A Shapley Value-Based Approach to Discover Influential Nodes in Social Networks

Ramasuri Narayanam and Yadati Narahari, *Fellow, IEEE*

***Abstract*—Our study concerns an important current problem, that of diffusion of information in social networks. This problem has received significant attention from the Internet research community in the recent times, driven by many potential applications such as viral marketing and sales promotions. In this paper, we focus on the target set selection problem, which involves discovering a small subset of influential players in a given social network, to perform a certain task of information diffusion. The target set selection problem manifests in two forms: 1) top-k nodes problem and 2) λ-coverage problem. In the top-k nodes problem, we are required to find a set of k key nodes that would maximize the number of nodes being influenced in the network. The λ-coverage problem is concerned with finding a set of key nodes having minimal size that can influence a given percentage λ of the nodes in the entire network. We propose a new way of solving these problems using the concept of Shapley value which is a well known solution concept in cooperative game theory. Our approach leads to algorithms which we call the ShaPley value-based Influential Nodes (SPINs) algorithms for solving the top-k nodes problem and the λ-coverage problem. We compare the performance of the proposed SPIN algorithms with well known algorithms in the literature. Through extensive experimentation on four synthetically generated random graphs and six real-world data sets (Celegans, Jazz, NIPS coauthorship data set, Netscience data set, High-Energy Physics data set, and Political Books data set), we show that the proposed SPIN approach is more powerful and computationally efficient.**

***Note to Practitioners*—In recent times, social networks have received a high level of attention due to their proven ability in improving the performance of web search, recommendations in collaborative filtering systems, spreading a technology in the market using viral marketing techniques, etc. It is well known that the interpersonal relationships (or ties or links) between individuals cause change or improvement in the social system because the decisions made by individuals are influenced heavily by the behavior of their neighbors. An interesting and key problem in social networks is to discover the most influential nodes in the social network which can influence other nodes in the social network in a strong and deep way. This problem is called the target set selection problem and has two variants: 1) the top-k nodes problem, where we are required to identify a set of k influential nodes that maximize the number of nodes being influenced in the network and 2) the λ-coverage problem which involves finding a set of influential nodes having minimum size that can influence a given percentage λ of the nodes in the entire network. There are many existing algorithms in the literature for solving these problems. In this paper, we propose a new algorithm which is based on a novel interpretation of information diffusion in a social network as a cooperative game. Using this analogy, we develop an algorithm based on the Shapley value of the underlying cooperative game. The proposed algorithm outperforms the existing algorithms in terms of generality or computational complexity or both. Our results are validated through extensive experimentation on both synthetically generated and real-world data sets.**

***Index Terms*—Diffusion of information, influential nodes, Shapley value, social networks, target set selection, top-k nodes, λ-coverage.**

I. Introduction

A SOCIAL network is a social structure made of individuals or organizations that are tied by one or more specific types of interdependencies, such as friendship, coauthorship, collaboration, etc. Real-world examples for web-based social networks include Myspace.com, Facebook, Orkut, etc. In each of these social networks, the underlying representation is a graph model where typically each individual is represented by a node, and there is an edge between two nodes if there exists a social interaction between them. Analyzing the structure of complex relationships that exist among the nodes in a social network is helpful in several ways such as determining how the information spreads in the network. This study allows us to know the critical role social networks play in several applications [51], [52]. Recently, social networks have received a high level of attention due to their ability in improving the performance of web search, recommendations in collaborative filtering systems, spreading a technology in the market using viral marketing techniques, etc. It is the interpersonal relationships (or ties or links) between individuals that cause change or improvement in the social system because the decisions made by individuals are influenced heavily by the behavior of their neighbors [23]. Among all nodes in a given social network, it is important and interesting to discover nodes which can affect the behavior of their neighbors and, in turn, all other nodes in a stronger way than the remaining nodes. We call such nodes *influential nodes*. We present two motivating examples to understand the importance of finding the influential nodes in real-world social networks.

Our first example deals with the diffusion of information in social networks. In general, social networks play a key role for the spread of an innovation or technology or information within a population of individuals. Suppose that we have data on a social network, with estimates on the extent to which individuals influence one another, and we would like to market a new product that we hope will be adopted by a large fraction of the

Manuscript received December 13, 2009; revised April 07, 2010 and April 25, 2010; accepted May 15, 2010. Date of publication July 01, 2010; date of current version January 07, 2011. This paper was recommended for publication by Associate Editor S. Zhou and Editor M. Zhou upon evaluation of the reviewers' comments. The work of R. Narayanam is supported by a Microsoft Research India Ph.D. Fellowship.

The authors are with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India (e-mail: nrsuri@csa.iisc.ernet.in; hari@csa.iisc.ernet.in).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2010.2052042

network. Which set of the individuals should we target? The idea is to initially target a few influential individuals in the network who will trigger a massive cascade of influence through which friends will recommend the product to other friends, and many individuals will ultimately try it. Given such a system, a natural question that emerges is to find a target set of desired cardinality that consists of influential nodes for maximizing the volume of the information cascade [14], [28], [29]. An effective answer to this question has significant applications in marketing, politics, economics, epidemiology, sociology, computer networking, and databases.

The second example is based on coauthorship networks and is concerned with the collaboration patterns among research communities. There exists a natural social network among the researchers where nodes correspond to researchers and an edge exists between two nodes if the two corresponding researchers have coauthored a paper. Using a coauthorship social network, it may be of interest to find the most prolific researchers since they are most likely to be the trend setters for breakthroughs.

In the above two examples, the common goal is to find a set of influential nodes given a well defined context in the social network. We call such a set of influential nodes as *target set*. We call the problem of determining a target set of particular cardinality to perform a given task in the social network as *target set selection problem* [5]. In this paper, we address two natural forms of the target set selection problem which we call the *top-k nodes problem* and the *λ-coverage problem*. The top-k nodes problem requires to determine a set of influential nodes to target for maximizing the volume of information cascade in the social network. More precisely, this problem deals with identifying a set of k most influential nodes to maximize the (expected) number of nodes that are influenced in the social network where k is a given parameter. The λ-coverage problem is concerned with finding a set of influential nodes having minimum cardinality with which we can influence a fixed percentage λ of the nodes in the social network through the process of diffusion. This problem is important in many contexts, for example, see [13], [18], and [42].

We address these two problems in the context of diffusion of information in social networks. The following is a brief discussion on diffusion of information.

A. Diffusion of Information

The conceptual framework of diffusion of information refers to the spread of abstract ideas or technical information within a social system, where the spreading denotes flow or movement from a source to an adopter, typically via a communication link [43], [7]. Such a communication can influence and alter an adopter's probability of adopting an innovation, where an adopter may be an individual, a group, or an organization.

There are two popular operational models in the literature that capture the underlying dynamics of the diffusion process. They are the *linear threshold model* [24], [45] and the *independent cascade model* [21].

Linear Threshold Model: This is proposed by Granovetter [24] and generalized by Watts [50]. We call a node active if it has adopted the information and inactive otherwise. In this model, initially every node is inactive. That is, no node has adopted the information that is being propagated. As time elapses, the neighbors of a node become active and at some point of time, this causes the given node also to become active. Granovetter [24] and Schelling [45] proposed models to capture this kind of node behavior based on the thresholds of the individual nodes. Let us consider a node i and represent its neighbors by the set N_i. Node i is influenced by a neighbor node j according to a weight w_{ij}. Assume these weights are normalized in such a way that $\sum_{j \in N_i} w_{ij} \leq 1$. The decision of a node i to become active is based on a threshold function (f_i) of the set of active neighbors of i and a threshold (call it θ_i) chosen uniformly at random by node i from the interval [0,1]. Note that $f_i : 2^{N_i} \rightarrow [0,1]$ is defined as $f_i(T) = \sum_{j \in T} w_{ij}, \forall T \subseteq N_i$. The threshold θ_i represents the weighted fraction of the neighbors of node i that must become active in order for node i to become active.

Given a choice of thresholds and an initial set (call it S) of active nodes, the diffusion process propagates as follows: In time step t, all nodes that were active in step $(t-1)$ remain active, and we activate every node i for which the total weight of its active neighbors is at least θ_i. In other words, if $A(i)$ is assumed to be the set of active neighbors of node i, then i gets activated if $f_i(A(i)) = \sum_{j \in A(i)} w_{ij} \geq \theta_i$. This process stops when there is no new active node in a particular time interval.

Independent Cascade Model: This model is investigated recently in the context of marketing by [21] and [22]. Here, we start with an initial set of active nodes A_0, and the process unfolds in discrete steps according to the following randomized rule. When a node i first becomes active in step t, it is given a single chance to activate each currently inactive neighbor j; it succeeds with a probability that is independent of the history thus far. If j has multiple newly activated neighbors, their attempts are sequenced in an arbitrary order. If i succeeds, then j will become active in step $(t+1)$; but whether or not i succeeds, it cannot make any further attempts to activate j in subsequent rounds. Again, the process runs until no more activations are possible.

In this paper, we consider the linear threshold model.

B. Target Set Selection Problem

We now describe the top-k nodes problem and the λ-coverage problem more precisely.

Top-k Nodes Problem: Define an objective function (or influence function) $\sigma(.)$ as follows. If S is the set of initially active nodes (also called the target set), then $\sigma(S)$ is the expected number of active nodes at the end of the diffusion process. For economic reasons, we want to limit the size of the initial active set S. For a given constant k, the top-k nodes problem seeks to find a subset of nodes S of cardinality k that maximizes the value of $\sigma(S)$.

λ-Coverage Problem: We are given $\lambda \in [0, 100]$. The problem is to find a subset S of influential nodes having minimal size such that $\sigma(S)$ contains at least λ percent of the nodes in the network.

We now briefly discuss different solution methods and algorithms from the existing literature that address these problems.

C. Relevant Work

We first focus on the top-k nodes problem. Domingos and Richardson [14], and Richardson and Domingos [44] were

the first to study the top-k nodes problem as an algorithmic problem. They modeled social networks as Markov random fields where the probability of an individual adopting a technology (or buying a product) is a function of both the intrinsic value of the technology (or the product) to the individual and the influence of neighbors. The authors proposed three algorithms that approximately determine the influential users and showed that selecting the right set of users for a marketing campaign can make a substantial difference.

The algorithmic and computational aspects of the top-k nodes problem are investigated by [28], [29], and [31]. The authors show that the optimization problem of selecting the most influential nodes is NP-hard and derive the first provable approximation guarantees for the proposed algorithm. Recall that the natural objective function, $\sigma(.)$, for information diffusion is the expected number of nodes that become active at the end of the diffusion process given a set of initial active nodes. The authors first show that this objective function is a submodular function under the linear threshold model and the independent cascade model. A function $g(.)$ is called submodular if it satisfies $g(S \bigcup \{i\}) - g(S) \geq g(T \bigcup \{i\}) - g(T)$, for all elements i and all pairs of sets $S \subseteq T$. The authors propose a greedy algorithm described in Algorithm 1. Using the property of submodularity, the authors show that the greedy algorithm achieves an approximation guarantee of $(1 - (1/e))$ where $e = \sum_{r=1}^{\infty}(1/r!)$.

Algorithm 1: Greedy Algorithm of Kempe *et al.* [28]. N is the set of nodes and k is a positive integer such that $k \leq |N|$

1: Set $A \leftarrow \phi$.
2: **for** $i = 1$ to k **do**
3: $-$ Choose a node $n_i \in N \setminus A$ maximizing $\sigma(A \cup \{n_i\}) - \sigma(A)$
4: $-$ Set $A \leftarrow A \cup \{n_i\}$.
5: **end for**

We note that Kleinberg, and Tardos [28] proposed a conjecture which states that, whenever the threshold functions f_i at every node are monotone and submodular, the resulting influence function $(\sigma(.))$ is monotone and submodular as well. Later, this conjecture was proved by Mossel and Roch [35]. For this result to hold, the submodularity of f_i is necessary in the following sense [35]: any function f_i which is not submodular admits a network with activation function f_i where the influence function is not submodular.

Leskovec *et al.* [34] address the top-k nodes problem while focusing on two specific applications, namely: i) given a water distribution network, where should we place sensors to quickly detect contaminants? and ii) which blogs should we read to avoid missing important stories? The authors develop an efficient algorithm based on the submodularity of the underlying objective function that scales to large problems and is reportedly 700 times faster than the greedy algorithm of Kempe *et al.* [28]. There are two aspects to this speed up: i) by speeding up function evaluations using the sparsity of the underlying problem and (ii) by reducing the number of function evaluations using the submodularity of the influence functions.

Chen *et al.* [10] present an efficient algorithm to find the top-k nodes in a social network and this algorithm improves upon the greedy algorithm of Kempe *et al.* [28] and also the algorithm of Leskovec *et al.* [34] in terms of its running time. The authors also design a new heuristic algorithm, which they call degree discount heuristic, that achieves much better influence spread than classic degree and centrality-based heuristics. They also note that the performance of this heuristic algorithm is comparable to that of the greedy algorithm while its running time is much less than that of the greedy algorithm.

Kimura and Saito [30] propose a shortest-path-based influence cascade model and provide efficient algorithms for finding the most influential nodes under these models. As the information cascade models are different, they do not directly address the efficiency issue of the greedy algorithm for the linear threshold model and the independent cascade model. This same issue is pointed out in [10].

Even-Dar and Shapira [16] study the top-k nodes problem in the context of probabilistic voter model ([26]). The authors present simple and efficient algorithms for solving this problem. Furthermore, in a special case, the popular heuristic which picks nodes in the network with the highest degree turns out to be an optimal solution.

Recall that the top-k nodes problem is a hard problem computationally. The only case for which the problem is known to have an acceptable worst-case solution is where the given social network has a bounded tree-width and the problem becomes polynomial-time solvable. For this kind of tractable instances, Ben-Zwi *et al.* [5] propose algorithms by considering tree-width parameter of the graphs.

Information diffusion models and the top-k nodes problem are also appropriately considered from the view of the blogspace where a blogger may have a certain level of interest in a topic and is thus susceptible to talking about it. By discussing the topic, the blogger may influence other bloggers. Adar and Adamic [1] and Gruhl *et al.* [25] model and study the dynamics of diffusion of information in the blogspace. Java *et al.* [27] and Agarwal *et al.* [2] discuss algorithms to identify influential blog posts and influential bloggers in a blogspace.

There are two popular heuristics that measure the influential capabilities of the nodes in social networks [48]. These heuristics differ in terms of the approach by which they choose the target set. Given this target set, the linear threshold model could be used to determine the expected number of active nodes at the end of the diffusion process.

1) *Maximum Degree Heuristic (MDH)*: The concept of centrality is well addressed in social networks [17]. One of the simplest and best-known measures of centrality is degree centrality, which is a count of the number of edges incident upon a given node. The maximum degree heuristic chooses the k nodes having the k highest degrees as the top-k nodes.
2) *High Clustering Coefficient Heuristic (HCH)*: Clustering of a node is a measure of the likelihood that two neighbors of the node are neighbors themselves [48]. Clustering coefficient of a node is defined as the fraction of number of pairs of its neighbors that are connected themselves as well. This indicates how dense the neighborhood of a node is in a network. Following this clustering heuristic, we choose

the first k nodes with high clustering coefficient as the initial target set.

We now turn to the relevant literature on the λ-coverage problem. Chen [9] shows that the problem is hard to approximate within a polylogarithmic factor using a deterministic thresholds model. It is possible to design optimal polynomial time algorithms for some restricted topologies of the networks. In particular, when the network is a tree, Chen [9] presents an optimal polynomial time algorithm for deterministic thresholds model.

D. Contributions and Outline of the Paper

We are motivated by the following reasons in seeking a new algorithm for the target set selection problem:

- The complexity of the existing algorithms for the top-k nodes problem explicitly depends on k.
- Algorithms that work for the top-k nodes problem do not immediately work for the λ-coverage problem.
- Existing algorithms seem to work well (in terms of solution quality) only when the node threshold functions are submodular and monotone increasing.

Our approach to solving the target set selection problem is fundamentally different from the existing approaches in the literature. Our approach is to map the information diffusion process in a social network to the formation of coalitions in an appropriately defined cooperative game. The Shapley values of the nodes in this game represent the marginal contributions that the nodes make to the information diffusion process. We use this fact to design an algorithm to discover influential nodes in the social network. We call our algorithm the ShaPley value-based Influential Nodes (SPINs) algorithm.

We first propose a SPIN algorithm for the top-k nodes problem. To compute the Shapley values required by the SPIN algorithm, we use a simple sampling technique to obtain a computationally efficient scheme. The SPIN algorithm can also be used for solving the λ-coverage problem with a minor modification. We carry out detailed experiments using four synthetic datasets of random graphs and six real-world social network datasets (Celegans, Jazz, NIPS coauthorship data set, Netscience data set, High-energy physics data set, and Political Books data set) and these results establish the SPIN algorithm as a powerful and effective approach to solve the target set selection problem. Comparison of performance with existing benchmark algorithms shows the following specific advantages:

1) The SPIN algorithm produces target sets which are as good as the target sets produced by: i) the greedy algorithm of Kempe *et al.* [28] (call this the KKT algorithm); ii) the CELF algorithm by Leskovec *et al.* [34] (call this the LKG algorithm); and iii) the more recent and faster algorithm by Chen *et al.* [10] (call this the CWY algorithm). It is known that the LKG algorithm is about 700 times faster than the KKT algorithm [34] for certain problem instances. We show that the proposed SPIN algorithm turns out to be more efficient than the LKG algorithm, and hence the SPIN algorithm clearly outperforms the KKT algorithm.
2) An important salient feature of the SPIN algorithm is that its execution time is (almost) independent of the value of k (regarding the top-k nodes problem). We note that the KKT, LKG, and CWY algorithms are heavily dependent on the value of k. For more discussion on this, we refer to Section III-C.
3) The maximum degree heuristic (MDH)-based and the high clustering heuristic (HCH)-based algorithms are much faster compared to the KKT, LKG, CWY, and SPIN algorithms, however, their performance in terms of the target sets produced tends to be very poor. This is strikingly observed in all the experiments.
4) It is known that the objective function $\sigma(.)$ is monotone increasing and submodular whenever the node threshold function at each node is monotone increasing and submodular [35]. Thus, the approximation guarantee of $(1-(1/e))$ holds when the node threshold function at each node is monotone increasing and submodular. However, in many settings, it is quite possible that the node threshold functions are monotone decreasing and nonsubmodular and in such contexts none of the algorithms have approximation guarantees. In Section IV of this paper, we propose two new models of diffusion of information and show that the corresponding node threshold functions are monotone decreasing and nonsubmodular. In these two settings, we experimentally show that the SPIN algorithm outperforms the KKT algorithm in terms of quality of the solution.

E. Organization of This Paper

This paper is organized as follows. Section II describes the Shapley value-based approach for the top-k nodes problem and the λ-coverage problem. We first present some essential preliminaries from cooperative game theory. Next, we describe the SPIN algorithm for the top-k nodes problem using an illustrative example. Then, we describe the SPIN algorithm for the λ-coverage problem. In Section III, we present detailed experimentation (where the dynamics of diffusion process are captured by the linear threshold model). First, we describe the data sets employed—both synthetic random graphs and also six different real-world data sets. Then, we describe the experimental setup and the experimental results. We provide a detailed comparison of performance of our algorithms with that of the KKT, LKG, MDH, and HCH algorithms. We also include a discussion on the statistical significance of the results obtained in our experiments. In Section IV, we experimentally show that the proposed algorithm outperforms the existing algorithms in terms of the quality of the solution when the threshold function at each node is monotone decreasing and nonsubmodular. We conclude this paper in Section V with a summary and several interesting directions for future work.

II. SPIN: Shapley Value-Based Discovery of Influential Nodes

A. Preliminaries

A cooperative game with transferable utility (TU game) [36] is defined as the pair (N, v), where $N = \{1, 2, \ldots, n\}$ is the set of players and $v : 2^N \rightarrow \mathbb{R}$ is a real-valued mapping with $v(\phi) = 0$. Note that 2^N is the set of all possible subsets of N. The mapping v is called the characteristic function or the value function. Given any subset S of N, $v(S)$ is called the value

of the coalition S and represents the total transferable utility that can be achieved by the players in S, without help from the players in $N \setminus S$. The set of players N is called the grand coalition and $v(N)$ is called the value of the grand coalition. In the sequel, we use the phrases cooperative game, coalitional game, and TU game interchangeably.

A cooperative game can be analyzed using a *solution concept*, which provides a method of dividing the total value of the game among individual players. There are many solution concepts such as the core, the Shapley value, the nucleolus, etc. We describe below the solution concept, the *Shapley value*, that is relevant to our work here. The Shapley value is a solution concept that provides a unique expected payoff allocation for a given coalitional game (N, v). It describes an effective approach to the fair allocation of gains obtained by cooperation among the players of a cooperative game. Since some players may contribute more to the total value than others, an important requirement is to distribute the gains fairly among the players. The concept of Shapley value, which was developed axiomatically by Shapley [46], takes into account the relative importance of each player to the game in deciding the payoff to be allocated to the players. We denote by

$$\Phi(N, v) = (\Phi_1(N, v), \Phi_2(N, v), \ldots, \Phi_n(N, v))$$

the Shapley value of the TU game (N, v). Mathematically, the Shapley value, $\Phi_i(N, v)$, of a player i is given by

$$\Phi_i(N,v) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!\,(n-|C|-1)!}{n!} \{v(C \cup \{i\}) - v(C)\}$$

where $\Phi_i(N, v)$ is the expected payoff to player i. There are several equivalent alternative formulations for the Shapley value. We present one equivalent formulation of the Shapley value in the following.

Given a node $i \in N$ and a subset $C \subseteq N$ such that $i \notin C$, the marginal contribution of node i to the coalition is defined as $v(C \bigcup \{i\}) - v(C)$, $\forall C \subseteq N \setminus \{i\}$. Now, consider the set Ω of all possible $n!$ permutations on N. Let π be a permutation in Ω and define $C_i(\pi)$ to be the set of all nodes appearing before node i in the permutation π. We compute the average marginal contribution of node i to the given coalitional game as

$$\frac{1}{n!} \sum_{\pi \in \Omega} \left[v\left(C_i(\pi) \cup \{i\}\right) - v\left(C_i(\pi)\right) \right].$$

The above expression is exactly the Shapley value of player i. It is clear that we have to work with $n!$ permutations to determine the marginal contributions of nodes. It is easy to see (using the Stirling approximation [11] for $n!$) that the computational complexity of this approach is $O((n/e)^n)$. Hence, the direct, naive approach for computing the Shapley values of the players is not a tractable one. We therefore resort to computing the Shapley values approximately in this paper.

The Shapley value is the unique mapping that satisfies three key properties: linearity, symmetry, and carrier property [36]. The three properties imply that the Shapley value provides a *fair* way of distributing the gains of cooperation among the players in the game. A natural way of interpreting the Shapley value $\Phi_i(N, v)$ of player i is in terms of the average marginal contribution that player i makes to any coalition of N assuming all orderings are equally likely. Thus, the Shapley value takes into account all possible coalitional dynamics and negotiation scenarios among the players and comes up with a single unique way of distributing the value $v(N)$ of the grand coalition among all the players. The Shapley value of a player accurately reflects the bargaining power of the player and the marginal value the player brings to the game.

In this paper, given a social network, our approach is to define a cooperative game that captures the information diffusion process in the social network. The nodes of the social network happen to be the players in this cooperative game and we use the Shapley values of the nodes to discover the influential nodes.

B. The SPIN Heuristic Algorithm for Finding Top-k Nodes

The fundamental idea of the greedy algorithm (that is, Algorithm 1) is to pick, in each iteration, a remaining node having maximum influence on the spread of information. Essentially, this means picking a node (from the remaining nodes) that makes the maximum marginal contribution to the process of spread of information. This idea of marginal contribution is what we use in the proposed SPIN algorithm also. However, to compute the marginal contributions, we use a well known solution concept, Shapley Value, from cooperative game theory. The Shapley value of a coalitional game provides the marginal contributions of the individual players to the overall value that can be achieved by the grand coalition of all the players. The critical idea behind the SPIN algorithm is to model the nodes in the social network as players in a coalitional game and to capture information diffusion process as the process of coalition formation in the coalitional game. The Shapley value then automatically provides the marginal contributions.

There are two main steps in this algorithm:

1) computing a ranking list of the nodes based on the Shapley value;
2) choosing the top k nodes from the rank list.

We describe these two steps in the following. Recall that a node is said to be *active* if it adopts the information or the technology. We assume that when a node becomes active, it continues to be active later on. Our approach assigns a credit for each node based on its *influence*. We say node i is more influential than node j when node i can cause more number of nodes to be activated than that of node j. If we know the influential capabilities of the nodes in the underlying diffusion process, we can identify the most influential nodes for the top-k nodes problem. Our idea is to identify the influential nodes through the marginal contribution the nodes make to the diffusion process. For this we use the Shapley value of an appropriate cooperative game.

We now define the following cooperative game (N, v) as follows. Let N be the set of nodes in the social network. Let $n = |N|$ and 2^N the set of all subsets of N. We define the characteristic function $v : 2^N \rightarrow \mathbb{R}$ as follows. For each $S \subseteq N$, if all the nodes in S are initially activated, then $v(S)$ represents the expected number of active nodes at the end of the diffusion process. In other words $v(S) = \sigma(S)$, $\forall S \subseteq N$. We assume

that $v(\phi) = 0$. This mapping is natural and intuitive. We use v and σ interchangeably through the rest of this paper.

Since computing the Shapley values of the nodes exactly is a hard problem computationally, we compute Shapley values of the nodes approximately using a sampling-based approach that works in polynomial time. We do this by using a randomly sampled set, call it $\hat{\Omega}$, of permutations where the cardinality of $\hat{\Omega}$ is a polynomial in n. Let $t = |\hat{\Omega}|$ so that $t = O(n)$. Using this sampled set $\hat{\Omega}$, we compute the Shapley values of the nodes approximately (Algorithm 2). In this algorithm:

- $S_i(\pi_j)$ represents the set of nodes that occur before node i in the permutation π_j;
- $MC[i]$ represents the marginal contribution of node i;
- $\Phi[i]$ represents Shapley value of a node i.

Algorithm 2: RankList[] Construction (SPIN)

1: Let π_j be the j-th permutation in $\hat{\Omega}$.
2: Also let R be the number of repetitions of the experiment.
3: **for** $i = 1$ to n **do**
4: set $MC[i] \leftarrow 0$
5: **end for**
6: **for** $j = 1$ to t **do**
7: **for** $i = 1$ to n, **do**
8: $temp[i] \leftarrow 0$;
9: **end for**
10: **for** $r = 1$ to R, **do**
11: assign random thresholds to nodes;
12: **for** $i = 1$ to n, **do**
13: $temp[i] \leftarrow temp[i] + v(S_i(\pi_j) \cup \{i\}) - v(S_i(\pi_j))$
14: **end for**
15: **end for**
16: **for** $i = 1$ to n, **do**
17: $MC[i] \leftarrow temp[i]/R$;
18: **end for**
19: **end for**
20: **for** $i = 1$ to n, **do**
21: compute $\Phi[i] \leftarrow (MC[i]/t)$
22: **end for**
23: use an efficient sorting algorithm to sort the nodes in nonincreasing order based on average marginal contribution values of the nodes

We sort the nodes in nonincreasing order of their Shapely values and construct a rank list, call it RankList[]. The ties could be resolved randomly. The subroutine presented in Algorithm 2 constructs the RankList[] of the nodes.

The implementation details of Algorithm 2 are as follows. We randomly generate t permutations of the nodes in the network. Let π be a permutation from the set of t randomly generated permutations. Assume that $\pi(i)$ represents the ith node in the permutation π. Now, following the order of the nodes as dictated by π, we compute the contribution of each node to the spread of influence as follows. *Initially, all nodes in the network are inactive and we randomly assign a threshold to each node. First, we activate $\pi(1)$ to determine how many nodes are activated because of its activation and this becomes the contribution of $\pi(1)$. Next, we consider $\pi(2)$ and if it is already activated due to the activation of $\pi(1)$, then the contribution of $\pi(2)$ is 0. Otherwise, we activate $\pi(2)$ to determine the number of nodes activated because of its activation and this becomes the contribution of node $\pi(2)$. Likewise, we continue up to $\pi(n)$.* We repeat the above process R times (for example 10 000 times) using the same π. Furthermore, we repeat all the above steps for each permutation in the set of sampled permutations and we determine the average contribution of each node towards the spread of information. Finally, we sort the nodes in nonincreasing order of their contribution values to construct the rank list of nodes.

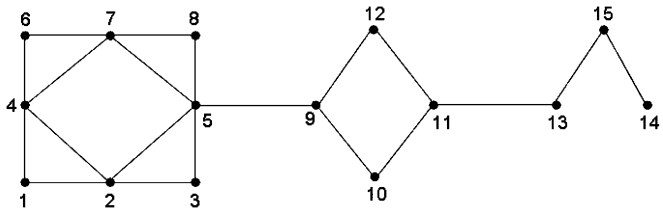

Fig. 1. An example network.

Now, we come to the important question of how to choose the top-k nodes from the RankList[]. The naive approach is to choose the first k in the RankList[] as the top-k nodes. This approach suffers from the following drawback: the chosen nodes may be clustered at one place. For example, consider the network shown in Fig. 1. Here, we note that $w_{ij} = (1/d_i)$, $\forall j \in N_i, \forall i \in N$. More specifically, $w_{12} = w_{14} = (1/2)$, $w_{21} = w_{23} = w_{24} = w_{25} = (1/4)$, etc. For this network, RankList[] $=\{5, 4, 2, 7, 11, 15, 9, 13, 12, 10, 6, 14, 3, 1, 8\}$. If $k = 4$, then we choose the nodes 5, 4, 2, 7 as the top four nodes following the naive idea of choosing the top four nodes from the RankList[]. Observe that these four nodes are located in the same cluster of the network. On the other hand, if these nodes are appropriately spread over the network, then such a situation will increase the number of active nodes or the cascade of information.

Motivated by the above situation, we choose the top-k nodes from the RankList[] as follows. We consider the nodes in the order given by RankList[] and initially we add the nodes to the list of top-k nodes that are not adjacent. More precisely: i) we take the first node from the RankList[] and add it to the list of top-k nodes; ii) we take the second node from RankList[] and add it to the list of top-k nodes if it is not adjacent to the node in the list of top-k nodes, and so on; and iii) in general, when we consider a node from the RankList[], we add it to the list of top-k nodes if it is not adjacent to any node in the list of top-k nodes. In this process, after adding a certain number of nodes to the list of top-k nodes, we may not find any node from the RankList[] that is not adjacent to any node in the list of top-k nodes. In other words, any node in the network is either added to the list of top-k nodes or adjacent to some node in the list of top-k nodes. Then, we consider nodes with the highest Shapley

values that are still not included to the list of top-k nodes and add them to the list. We stop the above process when the size of the list of top-k nodes is k. The algorithm presented in Algorithm 3 chooses the top-k nodes from the RankList[] as described above.

Algorithm 3: Choosing The Top-k Nodes (SPIN)

1: Initialize index := 1 and status[1 . . . n] to 0;
2: **for** $j = 1$ to n, **do**
3: flag $\leftarrow 0$;
4: **for** $i = 0$ to index, **do**
5: **if** topknodes$[i]$ = RankList$[j]$ *or*
6: topknodes$[i]$ is adjacent to RankList$[j]$,
7: **then**
8: flag $\leftarrow 1$;
9: **break**;
10: **end if**
11: **end for**
12: **if** flag $= 0$, **then**
13: **if** index $= k$
14: **then** goto Step 30:
15: **end if**
16: topknodes[index] $\leftarrow$ RankList$[j]$;
17: status$[j] \leftarrow 1$;
18: index $\leftarrow$ index $+ 1$;
19: **end if**
20: **end for**
21: **for** $j = 0$ to n, **do**
22: **if** status$[j] \neq 1$
23: **if** index $= k$
24: **then** goto Step 30:
25: **end if**
26: topknodes[index] $\leftarrow$ RankList$[j]$
27: index $\leftarrow$ index $+ 1$;
28: **end if**
29:**end for**
30: Declare the nodes in topknodes[] as top-k nodes

In Algorithm 3, the list of top-k is represented by *topknodes[]* and *index* refers to how many nodes are added to *topknodes[]*. Initially, the status of each node is set to 0 indicating the fact that each node i is not added to *topknodes[]* and this task is performed at step *1* in the algorithm. Next, we do the following for each node j in the RankList[]: (i) We examine whether the node in RankList$[j]$ is already added to *topknodes[]*; also we examine whether the node in RankList$[j]$ is adjacent to any node in *topknodes[]* (step 4 to step 11); ii) If the above two conditions are not satisfied, then we add the node in RankList$[j]$ to *topknodes[]* (step 12 to step 19). We also make a consistency check whether we have already found the top k nodes (step 13 to step 15); (iii) If k nodes are not added to *topknodes[]* at the end of step 20, then we add the nodes (among the nodes that are still not added) with highest Shapley values to *topknodes[]* and we stop the process when *topknodes[]* contains exactly k nodes (step 21 to step 29).

TABLE I
NUMBER OF ACTIVE NODES AT THE END OF THE DIFFUSION PROCESS WHEN THE SIZE (k) OF THE INITIAL TARGET SET TAKES VALUES 1,2,. . .,15, RESPECTIVELY

k *value*	*Greedy Algorithm*	*SPIN Algorithm*	*MDH Algorithm*	*HCH Algorithm*
1	4	4	4	2
2	8	7	7	4
3	10	10	8	6
4	12	12	8	7
5	13	13	10	8
6	14	14	13	8
7	15	15	13	8
8	15	15	13	8
9	15	15	13	10
10	15	15	13	11
11	15	15	13	13
12	15	15	13	13
13	15	15	14	14
14	15	15	15	15
15	15	15	15	15

To illustrate the proposed approach, let us consider again the graph shown in Fig. 1. Table I summarizes the number of active nodes using: 1) the greedy algorithm (Algorithm 1); 2) the SPIN algorithm; 3) the maximum degree heuristic (MDH)-based algorithm; and 4) the high clustering heuristic (HCH)-based algorithm when the size of the initial target set (k) takes values $1, 2, \ldots, 15$ respectively. The results of the greedy algorithm are averages over 10 000 repetitions of the experiment. It is clear that the performance of the SPIN algorithm almost matches that of the greedy algorithm. For example, when $k = 4$, i) greedy algorithm chooses {5,11,2,15} as the target set; ii) the SPIN algorithm chooses {5,4,11,15} as the target set; iii) the maximum degree heuristic chooses {5,4,2,7} as the target set; and iv) the high clustering heuristic chooses {1,3,6,8}. Note that the nodes in the target sets chosen by the greedy algorithm and the SPIN algorithm are spread over the network whereas the nodes chosen by the maximum degree heuristic are clustered. For this reason, the performance of the maximum degree heuristic is inferior to that of the greedy algorithm and the SPIN algorithm.

1) Computational Complexity: Recall that the running time of the Shapley value-based algorithm depends on two main steps namely RankList[] construction and the choice of top-k nodes from RankList[]. We have to compute the marginal contribution of each node and it takes $O(t(n+m)R)$ time where R is the number of repetitions of the experiment and m is the number of edges in the graph. It takes $O(n\ log(n))$ time to construct the RankList[]. Then, choosing the top k nodes from the RankList[] takes at most $O(kn)$ time. Finally, with these top k nodes in hand, we repeat the experiment R times in order to determine the expected number of active nodes at the end of the diffusion process and this takes $O(kRm)$. The overall running time of the Shapley value-based algorithm is $O(t(n+m)R + nlog(n) + kn + kRm)$, where t is a polynomial in n.

Note: From the view point of practical graphs (or real-world graphs), it is reasonable to assume that $n < m$. With this, the overall running time of the SPIN heuristic algorithm is $O(tmR)$ where t is a polynomial in n.

C. The λ-Coverage Problem

Though the λ-coverage problem is different from the top-k nodes problem, a very minor modification to the SPIN algorithm turns out to be an efficient heuristic algorithm for this problem. In particular, our approach is the following:

1) We invoke Algorithm 3 with $k = n$. This yields an ordering of nodes as provided by the list of topknodes[] from Algorithm 3.
2) Now, we determine the smallest value of x for which initially activating the first x nodes in the list of topknodes[] results in at least λ percent of the nodes in the network become active.

For example, recall the graph shown in Fig. 1 and Table I shows the expected number of active nodes at the end of the diffusion process when the size of the initial target set takes values 1,2,. . .,15, respectively. Now, we make the following observations: i) to cover the whole graph, we need seven nodes using both the greedy algorithm and the SPIN algorithm; in particular, we find that both these algorithms choose {5, 4, 11, 15, 2, 7, 9} as the nodes to be initially targeted to cover the whole network; ii) if we wish to cover 2/3 of the network, i.e., 10 nodes, the greedy algorithm requires 3 nodes and it picks {5, 11, 2} as the initial target set. The SPIN algorithm also requires three nodes and it picks {5, 4, 11} as the initial target set.

III. Experimental Results With Submodular Influence Functions

Throughout this section, we consider the linear threshold model as the model of diffusion of information. Recall that the influence function $(\sigma(.))$ is submodular under the linear threshold model [28]. We evaluate the performance of the SPIN algorithm experimentally. For our experiments, we consider both synthetic data sets and real-world data sets.

All the experiments are executed on a desktop computer with: i) Intel(R) Pentium(R) 4 CPU (3.20 GHz speed) and 1 GB of RAM and ii) 32-bit Windows operating system. Each experimental result on each data set is taken as the average over 10 000 or 4000 repetitions depending on the type of data set. In other words, we consider that $R = 10\,000$ or $R = 4000$ depending on the data set. Further, we note that all the experiments are carried out using JAVA.

We first describe the data sets and the experimental setup. We then demonstrate the performance of the proposed algorithm on various data sets in comparison with the greedy algorithm (or the KKT algorithm), the LKG algorithm, MDH, and HCH. We now make a note regarding the implementation of the LKG algorithm. Recall (from Section I-C) that there are two aspects to the speed up of the LKG algorithm. Since the first aspect (i.e., sparsity) is not relevant to the context of the target set selection problem, we only leverage the second aspect in the implementation of the LKG algorithm.

A. Data Sets and Experimental Setup

1) Synthetic Data Sets: There are several popular models to generate synthetic data sets with specific structural characteristics. We consider the following two types of synthetic data sets.

Sparse Random Graphs: The basic random graph model [19] is defined by two parameters: the number of vertices (n) and common the edge probability $(0 < p < 1)$. The edge between any pair of nodes is created with probability p independently of any other edges. Clearly, the number of edges in such a graph is binomially distributed with mean $p\binom{n}{2}$. Graphs generated from this model are balanced with similar vertex degrees, low level of clustering, and relatively short distances. We efficiently generate sparse random graphs with 500 nodes using the algorithm presented by Batagelj and Brandes [4].

Scale-free Networks: Preferential attachment is a model proposed by Barabasi and Albert [3] for generating random graphs with heavy-tailed degree distribution. Consider a graph with n vertices. The n vertices of the graph are added one at a time, and for each of them, a fixed number of edges connecting to previously created vertices with probability proportional to their degree are added.

2) Real-World Social Network Datasets: We now present details about six real-world datasets that we have experimented with.

Celegans Data Set: This data set describes the neural network of the worm Caenorhabditis elegans and is used by Watts and Strogatz [49]. This is a weighted, directed, graph. There are 306 nodes in this dataset.

Jazz Data Set: This dataset consists of the list of links of the network of Jazz musicians [20]. There are 198 nodes in this network.

Netscience Data Set: This is a coauthorship network of scientists working on network theory, compiled by Newman [40] in May 2006. The vertices of the network represent authors of papers and edges join every pair of individuals whose names appear together as authors of a paper. This network has a total of 1589 scientists from a broad variety of fields in network theory.

NIPS coauthorship Data Set: This is an unweighted coauthorship network of NIPS (Conference on Neural Information Processing Systems) papers (Estevez *et al.* [41]). This network consists of 1061 nodes.

High-Energy Physics (HEP) Data Set: This is a weighted network of coauthorship between scientists posting preprints on the High-Energy Theory E-Print Archive between January 1, 1995 and December 31, 1999. This is compiled by Newman [39]. This network has 10 748 nodes.

Political Books Data Set: Nodes represent books about US politics sold by the online bookseller Amazon.com. Edges represent frequent co-purchasing of books by the same buyers, as indicated by the "customers who bought this book also bought these other books" feature on Amazon. The number of nodes in this data set is 105. This data set is compiled by V. Krebs and is available on Krebs' web site (http://www.orgnet.com/).

A summary of all the datasets described above is given in Table II.

3) Experimental Setup: We follow the linear threshold model of information diffusion. We determine the probabilities (or weights) on the edges with the help of the multiplicity of the edges between two nodes. If there are $l(x, y)$ multiple edges

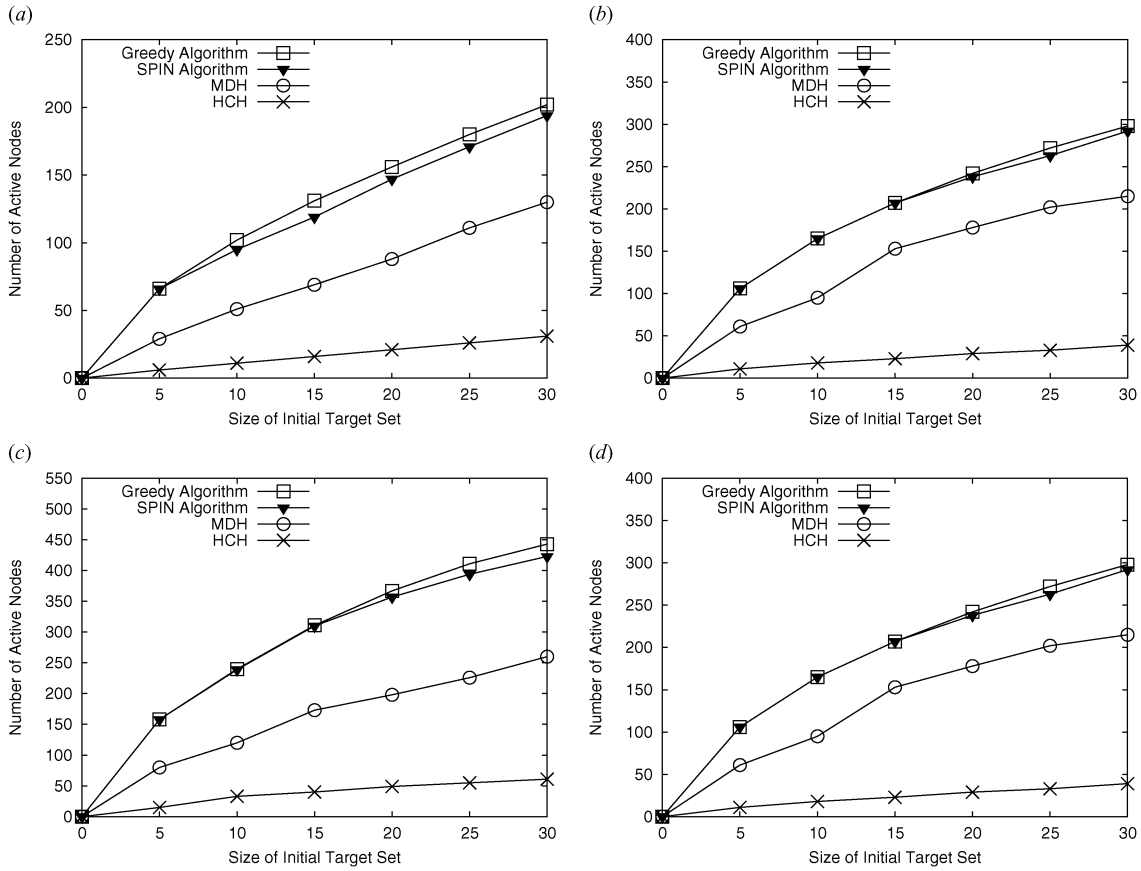

Fig. 2. Number of active nodes versus the size of the initial target set using: (a) sparse random graph with $p = 0.005$; (b) sparse random graph with $p = 0.01$; (c) sparse random graph with $p = 0.02$; and (d) scale-free graph.

TABLE II
SUMMARY OF THE DATASETS USED IN THE EXPERIMENTS

Data Set	Number of Nodes	Number of edges
Sparse Random Graph	500	5000 (approx.)
Scale-free Graph	500	1250 (approx.)
Political Books	105	441
Jazz	198	2742
Celegans	306	2345
NIPS	1061	4160
Netscience	1589	2742
HEP	10748	52992

between nodes x and y, and the degrees of the nodes are d_x and d_y, respectively, then the directed edge from x to y has weight $l(x,y)/d_y$ and the directed edge from y to x has weight $l(x,y)/d_x$.

We use the following convention while plotting the performance curves for various algorithms: the X axis represents the size of the initial target set, and Y axis represents the number of active nodes, that is, the number of influenced nodes at the end of the diffusion process.

B. Experimental Results and Analysis

Fig. 2 shows the performance of the SPIN algorithm *vis-a-vis* other algorithms on synthetic data sets consisting of 500 nodes each. In particular, Fig. 2(a) corresponds to a sparse random graph with $p = 0.005$, Fig. 2(b) corresponds to a sparse random graph with $p = 0.01$, Fig. 2(c) corresponds to a sparse random graph with $p = 0.02$, and Fig. 2(d) corresponds to a scale free graph. It is clear that the efficacy of the SPIN algorithm is good as that of the greedy algorithm (Algorithm 1). We note that the performance of MDH and HCH is quite poor.

The performance of various algorithms on the Celegans data set is shown in Fig. 3(a). Note that the performance of the SPIN algorithm inferior to that of the greedy algorithm by 4.9% whereas the performance of MDH is inferior by 14.3% to that of the greedy algorithm. HCH efficiency is much worse than that of the remaining three algorithms. The results of the greedy algorithm on this data set are computed by taking averages over 10 000 repetitions of the experiment.

For each possible size of the initial target set, the following is easy to observe from Fig. 3(b) and (c): the SPIN algorithm produces almost the same number of active nodes as that of the greedy algorithm. Note that Fig. 3(b) corresponds to the jazz musicians data set and Fig. 3(c) corresponds to the netscience coauthorship data set. The performance of MDH is inferior by about 37.5% and 53% to that of the greedy algorithm on the jazz data set and the netscience data set, respectively. HCH performs quite poorly on both the data sets. The results of the greedy algorithm on the jazz and netscience data sets are computed by taking averages over 10 000 repetitions of the respective experiments.

Fig. 3(d) shows the performance of the SPIN algorithm on the high-energy Physics coauthorship data set. The performance of the SPIN algorithm is inferior by about 4.4% to that of the

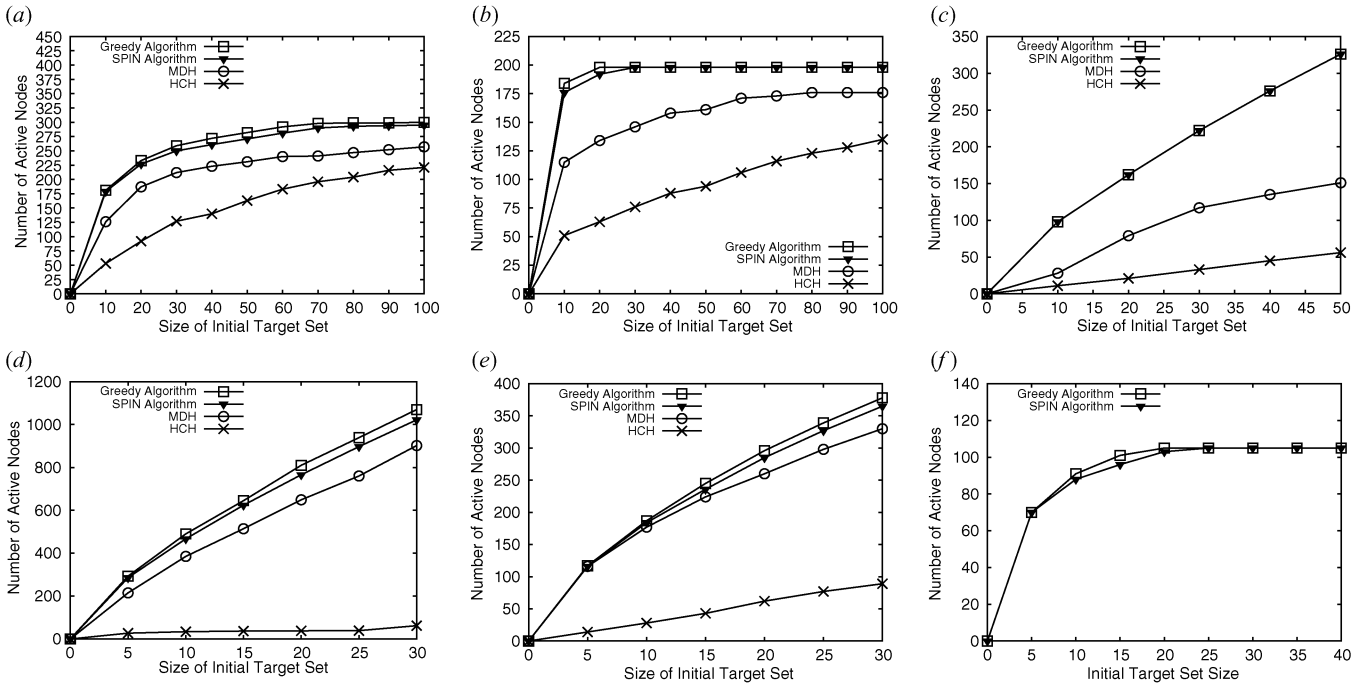

Fig. 3. Number of active nodes versus the size of the initial target set using: (a) Celegans data set; (b) Jazz musicians data set; (c) Netscience coauthorship data set; (d) High-energy physics coauthorship data set; (e) NIPS coauthorship data set; and (f) Political Books data set.

greedy algorithm and the performance of MDH is inferior by 15.7%. For the greedy algorithm, the number of active nodes for each size of the initial target sets is computed by taking average over 10 000 repetitions of experiment.

From Fig. 3(e), we see that the performance of the SPIN algorithm is inferior by 3.4% to that of the greedy algorithm on NIPS coauthorship data set. The efficacy of MDH is inferior by 12.6% to that of the greedy algorithm. In particular, the greedy algorithm chooses the following set of nodes as the top ten nodes: {479, 836, 376, 855, 68, 298, 331, 644, 440, 866}; the SPIN algorithm chooses the following set of nodes as the top ten nodes: {479, 836, 376, 855, 298, 68, 331, 459, 924, 852}; and MDH chooses the following set of nodes as the top ten nodes: {479, 836, 298, 331, 376, 206, 517, 644, 440, 855}. The experiment is repeated 4000 times in order to determine the average number of active nodes for each possible size of the target set using the greedy algorithm.

Fig. 3(f) shows the performance of the SPIN algorithm on the political books data set. The performance of the SPIN algorithm is inferior by about 1.9% to that of the greedy algorithm. For the greedy algorithm, the number of active nodes for each size of the initial target sets is computed by taking average over 10 000 repetitions of experiment.

Recall that the SPIN algorithm works with a sampled set $(\hat{\Omega})$q consisting of t number of permutations. The running time of the algorithm clearly depends on the value of t. Table III shows the values of t on various data sets using the SPIN algorithm.

From these experiments, we note that there exists a small performance gap in terms of the quality of the answer between the SPIN algorithm and the KKT algorithm. The following observations clarify this.

1) We work with approximate Shapley values of the nodes in order to rank them.

TABLE III
SIZE OF THE SAMPLED SET (t) USING THE SPIN ALGORITHM FOR VARIOUS DATA SETS

Data Set	t
Sparse Random Graph	1000
Scale-free Graph	700
Political Books	500
Jazz	800
Celegans	1000
NIPS	700
Netscience	1000
HEP	10000

2) We notice that the approximate Shapley values of the nodes follow a power-law-like distribution. That is, a few nodes have very high Shapley values and a large fraction of the nodes have very low Shapley values. Thus, to identify top-k nodes for small values of k, it is required to obtain the approximate Shapley values of nodes with high accuracy.

C. Computational Efficiency of the SPIN Algorithm

Here we explain certain implementation details of the proposed approach and the KKT and LKG algorithms. This highlights the reasons behind the computational efficiency of our approach.

1) The KKT Algorithm: The following is the procedure to find a new influential node using the KKT algorithm. *Initially, all nodes in the network are inactive and assume that A is the current set of influential nodes. Also, we randomly assign a threshold to each node. Then, we activate all the nodes in $A \cup \{n_i\}$, where $n_i \in N \backslash A$ and determine the number of nodes that are activated.* We repeat this step about R times (for example 10 000 times) and

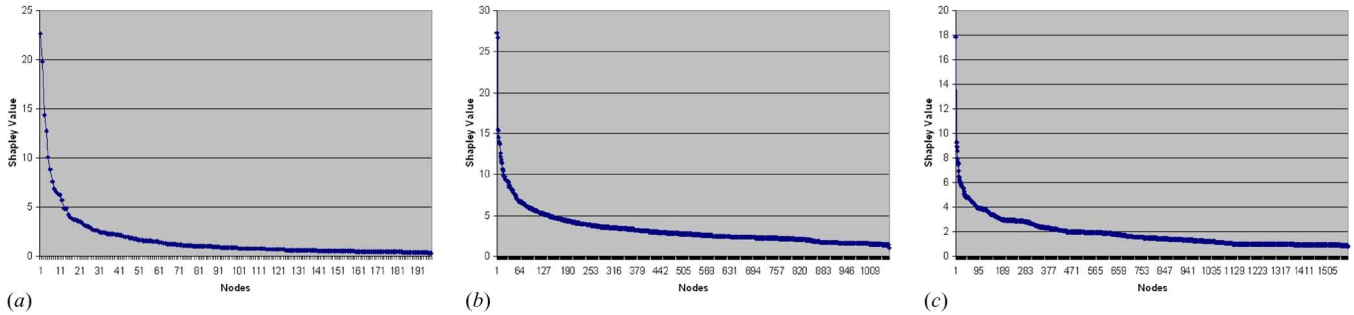

Fig. 4. Nodes ranked by the (approximate) Shapley values (or marginal contributions) tend to follow a heavy-tailed distribution. A few example data sets are: (a) Jazz data set; (b) NIPS coauthorship data set; and (c) Netscience coauthorship data set.

determine the expected number of nodes activated using $A \cup \{n_i\}$ as the initial target set (i.e., $\sigma(A \cup \{n_i\})$. We repeat the above procedure for all $n_i \in N \setminus A$ and we choose a node (as influential node) that maximizes $\sigma(A \cup \{n_i\}) - \sigma(A), \forall n_i \in N \setminus A$. That is, if our interest is to find k influential nodes, we have to repeat the above process k times (with appropriate A).

2) The LKG Algorithm: We implement this algorithm in the same fashion as the KKT algorithm except the fact that we take into account the submodularity property of the influence function. The submodularity of the influence function avoids the calculation of the (expected) value of the influence function with certain target sets.

3) The SPIN Approach: We implement our proposed approach as follows. As described previously, there are two steps in our approach: i) rank list generation and ii) choosing top-k nodes. We emphasize that the first step is completely independent of the second step. The implementation details of the first step (i.e., rank list generation) are clearly explained in Section II-B.

In fact, to make the algorithm even more efficient, we do not work with permutations of size n. We rather work with *perturbed permutations of size* q, where q is chosen appropriately based on the data set. We call a permutation as a perturbed permutation of size q if it is a sequence of q randomly selected nodes from the set of n nodes, where $q < n$. We randomly generate t perturbed permutations of size q to construct the rank list of nodes. Interestingly, we notice that the average contributions of the nodes using perturbed permutations of size q are almost same as that of using permutations on n nodes. The reason for this important observation is that the nodes ranked by the (approximate) Shapley values (or marginal contributions) tend to follow a *heavy-tailed* distribution. That is, a few nodes have high marginal contribution values and a large number of nodes have small marginal contribution values. For example, Fig. 4 shows the same for three real-world data sets. Furthermore, Table III shows the number of perturbed permutations of order q considered for various data sets. Table IV shows the value of q we have chosen to generate these perturbed permutations of size q for various data sets.

Next, we consider the second step in our proposed approach (i.e., choosing the top-k nodes). Given the value of k, we use Algorithm 3 to find out the set of k influential nodes. This step takes very short time (a few seconds even for data sets of size about 10 000 nodes) to execute. Thus, the execution time of the SPIN algorithm is (almost) independent of the value of k.

TABLE IV
THE VALUE OF q CONSIDERED IN GENERATING PERTURBED PERMUTATIONS OF SIZE q FOR VARIOUS DATA SETS

Data Set	n	q
Sparse Random Graph	500	200
Scale-free Graph	500	200
Political Books	105	70
Jazz	198	100
Celegans	306	100
NIPS	1061	150
Netscience	1589	200
HEP	10748	400

TABLE V
SPEEDUP OF THE SPIN ALGORITHM TO FIND TOP-30 NODES ON VARIOUS DATA SETS COMPARED TO THAT OF KKT ALGORITHM

Dataset	**Nodes**	**SPIN (Time in (MIN)**	**KKT (Time in) (MIN)**	**Speed-up of SPIN over KKT**
Random graph ($p = 0.005$)	500	13.9	824.93	59
Random graph ($p = 0.01$)	500	14.8	1123.16	75
Random graph ($p = 0.02$)	500	16.3	1302.46	79
Political Books	105	0.89	44.64	50
Jazz	198	1.1	366	332
Celegans	306	14.02	901	64
NIPS	1062	15.2	7201.54	473
Network-Science	1589	28.25	8539.48	302

Because of these reasons, the proposed approach is practically faster than the asymptotic running time given in Section II-B1. We note that the execution time of the KKT algorithm depends quite crucially on the value of k. Similarly, the running time of the LKG algorithm also depends on the value k.

The above implementation details of the KKT algorithm and our proposed approach clearly bring out the differences in the computation of the influence function. We notice that this leads to significant difference in the running time of these algorithms on various data sets as shown below.

Table V shows the running times of the SPIN algorithm and the KKT algorithm (Greedy algorithm) to find top 30 influential nodes on various data sets that we considered above in our experiments. This table shows that the speedup of the SPIN algorithm over the KKT algorithm is quite significant and even achieves a speed up of about 470 times on the netscience data

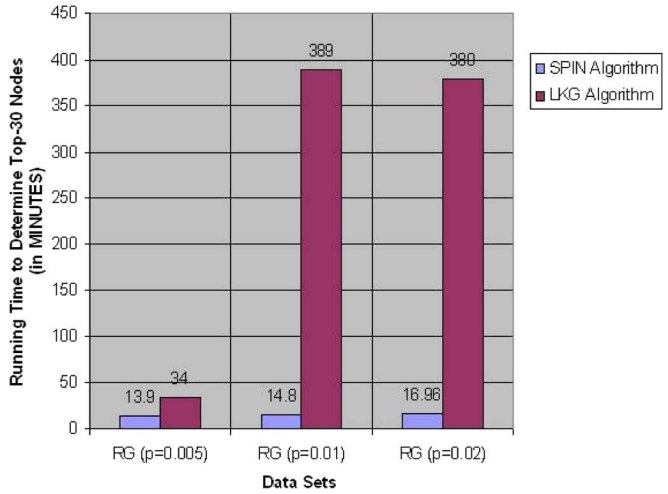

Fig. 5. Running times of the SPIN algorithm and the LKG algorithm to identify the top-30 influential nodes on three data sets, namely: i) sparse random graph with $p = 0.005$ and ii) sparse random graph with $p = 0.01$ (i) sparse random graph with $p = 0.02$.

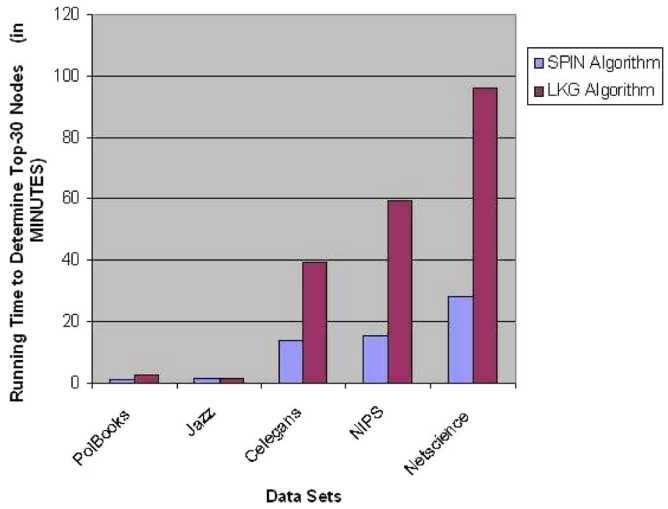

Fig. 6. Running times of the SPIN algorithm and the LKG algorithm to identify the top-30 influential nodes on various real-world data sets.

set. Since the KKT algorithm runs very slow on the large data sets, we did not run the KKT algorithm on the HEP data set.

We also observe that the SPIN algorithm is efficient than the LKG algorithm as well and the following experimental results reveal this fact. Fig. 5 shows the running times of the SPIN algorithm and the LKG algorithm to find top 30 influential nodes on three sparse random graphs with $p = 0.005$, $p = 0.01$, and $p = 0.02$, respectively. Similarly, Fig. 6 shows the running times of the SPIN algorithm and the LKG algorithm to find top 30 influential nodes on various real-world data sets (except HEP data set) that we considered previously. In the case of the HEP data set, to find the top 30 influential nodes, the SPIN algorithm takes 887 min, whereas the LKG algorithm takes 1730 min.

To exemplify the fact that the running time of our proposed approach does not depend on the value of k, we report experiments on two data sets. Table VI shows that the speed up of the SPIN algorithm (on the Celegans data set) increases significantly compared to that of the KKT algorithm as the value of k increases. Table VII shows clearly that the speed up of the

TABLE VI
RUNNING TIMES OF THE SPIN, KKT, AND LKG ALGORITHMS ON THE DATA SET ($n = 306$) TO DETERMINE TOP-k NODES WHERE $k = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$ AND THE SPEED UP OF THE SPIN ALGORITHM OVER THE KKT ALGORITHM

Top-k Nodes	Running Time (in MINUTES)			Speed-up of SPIN over KKT
	SPIN Algorithm	KKT Algorithm	LKG Algorithm	
$k = 10$	14	236	13.97	16
$k = 20$	14.01	572	28.26	40
$k = 30$	14.02	901	32.55	64
$k = 40$	14.04	1192.68	56.18	85
$k = 50$	14.07	1479.65	62.33	105
$k = 60$	14.09	1757.49	63.86	125
$k = 70$	14.10	2027.37	65.95	144
$k = 80$	14.11	2287.93	67.64	163
$k = 90$	14.12	2535.76	68.76	181
$k = 100$	14.13	2773.07	69.83	198

TABLE VII
RUNNING TIMES OF THE SPIN, KKT, AND LKG ALGORITHMS ON THE NETSCIENCE DATA SET ($n = 1589$) TO DETERMINE TOP-k NODES WHERE $k = 10, 20, 30, 40, 50$ AND THE SPEED UP OF THE SPIN ALGORITHM OVER THE KKT ALGORITHM

Top-k Nodes	Running Time (in MINUTES)			Speed-up of SPIN over KKT
	SPIN Algorithm	KKT Algorithm	LKG Algorithm	
$k = 10$	28.04	1341.29	77.07	47
$k = 20$	28.09	4297.02	79.75	152
$k = 30$	28.13	8539.48	85.04	302
$k = 40$	28.18	13949.9	90.33	493
$k = 50$	28.25	20411.1	99.03	722

TABLE VIII
NUMBER OF NODES REQUIRED TO INFLUENCE A GIVEN PERCENTAGE λ OF NODES USING JAZZ DATA SET

Algorithm	Jazz		
	$\lambda = 25\%$	$\lambda = 50\%$	$\lambda = 75\%$
Greedy Algorithm	2	3	5
SPIN Algorithm	2	3	5

TABLE IX
NUMBER OF NODES REQUIRED TO INFLUENCE A GIVEN PERCENTAGE λ OF NODES USING CELEGANS DATA SET

Algorithm	Celegans		
	$\lambda = 25\%$	$\lambda = 50\%$	$\lambda = 75\%$
Greedy Algorithm	3	8	20
SPIN Algorithm	3	8	24

SPIN algorithm (on the netscience data set) increases significantly compared to that of the KKT algorithm as k value increases. We note that the SPIN algorithm obtains a speed up of about 700 times, when $k = 50$, over the KKT algorithm on the netscience data set.

D. Experimental Results for the λ-Coverage Problem

Here, we compare the performance of the SPIN algorithm for the λ-coverage problem with that of the greedy algorithm. Since the greedy algorithm takes several hours of time to run on large data sets [10], we consider two moderate size real-world data sets, namely, Jazz data set and Celegans data set (refer to Section III-A for more details about the data sets). Tables VIII and IX show the number of nodes required to cover a given percentage λ of the nodes in the network using the greedy algorithm

and the SPIN algorithm on the Jazz data set and the Celegans data set, respectively. In our experiments, we consider that λ takes values 25%, 50%, and 75%.

E. Comparison of SPIN With Other Algorithms

The SPIN algorithm proposed in this paper follows a fundamentally new approach to solve the target selection problem. It has several advantages and a few limitations when compared to the algorithms existing in the literature. We first discuss the advantages followed by the limitations.

1) Advantages of the SPIN Algorithm: The proposed SPIN algorithm produces target sets which are as good as the target sets produced by the greedy algorithm of Kempe *et al.* [28] (KKT algorithm) and the more recent, faster versions of Leskovec *et al.* [34] (LKG algorithm), and Chen *et al.* [10] (CWY algorithm). The SPIN algorithm is in general more computationally efficient than the LKG algorithm and hence much more efficient than the KKT algorithm.

It is known that the objective function $\sigma(.)$ is monotone increasing and submodular whenever the node threshold function at each node is monotone increasing and submodular [35]. Thus the approximation guarantee of $(1-(1/e))$ holds when the node threshold function at each node is monotone increasing and submodular. However, in many settings, it is quite possible that the node threshold functions are monotone decreasing and nonsubmodular and in such contexts none of the algorithms have approximation guarantees. In two such specific settings as shown in Section IV, it is experimentally verified that the SPIN algorithm outperforms in terms of quality of the solution over the KKT algorithm (as well as the LKG algorithm and the CWY algorithm).

The other advantages of the SPIN algorithm over the KKT algorithm are: a) SPIN can solve both the top-k nodes problem and the λ-coverage problem equally effectively while the KKT algorithm runs slower on both these problems and b) the execution time of the SPIN algorithm is (almost) independent of the value of k, whereas the execution time of the KKT, LKG, and CWY algorithms heavily depends on the value of k.

The maximum degree heuristic (MDH)-based and the high clustering heuristic (HCH)-based algorithms are much faster compared to the KKT, LKG, and CWY algorithms or the SPIN algorithm, however their performance in terms of the target sets produced tends to be quite poor. This is strikingly observed in all the experiments. This is a serious limitation of the MDH and HCH algorithms.

2) Limitations of the SPIN Algorithm: The target sets produced by the SPIN algorithm are occasionally a little inferior to the ones produced by the KKT, LKG, and CWY algorithms when the $\sigma(.)$ function is submodular. This happens because the Shapley values computed are approximate.

F. Statistical Significance of Experimental Results

We evaluate the quality of the approximate Shapley values of the nodes that are computed using Algorithm 2. For the experiments in this section, we use the high-energy physics data set as it is a large data set.

We rely on statistical techniques for bounding the corresponding errors [47], [32]. Recall from Algorithm 2 that $\hat{\Omega}$ is a set of randomly sampled permutations. Let $\{X_i^1, X_i^2, \ldots, X_i^t\}$ be the random sample representing the marginal contributions of a specific node i due to the permutations in $\hat{\Omega}$. Following standard methods of statistical inference, let us define an estimator $\bar{X}_i$ for the actual Shapley value of the node i as $\bar{X}_i = (\sum_{j=1}^{t} X_i^j/t)$. If we directly use $\bar{X}_i$ as an estimator of the actual Shapley value, the problem is that we do not know how close $\bar{X}_i$ is to the original Shapley value. In this process, we first need to determine the variance of the random variable $\bar{X}_i$ given by $Var(\bar{X}_i) = ((\sum_{j=1}^{t} \{X_i^j - \bar{X}_i\}^2)/(t(t-1)))$. Table X gives the estimated Shapley values and the corresponding variances of the top ten nodes in the high-energy physics data set where the sampled set size $t = 60$.

TABLE X
ESTIMATED SHAPLEY VALUE AND VARIANCE OF TOP TEN NODES IN THE HIGH-ENERGY PHYSICS DATA SET WHEN SAMPLED SET SIZE $t = 60$

Node ID	$\bar{X}_i$	$Var(\bar{X}_i)$
1	62.23	7.32
5	57.10	5.88
2	56.18	2.01
3	55.36	4.24
10	51.50	3.13
6	47.23	3.77
4	46.83	1.95
14	46.35	2.32
26	43.01	2.44
7	40.43	3.83

Now, we need to assess the quality of the estimator $\bar{X}_i$. For this, we construct the confidence interval, which is an open interval on the real line such that we have a given level of confidence that it contains the actual Shapley value. More precisely, if we can ascertain that the estimator $\bar{X}$ satisfies the condition

$$P(\bar{X}_i - \delta < \Phi_i < \bar{X}_i + \delta) = \gamma$$

then, it is possible to say that the probability is γ (or the confidence is γ) that the interval $(\bar{X}_i - \delta, \bar{X}_i + \delta)$ will contain the actual Shapley value (Φ_i). Using the standard results in this theory [47], [32], we construct $(1-\alpha)100$ percent confidence intervals of the form $(\bar{X}_i - z_{\alpha/2}[s/t^{0.5}], \bar{X}_i + z_{\alpha/2}[s/t^{0.5}])$. Here, s stands for the standard deviation of the random sample and z is the standard normal random variable. Since we do not know the actual standard deviation of the underlying distribution, we use the standard deviation (s) of the random sample. This method gives good approximate confidence intervals for large values of t $(t > 30)$. If $\alpha = 0.01$, then we get a 99% confidence interval. Similarly, if $\alpha = 0.05$, then we obtain a 95% confidence interval. We consider the first ten nodes in the high-energy physics data set and for each node we consider a random sample of size $t = 60$ and determine the 99%, 95%, and 90% confidence intervals for their corresponding actual Shapley values. Table XI shows these results.

From the definition of confidence interval, we have $100(1-\alpha)$ percent confidence that the estimator $\bar{X}_i$ deviates from the actual Shapley value by less than $z_{\alpha/2}(s/n^{0.5})$ and call this error by e. Then, the sample size required in order to produce a symmetrical $100(1-\alpha)$ percent confidence interval of width

TABLE XI
VARIOUS CONFIDENCE INTERVALS FOR TOP TEN NODES IN THE HIGH-ENERGY PHYSICS DATA SET

Node ID	90 Percent Interval	95 Percent Interval	99 Percent Interval
1	(57.78,66.68)	(56.93,67.53)	(55.28,69.18)
3	(53.12,61.08)	(52.36,61.84)	(50.87,63.33)
2	(53.85,58.52)	(53.42,58.94)	(52.56,59.80)
3	(51.98,58.74)	(51.35,59.37)	(50.58,60.64)
10	(48.59,54.4)	(48.06,54.94)	(46.65,56.35)
6	(44.04,50.42)	(43.43,51.03)	(42.24,52.22)
4	(44.54,49.12)	(44.11,49.55)	(43.25,50.41)
14	(43.85,48.85)	(43.38,49.32)	(42.44,50.26)
26	(40.44,45.57)	(39.95,46.06)	(38.99,47.02)
7	(37.22,43.64)	(36.61,44.25)	(35.43,45.23)

TABLE XII
REQUIRED CARDINALITY OF THE SAMPLE SIZE t TO CONSTRUCT 99% AND 95% CONFIDENCE INTERVALS WITH ERROR $e = 0.05$ AND $e = 1$ FOR TOP TEN NODES IN THE HIGH-ENERGY PHYSICS DATA SET

Node ID	99% Interval with $e = 0.05$	99% Interval with $e = 1$	95% Interval with $e = 0.05$	95% Interval with $e = 1$
1	11602	2901	6718	1679
5	10865	2717	6294	1573
2	3745	936	2168	542
3	6465	1616	3744	935
10	4603	1150	2664	667
6	5587	1396	3234	809
4	3072	768	1778	445
14	3614	903	2092	523
26	3352	838	1941	485
7	5750	1437	3329	832

$2e$ (i.e., $(\bar{X}_i - e, \bar{X}_i + e)$) for the actual Shapley value is given by

$$t \approx \left(\frac{z_{\alpha/2}s}{e}\right)^2$$

Table XII shows the required cardinality of the sample size t to construct 99% and 95% confidence intervals with error $e = 0.05$ and $e = 1$ for the top ten nodes in the high-energy physics data set. Clearly, the respective sizes of the cardinality decrease when we go to 95% confidence interval from 99% confidence interval.

From the above results, it is clear that we can approximate the marginal contributions of the nodes even with a small or moderate size of the sampled set using the subroutine in Algorithm 2. There is a clear reason for this phenomenon. The Shapley values of the nodes for most real-world networks tend to follow a somewhat power law like distribution. That means, only a small fraction of the nodes have high marginal contribution values and a large fraction of the nodes have very small marginal contribution values.

This observation helps us in the following way. If a node is really a candidate node for the diffusion process, its marginal contribution value is high even if it occurs in a small number of randomly sampled set of permutations. On the other hand, if a node is not a candidate node for the diffusion, its marginal contribution value is low even if it occurs in a large number of permutations in the sampled set. In our experimentation, the sample size chosen guarantees a 99% confidence interval.

IV. EXPERIMENTAL RESULTS WITH NONSUBMODULAR NODE THRESHOLD FUNCTIONS

In this section, we demonstrate the efficacy of the SPIN algorithm when the threshold function at each node is monotone decreasing and nonsubmodular. Towards this end, we progress as follows: a) we first propose two new models of diffusion of information which we call *multiplication threshold model* and *minimum threshold model*; b) we then show that the threshold function at each node is monotone decreasing and nonsubmodular for each of these two new threshold models; and c) we then experimentally show the superior performance of the SPIN algorithm (in terms of the quality of the solution) compared to that of the KKT algorithm.

A. Multiplication Threshold Model

The process of diffusion in this model is described as:

- Each node i initially chooses a threshold θ_i uniformly at randomfrom the interval [0,1].
- Node i becomes active in time step t if $f_i(S) \geq \theta_i$, where $S \subseteq N_i$ is the set of active neighbors of i in step $(t-1)$.
- We define the threshold function f_i as follows:

$$f_i(S) = \prod_{j \in S} w_{ij} \tag{1}$$

where w_{ij} is the normalized weight representing the level of influence of node j on node i such that $\sum_{j \in N_i} w_{ij} \leq 1$.

We now describe an example where the multiplication threshold model can be applied. Consider a social network consisting of individuals. Call a node active if it has seen the movie and inactive if it has not seen the movie. Suppose node i is currently inactive. We call a neighbor of node i active if it already saw the movie. An active neighbor j of i attempts to make node i disinterested in the movie. The number θ_i could be viewed as the threshold beyond which the node i becomes active. The joint influence of the active neighbors could be modeled as the product of the w_{ij} values—this is because as more and more neighbors dissuade player i, the movie (or the product) becomes lower and lower valued and player i remains inactive. Assume that $S \subseteq N_i$ is the set of active neighbors of node i. We can capture this by saying that node i changes from inactive to active if $\Pi_{j \in S} w_{ij} \geq \theta_i$ and continues to be inactive, otherwise.

In what follows, we show that the node threshold function f_i for each $i \in N$ is monotone decreasing and nonsubmodular. In the following lemma, we in fact show that the function f_i is monotone decreasing and supermodular.

Lemma 1: Given the multiplication threshold model and any node $i \in N$, the threshold function f_i is monotone decreasing and supermodular.

Proof: Consider the threshold function f_i of node i defined as in the expression (1). It is immediate to see that f_i is monotone decreasing because $\prod_{j \in T} w_{ij} \leq \prod_{j \in S} w_{ij}$ whenever $S \subseteq T \subseteq N_i$. This is because $0 < w_{ij} \leq 1, \forall j \in N_i$.

We now show that f_i is supermodular. Let S and T be any two arbitrary subsets of N_i such that $S \subseteq T \subseteq N_i$. It is clear that

$$\prod_{j \in S} w_{ij} \geq \prod_{j \in T} w_{ij}. \tag{2}$$

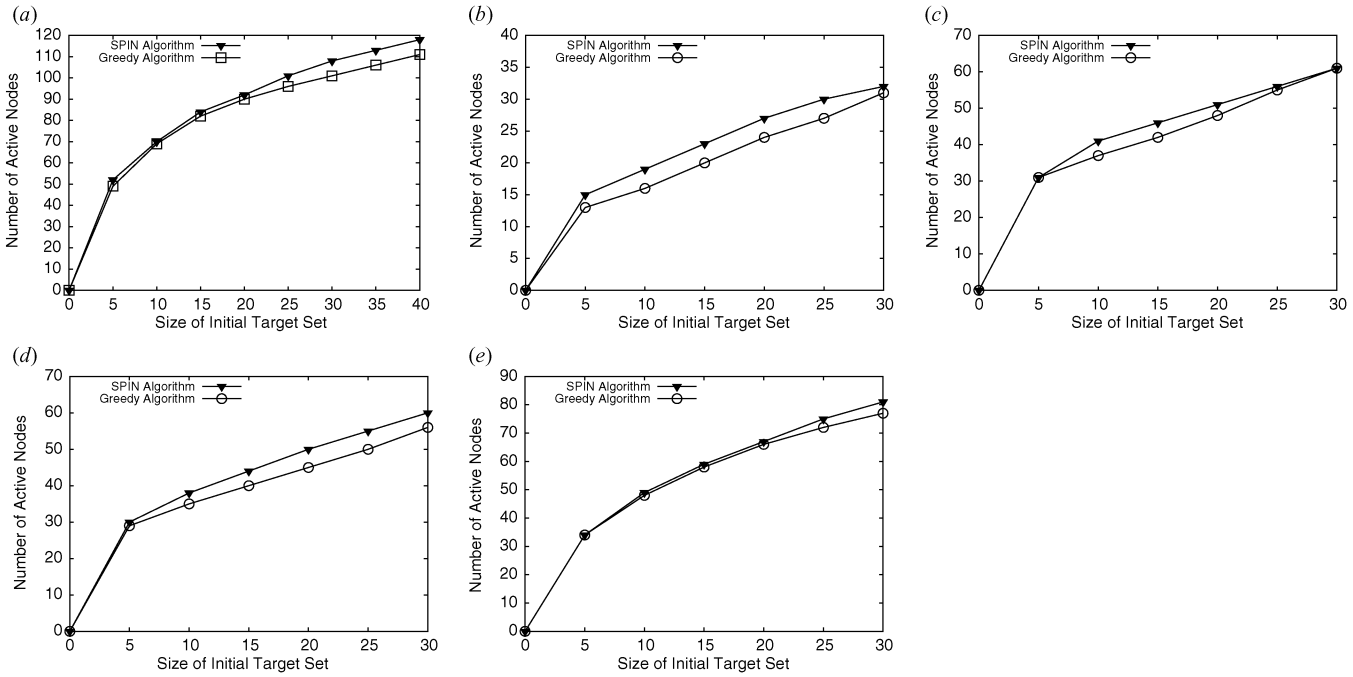

Fig. 7. Number of active nodes versus the size of the initial target set using multiplication threshold model with: (a) sparse random graph with $p = 0.01$ (nodes $n = 500$); (b) Karate data set (nodes $n = 34$); (c) Political Books data set (nodes $n = 105$); (d) Adjacency of nouns data set (nodes $n = 112$); and (e) Celegans data set (nodes $n = 306$).

Now, consider $z \in N_i$ such that $z \notin S$ and $z \notin T$. Also, since $(w_{iz} - 1) < 0$, we now get from expression (2) that

$$(w_{iz} - 1) \prod_{j \in S} w_{ij} \leq (w_{iz} - 1) \prod_{j \in T} w_{ij} \tag{3}$$

$$\Rightarrow \prod_{j \in S \cup \{z\}} w_{ij} - \prod_{j \in S} w_{ij} \leq \prod_{j \in T \cup \{z\}} w_{ij} - \prod_{j \in T} w_{ij} \tag{4}$$

$$\Rightarrow f_i(S \cup \{z\}) - f_i(S) \leq f_i(T \cup \{z\}) - f_i(T). \tag{5}$$

Hence, f_i is supermodular. ■

The performance of the SPIN algorithm using these monotone decreasing and nonsubmodular node threshold functions on various data sets is shown in Fig. 7. For these experiments, we use five data sets, namely: a) sparse random graph data set $(p = 0.01)$; b) karate data set; c) political books data set; d) adjacency nouns data set; and e) celegans data set. The last four data sets here are real-world data sets and they are taken from the home page of MEJ Newman (http://www-personal.umich.edu/~mejn/netdata/). From these experimental results, it is clear that the SPIN algorithm outperforms in terms of the quality of the solution over the KKT algorithm, when the node threshold functions are nonsubmodular.

B. Minimum Threshold Model

Here, our approach is similar to what we did in the case of multiplication threshold model. To start with, we first define the minimum threshold model as follows. The process of diffusion in this model is described as:

- Each node i initially chooses a threshold θ_i uniformly at random from the interval [0,1].
- Node i becomes active in time step t if $f_i(S) \geq \theta_i$, where $S \subseteq N_i$ is the set of active neighbors of i in step $(t-1)$.
- We define the threshold function f_i as follows:

$$f_i(S) = \min_{j \in S} \{\alpha_j w_{ij}\} \tag{6}$$

where $\alpha_j \geq 0, \forall j \in N_i$.

We present an example setting where this model is applicable. Consider the example of a node i to which the same message is being communicated by each of its active neighbors. Assume that the communication channels are noisy. If j is an active neighbor of i, the number w_{ij} gives the normalized value of the reliability of the channel between j and i. The number w_{ij} could be viewed as indicating a normalized probability of the message being transmitted in an error free way from j to i. The number θ_i could be viewed as the threshold for accepting a message as error free. Assume that $S \subseteq N_i$ is the set of active neighbors of node i. Under this setting, player i will consider the message received as error free only if $\min_{j \in S} w_{ij} \geq \theta_i$, which means the threshold function is the minimum function.

We now show that the node threshold function f_i for each $i \in N$ is monotone decreasing and nonsubmodular. In the following lemma, we in fact show that f_i is monotone decreasing and supermodular.

Lemma 2: Given the minimum linear threshold model and for any node i, the threshold function f_i is monotone decreasing and supermodular.

Proof: Given that the minimum linear threshold model and a node i. Now, consider the threshold function f_i of node i as defined in expression (6). It is immediate to see that f_i is monotone decreasing as $\min_{j \in T} \{\alpha_j w_{ij}\} \leq \min_{j \in S} \{\alpha_j w_{ij}\}$ whenever $S \subseteq T \subseteq N_i$.

It is not difficult to see that f_i is supermodular using the standard results from the literature. However, here we present

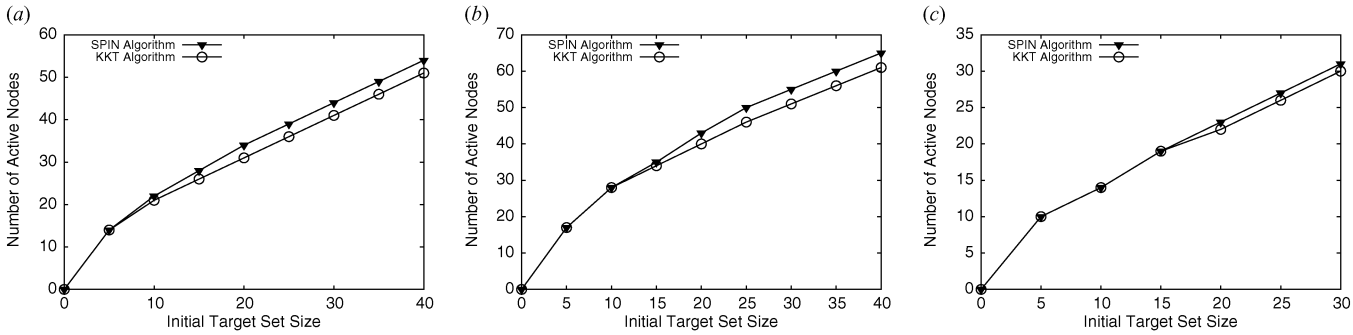

Fig. 8. Number of active nodes versus the size of the initial target set using minimum linear threshold model with: (a) adjacency of nouns dataset; (b) Celegans dataset; and (c) Karate dataset.

a proof for completeness. Let N_i be the set of neighbors of node i. Let S and T be any two arbitrary subsets of N_i such that $S \subseteq T \subseteq N_i$. Consider $z \in N_i$ such that $z \notin S$ and $z \notin T$. Let $x_1 = \min_{j \in S}\{\alpha_j w_{ij}\}$, $x_2 = \min_{j \in S \cup \{z\}}\{\alpha_j w_{ij}\}$, $x_3 = \min_{j \in T}\{\alpha_j w_{ij}\}$ and $x_4 = \min_{j \in T \cup \{z\}}\{\alpha_j w_{ij}\}$. It is clear that $x_1 \geq x_3$ and $x_2 \geq x_4$.

We now consider the following the three cases.

- *Case 1*: If $x_1 = x_2$, then we get that $x_3 = x_4$. This leads to the following:

$$f(S \cup \{z\}) - f(S) = f(T \cup \{z\}) - f(T).$$

- *Case 2*: If $x_3 \leq x_2 < x_1$, then it is clear that $x_3 = x_4$. This leads to the following:

$$f(S \cup \{z\}) - f(S) < f(T \cup \{z\}) - f(T).$$

- *Case 3*: If $x_2 < x_3$, then it is clear that $x_2 = x_4 < x_3 \leq x_1$. This leads to the following:

$$f(S \cup \{z\}) - f(S) \leq f(T \cup \{z\}) - f(T).$$

Hence, f_i is supermodular. ■

The performance of the SPIN algorithm using these monotone decreasing and nonsubmodular node threshold functions on various data sets is shown in Fig. 8. For these experiments, we use three data sets, namely: a) adjacency nouns data set; b) celegans data set; and c) karate data set. All these data sets are real-world data sets and they are taken from the home page of MEJ Newman. From these experimental results, it is clear that the SPIN algorithm outperforms in terms of the quality of the solution over the KKT algorithm, when we use the above nonsubmodular influence function.

V. Conclusion and Future Work

In this paper, we considered the target set selection problem where the objective is to find influential nodes to perform the information diffusion task. In particular, we focused on two variants of this problem namely the top-k nodes problem and the λ-coverage problem. We have proposed an efficient heuristic algorithm which we called the SPIN algorithm to address these problems. Our approach uses the novel idea of modeling the information diffusion process as a cooperative game and using the Shapley values of the nodes to compute their network value or influence in the network. We have experimented with four synthetic data sets and six real-world data sets and shown that the proposed algorithm outperforms the greedy algorithm and the LKG algorithm in terms of the running time.

The following are a few interesting and potential directions to enhance the performance of the SPIN algorithm.

- Currently, the SPIN algorithm sieves only the single-hop neighbors in the neighborhood, while selecting the top-k nodes. A more intelligent way of sieving the nodes in the immediate neighborhood will improve the performance of the SPIN algorithm.
- Performance improvement can be obtained by focusing on the bridge nodes (or cut vertices) in the network since such nodes enable the communication between the nodes belonging to different groups (or components). Our current approach may not effectively capture the bridging roles that some nodes play in the network.
- The SPIN algorithm for the top-k nodes problem assumes that the probabilities with which nodes influence their neighbors are computed from the structure of the network directly. Such an approach is often not practical because the nodes in the social network are individual entities or organizations and they behave strategically with self-interest. Hence, the probabilities with which a node influences its neighbors depend not only on the structure of the network but also on the private information that the node may have about its neighbors. Such scenarios can be effectively modeled using mechanism design.
- The existing approaches including the one in this paper do not exactly exploit the possible presence of communities. A community is a group of nodes that have a high number of connections to the nodes within the group and much less number of connections to the nodes outside the group. The presence of communities strongly affects the process of identifying influential nodes. It would be interesting to enhance existing algorithms to exploit the presence of communities in a more effective way.
- We have assumed the linear threshold model as the model of information diffusion. The proposed algorithms will work with the independent cascade model also. Experiments could be carried out with the latter model.
- In cooperative game theory, there are other solution concepts such as nucleolus which possibly have important im-

plications for discovering influential nodes. This would be another interesting direction for future work.

- We wish to mention that there is a generalization of the Shapley value for graph theoretic settings. This is the Myerson Value [37]. What we have implicitly employed in this paper is conceptually the same as the Myerson value. It would be interesting to formalize this.

ACKNOWLEDGMENT

The author Y. Narahari wishes to acknowledge the support of GM R&D India Science Laboratory for his research work.

REFERENCES

[1] E. Adar and L. A. Adamic, "Tracking information epidemics in Blogspace," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intelligence (WI)*, 2005, pp. 207–214.

[2] N. Agarwal, H. Liu, L. Tang, and P. S. Yu, "Identifying influential bloggers in a community," in *Proc. 1st ACM Int. Conf. Web Search and Data Mining (WSDM)*, 2008, pp. 207–217.

[3] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.

[4] V. Batagelj and U. Brandes, "Efficient generation of large random networks," *Phys. Rev. E*, vol. 71, p. 036113, 2005.

[5] O. Ben-Zwi, D. Hermelin, D. Lokshtanov, and I. Newman, "An exact almost optimal algorithm for target set selection in social networks," in *Proc. 10th ACM Conf. Electron. Commerce (EC)*, 2009.

[6] , U. Brandes and T. Erlebach, Eds., *Network Analysis*. Berlin, Germany: Springer-Verlag, 2005, vol. 3418, Lecture Notes in Computer Science.

[7] R. Bryce and N. C. Gross, "The diffusion of hybrid seed corn in two Iowa communities," *Rural Sociology*, vol. 8, no. 1, pp. 15–24, 1943.

[8] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the Flickr social network," in *Proc. 18th Int. Conf. World Wide Web (WWW)*, 2009, pp. 721–730.

[9] N. Chen, "On the approximability of influence in social networks," in *Proc. 19th Annual ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2008, pp. 1029–1037.

[10] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining (KDD)*, 2009, pp. 937–944.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[12] X. Deng and C. H. Papadimitriou, "On the complexity of cooperative solution concepts," *Math. Oper. Res.*, vol. 19, no. 2, pp. 257–266, 1994.

[13] Z. Dezso and A.-L. Barabasi, "Halting viruses in scale-free networks," *Phys. Rev. E*, vol. 65, p. 055103, 2002.

[14] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining (KDD)*, 2001, pp. 57–66.

[15] P. A. Estevez, P. Vera, and K. Saito, "Selecting the most influential nodes in social networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, 2007, pp. 2397–2402.

[16] E. Even-Dar and A. Shapira, "A note on maximizing the spread of influence in social networks," in *Proc. 3rd Workshop on Internet and Network Economics (WINE)*, 2007, pp. 281–286.

[17] L. C. Freeman, "Centrality in networks: I. Conceptual clarification," *Social Networks*, vol. 1, pp. 215–239, 1979.

[18] A. Ganesh, L. Massouli, and D. Towsley, "The effect of network topology on the spread of epidemics," in *Proc. IEEE INFOCOM*, 2005, pp. 1455–1466.

[19] E. N. Gilbert, "Random graphs," *Ann. Math. Stat.*, vol. 30, no. 4, pp. 1141–1144, 1959.

[20] P. M. Gleiser and L. Danon, "Community structure in Jazz," *Advanced Complex Systems*, vol. 6, pp. 565–573, 2003.

[21] J. Goldenberg, B. Libai, and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth," *Marketing Lett.*, vol. 12, no. 3, pp. 211–223, 2001.

[22] J. Goldenberg, B. Libai, and E. Muller, "Using complex systems analysis to advance marketing theory developments," *Acad. Market. Sci. Rev.*, vol. 1, no. 9, 2001.

[23] M. Granovetter, "The strength of weak ties," *Amer. J. Sociol.*, vol. 78, pp. 1360–1380, 1973.

[24] M. Granovetter, "Threshold models of collective behavior," *Amer. J. Sociol.*, vol. 83, pp. 1420–1443, 1978.

[25] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through Blogspace," in *Proc. 13th Int. Conf. World Wide Web (WWW)*, 2004, pp. 491–501.

[26] R. A. Holley and T. M. Liggett, "Ergodic theorems for weakly interacting infinite systems and the voter model," *Ann. Probability*, vol. 3, pp. 643–663, 1975.

[27] A. Java, P. Kolari, T. Finin, and T. Oates, "Modeling the spread of influence of the blogosphere," in *Proc. 15th Int. Conf. World Wide Web (WWW)*, 2006.

[28] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2003, pp. 137–146.

[29] D. Kempe, J. Kleinberg, and E. Tardos, "Influential nodes in a diffusion model for social networks," in *Proc. 32nd Int. Colloquium on Automata, Languages and Programming (ICALP)*, 2005, pp. 1127–1138.

[30] M. Kimura and K. Saito, "Tractable models for information diffusion in social networks," in *Proc. 10th Eur. Conf. Principles and Practice of Knowl. Discovery in Databases (PKDD)*, 2006, pp. 259–271.

[31] J. Kleinberg, "Cascading behavior in networks: Algorithmic and economic issues," in *Algorithmic Game Theory*, N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2007, pp. 613–632.

[32] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*. New York: McGraw-Hill, 2000.

[33] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Trans. Web (TWEB)*, vol. 1, no. 1, 2007, Article 5, 39 Pages.

[34] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining (KDD)*, 2007, pp. 420–429.

[35] E. Mossel and S. Roch, "On the submodularity of influence in social networks," in *Proc. 39th Annu. ACM Symp. Theory Comput. (STOC)*, 2007, pp. 128–134.

[36] R. B. Myerson, *Game Theory: Analysis of Conflict*. Cambridge, MA: Harvard Univ. Press, 1997.

[37] R. B. Myerson, "Graphs and cooperation in games," *Math. Oper. Res.*, vol. 2, no. 3, pp. 225–229, 1977.

[38] M. E. J. Newman, "The structure of scientific collaboration networks," *Proc. Nat. Acad. Sci. (PNAS)*, vol. 98, pp. 404–409, 2001.

[39] M. E. J. Newman, "The structure of scientific collaboration networks," *Proc. Nat. Acad. Sci. (PNAS), USA*, vol. 98, pp. 404–409, 2001.

[40] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, p. 036104, 2006.

[41] P. A. Estevez, P. Vera, and K. Saito, "Selecting the most influential nodes in social networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, 2007, pp. 2397–2402.

[42] R. Pastor-Satorras and A. Vespignani, "Epidemics and immunization in scale-free networks," in *Handbook of Graphs and Networks: From the Genome to the Internet*, S. Bornholdt and H. G. Schuster, Eds. New York: Wiley, 2003, pp. 111–130.

[43] E. M. Rogers, *Diffusion of Innovations*. New York: Free Press, 1962.

[44] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining (KDD)*, 2002, pp. 61–70.

[45] T. C. Schelling, *Micromotives and Macrobehavior*. New York: Norton and Company, 1978.

[46] L. S. Shapley, "A value for N-person games," in *Contributions to the Theory of Games, Volume I*, H. W. Kuhn and A. W. Tucker, Eds. Princeton, NJ, USA: Princeton Univ. Press, 1950.

[47] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*. New Delhi, India: Prentice-Hall, 1997.

[48] S. Wasserman and K. Faust, *Social Network Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1994.

[49] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[50] D. J. Watts, "A simple model of global cascades in random networks," *Proc. Nat. Acad. Sci. (PNAS)*, vol. 99, no. 9, pp. 5766–5771, 2002.

[51] D. J. Watts, *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton, NJ: Princeton Univ. Press, 2003.

[52] D. J. Watts, *Six Degrees: The Science of a Connected Age*. New York: Norton and Company, 2004.

Ramasuri Narayanam received the M.Sc.(Engg.) degree in computer science from the Indian Institute of Science, Bangalore, in 2006. He is currently working towards the Ph.D. degree at the Indian Institute of Science.

His current research interests include game theory, mechanism design, social networks, ad hoc wireless networks, electronic commerce.

Mr. Narayanam is a recipient of the Microsoft Research India Ph.D. Fellowship.

Yadati Narahari (M'88–SM'03–F'08) is currently a Professor and Chair at the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. He is the Lead Author of a research monograph entitled "*Game Theoretic Problems in Network Economics and Mechanism Design Solutions*" (Springer, London, U.K.). He is a J. C. Bose National Fellow, and a Fellow of all leading science and engineering academies in India. The global companies with whom he has collaborated in the recent past include GM R&D, Intel, Infosys Technologies, and Xerox Corporation. The focus of his current research is to apply game theory and mechanism design to problems in Internet and Network Economics, Electronic Commerce, and Social Networks.

Prof. Narayanam is currently a Senior Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.