0.4755	0.4688	0.4410	0.5081	0.5061	0.5035
web-Stanford	0.3595	0.3908	0.3896	0.3791	0.3772	0.3673	0.3908	0.3890	0.3866
Amazon0601	0.1768	0.2649	0.2615	0.2336	0.2269	0.1999	0.2650	0.2637	0.2621
P2P-Gnutella31	0.0009	0.0522	0.0503	0.0343	0.0301	0.0146	0.0523	0.0523	0.0523
RoadNet-CA	0.0212	0.1690	0.1633	0.1168	0.1053	0.0603	0.1692	0.1680	0.1664
Wiki-Vote	0.0266	0.2093	0.2023	0.1451	0.1306	0.0752	0.2095	0.2090	0.208

Table 6. Comparison of our approaches with other best methods in terms of time.

Name	Spectral	SDP	GT	WT	LP	GBM	NN-Search	M-Tree	LSH
Facebook	6	7	11	13	7	8	6	4	1
Gplus	797	832	1342	1512	877	948	661	390	115
Twitter	462	485	786	886	509	554	398	235	68
Epinions1	411	419	667	749	452	475	292	174	56
LiveJournal1	1297	1332	2129	2394	1427	1514	969	576	179
Pokec	1281	1305	2075	2330	1410	1480	901	538	173
Slashdot0811	552	561	891	1000	608	636	382	228	74
Slashdot0922	561	570	906	1017	618	647	389	232	75
Friendster	2061	2105	3352	3766	2269	2390	1477	880	280
Orkut	1497	1529	2435	2736	1647	1735	1074	640	203
Youtube	829	844	1340	1505	913	957	578	345	111
DBLP	381	403	655	739	420	461	341	201	57
Arxiv-AstroPh	217	230	375	423	239	263	197	116	33
web-Stanford	498	525	852	960	549	600	437	258	74
Amazon0601	653	678	1089	1225	719	771	520	308	93
P2P-Gnutella31	182	184	293	328	200	209	124	74	24
RoadNet-CA	758	785	1261	1419	834	894	599	355	107
Wiki-Vote	54	55	88	99	59	63	39	23	7

Experimental steps are as follows:

Design of experiment: In this experiment we have compared three groups of algorithms for network community detection with one based on nearest neighbor search, described above. Two versions of the experiment are developed for comparison purposes based on modularity and time taken in seconds.

Best of literature: Regarding the three groups of algorithms; the first group contain algorithms based on semi-definite programming and the second group contain algorithms based on graph traversal approaches. For each group, we have taken the best value of modularity in Table 5 among all the algorithms in the groups. All the algorithms considered in this experiment are detailed in Section 5.

Other distance based methods: Three different methods of network community detection are also considered for our comparison which indirectly use the influence between the nodes in their algorithms. These methods are walktrap (WT), label propagation (LP) and geometric brownian motion (GBM) and already discussed in Section 5 along with their references and complexities.

Proposed methods: Three versions of proposed algorithm are compared with other algorithms, the proposed algorithm based on exact nearest neighbor, approximated nearest neighbor computed using metric tree and approximate nearest neighbor computed using locality sensitive hashing.

Network data: A long list of real network data is taken for evaluation of modularity and timedescribed in Table 4.

Efficiency and time: Compared the community structure obtained in terms of modularity and time (seconds) taken by the algorithms, shown in the Tables 5 and 6, respectively.

The results obtained with our approach are very competitive with most of the well known algorithms in the literature and this is justified over the large collection of datasets. On the other hand, it can be observed that time (second) taken (Table 6) by our algorithm is quite less compared to other methods and justify the theoretical findings.

6.7. Results Analysis and Achievements

In this subsection, we have described the analysis of the results obtained in our experiments shown. The results obtained in the first experiment justify that the proposed distance is more useful for complex network to extract the community structure compared to other measures of similarity. The results obtained in the second experiment verify that the approximate distance is also useful for network community detection especially for large data where time is a major concern. The results obtained in the third experiment justify that the proposed algorithm for community detection is very efficient compared to other existing methods in terms of modularity and time.

7. Conclusions

In this paper, we studied the interesting problem of the nearest neighbor within the nodes of a complex networks and applied this for community detection. We have used a geometric framework for network community detection instead of the traditional graph theoretic approach or spectral methods. Processing the nearest neighbor search in complex networks cannot be achieved straightforwardly; we presented the transformation of the graph to metric space and efficient computation of the nearest neighbor therein using metric tree and locality sensitive hashing. To validate the performance of proposed nearest neighbor search designed for complex networks, we applied our approaches on a community detection problem. Through several experiments conducted in this regard and we found community detection using nearest neighbor search is very efficient and time saving for large networks due to good approximations. The results obtained on several network data sets prove the usefulness of the proposed method and provide motivation for further application of other structural analysis of complex network using the nearest neighbor search.

Acknowledgments: This work is supported by the Jaypee University of Information Technology.

Author Contributions: Suman Saha proposed the algorithm and prepared the manuscript. Satya P. Ghrera was in charge of the overall research and critical revision of the paper. Both authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Uhlmann, J.K. Satisfying general proximity/similarity queries with metric trees. *Inf. Proc. Lett.* **1991**, *40*, 175–179.
2. Ruiz, E.V. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognit. Lett.* **1986**, *4*, 145–157.
3. Panigrahy, R. Entropy based nearest neighbor search in high dimensions. In Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm (SODA '06), Miami, FL, USA, 22–24 January 2006.
4. Indyk, P.; Motwani, R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC '98), Dallas, TX, USA, 23–26 May 1998.
5. Gionis, A.; Indyk, P.; Motwani, R. Similarity search in high dimensions via hashing. In Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99), Edinburgh, UK, 7–10 September 1999.
6. Dasgupta, S.; Freund, Y. Random projection trees and low dimensional manifolds. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC '08), Victoria, BC, Canada, 17–20 May 2008.
7. Akoglu, L.; Khandekar, R.; Kumar, V.; Parthasarathy, S.; Rajan, D.; Wu, K.L. Fast nearest neighbor search on large time-evolving graphs. In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, Nancy, France, 15–19 September 2014.
8. Liu, T.; Moore, A.W.; Gray, E.; Yang, K. *An Investigation of Practical Approximate Nearest Neighbor Algorithms*; MIT Press: Cambridge, MA, USA, 2004; pp. 825–832.
9. Weiss, R.; Jacobson, E. A method for the analysis of complex organisations. *Am. Sociol. Rev.* **1955**, *20*, 661–668.
10. Schaeffer, S.E. Graph clustering. *Comput. Sci. Rev.* **2007**, *1*, 27–64.
11. Fortunato, S. Community detection in graphs. *Phys. Rep.* **2010**, *486*, 75–174.
12. Newman, M.E.J.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **2004**, *69*, 026113.
13. Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.* **2007**, *17*, 395–416.
14. Pons, P.; Latapy, M. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.* **2004**, *10*, 284–293.
15. Duch, J.; Arenas, A. Community detection in complex networks using Extremal Optimization. *Phys. Rev. E* **2005**, *72*, 027104.
16. Chakrabarti, D. AutoPart: Parameter-Free Graph Partitioning and Outlier Detection. *Knowledge Discovery in Databases: PKDD 2004*; Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3202, pp. 112–124.
17. Macropol, K.; Singh, A.K. Scalable discovery of best clusters on large graphs. *Proc. VLDB Endow.* **2010**, *3*, 693–702.
18. Levorato, V.; Petermann, C. Detection of communities in directed networks based on strongly p-connected components. In Proceedings of the 2011 International Conference on Computational Aspects of Social Networks (CASoN), Salamanca, Spain, 19–21 October 2011; pp. 211–216.
19. Brandes, U.; Gaertler, M.; Wagner, D. Experiments on Graph Clustering Algorithms. *Algorithms - ESA 2003*; Battista, G.D., Zwick, U., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2832, pp. 568–579.
20. Bullmore, E.; Sporns, O. Complex brain networks: Graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci.* **2009**, *10*, 186–198.
21. Saha, S.; Ghrera, S.P. Network community detection on metric space. *Algorithms* **2015**, *8*, 680–696.
22. Freeman, L.C. Centrality in social networks conceptual clarification. *Soc. Netw.* **1978**, *1*, 215–239.
23. Carrington, P.J.; Scott, J.; Wasserman, S. (Eds.) *Models and Methods in Social Network Analysis*; Cambridge University Press: Cambridge, UK, 2005.

24. Newman, M. The structure and function of complex networks. *SIAM Rev.* **2003**, *45*, 167–256.
25. Radicchi, F.; Castellano, C.; Cecconi, F.; Loreto, V.; Parisi, D. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA* **2004**, *101*, 2658–2663.
26. Ciaccia, P.; Patella, M.; Zezula, P. M-tree: An efficient access method for similarity search in metric spaces. In Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97), Athens, Greece, 25–29 August 1997.
27. Ciaccia, P.; Patella, M.; Zezula, P. A cost model for similarity queries in metric spaces. In Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '97), Seattle, WA, USA, 1–4 June 1998.
28. Motwani, R.; Naor, A.; Panigrahy, R. Lower Bounds on Locality Sensitive Hashing. *SIAM J. Discret. Math.* **2007**, *21*, 930–935.
29. Paulevé, L.; Jégou, H.; Amsaleg, L. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognit. Lett.* **2010**, *31*, 1348–1358.
30. Joly, A.; Buisson, O. A posteriori multi-probe locality sensitive hashing. In Proceedings of the 16th ACM International Conference on Multimedia, Vancouver, BC, Canada, 27–31 October 2008; pp. 209–218.
31. Datar, M.; Immorlica, N.; Indyk, P.; Mirrokni, V.S. Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the Twentieth Annual Symposium on Computational Geometry (SCG '04), Brooklyn, NY, USA, 9–11 June 2004.
32. Andoni, A.; Indyk, P. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM* **2008**, *51*, 117–122.
33. Charikar, M.S. Similarity Estimation Techniques from Rounding Algorithms. In Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing (STOC '02), Montreal, QC, Canada, 19–21 May 2002.
34. Indyk, P. A Sublinear Time Approximation Scheme for Clustering in Metric Spaces. In Proceedings of the 40th Annual Symposium on Foundations of Computer Science, New York, NY, USA, 17–19 October 1999; pp. 154–159.
35. Leskovec, J.; Lang, K.J.; Mahoney, M.W. Empirical comparison of algorithms for network community detection. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010.
36. Yang, J.; Leskovec, J. Defining and Evaluating Network Communities Based on Ground-Truth. In Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, Beijing, China, 12–16 August 2012.
37. Zarei, M.; Samani, K.A.; Omidi, G.R. Complex eigenvectors of network matrices give better insight into the community structure. *J. Stat. Mech. Theory Exp.* **2009**, *2009*, P10018.
38. Pan, G.; Zhang, W.; Wu, Z.; Li, S. Online community detection for large complex networks. *PLoS ONE* **2014**, *9*, e102799.
39. Lee, C.; Cunningham, P. Community detection: Effective evaluation on large social networks. *J. Complex Netw.* **2014**, *2*, 19–37.
40. Aldecoa, R.; Marin, I. Exploring the limits of community detection strategies in complex networks. *Sci. Rep.* **2013**, *3*, doi:10.1038/srep02216.
41. De Meo, P.; Ferrara, E.; Fiumara, G.; Provetti, A. Mixing local and global information for community detection in large networks. *J. Comput. Syst. Sci.* **2014**, *80*, 72–87.
42. De Meo, P.; Nocera, A.; Terracina, G.; Ursino, D. Recommendation of similar users, resources and social networks in a social internetworking scenario. *Inf. Sci.* **2011**, *181*, 1285–1305.
43. Becker, H.; Naaman, M.; Gravano, L. Learning similarity metrics for event identification in social media. In Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM '10), New York, NY, USA, 3–6 February 2010.
44. Van Dongen, S. *A Cluster Algorithm For Graphs*; Technical Report INS-R 0010; CWI: Amsterdam, The Netherlands, 2000.
45. Eckmann, J.P.; Moses, E. Curvature of co-links uncovers hidden thematic layers in the World Wide Web. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 5825–5829.
46. Girvan, M.; Newman, M.E.J. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826.

47. Zhou, H.; Lipowsky, R. Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. *Computational Science - ICCS 2004*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3038, pp. 1062–1069.
48. Reichardt, J.; Bornholdt, S. Detecting fuzzy community structures in complex networks with a Potts model. *Phys. Rev. Lett.* **2004**, *93*, 218701.
49. Clauset, A.; Newman, M.E.J.; Moore, C. Finding community structure in very large networks. *Phys. Rev. E* **2004**, doi:10.1103/PhysRevE.70.066111.
50. Wu, F.; Huberman, B. Finding communities in linear time: A physics approach. *Eur. Phys. J. B* **2004**, *38*, 331–338.
51. Fortunato, S.; Latora, V.; Marchiori, M. Method to find community structures based on information centrality. *Phys. Rev. E* **2004**, *70*, 056104.
52. Donetti, L.; Muñoz, M.A. Detecting network communities: A new systematic and efficient algorithm. *J. Stat. Mech. Theory Exp.* **2004**, *2004*, P10012.
53. Guimera, R.; Amaral, L.A.N. Functional cartography of complex metabolic networks. *Nature* **2005**, *433*, 895–900.
54. Capocci, A.; Servedio, V.D.P.; Caldarelli, G.; Colaiori, F. Detecting communities in large networks. *Physica A* **2004**, *352*, 669–676.
55. Bagrow, J.P.; Bollt, E.M. Local method for detecting communities. *Phys. Rev. E* **2005**, *72*, 046108.
56. Palla, G.; Derenyi, I.; Farkas, I.; Vicsek, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **2005**, *435*, 814–818.
57. Raghavan, U.N.; Albert, R.; Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **2007**, *76*, 036106.
58. Rosvall, M.; Bergstrom, C.T. Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci.* **2008**, *105*, 1118–1123.
59. Ronhovde, P.; Nussinov, Z. Multiresolution community detection for megascale networks by information-based replica correlations. *Phys. Rev. E* **2009**, *80*, 016109.
60. Jin, F.; Khandpur, R.P.; Self, N.; Dougherty, E.; Guo, S.; Chen, F.; Prakash, B.A.; Ramakrishnan, N. Modeling mass protest adoption in social network communities using geometric brownian motion. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14), New York, NY, USA, 24–27 August 2014.