Social Structure Detection

Kai Yang, Weili Wu, Wei Zhang, Tzuwei Hsu and Lidan Fan

Contents

Abstract

This chapter mainly studies four problems proposed in social networks: they are link prediction problem, community detection problem, influence maximization problem, and routing problem for networks that satisfy power-law distribution property. As for these problems, corresponding efficient approximation algorithms with different aspects are introduced. Except numerical results of

K. Yang (✉) • W. Wu • W. Zhang • T. Hsu • L. Fan
Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA
e-mail: yangkai2011@gmail.com; weiliwu@utdallas.edu; dc.zhangwei@gmail.com; TzuweiHsu@utdallas.edu; ldfan28@gmail.com

P.M. Pardalos et al. (eds.), *Handbook of Combinatorial Optimization*,
DOI 10.1007/978-1-4419-7997-1_74,

experimental simulations of these algorithms, theoretical analysis for influence maximization problem and routing problem is provided.

1 Introduction

A social network is a kind of social structure that is made up of a finite set of individuals (called "nodes") and relationships (friendship, kinship, common interest) defined on them. Social network existed before the emergence of internet. In a social network, a person can use existing contacts as potential social links to extend their relationships to more people. Recently, online social network, which appears after internet, becomes popular among the public. Most online social network such as MySpace, LinkedIn, and Facebook provide a convenient platform for people to communicate with each other, exchange their ideas or spread information, and so on. Therefore, based on the relationships established between people, communities are shown to be an important local structure, which can be used to efficiently predict new links or disseminate information to their members by already known information.

This chapter consists of four parts: social network structure, local structure, influence maximization problem, and routing scheme for power-law graphs in social networks. In Sect. 2, a link prediction problem, which is a basic computational problem underlying social network evolution, is introduced. Section 3 focuses on local structural properties of social networks, also known as community structure. Some community properties and several algorithms for community identification are introduced. Sections 4 and 5 present two hot research topics in social networks. Influence maximization problem is addressed when a new idea is supposed to have more people to believe it or a company would like to make more customers to purchase a new product with the "word-of-mouth" effect in a social network. How to choose the people and take advantage of the relationships between people to promote your idea or product is what the influence maximization problem concerns about. Routing problem mainly focuses on compact routing scheme by using the theory of unweighted random power-law graphs with fixed expected degree sequence. The method discussed has the first theoretical bound coupled to the parameter of the power-law graph model for a compact routing scheme.

2 Social Network Structure

In recent years, people have witnessed the rapid growth of interests in networks, which are pervasive in the real world. With extensive studies, many researchers point out that the real-world networks demonstrate certain surprising consistent structure properties across different fields. Particularly, studies on three major characteristics of the networks, which are small world, clustering, and scale-free, are of great popularity. And based on those structural properties, three basic topological measurements including average path length, clustering coefficient, and degree

distribution are proposed to study the network from the topological perspective. In general, the average path length is the average distance between any pair of nodes. The degree of a node is the number of edges connected to that node. Newman [51] defined the clustering coefficient measure as

$$C = \frac{3 \times \text{number of triangles in the graph}}{\text{number of connected triples}} \tag{1}$$

where a triangle is formed by a set of three vertices, among which each node is connected to both of the others. A connected triple is three vertices u − v − w, with both vertices u and w connected with v (u − v − w and w − v − u are considered as the same connected triple). According to topological features, networks fall into mainly four different classes, which are information network, social network, technological networks, and biological networks. In this section, we focus on social networks.

A social network is a set of people or groups of people with some pattern of contacts or interactions among them under certain situation. From a graph theory point of view, a social network structure is defined as this: among a set of nodes and edges, the nodes represent people or other entities embedded in an underlying social context, and the edges represent interaction, collaboration, or influence between two entities. For instance, in the academic domain, nodes represent scientists in a particular discipline, and an edge is generated between two individuals if they have coauthored papers.

On one hand, from a static perspective that all edges are considered simultaneous, a social network has many classical invariant structural properties, such as small world phenomenon, clustering, scare-free, and so on. All of these characteristics provide us important information for probing other features of networks. On the other hand, in reality, social networks are highly dynamic, that is, they grow and change over time by adding new edges between nodes, which means that new connections occur in underlying networks. In order to realistically model the real world networks, it is of great advantage to consider the structure of social networks from a dynamic perspective.

Nowadays, along with the development of online large networks and the availability of the dataset collected from social networks, people are able to use the existing information to understand social network evolution or the potential structure pattern already existing but unobserved in the whole network. In this section, we would like to introduce the link prediction problem, which is a basic computational problem concerned with social network evolution.

2.1 Link Prediction Problem on Social Network

As a subfield of social network analysis, unlike other problems such as influence maximization, community detection, and small network, link prediction problem is concerned with the problem of predicting the future existence of links among nodes in a social network.

The link prediction problem defined in [66] is as follows: Given a social network $G = (V, E)$, where V denotes the set of entities and E is the set of observed links among those groups, then the aim of the problem is to predict how likely an unobserved link $e_{ij} \notin E$ exists between an arbitrary pair of nodes (v_i, v_j) in the network. In general, the link prediction problem can be studied from three different perspectives: they are link existence prediction (Does a link exist between two nodes?), link classification (What type of the relationship between a pair of nodes?), and link regression (How does the user estimate the item?), separately.

Until recently, link prediction has obtained a wide variety of applications among areas including bibliographic domain, molecule biology, criminal investigations, recommendation systems, and so on.

The authors in [66] summarized that the techniques used in the link prediction problem for social network fall into three categories according to the types of models: node-wise similarity-cased approaches, topological pattern-based approaches, and probabilistic model-based approaches. According to the resources used to predict links between nodes, methods are developed from two aspects: one is based on the attributes of nodes, and the other takes advantage of the structural properties of networks. Since it is hard to collect accurate individual attributes in real world, the structural properties make more contribution to the prediction problem.

In the following, we introduce approaches that use the topological features of social networks in link prediction problem.

2.2 Graph Proximity Approaches

The graph proximity measures for link prediction problem depend on structural features of the given network. The basic and intuitive method for predicting links is to rank all node pairs according to their graph topology measurements. That is, with regard to the input graph, a weight $\text{Score}(x, y)$, which is a subtle predictor based on node neighborhood or path information, is assigned to a pair of nodes (x, y); after each pair of nodes get their own scores, these nodes are ranked in decreasing order according to the values of their associated scores, and the top-ranked pairs of nodes are predicted to have high probability to have links between them. In addition, the meanings of these scores vary according to certain contexts.

2.2.1 Original Graph Proximity Measures

In [39], Liben-Nowell et al. studied this problem through a number of proximity measures to predict the new collaborations in a coauthorship network. Given a network $G = (V, E)$, where V denotes the groups of authors, $e = (u, v) \in E$ represents the interactions(coauthorship) that take place at a particular time t(e) between author u and author v.

In order to predict the interactions that will occur among those people, firstly, the authors in [39] chose four distinct time stamps $t_0 < t_0' < t_1 < t_1'$. Assume $[t_0, t_0']$ as

the training interval, in which the subgraph $G[t_0, t_0']$ contains the edges that appear between the time period from t_0 to t_0'. Meanwhile, regard $[t_1, t_1']$ as the test interval, which is used to validate the prediction accuracy. Then, a variety of predictors modified from techniques adopted in graph theory and network analysis are applied to predict the similarity of pairs of nodes. These predictors are common neighbors, graph distance, Jaccard's coefficient, and so on, among which Score(x, y) plays a vital role for predicting links that are likely to appear in the future. In this section, Score(x, y) denotes the proximity or similarity between nodes x and y with respect to the network topology.

To estimate the validity of these predictors, the authors in [39] brought in two parameters, $k_{training}$ and k_{test}, and focused on a set Core, which contains all nodes that are incident to at least $k_{training}$ edges in $G[t_0, t_0']$ and k_{test} edges in $G[t_1, t_1']$. For each link predictor p, the ranked list L_p of pairs of nodes in $V \times V - E_{old}$ (the set of edges developed in training interval) is made up of predicted links, which are arranged in the order of decreasing probability to appear in the future. Denote $E^*_{new} := E_{new} \cap (Core \times Core)$ and $n =: |E^*_{new}|$, and take the initial n pairs of nodes in the set of $Core \times Core$; here, the size of the intersection component of these nodes with set E^*_{new} is used to measure the performance of predictor p. According to the experiment analysis of predictors on coauthorship networks from different conferences, it was suggested that the datasets applied to this kind of networks are less noisy compared to other online networks, that is, these coauthorship networks rely on their own topological features rather than other external factors. Moreover, the experiment results showed that the approaches based on the structural features outperform those ones obtained from random prediction. Meanwhile, the score proved to be better than others.

2.2.2 Weighted Graph Proximity Measures

Unlike previous methods based on structural features such as Newman's common neighbors [48], Adamic and Adar method [4], and preferential attachment [45, 46] took into account of the weight of a link, here, the weight can be viewed as the number of encounters of a user on QABB (question-answering bulletin boards) corresponding to the number of times they meet or communicate.

The authors in [46] defined new scores that combine both structure proximity and link weight as follows:

The score of weighted common neighbor:

$$\text{Score}(x, y) = \sum_{z \in N(x) \cap N(y)} \frac{w(x, z) + w(y, z)}{2}, \tag{2}$$

where w(x, z) denotes the weight of the link between node x and node z.

The score of weighted Adamic and Adar method:

$$\text{Score}(x, y) = \sum_{x' \in N(x) \cap N(y)} \frac{w(x, z) + w(y, z)}{2} \times \frac{1}{\log(\sum_{z' \in N(z)} w(z', z))}. \quad (3)$$

The score of weighted preferential attachment:

$$\text{Score}(x, y) = \sum_{x' \in N(x)} w(x', x) \times \sum_{y' \in N(y)} w(y', y). \quad (4)$$

Murate and Moriyasu [46] collected the data information from the perspectives of encrypted user ID, categories, date, and time. The main idea is first, group QABB Data into two categories, that is, the early time data is used for training and the later one for testing, which aimed at making computation valid. Then, model a social network by adding links to all pairs of the answers in each question for each category. Finally, predict links that are possible to occur based on proximity measures. The validity of the predictors is measured by

$$\text{accuracy} = \frac{\text{number of correctly predicted links}}{\text{number of new links}}. \quad (5)$$

According to the experiment results in [46], it is shown that Adamic and Adar method performs better than the measure of common neighbors, and for networks whose degree distributions are almost uniform, preferential attachment performs bad. To be excited, the weighted Adamic and Adar method outperforms the original Adamic and Adar approach, weighted common neighbor measure also outperforms original common neighbors over almost all cases, and weighted preferential attachment works slightly better than original preferential attachment only when social networks are relatively dense. In general, when the weighted case is considered, the performances of link predictors are improved compared to previous pure proximity measures. The most important contribution of this measure is that it is fairly effective for open and dynamic online social networks, especially when the network is sufficiently dense.

2.2.3 Generalized Clustering Coefficient-Based Measures

The above two approaches solve the prediction problem under the condition that a network model has been given. However, when the model is not provided, how to build the model of a network? Moreover, it is shown that the methods for prediction problem have close relation with the topological structure of large-scale networks. Thus, it is plausible to generate parsimonious graph models, whose characteristics can be used to describe the significant mechanisms governing the structure of graphs.

Huang [33] showed that structural predictors summarize graph data categories with respect to graph generation model and explain link occurrences in an observed graph. Thus, those measures are of great value to predict the link appearance in the future. As for other predictors, more attention was paid on analyzing generalized

clustering coefficient, which is supposed to describe a cycle formation model. Based on the model, a new method for prediction problem was proposed.

Firstly, Huang [33] introduced relevant notations of a graph and defined the generalized clustering coefficient. Given $G = (V, E)$ a finite undirected graph with only simple edges, that is, there are no multiple edges or self-loops in the graph. $V = (1, 2, \cdots, N)$ is the set of vertices of G, and $E = (e_1, e_2, \cdots, e_M)$ is the set of edges of G, each e_s corresponds to a sequence of two vertices (i, j), and the terms of link and edge can be viewed as the same one. A path of length k is denoted as $p = (v_0, v_1, \cdots, v_k)$, where (v_i, v_{i+1}) is an edge of the graph for all $0 \leq i \leq k-1$. Define a cycle of length k as a list of vertices $p = (v_0, v_1, \cdots, v_k, v_0)$, where (v_i, v_{i+1}) is an edge of G for all $0 \leq i \leq k-1$. P_{ijk} represents the set of paths of length k starting at i and ending at j, and $|P_{ijk}|$ is the number of such paths. A generalized clustering coefficient C(k) of degree k was defined as

$$C(k) = \frac{\text{number of cycles of length k in the graph}}{\text{number of paths of length k}}. \tag{6}$$

Then the algorithm applied to the cycle formation link probability model is introduced, in which the occurrence probability of a link is determined by the number of cycles (of different lengths) that will be formed by adding this link. The algorithm made the assumption that the clustering coefficient (of deferent degrees) is static.

The Algorithm [33]

In this algorithm, a cycle formation model of degree $k(k \geq 1)$ is denoted as CF(k), and a list of parameters $c_1, \cdots, c_k$ are adopted for corresponding link generation mechanisms $g(1), \cdots, g(k)$, which are used to determine the probability that a link will appear. Here, g(1) is a mechanism similar to a random link generation process. Other mechanisms $g(k)'s$ $(k > 1)$ are consistent with link probability governed by the paths of length k, where length one means one edge between two nodes. $c_k = Pr((i, j) \in E||Pr_{ijk}| = 1)$ is used to measure the probability that a length-k path will become a length-k cycle. Derive from an instance, an equation for c_k was generalized:

$$Pr((i, j) \in E||P_{ijk}| = m) = \frac{c_k^m}{(c_k^m + (1 - c_k)^m)}, \; k > 1. \tag{7}$$

Considering the effects coming from multiple mechanisms, the above equation can be furthered as follows:

$$\begin{aligned} P_{m_2,\cdots,m_k} &= Pr((i, j) \in E||P_{ij2} = m_2, \cdots, |P_{ijk}| = m_k) \\ &= \frac{c_1 c_2^{m_2} \cdots c_k^{m_k}}{c_1 c_2^{m_2} c_k^{m_k} + (1 - c_1)(1 - c_2)^{m_2}(1 - c_k)^{m_k}}, \end{aligned} \tag{8}$$

where $P_{m_2,\cdots,m_k}$ denotes the total link occurrence probability under the cycle formation model of degree k.

Then we introduce the approaches adopted to estimate the parameters of the cycle formation model based on the generalized clustering coefficients. Actually, when the parameters are figured out, to obtain the link probabilities, it only needs to apply the estimated parameters to Eq. (7). The operations used in this algorithm are iterative.

Firstly, according to the degree distribution, start the computation with a complete random model, then get the cycle formation probability c_1. C(k) is regarded as the generalized clustering coefficient and is calculated one by one. Secondly, compare C(2) with c_1, if C(2) cannot be expressed by c_1, then new formation mechanism for length-2 cycle will be generated. Then, continue to compare the observed C(3) with the expected C(3) under the length-2 cycle formation model, so the estimator c_3 can be derived. When it satisfies the degree of the model, the procedure terminates. The primary component of this method depends on the fact that C(k) is a function of c_1 and has no relation with $c_{k'}$, here, $k' > k$. The method is formally described as follows:

1. Input information of $G = (V, E)$.
2. Compute the generalized clustering coefficients $C(2), \cdots, C(k)$ through Eq. (6).
3. Compute the connecting probability under random graph with the degree distribution of G as c_1, which is the cycle formation probability.
4. Denote c_2 as

$$c_2 = \frac{(1 - c_1)C(2)}{(c_1 - 2c_1C(2) + C(2))}.$$

5. Set $c_i = 0.5$, where $i = 3, \cdots, k$.
6. For $i = 3, \cdots, k$, iteratively apply the following equation:

$$c_i = \text{argmin}_{c_i'}(|C(i) - f(c_1, \cdots, c_i', \cdots, c_k)|).$$

7. Output the values of $c_1, \cdots, c_k$ that have been figured out.

The function which is denoted as

$$f(c_1, \cdots, c_i', \cdots, c_k) = \sum_i \#(G_i)\Pr(G_i)\Pr((1, k+1) \in E|G_i) \tag{9}$$

is the total probability of link $(1, k+1)$'s occurrence conditional on a path $p = (1, 2, \cdots, k+1)$. It is also the theoretical prediction of the expected clustering coefficient of degree k, denoted as E[C(K)] based on the cycle formation model. For a given path of length k, G_i is supposed to be a possible graph pattern, $\#(G_i)$ denotes the number of subgraphs corresponding to this graph pattern, and $\Pr(G_i)$ is the probability for one of the subgraphs to occur, the probability for edge $(1, k+1)$ to occur under certain condition of G_i is denoted as $\Pr((1, k+1) \in E|G_i)$.

Experiment Evaluation ([33])
The experiments were carried out on Enron email dataset, which is a large email collection from a real organization over the course covering a 3.5 years period. In the experiment, the author mainly evaluated the performances of the cycle formation link probability model and the corresponding link prediction algorithm. The dataset analyzed contains 40,489 emails during May 11, 1999, to June 21, 2002. And this paper implements the link prediction analysis on the monthly email graph in 2001. The email graph is undirected and unweighted, and the edges connect senders and recipients of emails during the corresponding time periods. An edge (a, b) means that there is at least one email communication between a and b, that is, either a sends at least one email to recipients including b or b sends at least one email to recipients including a. And month t was set in 2001 to build the initial graph G_{tb} through the emails in the previous 3 months $(t-3, t-2, t-1)$. This graph is regarded as the input for prediction of email links in G_t, thus, the primary goal is to predict the occurrence of links in G_t that do not exist in G_{tb}.

To evaluate the performance of link prediction, construct a receiver operating characteristics (ROC)-style curve with x-axis as the percent of total possible new links selected and y-axis as the percent of actual new links that are in the selected links, respectively. Then area under curve (AUC) measure is applied to estimate the link prediction performance. It is shown that the algorithm based on the cycle formation model performs better than others that already existed, and has great power to predict the probability of link occurrence.

2.3 Supervised Learning-Based Approaches

Facing the challenge of a wide application of the link prediction, it is necessary to construct both a powerful and universal framework. Although the work in [39] demonstrated improvement of measures for prediction problem over random predictors, they used the properties only based on network intrinsic topological structure, while there are other non-topological properties that proved to enhance the performance of methods for link prediction problem. In [32, 40, 43], the authors regarded the prediction problem as a classification modeling, in which they extracted the features both from topology and non-topology. Throughout their studies, supervised learning played a vital role, and [40] improved the predictive accuracy of supervised learning.

Firstly, we introduce the concept of supervised learning. Supervised learning is a task to infer a function from the supervised training data, which contains a big volume of samples, where each sample consists of an input object and a desired output value. Based on the data information, a supervised learning algorithm is used to analyze the training instances to produce the expected function, which will be applied to predict the output value of any input instances. In general, supervised learning contains the following steps:

1. Determine the types of training examples.
2. Collect a training set.
3. Determine the input feature representation for the learned function.
4. Determine the structure of the learned function and its corresponding learning algorithm.
5. Complete the design process.
6. Evaluate the performance accuracy of the learned function.

During supervised learning, four major issues should be considered. The first is the amount of training data available relative to the complexity of the "true" function (classifier or regression function), the second is the trade-off between bias and variance, the third is the dimensionality of the input space, and the fourth is the degree of noise in the desired output values (the supervisory targets).

2.3.1 Work of Madadhain et al. [43]

In this article, the authors studied an event-based network dataset, which consists of sets of events over time. They paid special attention to the temporal feature of the data, and two specific problems related to the event network data were extensively studied. Here, we only introduce one of them: predicting future event co-participation of entities, which aims to estimate to what extent that a given pair of individuals will co-participate in at least one event during some specific time period in the future.

To simplify the following analysis, we introduce the definition in [43], where $\mathcal{E} = \{e_1, \cdots, e_m\}$ denotes a set of events and $\mathcal{V} = \{v_1, \cdots, v_n\}$ represents the set of participating entities. The set of entities that participate in event e_i is named as P_i. Each event and each entity can have a set of attributes or covariates. And the covariates for e_i and v_j are denoted as y_i and x_j, respectively. The sets of all events and entity covariates are represented by $\mathbf{Y}$ and $\mathbf{X}$, respectively. The time that an event e_i occurrs is denoted as t_i, and i means that e_i is the ith occurred events. $v_j, v_k \in P_i$ means that v_j and v_k are co-participants in event e_i, and $v_j, v_k \in P_{t,t+\Delta t}$ means that v_j and v_k are co-participants in one or more events in the interval $[t, t + \Delta t]$. Meanwhile, name the subset of events taking place in the interval $\mathcal{E}_{t,t+\Delta}$. Usually, a vertex corresponds to an entity in a network derived from such a dataset, and edges connect vertices that participate in the same events.

The prediction problem was considered as a data-driven classification problem, in which there are two classes consisting of co-participating and not co-participating. Firstly, probabilistic classifiers were used to offer a probability to each class according to the values of some specified features. The probability is defined as follows:

$$p(v_j, v_k \in P_{t,t+\Delta t} | f(\mathcal{E}_{1,t}, \mathcal{V}, \mathbf{X}, \mathbf{Y}) = \mathbf{w}), \tag{10}$$

where $v_j, v_k \in P_{t,t+\Delta t}$ is a binary suggestion determining whether entities v_j and v_k co-participate in any event in the time period $[t, t + \Delta t]$, f is a function used to produce a vector $\mathbf{w}$ of feature values, $\mathcal{E}_{1,t}$ is the historical event data through to time t, and $\mathbf{X}$,$\mathbf{Y}$ are the relevant entity and event covariate data, respectively. Then based on

the formulation, the problem can be regarded as learning the mapping from feature vectors to class probabilities, and it can be computed by standard "off-the-shelf" prediction algorithms.

To construct a classification model, the authors proposed foundational components for the operation of classification, which include feature selection, training sets and test sets, classification method, and evaluation metric. Then based on the model, experiments were implemented and indicated that relatively standard machine learning approaches could be used to draw predictive information from the event data, which could be regarded as a ranking machine to detect the individual pairs that have high probability to co-participate in future event.

2.3.2 Work of Lichtenwalter et al. [40]

As for previous supervised methods, the researchers were interested in a domain of the prediction problems with highly imbalanced class distributions [32], although the measures adopted work well, the results of the tests operated on the modified data cannot provide the precise information of the real world, and the performance measures in tests were not available to present the strength and weakness of the models. On the other hand, these models mainly use both the semantic and contextual information related almost exclusively with the bibliographic realm. In addition, the surprising influence of geodesic distance and the complexity of class imbalance specific to the task of link prediction were neglected in those works. Furthermore, many factors, which played significant role in influence and guide classification, had not been explored previously by supervised learning.

In [40], the authors first proposed a supervised framework, which considered the factors of the observational period in the network: generality of existing methods, variance reduction, topological causes, and degrees of imbalance. The attractive merit of the framework is its universality, which means that it can be used on any kind of networks, that is, whether a network is weighted, unweighted, directed, or undirected, the framework can be applied to it. Additionally, the framework has the capability of accepting vertex attributes even though it does not need them. Besides studies in supervised learning for prediction problems, an intuitive flow-based metric was applied to the unsupervised measures to obtain more predictive results.

Next, we would like to introduce the PropFlow method for unsupervised prediction, which was proved to be an efficient predictor. The procedure of the algorithm is given as follows.

The PropFlow method [40] corresponds to the possibility that a restricted random walker starts at v_i and terminates at v_j in k steps or fewer by using link weights as transition probabilities. The walker chooses links based on their weights, and the score in the method is applied to predict the occurrence of new links in the future. PropFlow is somewhat like rooted PageRank, but it is a more localized approach of diffusion and is insensitive to topological noise that is far from the source node. Furthermore, PropFlow simply employs a modified breadth-first search restricted to height k, by which it computes faster.

Algorithm 1 PropFlow predictor [40]

Input information of network G = (V, E), node v_s and maximal length l
Add node v_s to a set denoted as *Found*
Put node v_s to another set called *NewSearch*
Add (v_s, 1) into S
For CurrentDegree, which ranges from 0 to l, **do**
  Assign the value of *NewSearch* to a set *OldSearch*
  Make *NewSearch* as a empty set
  While *OldSearch* is not empty, **do**
    Bring out v_i from *OldSearch*
    Find *NodeInput* through v_i in S
    Assign *SumOutput* 0
    For each v_j in v_i's neighborhood **do**
      Add weight of e_{ij} to *SumOutput*
    End for
    Set the value of *Flow* to be 0
    For each v_j in v_i's neighbors **do**
      Assign the weight of e_{ij} to w_{ij}
      Assign *Flow* the value of *NodeInput* $\times w_{ij}$ *SumOutput*
      Sum (v_j, *Flow*) and put it into S
      If v_j is not in *Found* **then**
        Add v_j into *Found*
        Put v_j to *NewSearch*
      end if
    end for
  end while
end for
Output score S_{sd} for all $n \leq$ l-degree neighbors v_d of v_s

Supervised learning, which includes dataset collection, generalization, variance reduction, and sampling, was investigated. Based on the detailed study of those above operations, class imbalance is proposed. Combining the nature of the prediction problem and the benefit of supervised learning, the author implemented the supervised framework by classification including general feature extraction, ensemble of classifiers, and overcoming imbalance.

The experiments in [40] were carried out on two datasets: one is a stream of 712 million cellulars from a major non-American cellular phone service provider, and the other is a stream of 19,464 multi-agent events representing condensed matter physics collaborations from 1995 to 2000. For the former dataset, a weighted, directed network *(phone)* was built, in which a node v_i represented a caller, and if v_i called v_j, a weighted and directed link e_{ij} would connect v_i with v_j, where weight was the number of calls on the link. For the latter dataset, a weighted, undirected network *(condmat)* from the collaborations was constructed, in which a node represented an author in the event, and a weighted, undirected link was used to denote the interactions between each pair of authors.

Based on those above experiments, it is shown that the general supervised framework outperformed other existing approaches, and the framework was demonstrated to be entirely general, that is, it is able to operate on any kinds of networks whether it

is weighted, unweighted, directed, or undirected. It has the capacity to accept vertex attributes though it does not need to consider them. Get more useful results reading [40].

3 Local Structure

One of the most significant characters of social networks is the local structure, also known as community structure. That is, a massive social network always contains many very dense subgraphs. Identifying these dense subgraphs or communities will help reveal the underlying network structure and analyze the common activities of individuals. In this section, we will introduce the community properties and review several algorithms for community identification.

3.1 Social Community

Several significant characters of social networks have drawn the attentions of many researchers, such as the small-world property, power-law degree distributions, and local structures, also known as communities. In this section, we will focus on this community property.

In a social network, individuals are shown as nodes, and interactions between the individuals are presented by edges, like relationships and influence. The individuals tend to form communities. A community is a group of nodes that are similar to each other and dissimilar from the rest nodes in the network. In a network, it is usually thought as a group where nodes are densely interconnected and sparsely connected to other parts of the network.

So far, several definitions for community structure have been derived. One of the most intuitive ways is to define community in terms of cliques. A clique in a graph is a subgraph in which any pair of nodes are linked. Based on this, the authors in [58] regard maximal cliques in a graph as communities. A maximal clique is a clique that is not contained by any larger clique. Abello et al. [1] generalized the definition of clique and proposed a structure named quasi-clique. A connected subgraph is considered to be a quasi-clique when it is dense enough. Filippo et al. [54] proposed two definitions for community structure. Consider a subgraph V of graph G. For any node $i \in V$, let K_i^{in} be the inside degree of node i, for example, the number of links toward nodes in V. And let K_i^{out} be the outside degree of node i. Then they defined that the subgraph V is a community in a strong sense if

$$K_i^{in} > K_i^{out}, \quad \forall i \in V. \tag{11}$$

In this strong community, each node has more connections within the community than with the rest of the graph. Also, they defined the community in a weak sense by

$$\sum_{i \in V} K_i^{in} > \sum_{i \in V} K_i^{out}. \tag{12}$$

In this weak community, the sum of all degrees within V is larger than the sum of all degrees toward the rest of the network.

Although none of the above definitions can be considered universal, they are all capable to capture the underlying properties of social networks.

3.2 Social Community Identification

Community identification is to find groups that either have an inherent or an externally specified notion of similarity among their nodes. As mentioned earlier, people in the same community are more similar than people from different communities. Which means that they might share more on information, interests, experiences, and other useful resources. So discovering the underlying community structures has direct impacts on optimizing and managing activities in a social network. Agarwal and Kempe [5] showed that identifying communities might help people study the whole massive network by communities. That is, after partitioning the graph into communities, one can start focusing on single community. Since nodes of the same community have considerable overlapping on their characters, the analysis inside one community would be more convenient and meaningful. They also claimed that by contracting each community into a node, one can simplify the original large-scale network considerably. This will help people get to know the network from a very high point of view.

Due to the above reasons, community identification has become a very important issue in the social network study. However, nowadays, social networks are usually huge massive networks consisting of millions of nodes, so in general, very little is known about the community structure of a graph; thus, one of the challenges in community identification is that the number of communities in a network and their sizes are not known beforehand. Furthermore, the communities need to be established by the community detection algorithm. Next, we will introduce algorithms within different categories and show how they deal with the challenge and generate communities effectively.

3.2.1 Hierarchical Clustering

Hierarchical clustering is the most widely used method among traditional community identification methods. The core of any hierarchical clustering method is the definition of a similarity measure between vertices. Once such a measure is set, one can compute and sort the similarity values for all pairs of nodes in the given network. Based on the direction of a community formulation procedure, the corresponding methods fall into the following two categories:

Agglomerative algorithms: start with a non-edge graph with only node set. Edges are iteratively added into this graph by decreasing order of similarity values until the original graph is formed.

Divisive algorithms: run in the opposite direction of agglomerative algorithms. They start with the original graph, and edges are deleted based on the increasing order of the similarity values.

The two categories of algorithms will finally generate a tree or dendrogram for the original graph. All the nodes of the given graph are presented by leaves. A non-leaf node on the tree denotes the community resulted from the merging of two smaller communities. The root would be the original graph.

The hierarchical clustering method does not provide any measure of when the clustering procedure should be terminated. So as for the size and number of communities, the terminating point should be manually decided according to the application scenario.

Several Similarity Measures

Structural equivalence is a property derived from sociological studies. Two nodes are of structural equivalence if they have the same set of neighbors. A measure called correlation coefficient was proposed in [63] based on structural equivalence. Suppose $A_{i,j}$ is the adjacent matrix for the given graph. Define means and variances of the columns as

$$\mu_i = \frac{1}{n}\sum_j A_{ij}, \quad \sigma_i^2 = \frac{1}{n}\sum_j (A_{ij} - \mu_i)^2; \tag{13}$$

the correlation coefficient is

$$x_{ij} = \frac{\frac{1}{n}\sum_k (A_{ik} - \mu_i)\sum_k (A_{jk} - \mu_j)}{\sigma_i \sigma_j}. \tag{14}$$

Vertices that have a high degree of structural equivalence will have high values of this similarity measure, and meanwhile, those that do not have a high degree of structural equivalence will have low values.

Another similarity measure is the number of edge (vertex)-independent paths between two nodes [64]. Intuitively, the more independent paths between two nodes, the more related they are. This measure is easy to calculate through augmenting path algorithm.

A measure called "edge betweenness" was proposed in [26]. Recall that in hierarchical clustering method, the algorithm iteratively finds the most "central" edges and adds them into communities. Different from those traditional similarity measures, "edge betweenness" describes how an edge is "between" communities and the algorithm will run in a divisive manner.

For edge e, the edge betweenness of e is defined to be the number of shortest paths between pairs of other vertices that run through e. The idea behind this definition is that if a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness.

Their algorithm is proposed as follows:

Algorithm 2 Algorithm based on edge betweenness [26]

1: For each edge in the graph, calculate its edge betweenness. Sort all the edges by decreasing order of the edge betweenness.
2: Recalculate betweennesses for all edges (or for time efficiency, only those whose betweenness value are changed after the removal). Perform another sorting process.
3: Repeat steps 2 and 3 until there is no edge left.
4: Based on the removal, a hierarchical dendrogram can be generated. Then a community partition can be decided manually.

3.2.2 Modularity-Based Algorithm

As mentioned earlier, so far, there is no universal definition for communities. But a quantity known as modularity, derived by Newman and Girvan [50], can measure how good a community partition is.

For a given graph $G = (V, E)$, let A be its adjacent matrix; then

$$A_{vw} = \begin{cases} 1 & \text{when } (v, w) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Suppose all the vertices have been divided into communities. Let c_v denote the community node v belongs to. Define function δ to be

$$\delta(i, j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Then the fraction of edges that fall within communities, that is, edges that connect vertices that both lie in the same community, is

$$\frac{\sum_{vw} A_{vw}\delta(c_v, c_w)}{\sum_{vw} A_{vw}} = \frac{1}{2m}\sum_{vw} A_{vw}\delta(c_v, c_w), \tag{17}$$

where $m = \frac{1}{2}\sum_{vw} A_{vw}$ is the number of edges of graph G. This quantity is not suitable to measure the strength of community partitions for the case that all the nodes form the same community. Under this case, the quantity is maximized while there is no community structure information proposed at all. However, if the expected value of the same quantity is subtracted from it in the case that every edge is generated randomly, an effective measure can be obtained.

Set k_v to be the degree of node v:

$$k_v = \sum_{v} A_{vw}. \tag{18}$$

Then the probability that there is a random edge between node v and node w is

$$p_{vw} = \frac{k_v \cdot k_w}{2m}. \tag{19}$$

The modularity Q is defined to be

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w). \tag{20}$$

By defining modularity in this way, the case that the whole graph forms one community can be avoided since the corresponding quantity is 0. This modularity expression can be reformulated through the following two quantities:

$$e_{ij} = \frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, j) \delta(c_w, j), \tag{21}$$

which denotes the fraction of edges between community i and community j. Another quantity is

$$a_i = \frac{1}{2m} \sum_{v} k_v \delta(c_v, i), \tag{22}$$

which is the fraction of edges that have one endpoint falling in community i. Notice that

$$\delta(c_v, c_w) = \sum_{i} \delta(c_v, i) \delta(c_w, i), \tag{23}$$

then

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w) \\ &= \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \sum_{i} \delta(c_v, i) \delta(c_w, i) \\ &= \sum_{i} \left[\frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, i) \delta(c_w, i) - \frac{1}{2m} \sum_{v} k_v \delta(c_v, i) \frac{1}{2m} \sum_{w} k_w \delta(c_w, i) \right] \\ &= \sum_{i} (e_{ii} - a_i^2). \end{aligned} \tag{24}$$

Newman and Girvan [50] showed that, in practice, values being greater than about 0.3 appear to indicate significant community structure. Based on modularity, many algorithms have been designed, which makes modularity currently the most widely used measurement for community identification. Next, we start introducing several of these algorithms.

Greedy Algorithms

One most intuitive algorithm is a greedy strategy derived by Newman [49]. At the beginning, each node forms a sole community. Then communities are integrated together in pairs repeatedly. Which pair to be combined is based on the increase

that they make on Q. The procedure continues until all the nodes are contained in one community. Clearly when the algorithm terminates, a dendrogram is generated, showing the order of these combinations. Then as introduced earlier, communities partition can be manually determined. Each step of the algorithm takes time $O(m + n)$ in worstcase. At most, $n - 1$ combination operations are needed to construct the complete dendrogram. So the algorithm runs in time $O((m + n)n)$. When it comes to sparse graph, it is $O(n^2)$.

Clauset et al. [16] improved the above algorithm and proposed another similar algorithm but has running time O(mdlogn), where d is the depth of the dendrogram describing the community structure. In their algorithm, instead of maintaining the adjacency matrix and calculating ΔQ_{ij} for each pair (i, j), a matrix of value of ΔQ_{ij} was directly maintained and updated. Communities that have no edges between them will not make changes to the value of Q. Hence, the algorithm will only focus on communities that are linked by some edges. Additionally, another two data structures, which will further save both memory and time consumption of the algorithm, were maintained to track the largest Q_i. These two data structures are described as follows:

1. A vector array H recording the largest element of each row of matrix ΔQ_{ij} and the corresponding labels i,j
2. A vector array with elements a_i

The algorithm also starts with the state that each vertex forms a sole community. Initially

$$e_{ij} = \begin{cases} 1/2m & \text{if } (i,j) \in E, \\ 0 & \text{otherwise}, \end{cases} \tag{25}$$

and $a_i = k_i/2m$. So

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j/(2m)^2 & (i,j) \in E, \\ 0 & \text{otherwise}. \end{cases} \tag{26}$$

The algorithm then proceeds as follows:

Algorithm 3 Greedy algorithm [16]

1: Calculate the values of ΔQ_{ij} and a_i, and construct vector array H.
2: Choose the largest ΔQ_{ij} from H, merge the corresponding communities, and update the matrix ΔQ, vector array H, Q, and a_i.
3: Repeat step 2 till there is only one community left.

Step 2 is supposed to be carried out faster according to the definition of H and a_i. If communities i and j are merged, then the ith row and column of ΔQ are removed and the jth row and column are updated. The update rules are as follows:

If community k is connected to both i and j, then

$$\Delta Q'_{jk} = \Delta Q_{ik} + \Delta Q_{jk}. \tag{27}$$

If k is only connected to i, then

$$\Delta Q'_{jk} = \Delta Q_{ik} - 2a_j a_k. \tag{28}$$

If k is only connected to j, then

$$\Delta Q'_{jk} = \Delta Q_{ik} - 2a_i a_k. \tag{29}$$

Searching Strategies

Simulated annealing [7] is an optimization technique that stochastically avoids local peaks by introducing a computational temperature T. When T is high, the searching direction is flexible. Which means that the algorithm is allowed to reach an area with an objective value worse than the current spot. When T decreases, such flexibility is weakened. And after T reaches 0, the algorithm will stop and output the current solution.

In [31], the authors applied the simulated annealing technique for modularity-based identification algorithm to maximize the modularity value. So here, the objective of the simulated annealing procedure is

$$C = -Q, \tag{30}$$

where Q is still the value of the modularity.

There are three operations during the annealing process:

1. Movements of nodes between communities
2. Merging two communities into a new community
3. Splitting one community into two new communities

These operations are performed with probability

$$p = \begin{cases} 1 & \text{if } C_f \leq C_i, \\ \exp\left(-\frac{C_f - C_i}{T}\right) & \text{if } C_f > C_i. \end{cases} \tag{31}$$

The algorithm starts with an initial value for T and runs iteratively. In each iteration, certain numbers of operation are performed based on the probability defined above. Then T is decreased for a fixed value and another iteration starts. Once T reaches 0, the algorithm will terminate and the final communities will be outputted.

3.2.3 Fast Algorithms

The real-world networks are usually huge in size and always have massive structure thus, time efficiency of the community detection algorithm is an important performance measure. The above several algorithms we introduced are usually very time-consuming while facing such complex networks. In this section, we will introduce several algorithms that have near linear running time.

In [65], the authors presented a method that allows for the discovery of communities within graphs of arbitrary size in times that scale linearly with their size. The basic idea behind the algorithm is as follows.

Firstly, partition the given graph $G = (V, E)$ into two communities. At the very beginning, two nodes A and B are selected and assumed to belong to two different communities (the selection policy will be introduced later). Suppose the two communities are presented by G_1 and G_2. Then the whole graph is considered as an electric circuit. Edges of the graph are taken as resistors of the same resistance. A battery is placed with A and B as the poles, which will add constant voltages on them, suppose 1 on A and 0 on B. Then there will be flows flowing through all the edges. By solving Kirchhoff equations, the voltages V_i for each node i can be obtained, which lie between 0 and 1. Based on a threshold given beforehand, it will be easy to decide which community each node belongs to.

For a certain node C in the graph, suppose its neighbors are $D_1, \ldots, D_n$. According to Kirchhoff equations, let I_i denote the current flowing from D_i to C, the following equation holds

$$\sum_{i=1}^{n} I_i = \sum_{i=1}^{n} \frac{V_{D_i} - V_c}{R} = 0, \tag{32}$$

which means that the total current flowing into C should sum up to zero. Hence the voltage of a node equals to the average of its neighbors, that is,

$$V_C = \frac{1}{n} \sum_{i=1}^{n} V_{D_i}. \tag{33}$$

According to the above equation, the Kirchhoff equations of G can be formulated as follows:

$$V_1 = 1, \tag{34}$$

$$V_2 = 0, \tag{35}$$

$$V_i = \frac{1}{k_i} \sum_{(i,j)\in E} V_j = \frac{1}{k_i} \sum_{j\in G} V_j a_{ij} \quad i = 3, 4, \ldots, n, \tag{36}$$

where k_i is the degree of node i and a_{ij} is the element of the adjacency matrix of the graph. Equation (36) can be further written as

$$V_i = \frac{1}{k_i} \sum_{j=3}^{n} V_j a_{ij} + \frac{1}{k_i} a_{i1} \quad i = 3, 4, \ldots, n. \tag{37}$$

By defining

$$V = (V_3, \ldots, V_n)^T, \tag{38}$$

$$B = \begin{pmatrix} \frac{a_{33}}{k_3} \cdots \frac{a_{3n}}{k_3} \\ \vdots \quad \vdots \\ \frac{a_{n3}}{k_n} \cdots \frac{a_{nn}}{k_n} \end{pmatrix}, \tag{39}$$

$$C = \left(\frac{a_{31}}{k_3}, \ldots, \frac{a_{n1}}{k_n} \right)^T, \tag{40}$$

the Kirchhoff equations can be presented by a matrix form

$$V = BV + C. \tag{41}$$

The corresponding solution to the above equation is

$$V = (I - B)^{-1} C. \tag{42}$$

Furthermore, the equation can be simplified as

$$LV = D, \tag{43}$$

where

$$B = \begin{pmatrix} k_3 & -a_{34} & \cdots & -a_{3n} \\ -a_{43} & k_4 & \cdots & -a_{4n} \\ \cdots & & \cdots & \\ -a_{n3} & -a_{n4} & \cdots & k_n \end{pmatrix}, \tag{44}$$

$$D = (a_{31}, \ldots, a_{n1}). \tag{45}$$

To solve the equations, instead of applying the well-known spectral partitioning method, the authors derived a much faster technique that does not compute the eigenvectors of G. Their algorithm starts with a precomputing process which evaluates the initial values for each node. This process takes O(V) time. Then iteratively, based on Eq. (33), the algorithm starts updating the voltage value of each node. The final precision is related to the number of such rounds. Therefore, the performance of the algorithm is proportional to the time it consumes, and this procedure costs $O(\sum_{i=3}^{n} k_i) = O(E)$ time.

After the above processes terminate, each node gets a voltage value. Then all nodes are sorted according to the voltage value. The authors applied a spectrum presentation, that is, each node is illustrated as a vertical line at the abscissa which is equal to the corresponding voltage value.

Then two challenges come up. One is that if the initial two poles are actually in the same community, obviously, the algorithm will fail. In [65], the authors gave a strategy based on the idea that two nodes that are far away from each other belong to different communities with high probability. So while determining the pole nodes, two nodes with as long distance between them as possible are considered firstly.

This is done by starting with an arbitrary node. Find a node farthest from it by breadth-first search, then find a node farthest from this second node. After certain steps, choose the farthest pair. The other challenge is how to identify the two communities based on the voltage spectrum. Their idea is to find the reasonable gap in the voltage spectrum as large as possible. Here, a reasonable gap is mentioned because the largest gap usually appears at the two endings of the voltage spectrum, which does not make clear sense.

Having shown the algorithm that partitions a network into two parts, it would be easy to discuss the condition that more than two communities are required. Wu and Huberman [65] explained this by an example of partitioning the US college football data into 13 conferences. Here totally 115 teams are involved. The algorithm runs in 13 iterations, in each of which, one community is found. At the beginning of the first iteration, two poles are selected. Then the algorithm for two community identifications is applied to generate the spectrum. Based on this spectrum, two communities are found from the two endings of the spectrum according to the preestimated community size. The process is repeated for 50 times to generate 100 such communities, which are called candidates. Then the rest task is to find 13 communities out of these 100 candidates. This is done by firstly specifying a node(team) that appear in the maximum number of candidate sets. Then select another several nodes based on the number of times they appear in the same candidate set with the specific node. Together, they form the first community. Then find the second specific node that appears in the maximum number of candidate sets, ignoring the nodes in the first community. After another 12 iterations, all the 115 teams will be partitioned into 13 conferences.

Recently, Raghavan et al. [55] proposed a localized community detection algorithm based on label propagation. In their algorithm, initially, each node is assigned a unique label. Then at every iteration of the algorithm, each node scans its neighbors, finds the label that most of them take, and adopts it. It is easy to regard the process as a label propagation through the network. Finally, nodes sharing the same label are considered to be of the same community. Clearly, this algorithm is distributed and takes almost linear time.

4 Influence Maximization Problem

A social network is like a huge container, in which thousands of individuals build up their relationships and interactions. Within a social network, ideas or information among its members spread like a cascade and the influence of the information has practical value. When it comes to marketing, for example, a salesman wants to promote the new products of his company, how should he make his marketing strategy such that the products are purchased by as many customers as possible? This kind of problem is called influence maximization problem in social networks. In order to estimate the influence between individuals and the probability of customers' acceptance, some promotion (discount) will be given to certain customers for free to maximize the sales of the products. Therefore, it can

be seen that the most important step is to select proper target customers, then the question is that by what standard a group of initial target customers will be chosen to get the best influence result. In other words, it is valuable to maximize "word-of-mouth" effect [8, 10, 27, 28, 44].

Due to the strong network effect, only taking into consideration the intrinsic value (the value that an individual purchases a product based on his own desire) is not enough. A more important values, say, network value should be taken into account. A network value is used to evaluate the positive influence of one customer on the others around him/her. The combination of the intrinsic value and network value comes up to be the standard that is used to target the initial customers.

In this section, we will introduce the influence maximization problem and its corresponding algorithms. Domingos and Richardson [20] firstly addressed this problem as a fundamental algorithmic problem. To solve this problem, we introduce two probabilistic models and two operational diffusion models in next section. Except the simplest linear function probabilistic model, the other three are all NP-hard problems. Greedy approximation algorithms with $(1 - 1/e)$ – approximation performance are given.

4.1 Two Probabilistic Models

4.1.1 A General Model

Suppose there are n potential customers in the system. Define them as X_i, which is a Boolean variable (Table 1). If customer i purchases the product, X_i is 1; otherwise, X_i is 0. X_i corresponds to the ith customer. $\mathbf{N}_i$ is the set of all neighbors of X_i, i.e., $\mathbf{N}_i = \{X_{i,1}, \ldots, X_{i,ni}\} \subseteq \mathbf{X} - X_i$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$. Let $X^k(X^u)$ be the customers whose value is known(unknown), and let $N_i^k = N_i \cap X^k$ and $N_i^u = N_i \cap X^u$. Assume the product is described by a set of attributes $\mathbf{Y} = \{Y_1, \ldots, Y_m\}$. Define M_i as a variable showing the marketing action of customer i. For instance, M_i could be a Boolean variable, with $M_i = 1$ if the customer is offered a given discount, and $M_i = 0$ otherwise. Let $\mathbf{M} = \{M_1, \ldots, M_n\}$. Thus, for all $X_i \notin X^k$, there is

$$
\begin{aligned}
[20]P(X_i|X^k, Y, M) &= \sum_{C(N_i^u)} P(X_i, N_i^u|X^k, Y, M) \\
&= \sum_{C(N_i^u)} P(X_i|N_i^u, X^k, Y, M)P(N_i^u|X^k, Y, M) \\
&= \sum_{C(N_i^u)} P(X_i|N_i, Y, M)P(N_i^u|X^k, Y, M). \qquad (46)
\end{aligned}
$$

Table 1 Symbols in the system model

Item	Description
$\mathbf{X}$:	$\{X_1, X_2, X_3, \ldots, X_n\}$ denotes the set of customers or the purchase activity of customers
N_i:	$\{X_{i,1}, \ldots, X_{i,ni}\}$ denotes X_i's neighbors
N_i^k:	Neighbors in N_i which are known of purchasing the product
N_i^u:	Neighbors in N_i which are unknown of purchasing the product
$\mathbf{Y}$:	$\{Y_1, \ldots, Y_m\}$ denotes the set of product attributes
$\mathbf{M}$:	$\{M_1, \ldots, M_n\}$ denotes the marketing plan
$C(N_i^u)$:	The set of all possible configurations of the unknown neighbors of X_i

According to [53], $P(N_i^u|X^k, Y, M)$ is approximated by its maximum entropy estimate if the marginal $P(X_j|X^k, Y, M)$ is given for $X_j \in N_i^u$. Then

$$[20]P(X_i|X^k, Y, M) = \sum_{C(N_i^u)} P(X_i|N_i, Y, M) \prod_{X_j \in N_i^u} P(X_j|X^k, Y, M). \tag{47}$$

Because in Eq. (47), $P(X_i|X^k, Y, M)$ is expressed as the function of themselves, it can be applied iteratively with an initial value. One of the initial value is $P(X_i|\mathbf{Y,M})$, which is the network-less probabilities. Note that the number of terms in Eq. (47) is exponential in the size of N_i^u. If this number of the unknown neighbors of X_i is small, this should not be a problem, otherwise, an approximate solution is necessary. Gibbs sampling [25] is one of the standard methods for this problem, and another one is based on an efficient k-shortest-path algorithm presented by Chakrabarti et al. [12].

If N_i and $\mathbf{Y}$ are given, X_i should be independent of the marketing actions for other customers except its neighbors. Then a naive Bayesian model is used to present X_i as a function of $N_i, Y_1, \ldots, Y_m$, and M_i:

$$\begin{aligned}
[20]P(X_i|N_i, Y, M) &= P(X_i|N_i, Y, M_i) \\
&= \frac{P(X_i)P(N_i, Y, M_i|X_i)}{P(N_i, Y, M_i)} \\
&= \frac{P(X_i)P(N_i|X_i)P(M_i|X_i)}{P(N_i, Y, M)} \prod_{k=1}^{m} P(Y_k|X_i) \\
&= \frac{P(X_i|N_i)P(M_i|X_i)}{P(Y, M_i|N_i)} \prod_{k=1}^{m} P(Y_k|X_i).
\end{aligned} \tag{48}$$

It is known that

$P(Y, M_i|N_i) = P(Y, M_i|X_i = 1)P(X_i = 1|N_i) + P(Y, M_i|X_i = 0)P(X_i = 0|N_i)$.

The corresponding network-less probabilities are $P(X_i|Y, M) = P(X_i)$ $P(M_i|X_i) \prod_{k=1}^{m} \frac{P(Y_k|X_i)}{P(Y,M_i)}$ [20].

To compute Eq. (47), it only needs to know the following probabilities if Eq. (48) is given. These probabilities are $P(X_i|N_i)$, $P(X_i)$, $P(M_i|X_i)$, and $P(Y_k|X_i)$ for all k. All of these are easily obtained except probability $P(X_i|N_i)$. The form of $P(X_i|N_i)$ depends on how the system is built. In other words, due to the mechanism by which customers are influenced by others, this probability will vary from application to application.

4.1.2 Approximate Algorithms Based on the Probability Model

For the sake of simplicity, it is assumed that **M** is a Boolean vector. When marketing a product, suppose the cost is a constant. Use c to represent this number. Let r_0 and r_1 be the revenue from selling the product to the customer whether marketing action is performed. Let $f_i^1(\mathbf{M})$ show the result of setting M_i to 1 and leaving the rest of **M** unchanged and $f_i^0(\mathbf{M})$, the result of setting M_i to 0 and remaining the rest of the part in **M** the same as before. Thus, customer i′s expected lift in profit in isolation is

$$[20]\mathrm{ELP}_i(X^k, Y, M) = r_1 P(X_i = 1|X^k, Y, f_i^1(\mathbf{M})) - r_0 P(X_i = 1|X^k, Y, f_i^0(\mathbf{M})) - c. \quad (49)$$

Let M_0 be the null vector in which all the members are zero. The global lift in profit that after a particular choice **M** of customers to market is as

$$[20]\mathrm{ELP}(X^k, Y, M) = \sum_{i=1}^{n} r_i P(X_i = 1|X^k, Y, \mathbf{M}) - r_0 \sum_{i=1}^{n} P(X_i = 1|X^k, Y, \mathbf{M}) - |\mathbf{M}|c, \quad (50)$$

where $r_i = r_1$ if $M_i = 1, r_i = r_0$ if $M_i = 0$, and the number of 1s in **M** is $|\mathbf{M}|$. The goal is to find the assignment of values to **M** that maximizes ELP. Trying to find the optimal **M** is intractable. It needs to compute every possible combinations of the assignments to its members. There are three approximate methods that can finish this job. They are single-pass method, greedy search method, and hill-climbing method. Each method is much more expensive than the previous one but better solution. The algorithms are as follows:

Algorithm 4 Single pass [20]

1: For each i,
2: **if** $\mathrm{ELP}(X^k, Y, f_i^1(M_0)) > 0$
3: $M_i = 1$;
4: **else**
5: $M_i = 0$;
6: **end if**
7: **end for**

Algorithm 5 Greedy search [20]

1: set $M = M_0$;
2: Loop through the M_i's, until no change to M_i's;
3: **if** $ELP(X^k, Y, f_i^l(M_0)) > ELP(X^k, Y, M)$ **then**
4: $M_i = 1$;
5: **else**
6: $M_i = 0$;
7: **end if**

Algorithm 6 Hill-climbing search [20]

1: Set $\mathbf{M} = \mathbf{M}_0$;
2: Set $M_{i_1} = 1$, where $i_1 = \text{argmax}_i ELP(\mathbf{X}^k, \mathbf{Y}, f_i^l(\mathbf{M}))$;
3: Set $M_{i_2} = 1$, where $i_2 = \text{argmax}_i ELP(\mathbf{X}^k, \mathbf{Y}, f_i^l(f_{i_1}^l(\mathbf{M})))$;
4: Repeat until there is no i for which setting $M_i = 1$ increases ELP.

4.1.3 A Polynomial-Time Solvable Model

In last two subsections, we introduce a general probabilistic model for influence maximization problem, where the optimization problem cannot even be approximated to within a nontrivial factor. In this subsection, we introduce a similar but simpler model. Suppose the symbol is the same as listed in the last subsection, then, for all X_i, there is

$$[56]P(X_i|X - X_i, Y, M) = P(X_i|N_i, Y, M) = \beta_i P_0(X_i|Y, M_i) + (1 - \beta_i)P_N(X_i|N_i, Y, M). \quad (51)$$

$P_0(X_i|Y, M_i)$ is X_i's internal probability of buying the new product. $P_N(X_i|N_i, Y, M)$ is the effect that X_i's neighbors directly put on her. β_i is between 0 and 1 that measures how neighbor-reliant X_i is. In a general probabilistic model, these interactions between neighbors and X_i are modeled by a nonlinear function. In this model, a simpler linear model is used to approximate this effect instead, the probability is as follows:

$$[56]P_N(X_i = 1|N_i, Y, M) = \sum_{X_j \in N_i} w_{ij} X_j. \quad (52)$$

w_{ij} represents the extent that customer i is affected by her/his neighborj, with $w_{ij} \geq 0$ and $\sum_{X_j \in N_i} w_{ij} = 1$. Linear models often perform well, especially when data is sparse [19], and they provide significant advantage for computation. Thus, by combining the last two equations, there is

$$[56]P(X_i|N_i, Y, M) = \beta_i P_0(X_i|Y, M_i) + (1 - \beta_i) \sum_{X_j \in N_i} w_{ij} X_j. \quad (53)$$

Because the optimal marketing strategy for a product has not yet been introduced to the market, the state of the neighbors will not be known. Thus, a formula for computing $P(X_i = 1|Y, M)$ is listed as follows:

$$\begin{aligned}
[56]P(X_i = 1|Y, M) &= \sum_{\tilde{N}\in C(N_i)} P(X_i = 1|\tilde{N}, Y, M)P(\tilde{N}|Y, M) \\
&= \sum_{\tilde{N}\in C(N_i)} \beta_i P_0(X_i = 1|Y, M_i)P(\tilde{N}|Y, M) \\
&\quad + \sum_{\tilde{N}\in C(N_i)} (1-\beta_i) \sum_{X_j\in N_i} w_{ij}\tilde{N}_j P(\tilde{N}|Y, M) \\
&= \beta_i P_0(X_i = 1|Y, M_i) \\
&\quad +(1-\beta_i) \sum_{X_j\in N_i} \sum_{(\tilde{N}\in C(N_i))\text{with}N_j=1} w_{ij}P(\tilde{N}|Y, M). \qquad (54)
\end{aligned}$$

Denote the set of all possible configurations of X_i's neighbors as $C(N_i)$, and $\tilde{N}$ is one of the state assignments. $\tilde{N}_j$ is the value of X_j specified by $\tilde{N}$. Since the inner summation is over all possible values of $\tilde{N}$ whenever $\tilde{N}_j = 1$, it is equivalent to $w_{ij}P(X_j = 1|Y, M)$, therefore,

$$\begin{aligned}
[56]P(X_i = 1|Y, M) &= \beta_i P_0(X_i = 1|Y, M_i) \\
&\quad +(1-\beta_i) \sum_{X_j\in N_i} w_{ij}P(X_j = 1|Y, m). \qquad (55)
\end{aligned}$$

The equation above expresses the probabilities $P(X_i = 1|Y, M)$ as a function of themselves. It can be applied iteratively to find them, starting from a suitable initial assignment. A natural choice for initialization is to use the internal probabilities $P_0(X_i = 1|Y, M)$.

In this simple probabilistic model, both the propagation of influence and the effect of the initial targeting are linear. Thus, the influence can be maximized by solving a system of linear equations.

4.2 Two Operational Models

In the last section, we introduce two descriptive models which give a joint distribution over all vertex behaviors in a global view. In this section, operational models from mathematical sociology [29, 57] and interacting particle systems [21, 41], which show the dynamics of adoption step-by-step, are considered. Two basic diffusion models, linear threshold model and independent cascade model,

are presented. The influence maximization problem on these two models and their extension models are NP-complete, but can be approximated well. It is shown that the approximate algorithms for maximizing the spread of influence on these models can be developed in a general framework based on submodular functions. In the next subsections, the definition of submodular function and the description of these two models are given, and based on them, the influence function is shown to be submodular and the influence maximization problem is NP-hard on them.

4.2.1 Submodular Function

Consider a function $f(\cdot)$, which is submodular if it satisfies a natural "diminishing returns" property. This property states that the marginal gain from adding an element to a set A is at least as high as the marginal gain from adding the same element to a superset B. Formally, a submodular function satisfies

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B) \tag{56}$$

for all elements v and all pairs of sets $A \subseteq B$.

Suppose a function f has the following four attributes:

1. Submodular
2. Takes only non-negative values
3. Monotone
4. Not decreasing: $f(A \cup \{v\}) \geq f(A)$ for all elements v and sets A

The influence maximization problem is to find a k-element set A such that $f(A)$ is maximized. This problem is NP-hard, but Nemhauser, Wolsey, and Fisher [17, 47] showed that the greedy hill-climbing algorithm approximates the optimum to within a factor of $(1 - 1/e)$ (where e is the base of the natural logarithm): start with the empty set $A(0)$, and repeatedly add an element that gives the maximum marginal gain.

Theorem 1 ([17, 34, 47]) *For a nonnegative, monotone submodular function* f, *let* S *be a set of size* k *obtained by selecting elements one at a time, each time choosing an element that provides the largest marginal increase in the function value. Let* $S*$ *be a set that maximizes the value of* f *over all* k*-element sets. Then* $f(S) \geq (1 - 1/e)f(S*)$*; in other words, S provides a* $(1 - 1/e)$*-approximation.*

4.2.2 Independent Cascade Model

Regard A_0 as the initial node set, and the process proceeds in discrete steps according to the following randomized rule. When one node u wants to activate its neighbor v, the process can succeed only with a probability $p_{u,v}$. If u succeeds at step t, then v will become active in step $t + 1$, but whether or not u succeeds, it cannot make any further attempts to activate v in subsequent steps. When there are no more nodes that can be activated, the process terminates.

Theorem 2 ([34]) *For an arbitrary instance of the independent cascade model, the resulting influence function* $\sigma(\cdot)$ *is submodular.*

At first sight, it is hard to compute $\sigma(A \cup \{v\}) - \sigma(A)$ for arbitrary set A and node v. The increase of the diffusion effect is very difficult to analyze directly because of the hardness of computing the size of $\sigma(A)$. Fortunately, an equivalent view of the diffusion process can be formulated to an order-independent outcome. Since between each pair of nodes, for example, node v and node w, there is a probability $p_{v,w}$ between v and w, this can be regarded as a result of flipping a coin. From the aspect of process, no matter what is the sequence of the node being activated or whether the node is activated, the result is the same. Based on this fact, it can be assumed that for each pair of neighbors (v, w), a coin of bias $p_{v,w}$ is flipped at the beginning of the influence diffusion. Store the result for the later check whether w is activated with v already active. If the activation is successful, the edge between the two nodes becomes live, otherwise, it is blocked. So, if the initial active set A and the outcome of the coin flips are fixed, the set of all the activated nodes will be determined.

Let X be one of the outcome and A the initial node set. Denote $\sigma_X(A)$ as the total number of nodes activated when the outcome is X and the initial target set is A. Let $R(u, X)$ be the number of nodes that can be reached on live-edge paths from u, so $\bigcup_{u \in A} R(u, X) = \sigma_X(A)$.

Let A and B be two sets of nodes and $A \subseteq B$. Consider the quantity $\sigma_X(A \cup \{u\}) - \sigma_X(A)$. This is the number of elements in $R(u, X)$ that are not already in the union $\bigcup_{u \in A} R(u, X)$, it is at least as large as the number of elements in $R(u, X)$ that are not in the (larger) union $\bigcup_{u \in B} R(u, X)$. It follows that $\sigma_X(A \cup \{u\}) - \sigma_X(A) \geq \sigma_X(B \cup \{u\}) - \sigma_X(B)$, which is the definition of submodularity. Finally, there is

$$[34]\sigma(A) = \sum_{\text{outcomes } X} \text{Prob}[X]\sigma_X(A). \tag{57}$$

since the expected number of nodes activated is just the weighted average over all outcomes. And a nonnegative linear combination of submodular functions is also submodular, hence $\sigma(\cdot)$ is submodular.

Theorem 3 ([34]) *The influence maximization problem is NP-hard for the independent cascade model.*

The influence maximization problem on independent cascade model can be reduced from set cover problem. Consider an instance of the NP-complete set cover problem, define a collection of subsets $T_1, T_2, \ldots, T_m$ and a ground set $Q = \{q_1, q_2, \ldots, q_n\}$. The problem aims to find out whether there are k subsets whose union is equal to Q. It is assumed that $k < n < m$. This also can be viewed as a special case of the influence maximization problem.

Given an arbitrary instance of the set cover problem, define a corresponding directed bipartite graph with $n + m$ nodes: there is a node i corresponding to each set T_i, a node j corresponding to each element q_j, and a directed edge (i, j) with activation probability $p_{i,j} = 1$ whenever $q_j \in T_i$. The set cover problem is

equivalent to deciding whether there is a set A of k nodes in this graph with $\sigma(A) \geq n + k$. Note that for the instance we have defined, activation is a deterministic process, since all probabilities are 0 or 1. Initially activating the k nodes corresponding to sets in a set cover solution results in activating all n nodes corresponding to the ground set Q, and if any set Q of k nodes has $\sigma(A) \geq n + k$, then the set cover problem must be solvable.

4.2.3 Linear Threshold Model

In the linear threshold model, a node v is influenced by each neighbor w according to a weight $b_{v,w}$ and $\sum_{w \text{ neighbor of } v} b_{v,w} \leq 1$. The dynamic process proceeds as follows: each node v chooses a threshold θ_v uniformly at random from the interval [0,1], this shows that, in order to activate node v, all the weight between v and its active neighbors must be over the threshold. If there is an initial target set A_0 and a bunch of random thresholds between each pair of nodes, the diffusion process proceeds deterministically in discrete steps: at step t, all the active nodes are still active and inactive nodes whose neighbors' overall weight larger than their thresholds become active.

$$[34] \quad \sum_{w \text{ active neighbor of } v} b_{v,w} \geq \theta_v. \tag{58}$$

Theorem 4 ([34]) *For an arbitrary instance of the linear threshold model, the resulting influence function $\sigma(\cdot)$ is submodular.*

In the last subsection, the independent cascade model was transferred into the live-edge graph and the influence function on it proved to be submodular.

Similarly, in the linear threshold model, given a graph G, assume the active node set is A_t at step t, for $t = 0, 1, 2, \ldots$, and A_0 is the initial set. If node v is inactive in step t, then the probability of v to be activated by its neighbors in step $t + 1$ is $\frac{\sum_{u \in A_t \setminus A_{t-1}} b_{u,v}}{1 - \sum_{u \in A_{t-1}} b_{u,v}}$.

In the live-edge model, it also starts with an initial set A_0. At step t, if node v′s edge is among the live-edge set, then v is known, otherwise, v is unknown. Then, at step $t + 1$, the chance of v being known is equal to the chance that its live edge comes from $A_t \setminus A_t - 1$, given that its live edge has not come from any of the earlier sets. The probability is $\frac{\sum_{u \in A_t \setminus A_{t-1}} b_{u,v}}{1 - \sum_{u \in A_{t-1}} b_{u,v}}$, same as above. Thus, the two distribution processes are the same.

Once the equivalence between live-edge model and linear threshold model is proved to be true, it can be used in the last subsection to show that the influence function in linear threshold model is also submodular.

Theorem 5 ([34]) *The influence maximization problem is NP-hard for the linear threshold model.*

Consider an instance of the NP-complete vertex cover problem: given an undirected n-node graph $G = (V, E)$ and an integer k, the problem is to find whether there is a set S with k nodes in G such that every edge has at least one endpoint in S.

It is shown that this is a special case of the influence maximization problem. Given an instance of the vertex cover problem involving a graph G, define a corresponding instance of the influence maximization problem by directing all edges of G in both directions. If there is a vertex cover S of size k in G, then one can deterministically make $\sigma(A) = n$ by targeting the nodes in the set $A = S$, conversely, this is the only way to get a set A with $\sigma(A) = n$.

4.2.4 Greedy Algorithms for Operational Models

In this section, two approximate algorithms are given for the operational model. One is the basic greedy heuristic algorithm [47]; the other is CELF (cost-effective lazy forward selection) algorithm [38].

Basic Greedy Heuristic Algorithm [47]

The natural way to find the solution of a submodular function model is to start with the empty set and add new element which has the largest influence increase. Repeat the same selection process until the stop condition is satisfied. For influence maximization problem, define $d_v(A) = f(A \cup \{v\}) - f(A)$. Concrete description of the algorithm is as follows:

Algorithm 7 Greedy heuristic for submodular function model [47]

$A_0 = \text{NULL}$
$N_0 = N$ and $t = 1$.
Iteration t
Select $i(t) \in N_{t-1}$
Loop $d_{i(t)}(A_{t-1}) = \max_i \in N_{t-1}(A_{t-1})$
with connection settled arbitrarily. Set $d_{t-1} = d_{i(t)}(A_{t-1})$
$A_t = A_{t-1} \cup i(t)$ and $N_t = N_{t-1} - i(t)$.
if $t < K$ **then**
    $t = t + 1$.
end if
repeat until $t = K$. Here K is the number of the total iteration.

Let f_g be the solution of the greedy heuristic. There is

$$[47] f_g = f_0 + f_1 + \ldots + f_K \tag{59}$$

CELF Algorithm for Submodular Function Model [38]

The basic greedy heuristic algorithm has two drawbacks. One is time-consuming for the reason that at each iteration, it needs to reevaluate each node's marginal gain; the other is that greedy heuristic algorithm only applies to uni-cost problem (the cost of choosing each node is the same). In the case of different cost problem, this algorithm works badly. It will choose the node with highest marginal gain every time without considering the cost. For example, there are two nodes, say u and v. Choosing u will improve the influence at I and cost C, while selecting v will lead to influence increase at $I + \xi$ but cost $2 * C$. Here, I and C are two constant numbers. When $\xi \to 0$, one should definitely choose u rather than v with considering the cost.

A new greedy heuristic algorithm called benefit-cost algorithm was presented in [38]. Instead of only taking the benefit into account that argmaxf($A_{k-1} \cup \{v\}$)− f(A_{k-1}), the new algorithm computes the ratio of benefit and cost, which is

$$\frac{\text{argmaxf}(A_{k-1} \cup \{v\}) - f(A_{k-1})}{c(s)}, \tag{60}$$

where c(s) is the cost to choose node v.

But in fact, this new algorithm is much worse than the original one. For example, there are two nodes n1 and n2 with $c(n1) = \xi$ and c(n2) = B, the total budget is B. $R(n1) = 2 * \xi$ and R(n2) = B. $R((\{n1\}) - R(\emptyset))/c(n1) = 2$ and $R((\{n2\}) - R(\emptyset))/c(n2) = 1$. Here, R(·) is the benefit function, and c(·) is the cost function. The benefit-cost greedy algorithm would pick n1. After selecting n1, n2 cannot be satisfied anymore, and the total reward would be ξ. However, the optimal solution would pick n2, achieving total penalty reduction of B. As ξ approaches 0, the performance of the benefit-cost greedy algorithm becomes arbitrarily bad.

When combining the two algorithms into one algorithm, here comes the CELF algorithm [38]. It has two steps:

- Set (solution) R(A): use benefit-cost greedy algorithm
  Set (solution) R(B): use uni-cost greedy algorithm
- argmax(R(A), R(B))

It is shown that although both of the two solutions can be arbitrarily bad, there is at least one of them which is not too far away from optimum, and hence CELF provides a constant factor approximation.

Theorem 6 ([38]) *Let* R *be a nondecreasing submodular function with* $R(\emptyset) = 0$. *Then* $\max\{R(A), R(B)\} \geq 1/2(1 - 1/e)$OPT, *where* OPT *is the optimal value.*

In [35], a special case of the budgeted MAX-COVER problem is proved. For arbitrary nondecreasing submodular functions, the proof is shown in [30]. This theorem stated that the best solution of the basic greedy and benefit-cost greedy (which is returned by CELF) is at most a constant factor within $1/2(1 - 1/e)$ of the optimal solution. But CELF's running time is only O(T|V|) compared to the basic greedy algorithm which is $(T|V|^4)$. Here, |V| is the number of nodes and T is the budget. Sviridenko [60] showed that an approximation guarantee of $(1 - 1/e)$ can be achieved even in a nonconstant environment.

5 Compact Routing Scheme for Power-Law Graphs in Social Network

In the field of complex networks, a number of different characteristics [59] have been explored for social networks. These characteristics contain: (1) small-world property: although not all nodes are neighbors of one another, most nodes can be reached from each other through a small number of hops or steps; (2) heavy-tailed

degree distributions: degree distribution of this graph approximates a power-law distribution; (3) community structure: groups of nodes in a network are more densely connected internally in a community than with the rest in the network. Based on property (2), this section mainly focuses on compact routing scheme by using the theory of unweighted random power-law graphs with fixed expected degree sequence. The method here we introduce is the first theoretical bound coupled to the parameter of the power-law graph model for a compact routing scheme.

5.1 Compact Routing Scheme for Power-Law Graphs

For a network with n nodes, a routing scheme is only allowed to have routing tables with sizes sublinear in n and message header sizes polylogarithmic in n. In general, there are two classes of compact routing schemes: the first is the *labeled* scheme, which is allowed to add labels to node addresses to encode useful information for the sake of routing, where each label has length at most polylogarithmic in n. On the other hand, the second scheme, *name-independent* scheme, does not allow the renaming of node addresses, instead they must function with all possible addresses. Both of these two kinds of compact routing schemes have been studied widely recently. Therefore, this phenomenon demonstrates the relationship between compact routing scheme and power-law graphs, which will play an important role in social network routing problem.

Power-law graphs form an essential family of networks appearing in various real-world scenarios such as some collaboration networks and the famous World Wide Web, especially in social networks, which is a rapidly expanding network in social connectivity. In a power-law graph, for some constant τ, the number of nodes with degree x is proportional to $x^{-\tau}$. In general, the range of the power-law exponent τ for many real-world networks is between 2 and 3. However, power-law graphs do not seem to belong to any of the previous well-studied network families such as trees, planar graphs, or low doubling-dimension graphs, which means that the importance of the power-law graphs should be noticed, and this property is worth paying attention.

Recently, there are some other experimental studies which concentrate on compact routing in power-law graphs and Internet-like graphs. However, they all have some drawbacks. For example, although Krioukov et al. [37] evaluated the universal routing scheme of Thorup and Zwick (TZ) [61] on random power-law graphs and they also provided experimental evidence of much better performance than the theoretical worst-case bound, however, they did not provide a theoretical bound of the TZ scheme on power-law graphs such as stretch and table size. Some other papers such as Enahescu et al. [22] and Brady and Cowen [9], even though they contributed to compact routing schemes for power-law graphs, both of them did not provide rigorous analysis. Therefore, bridging the gap in the study of compact routing schemes for power-law graphs becomes a main optimization problem.

5.1.1 Related Work

Due to the comparison with Thorup and Zwick's routing schemes and other random power-law graph models, it is necessary to provide some related work. Thorup and Zwick [61] mainly contributed two different routing schemes. The first scheme is a stretch-3 scheme with an $O(n^{1/2}\log^{3/2} n)$-bit routing table per node and $O(\log n)$-bit labels and headers, which is based on Cowen's earlier scheme in [18]. They both used a small subset A of nodes, called *landmarks*, and they used the landmarks to route messages. In a graph G = (V,E), for every node u, define its cluster $C(u) = \{v \in V: d(v,u) < d(v,A)\}$, where $d(v,u)$ and $d(v,A)$ denote the graph distance from v to u and A, separately. Let l(u) denote the landmark in A, which is the closest one to node u (ties are arbitrarily resolved). The routing table of node u stores the port identifiers to route messages to all nodes in A and C(u). If a destination v is not in $A \cup C(u)$, u routes through l(v), which guarantees a stretch bound of 3 because of the definition of the cluster C(u). A resampling method was used by Thorup and Zwick to achieve $|A \cup C(u)| = O(n^{1/2}\log^{1/2} n)$ for every node u.

The second scheme of Thorup and Zwick [61] is on the basis of their approximate distance oracle in [62]. For any $k \geq 2$, they designed a compact routing scheme with the attributes of $\tilde{O}(n^{1/k})$-bit tables, $O(k\log^2 n/\log\log n)$-bit addresses, and $O(\log^2 n/\log\log n)$-bit headers (both the bounds on addresses and headers are for fixed-port schemes). The stretch $2k-1$ with a stretch $4k-5$ handshake is achieved by this scheme. In order to reduce the stretch to 3, a *handshake* is needed. The scheme used and appeared in this article is similar to the second scheme. However, there are two main differences even both the two schemes use balls and landmarks to route messages. One difference is that high-degree nodes are chosen as landmarks instead of the randomly selected nodes. By using this strategy, $|A \cup B(u)| = O(n^{\gamma})$ with $\gamma = \frac{\tau-2}{2\tau-3} + \varepsilon$ and $\varepsilon > 0$ is achieved. Another difference is that the improved scheme will directly encode the shortest path from l(v) to v in v's address. It is short within the probability of $1 - o(1)$ because of the distance properties in random power-law graphs. The result of the modified scheme shows smaller routing table size, and the address and header size of $O(\log n \log\log n)$ is better than the second scheme and near the result of the first scheme. However, the improvement of this scheme here is only tailored to unweighted power-law graphs.

5.1.2 Preliminaries

The method adopted in this section is the random graph model for fixed expected degree sequence as defined in [6, 14, 15, 42]. The expression fixed degree random graph (**FDRG**) is used to refer to the original random graph distribution. Consequently, from the previous related paper, a definition for the random power-law graph distribution **RPLG**(n,τ) is obtained.

Definition 1 ([13]) For a constant $\tau \in \{(2,3)\}$, the random power-law graph distribution **RPLG**(n,τ) is defined as follows. First, we let the sequence of generating

parameters $\mathbf{w} = \{w_1, w_2 \ldots, w_n\}$ obey a power law, which is $w_k = (\frac{n}{k})^{1/(\tau-1)}$ for $k \in \{1, 2, \ldots n\}$. Then we insert the edge between v_i and v_j into the random graph within the probability of $\min\{w_i w_j æ, 1\}$, where $æ = \frac{1}{\sum_k w_k}$.

In the **FDRG** model, the value w_i corresponds to the expected degree of vertex v_i, and $\mathbf{w}$ is referred to as the *expected degree sequence*. The graph is sampled here because of the generating parameter values w_i. Let D_i be the random variable denoting the degree of node v_i. In this adapted model, it can be found that the expected degree $E[D_i]$ of node v_i is smaller than or equal to the generated parameter w_i.

It is required that $n = \|V(G)\|$ is sufficiently large, satisfying the following:

$$n^{\frac{\varepsilon(2\tau-3)}{(\tau-1)}} \geq \frac{2(\tau-1)}{\tau-2} \ln n. \tag{61}$$

The results do not have any other implicit dependencies on ε. Furthermore, the *core* of a graph consists of nodes having large degrees. Let $\gamma = \frac{\tau-2}{2\tau-3} + \varepsilon$ for some $\varepsilon > 0$ and $\gamma' = \frac{1-\gamma}{\tau-1}$.

Definition 2 ([13]) For a power-law graph which has the degree sequence $\mathbf{w}$ and a graph G with n nodes, we can define the core with degree threshold $n^{\gamma'}$, $\gamma' \in (0, 1)$, this kind of equations form as follows:

$$\mathsf{core}_{\gamma'}(w) := \{v_i : w_i > n^{\gamma'}\}, \tag{62}$$

$$\mathsf{core}_{\gamma'}(G) := \{v_i : \deg_G(v_i) > n^{\gamma'}/4\}, \tag{63}$$

where $\deg_G(v_i)$ is the degree of v_i in G (the subscript G is omitted when the graph is clear from the context).

The meaning of $n^{\gamma'}$-Core is as what $\mathsf{core}_{\gamma'}(w)$ means in [42].

For each vertex u of graph G, the ball relative to the core can be defined as

$$B_G(u) := \{v \in V(G) : d(u, v) < \min_{v' \in \mathsf{core}_{\gamma'}(G)} d(u, v')\}. \tag{64}$$

5.2 The Adapted Compact Routing Scheme

The scheme adapted here is a fixed-port scheme, and it works with arbitrary permutations of port number assignments. Let the unweighted graph $G = (V, E)$ model the network. Each node v in the network has a unique $\lceil \log_2 n \rceil$-bit static name. Whenever v is written in a routing table, a message header, or a node address, represents $\lceil \log_2 n \rceil$-bit static name representation. Each node v has deg(v) ports connecting it with its neighbors. Number these ports by $0, 1, \ldots, \deg(v) - 1$, and

thus each port number of v requires $\lceil \log_2 \deg(v) \rceil$ bits. For every packet, the routing scheme needs to determine clearly which port the packet is to be forwarded to.

5.2.1 Routing Scheme

The routing algorithm is based on [18, 61]. A set of landmarks $A \subseteq V$ is used, but it is different from [18,61]. $\text{Core}_{\gamma'}(G)$ is used as landmarks instead of nodes sampled at random. For each node u in G, let l(u) denote u′s closest landmark, which means $l(u) := \arg\min_{v' \in \text{core}_{\gamma'}(G)} d(u, v)$. The local targets of node u are defined as the elements of its *ball* $B_G(u)$. Each node u stores the ports used to route messages along the shortest paths to all landmarks and its local targets. If the target v is neither a landmark nor a local target of u, the message is routed to v's closest landmark l(v) and from there to the target v.

The scheme used is a labeled scheme. For a node u knowing l(v) of any target v, the address of node v contains an encoding of l(v). Moreover, for a node w on the shortest path from l(v) to v($w \neq l(v)$ and $w \neq v$), v may not be in $B_G(w)$ and thus w may not know the port to route messages to v. To resolve this issue, it is necessary to extend the address of v by encoding the shortest path from the landmark l(v) to v.

Let $s = u_0, u_1, \ldots, u_m = t$ denote the sequence of nodes on a shortest path from s to t. Let SP(s, t) be the encoding of this shortest path as an array with m entries, where SP(s, t) can be encoded with $\sum_{i=0}^{m-1} \log_2 \lceil \deg(u_i) \rceil$ bits. The precise definitions of addresses, message headers, and local routing tables are provided as follows:

Definition 3 ([13]) The address of node $u \in V$ is $\mathsf{addr}(u) := (u, l(u), SP(l(u), u))$.

The header of a message from node s to node t is in one of the following formats:

1. header = (route,s,t), where route = `local`,
2. header = (route,s,addr), where route = `toLandmark` and addr = `addr(t)`,
3. header = (route,s,t,pos,SP), where route $\in$ `fromLandmark,direct`, pos is a nonnegative integer that may be modified along the route, and SP = SP(s,t) if route = `direct` or SP = SP(l(t),t) if route = `fromLandmark`,
4. header = (route,s,t,SP), where route = `handshake` and SP is a reversed shortest path from t to s to be encoded along the path from s to t.

The local routing table for each node u forms the information about routes to the core and the information about local routes:

$$\texttt{tbl}(u) := \{(v, \text{port}_u(v)) : v \in \text{core}_{\gamma'}(G)\} \cup \{(v, \text{port}_u(v)) : v \in B_G(u)\}, \quad (65)$$

where $\text{port}_u(v)$ is the local port of u to route messages toward node v along some shortest path from u to v.

5.2.2 Routing Algorithm

Algorithm 8 describes the routing procedure. It includes pseudocode for the source node s to determine the method of sending a message to target t which is based on whether t is local and whether a shortest path to t is known or not. It also describes

the details that an intermediate node u determines whether to forward the message using its local routing table, or forward the message using the shortest path encoded in the header, or to switch the routing direction from the landmark l(t) to the target t. The correctness of the algorithm is based on the simple observation $t \in B_G(w) \cup \text{core}_{\gamma'}(G)$.

Moreover, an additional handshake protocol described here handles the special case when t does not belong to $B_G(s)$ but s belongs to $B_G(t)$. In this situation, the basic LANDMARKBALLROUTING scheme only achieves worst case stretch 5 not stretch 3. However, t knows the reverse path from t to s. Since the graph is undirected, t can send a special handshake message back to s, and each node along the path encodes the reverse port number such that, in the end, s knows the shortest path from s to t. For simplicity of expression, the reasonable assumption in [2] that node u knows port q on which the message is received is used. If this assumption does not hold, this handshake protocol can be adapted accordingly as follows. In the routing table of node u, for all $v \in B_G(u) \cup \text{core}_{\gamma'}(G)$, store a $\text{rev-port}_u(v) = \text{port}_w(u)$, where w is the first node on the path from u to v. Then, when forwarding the handshake message from t to s, every node u on the path (including t) pretends $\text{rev-port}_u(s)$ to the SP in the header. This adds the routing table size by at most $\lceil \log_2 n \rceil$ bits per entry. In the description of this algorithm, the case of $s \in \text{core}_{\gamma'}(G)$ is also included in the case that stretch can be improved from 3 to 1.

Both the performance of Algorithm 1 and the statement described above are evaluated, we will induce the following theorem from the two previous algorithms in [13].

Theorem 7 ([13]) *LANDMARKBALLROUTING along with the handshake protocol is a routing scheme with the following properties: (1) the worst-case stretch is 5 and it is without handshaking, (2) the worst-case stretch is 3 after handshaking, and (3) every routing decision takes constant time. In addition, for random graphs sampled from* ***RPLG****(n, τ), the following properties hold: (4) the expected maximum table size is $O(n^{\gamma} \log n)$ bits; this bound also holds with probability at least $1 - 1/n$; (5) address length and message header size are $O(\log n \log \log n)$ bits with probability 1-o(1); and (6) addresses and routing tables can be generated efficiently in expected time $O(n^{1+\gamma} \log n)$, and this bound also holds with probability at least $1 - 1/n$.*

5.3 Analysis of Properties in Performance

This subsection analyzes the performance of LANDMARKBALLROUTING for random power-law graphs from different parts.

5.3.1 Analysis of Properties

In stretch part, LANDMARKBALLROUTING has worst-case stretch 5. After handshaking with stretch 5, LANDMARKBALLROUTING has worst-case stretch 3,

Algorithm 8 LANDMARKBALLROUTING on node u, with source s, target $t \neq s$, and header [61]

First, **if** $u = s$ **then**
if $t \in B_G(s)$ **then**
send packet with header = (local,s,t) using $port_s(t)$ stored in tbl(s)
else if u knows SP(s, t) **then**
send packet and the header = (direct,s,t,0,SP(s, t)) by using port SP(s, t)[0]
else
send packet and the header = (toLandmark,s,addr(t)) using $port_s(l(t))$ stored in tbl(s)
end if
exit
end if
Under the condition $u \neq s$, we decide **if** u = header.t **then**
exit as the packet arrived.
end if
if header.route = toLandmark **then**
if u = header.*addr*.l(t) **then**
header.*route* ← fromLandmark; header.*pos* ← 0; header.SP ← header.*addr*.SP(l(t), t);
forward packet with the new header using port header.SP[0]
else
forward the packet to $port_u$(header.*addr.l(t)*) stored in tbl(u)
end if
else if header.*route* ∈ { fromLandmark,direct} **then**
We should do the header.*pos* + 1, then we assign it to header.pos
after doing the assignment, we forward the packet by using port header.SP[header.pos]
else if header.*route* = local **then**
forward the packet using $port_u$(header.t) stored in tbl(u)
end if

which is proved by the triangle inequality as in [18, 61]. In random power-law graphs, some properties of the adapted random power-law graph model should be addressed. Let G be a random graph sampled from $\mathbf{RPLG}(n, \tau)$, the *volume* Vol(G) satisfies

$$n < \mathrm{Vol}(G) \leq \frac{\tau - 1}{\tau - 2} n. \tag{66}$$

In random power-law graphs and their cores and ball parts, the concentration results for the actual degree of a vertex and for the volume of a set of vertices in the adapted $\mathbf{RPLG}(n, \tau)$ model will be shown, and the corresponding results in the original **FDRG** model are also restated. For a random graph sampled from **FDRG(w)**, the random variable D_i measuring the degree of vertex v_i is around its expectation w_i as follows:

$$\Pr[D_i > w_i - c\sqrt{w_i}] \geq 1 - e^{-c^2/2} \tag{67}$$

$$\Pr[D_i < w_i + c\sqrt{w_i}] \geq 1 - e^{-\frac{c^2}{2(1+c/(3\sqrt{w_i}))}}. \tag{68}$$

For a random graph sampled from $\mathbf{FDRG}(w)$, a subset of vertices S and all $0 < c \leq \sqrt{\mathrm{Vol}(S)}$, there is

$$\Pr[|\mathrm{vol}(S) - \mathrm{Vol}(S)| < c\sqrt{\mathrm{Vol}(S)}] \geq 1 - 2e^{-c^2/6}. \tag{69}$$

Let $n \geq 4^{\frac{\tau-1}{(\tau-2)^2}}$. For a random graph sampled from $\mathbf{RPLG}(n, \tau)$, if $w_i \geq 32 \ln n$, for vertex v_i, the degree D_i satisfies the following condition: $\Pr[w_i/4 \leq D_i \leq 3w_i] > 1 - 2/n^4$. Let G be a random graph sampled from $\mathbf{RPLG}(n, \tau)$. For a subset of vertices S satisfying $\mathrm{Vol}(S) \geq 192 \ln n$, it holds with probability at least $1 - 2/n^3$ that $\mathrm{Vol}(S)/8 \leq \mathrm{vol}(S) \leq 4\mathrm{Vol}(S)$. From previous lemma, a corollary, in which the number of edges of a random graph sampled from $\mathbf{RPLG}(n, \tau)$ is at most $\mathrm{vol}(G)/2 \leq \frac{4(\tau-1)}{\tau-2} n$ with probability at least $1 - 1/n^2$, was obtained. For any two disjoint subsets S and T with $\mathrm{Vol}(S).\mathrm{Vol}(T) > c.\mathrm{Vol}(G)$, there is

$$\Pr[d(S, T) > 1] = \prod_{v_i \in S, v_j \in T} \max\{0, (1 - w_i w_j / \mathrm{Vol}(G))\} \leq e^{-c}. \tag{70}$$

Regarding to the issue of core size, let G be a random graph sampled from $\mathbf{RPLG}(n, \tau)$. The probability of the core size is at least $1 - 1/n^2$, which holds that $\mathsf{core}_{\gamma'}(w) = \{v_i : w_i > n^{\gamma'}\} \subseteq \{v_i : \deg(v_i) > n^{\gamma'}/4\} = \mathsf{core}_{\gamma'}(G)$. Moreover, $|\mathsf{core}_{\gamma'}(G)| = \Theta(n^{\gamma})$.

Now we about the ball sizes. According to the original definition of a ball by Eq. (64), let $\beta = \gamma'(\tau - 2) + \frac{(2\tau-3)\varepsilon}{\tau-1}$ be a constant. Assume Eq. (61) is satisfied. For a random graph G sampled from $\mathbf{RPLG}(n, \tau)$, with probability at least $1 - 3/n^2$, it holds that for all $u \in V(G)$, there is

$$\begin{aligned} |B_G(u)| &= |\{u' \in V(G) : d(u, u') < d(u, \mathsf{core}_{\gamma'}(w))\}| \\ &= O(n^{\beta}), |E(B_G(u))| = O(n^{\beta} \log n), \end{aligned} \tag{71}$$

where $E(B_G(u))$ is the set of internal edges among vertices in $B_G(u)$.

The *existence* of every edge in random graph G is determined only when it is needed. It is treated as a probability distribution before determining which is defined in the random graph model of this analysis. According to the probability distribution *revealing* the edge, it is easy to know when the existence of an edge is determined. Under the condition that given vertex $u \in V(G)$ and a sequence of balls $(B_0 = \{u\}, B_1, B_2, \ldots)$, let $V' = V \backslash \mathsf{core}_{\gamma'}(w)$ and $B_i = \{v : d_G(u, v) \leq i\}$. Circles $C_i = B_i \backslash B_i - 1$ for $i \geq 0$ with $B_{-1} = \emptyset$. Then, focus on the result of E_i when the condition E_i, which is the number of edges between C_i and $C_i \cup C_{i+1}$, is given. For circle C_i, the following holds with probability at least $1 - 2/n^3$: (1) If $\mathrm{Vol}(C_i) < 192 \ln n$, then $E_i \leq 4 \cdot 192 \ln n$ and (2) if $\mathrm{Vol}(C_i) \geq 192 \ln n$, then $E_i \leq 4\mathrm{Vol}(C_i)$.

For table sizes and their computation issue, it is known that core $\mathsf{core}_{\gamma'}(G)$ has size $\Theta(n^{\gamma})$ with probability at least $1 - 1/n^2$ and all balls $B_G(u)$ have size $O(n^{\gamma})$ with probability at least $1 - 3/n^2$. Therefore, for a random graph G sampled from $\mathbf{RPLG(n,\tau)}$, for all $u \in V(G)$, the expected table size is at most

$|\mathsf{tbl}(u)| = O(n^{\gamma})$, and all tables can be generated in expected time at most $O(n^{1+\gamma} \log n)$. These bounds also hold with probability at least $1 - 1/n$.

Last but not least, the address lengths should be considered seriously. Bounding the number of bits for the address of each vertex is: for a node u; its address contains the encoding of the shortest path $SP(u, l(u))$ from u to its landmark l(u). Moreover, bounding the diameter of a random power-law graph and the diameter of its core is also needed. Therefore, for a random graph sampled from **RPLG(n,τ)**, with probability at least $1 - o(1)$, the diameter of its largest connected component is $\Theta(\log n)$[14]. If it holds for all $s, t \in V(G)$, $SP(s, t)$ can be encoded with $O(\log n \log\log n)$ bits. In order to extend the core, a definition is given as follows:

Definition 4 ([13]) The extended core of a random graph from $\mathbf{RPLG}(n, \tau)$ contains all nodes v_i with w_i at least $n^{1/\log\log n}$, that is, $\mathsf{core}^{+}(w) = \{v_i \in V : w_i \geq n^{1/\log\log n}\}$.

From the fact that the *extended core* "contains" a dense random graph [24], let G be a random graph sampled from $\mathbf{RPLG}(n, \tau)$. The diameter of the subgraph induced by $\mathsf{core}^{+}(w)$ in G is $O(\log\log n)$ with probability at least $1 - 1/n$ [14]. Moreover, from [14], it is known that there exists a constant C, such that each vertex v_i with $w_i \geq \log^{C} n$ is at distance $O(\log\log n)$ from the extended core, with probability at least $1 - 1/n^2$. However, from the previous statement, if the probability changed with at least $1 - 1/n$, the distance between any two vertices v_i and v_j with $w_i \geq \log^{C} n$ and $w_j \geq \log^{C} n$ is $O(\log\log n)$.

5.3.2 Analysis of Approximate Distance Oracle

For social networking sites, the graph of this kind of application is *preprocessed*, and a special data structure is used for efficient *queries*. Precomputing all shortest paths by using an all-pairs shortest path algorithm and reading a shortest path from a distance table is one way to prepare for queries. However, this approach is impractical due to the constraints of time and memory. If it is supposed to efficiently preprocess a graph to allow for fast distance queries, an approximation method is needed because of general and directed graphs with n vertices, which is necessary to return the shortest distance by using $\Omega(n^2)$ space. The trade-off between approximation ratio, space, preprocessing, and query time can be addressed by approximation distance oracle, and this kind of scenario can be interpreted as a generalization of the all-pairs (approximate) shortest path problem.

To solve this kind of distance oracle for power-law graphs problem, let $\gamma = \frac{\tau-2}{2\tau+3} + \varepsilon$ be a constant and suppose Eq. (61) is satisfied. There exists a preprocessing algorithm for random power-law graphs from $\mathbf{RPLG}(n, \tau)$, which creates a distance oracle of expected size $O(n^{1+\gamma})$ and runs in expected time $O(n^{1+\gamma} \log n)$. This method in this chapter is modified from Thorup and Zwick's distance oracle, which guarantees stretch 3 when $k = 2$. Unlike Thorup and Zwick's preprocessing method, each node $v \in V$ is chosen as a landmark independently at random with probability $n^{1/2}$ for general graphs, a better balance is possible by

using high-degree nodes as landmarks for power-law graphs. In this way, fewer landmarks will be chosen, and smaller-sized balls can be obtained than original method at the same time. In preprocessing period, the first step is to compute the core for any $v \in V$ and $\deg(v) > n^{\gamma'}/4$, then for each $v \in$ core, run the breadth-first search from $v \in G$. Moreover, if for each node $u \neq v$, store $d(u, v)$ and set $\mathsf{port}_u(v)$ to be the penultimum node on the shortest path. As for each $u \in V$, compute and store $B_G(u)$ and its distances. Finally, for each $v \in B_G(u)$, set $\mathsf{port}_u(v)$ to be the first node on the shortest path to v.

As far as the exact distance(s, t) is concerned with, the distance query result $d(s, t)$ is exact if $s \in B(t)$ or $t \in B(s)$, which runs in time $O(1)$ and achieves stretch 3.

With the rapid development of society, people have more and more opportunities to communicate with each other. From the aspect of social networks, it is of great importance to find the shortest paths for pairs of nodes. In this section, we introduced some properties that theoretically justify the importance of high-degree nodes in power-law graphs. After analyzing the adapted compact routing scheme and its algorithm for random power-law graphs, it is shown that optimizing for power-law graphs may induce better algorithm performance for problems in social network. From both real-world graphs and random power-law graphs, the optimization algorithm shows the efficiency.

Apart from compacting routing scheme in power-law graphs, some techniques in universal graphs are proposed: in [3], the first optimal compact name-independent routing scheme for arbitrary undirected graphs is proposed, and in [52], tight upper and lower bounds for the efficiency of a routing scheme and its space requirement are presented; furthermore, the bounds are improved in [24]. In addition, new compacting schemes about special graphs appear for tree graphs [36], directed graphs [11], weighted graphs [61], and so on.

6 Conclusion

Social network has shown wide range applications in real world, therefore, it is practically necessary to probe into social network structure to find its special properties, based on which many optimization problems regarding social networks in real life can be solved efficiently. In this chapter, approximation algorithms for four problems including link prediction problem, community detection problem, influence maximization problem, and routing problem are investigated. It particular, it can be seen that almost all of these algorithms are designed by using the structure features of social networks. For example, the routing schemes in Sect. 5 take advantage of power-law link distribution property of certain social networks.

Cross-References

►Optimization Problems in Online Social Networks

Recommended Reading

1. J. Abello, M.G.C. Resende, S. Sudarsky, Massive quasi-clique detection, in *LATIN*, pp. 598–612. doi:10.1007/3-540-45995-2-5134, 35
2. I. Abraham, C. Gavoille, D. Malkhi, On space-stretch trade-offs: lower bounds, in *Proceedings of the eighteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, Cambridge, 30 July-02 Aug 2006 (ACM, New York, 2006), pp. 207–216
3. I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, M. Thorup, Compact name-independent routing with minimum stretch. *16th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, Barcelona, 2004
4. L.A. Adamic, E. Adar, Friends and neighbors on the web. Soc. Netw. **25**(3), 211–230 (2003)
5. G. Agarwal, D. Kempe, Modularity-maximizing graph communities via math- ematical programming. Eur. Phys. J. B **66**, 409–418 (2008)
6. W. Aiello, F.R.K. Chung, L. Lu, A random graph model for massive graphs, in *STOC*, 2000, pp. 171–180
7. R. Albert, A.L. Barabasi, Statistical mechanics of complex networks. Rev. Mod. Phys. **74**, 47–97 (2002)
8. F. Bass, A new product growth model for consumer durables. Manag. Sci. **15**, 215–227 (1969)
9. A. Brady, L. Cowen, Compact routing on power law graphs with additive stretch, in *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX)*, Miami, 2006, pp. 119–128
10. J. Brown, P. Reinegen, Social ties and word-of-mouth referral behavior. J. Consum. Res. **14**(3), 350–362 (1987)
11. P. Chandra, A.D. Kshemkalyani, Compact routing in directed networks with stretch factor of two, in *High Performance Computing (HiPC)*, London, vol. 2228/2001, 2001, pp. 24–35
12. S. Charkrabarti, B. Dom, P. Indyk, Enhanced hypertext categorization using hyperlinks, in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (ACM, Seattle, 1998), pp. 307–318
13. W. Chen, C. Sommer, S. Teng, Y. Wang, Compact routing scheme and approximate distance oracle for power-law graphs. J. ACM Trans. Algorithms 9(1), Article No. 4
14. F. Chung, L. Lu, The average distances in random graphs with given expected degrees. Internet Math. **99**, 15879–15882 (2002)
15. F.R.K. Chung, L. Lu, *Complex Graphs and Networks*. Volume 107 of CBMS Regional Conference Series in Mathematics (AMS, Providence, 2006)
16. A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks. Phys. Rev. E **70**, 066111 (2004)
17. G. Cornuejols, M. Fisher, G. Nemhauser, Location of bank accounts to optimize float. Manag. Sci. **23**, 789–810 (1977)
18. L. Cowen, Compact routing with minimum stretch. J. Algorithms **38**(1), 170–183 (2001)
19. P. Domingos, M. Pazzani, On the potimality of the simple Bayesian classifier under zero-one loss. Mach. Learn. **29**, 103–130 (1997)
20. P. Domingos, M. Richardson, Mining the network value of customers, in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, 26–29 Aug 2001, pp. 57–66
21. R. Durrett, *Lecture Notes on Particle Systems and Percolation* (Wadsworth, Pacific Grove, 1988)
22. M. Enachescu, M. Wang, A. Goel, Reducing maximum stretch in compact routing, in *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, Phoenix, 13–18 Apr 2008, pp. 336–340
23. P. Erdös, A. Rényi, On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci. **5**, 17–61 (1960)

24. P. Fraigniaud, C. Gavoille, Universal routing schemes. Distrib. Comput. **10**(2), 65–78. doi:10.1007/s004460050025
25. S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 721–741 (1984)
26. M. Girvan, M.E.J. Newman, Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA **99**(12), 7821–7826 (2002)
27. J. Goldenberg, B. Libai, E. Muller, Talk of the network: a complex systems look at the underlying process of word-of-mouth. Mark. Lett. **12**(3), 211–223 (2001)
28. J. Goldenberg, B., Libai, E., Muller, Using complex systems analysis to advance marketing theory development: modeling heterogeneity effects on new product growth through stochastic cellular automata. Acad. Mark. Sci. Rev. (2001)
29. M. Granovetter, Threshold models of collective behavior. Am. J. Sociol. **83**(6), 1420–1443 (1978)
30. C. Guestrin, A note on the budgeted maximization of submodular functions. Technical report, CMU-CALD-05-103, 2007
31. R. Guimera, L.N. Amaral, Functional cartography of complex metabolic networks. Nature **433**, 895–900 (2005)
32. M.A. Hasan, V. Chaoji, S. Salem, M. Zaki, Link prediction using supervised learning, in *Workshop on Link Analysis, Counter-terrorism and Security (SDM 2006 Workshop)*, Bethesda, MD, 2006
33. Z. Huang, Link prediction based on graph topology: the predictive value of the generalized clustering coefficient, in *Workshop on Link Analysis: Dynamics and Static of Large Networks, the 12th ACM SIGKDD*, Philadelphia, 2006
34. D. Kempe, J. Kleinberg, E. Tardos, Maximizing the spread of influence through a social network, in *KDD '03 Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, pp. 137–146 (ACM, New York, 2003)
35. S. Khuller, A. Moss, J. Naor, The budgeted maximum coverage problem. Inf. Process. Lett. **70**, 39–45 (1999)
36. K.A. Laing, Brief announcement: name-independent compact routing in trees, in *PODC '04: Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing*, New York (ACM, 2004), pp. 382–382
37. D.V. Krioukov, K.R. Fall, X. Yang, Compact routing on internet-like graphs, in *Proceedings of IEEE INFOCOM*, Hong Kong, Mar. 2004
38. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost-effective outbreak detection in networks, in *KDD '07 Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose (ACM, New York, 2007), pp. 420–429
39. D. Liben-Nowell, J. Kleinberg, The link prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. **58**(7), 1019–1031 (2007)
40. R.N. Lichtenwalter, J.T. Lussier, N.V. Chawla, New perspectives and methods in link prediction, in *KDD '10 Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington* (ACM, New York, 2010), pp. 243–252
41. T.M. Ligget, *Interacting Particle Systems* (Springer, New York, 1985)
42. L. Lu, Probabilistic methods in massive graphs and internet computing. Ph.D. thesis, University of California, San Diego, 2002
43. J.O. Madadhain, J. Hutchins, P. Smyth, Prediction and ranking algorithms for event-based network data. ACM SIGKDD Explor. **7**, 23–30 (2006)
44. V. Mahajan, E. Muller, F. Bass, New product diffusion models in marketing: a review and directions for research. J. Mark. **54**(1), 1–26 (1990)
45. M. Mitzenmacher, A brief history of generative models for power law and lognormal distributions. Internet Math. **1**(2), 226–251 (2004)
46. T. Murate, S. Moriyasu, Link prediction of social networks based on weighted proximity measures, in *2007 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2007)*, Silicon Valley, 2007

47. G. Nemhauser, L. Wolsey, M. Fisher, Analysis of the approximations for maximizing submodular set functions. Math. Program. **14**, 265–294 (1978)
48. M.E. Newman, Clustering and preferential attachment in growing networks. Phys. Rev. Lett. E **64**, 025102 (2001)
49. M.E. Newman, Fast algorithm for detecting community structure in networks. Phys. Rev. E **69**, 066133 (2004)
50. M.E. Newman, M. Girvan, Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026113 (2004)
51. M.E. Newman, S.H. Strogatz, D.J. Watts, Random graphs with arbitrary degree distributions and their applications. Phys. Rev. E **64**, 026118 (2001)
52. D. Peleg, E. Upfal, A trade-off between space and efficiency for routing tables. J. ACM **36**(3), 510–530 (1989)
53. L. Pelkowitz, A continuous relaxation labeling algorithm for Markov random fields. IEEE Trans. Syst. Man Cybern. **20**, 709–715 (1990)
54. F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks. Proc. Natl. Acad. Sci. USA **101**, 2658–2663 (2004)
55. U.N. Raghavan, R. Albert, S. Kumara, Phys. Rev. E **76**, 036106 (2007)
56. M. Richardson, P. Domingos, Mining knowledge-sharing sites for viral marketing, in *KDD '02 Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton (ACM, New York, 2002), pp. 61–70
57. T. Schelling, *Micromotives and Macrobehavior* (Norton, New York 1978)
58. J. Scott, *Social Network Analysis: A Handbook*, 2nd edn. (Sage, London, 2000)
59. S.H. Strogatz, Exploring complex networks. Nature **410**, 268–276 (2001). Macmillan Magazines Ltd
60. M. Sviridenko, A note on maximizing a submodular set function subject to knapsack constraint. Oper. Res. Lett. **32**, 41–43 (2004)
61. M. Thorup, U. Zwick, Compact routing schemes, in *SPAA '01 Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, Crete Island (ACM, New York, 2001), pp. 1–10
62. M. Thorup, U. Zwick, Approximate distance oracles. J. ACM **52**(1), 1–24 (2005)
63. S. Wasserman, K. Faust, *Social Network Analysis* (Cambridge University Press, Cambridge, 1994)
64. D.R. White, F. Harary, Sociol. Methodol. **31**, 305–359 (2001)
65. F. Wu, B. Huberman, Finding communities in linear time: a physics approach. Eur. Phys. J. B **38**, 331 (2004)
66. E.W. Xiang, A survey on link prediction models for social network Data. Ph.D. dissertation, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, 2008