Noname manuscript No.
(will be inserted by the editor)

Time-Bounded Positive Influence in Social Networks

Tuo Shi · Siyao Cheng · Zhipeng Cai · Yingshu Li · Jianzhong Li

Received: date / Accepted: date

Abstract The appearance of social networks provides great opportunities for people to communicate, share and disseminate information. Meanwhile, it is quite challenge for utilizing a social networks efficiently in order to increase the commercial profit or alleviate social problems. One feasible solution is to select a subset of individuals that can positively influence the maximum other ones in the given social network, and some algorithms have been proposed to solve the optimal individual subset selection problem. However, most of the existing works ignored the constraint on time. They either assume that the time is infinite or only suitable to solve the snapshot selection problems. Obviously, both of them are impractical in the real system. Due to such reason, we study the problem of selecting the optimal individual subset to diffuse the positive influence when time is bounded. We proved that such a problem is NP-hard, and a heuristic algorithm based on greedy strategy is proposed. The experimental results on both simulation and real-world social networks based on the trace data in Shanghai show that our proposed algorithm outperforms the existing algorithms significantly, especially when the network structure is sparse.

Keywords social network · influence · dominating set

Tuo Shi
Harbin Institute of Technology
E-mail: shituo@hit.edu.cn

Siyao Cheng
Harbin Institute of Technology
E-mail: csy@hit.edu.cn

Zhipeng Cai
Georgia State University
E-mail: zcai@cs.gsu.edu

Yingshu Li
Georgia State University
E-mail: yili@cs.gsu.edu

Jianzhong Li
Harbin Institute of Technology
E-mail: lijzh@hit.edu.cn

1 Introduction

Social network is a platform consisting of individuals who share same interests, activities and backgrounds or have real-life connections. With the exponential growth of online social networks, more and more people are involved. According to the recent statistics, there are more than a billion Facebook accounts in the world. Among the billions of online social network individuals, more than 27% and 40% Europeans access social networks at least once within a day and a week, respectively [1]. Such a platform provides great opportunities for people to communicate, share and disseminate information. The popularity of social networks and their influences among individuals have attracted a great deal of attention due to its great potential benefits on both economy and society [2] . For example, in viral marketing, commercial companies can utilize social network as a channel to publicize their products [3]. Many intervention and education programs take advantage of this platform to help with alleviating social problems, such as drinking, smoking and drug problems. The government can also use social networks to stop rumors [4][5] .

In general, a social network can be represented as a graph $G(V, E)$, where V is the set of nodes representing the individuals in a social network, E is the set of edges where each edge represents the social tie between two individuals. Many research works indicate that individuals have tendency to follow the behaviors of their friends and relatives rather than the information obtained from other ways, such as radio, TV and online advertisement [6], and this tendency is growing with the increase of the number of her friends having the same behavior. Thus, one promising way to take advantage of social networks is to select a subset of individuals to positively influence other nodes in a social network, that is, this subset is expected to result in large cascade of further adoptions. The major reason to choose a subset of individuals instead of nodes in a whole network is because of budget limitation which is the primary concern for many applications. For example, a commercial company can promote some individuals in a social network with a fixed budget and ask them to publicize their products. An intervention program can select a subset of binge drinkers to participate such that they will positively influence other binge drinkers subsequently over a social network. The government can also depend on a subset of individuals to spread the truth in order to stop rumors. One efficient way to address the aforementioned problems is to make use of a dominating set [7]. A node set D is called a dominating set of graph G if each node in G not in D has at least one neighbor in D. However, the approaches employing dominating sets only consider individual-to-individual social influence, which is not practical in social networks. It is indicated by [6] that positive (negative) influence from more neighbors/friends will result in more impact to an individual. That is, if one individual will turn from negative to positive with a high probability if she has more positive friends than negative friends. Wang *et al.* [8] take the group-to-individual interaction into account and introduce a more reasonable model in social networks. They studied the Positive Influence Dominating Set (PIDS) problem. A set D is called a PIDS if for each individual in a social network, there are more than half of its neighbors belonging to D. Wang *et al.* proposed a greedy approximation algorithm to derive the least possible set to influence all the nodes. In many previous works, the problems are always formulated as discrete optimization problems (influence maximization problems) with

different object functions and constraints. For example, Kempe *et al.* [9] proposed the first influence maximization problem, which is to select a fixed number of candidates such that the expected number of the influenced nodes can be maximized. However, one of the key factors, time constraint, has been overlooked by most existing studies. The goal of the influence maximization problem is to maximize the expected number of influenced nodes after an unbounded time period, while the purpose of the PIDS problem is to find a dominate set D such that all the other nodes can be influenced by D instantly. In many real-world applications, time is an inevitable factor that needs to be considered. For example, in viral marketing, commercial companies are expected to know the effect of popularization within a time limit in order to ensure the normal operation of the chain of funds. Government always wants to stop rumors within a certain time to reduce public panic. Another factor overlooked by most of the existing works is the negative influence. Note that people can have both positive and negative influence on their close friends in social networks and an individual could turn back and forth with the changing of her neighbors. For example, one can trust the truth at the beginning and will have positive impact on her direct friends, on the other hand, she may believe the rumors if most of her friends pass negative information to her.

Motivated by the aforementioned factors, we consider the problem of how to select as few initial users as possible to influence all of the users in a social network within a certain time limit in this paper. An individual is positively influenced if more than half of its neighbors have positive impact on its and vice versa. In this paper, we first introduce the problem of finding a time-bounded Positive Influence Dominating Set (tPIDS), and propose a greedy algorithm to find the smallest possible size of a subset of the nodes in a social network which can influence the whole network within time t, where t is a parameter specified by users. In summary, the main contributions of this paper are summarized as follows.

1)The time-bounded Positive Influence Dominating Set problem is defined and we show finding a minimum tPIDS is NP-hard.

2)A greedy algorithm is proposed.The algorithm is based on the tPIDS SPREAD graph built on an original graph. The Move-Down operation is defined to adjust a tPIDS in order to find a feasible solution. The complexity of our greedy algorithm is also analyzed.

3)The correctness of our proposed algorithm is proven, that is, the solution returned by our algorithm is valid tPIDS of social networks.

4)We conduct extensive experiments on randomly generated graphs and real-world social networks like trace data sets provided by [10]. On loosely structured networks with roughly uniform degree of nodes, the initial influence set can be reduced by 15%-20% and the running time of our algorithm is acceptable. The experimental results indicate that our greedy algorithm is effective and efficient to find a tPIDS.

The rest of the paper is organized as follows. Section II summarizes the state-of-art works on the influence maximization problem. Section III provides the problem definition. Section IV proposes a greedy algorithm and proves that the result returned by our algorithm is feasible tPIDS of social networks. Section V presents the experimental results. Section VI concludes the paper.

2 Related Works

Domingos and Richardson first proposed the influence maximization problem [11]. The milestone work for the influence maximization problem is [9]. The problem is formulated as a discrete optimization problem which is to find k most influential nodes such that the expected number of the influenced nodes by the k seeds is maximized. Two fundamental information diffusion models, the Independent Cascade (IC) model and the Linear Threshold (LT) model are introduced. They first showed the problem is NP-hard and proposed two greedy algorithms with performance ratio of $1 - \frac{1}{e}$ for both the IC and LT models. The experimental results demonstrate their greedy algorithms significantly outperform two straightforward algorithms, classic degree and centrality-based heuristics. Kimura and Saito [12] proposed an efficient algorithm to calculate influence spread under shortest-path based influence cascade models. Leskovec *et al.* [13] proposed a Cost-Effective Lazy Forward (CELF) scheme to select seeds. CELF significantly reduces the number of evaluations on the influence spread ability of nodes by utilizing the submodularity property of the influence maximization objective. CELF speeds up the previous greedy algorithm more than a hundred times. Chen *et al.* [14] further improved the CELF algorithm in terms of scalability. They proposed an efficient heuristic algorithm which can keep the same approximation ratio but improve the running time significantly. Liu *et al.* [15] utilized the supervised Monte Carlo method to estimate the influence spread for nodes with specified precision by the sampling technology. Sheldon *et al.* [16] proposed a mixed integer programming formulation to maximize spread of cascades in networks. Although the objective function is not submodular anymore, their proposed preprocessing techniques can significantly reduce the computation time. Jung *et al.* [17] proposed a novel algorithm IRIE that incorporates Influence Ranking (IR) with a fast Influence Estimation (IE) method in order to obtain scalability and robustness. The Independent Cascade(IC) model and its extended version, the Independent Cascade Negative (IC-N) model, were considered. Wang *et al.* [18] proposed a novel greedy algorithm for selecting top-K influential nodes based on the community structure of a graph. Due to the fact that individuals within the same community have close relationships thus are more likely to influence each other while individuals among different communities are less likely to influence each other, the community based greedy algorithm chooses influential nodes within communities. All the aforementioned models try to maximize the expected number of the influenced nodes and do not take the time factor into account.

Another vein of the influence maximization problem is to select a positive influence dominating set. Dominating set, a subset D of node set V in graph G such that every node not in D is adjacent to at least one member of D, has been widely studied and utilized in many real life applications [19]. A positive influence dominating set is a variation of a dominating set, which is more suitable for social networks. A PIDS is a subset of graph G, called D, such that any node v in G is dominated by at least $\lceil \frac{deg(v)}{2} \rceil$ nodes in D. Wang *et al.* [8] first defined the PIDS problem in general random graphs. They proved the NP-hard of the Minimum Positive Influence Dominating Set (MPIDS) problem and proposed a greedy algorithm with approximation ratio $H(\delta)$ to construct an MPIDS in $O(n^3)$ time, where H is the harmonic function and δ is the maximum node degree. Zhang *et al.* [20] investigated the MPIDS problem in power-law graphs and proposed a greedy al-

gorithm with a constant approximation ratio. Raei *et al.* [21] further improved the greedy algorithm with time complexity $O(n^2)$ for a graph with power-law degree distribution. Dihn *et al.* [22] extended the PIDS problem to a more general version, called the Total Positive Influence Dominating Set (TPIDS) problem. This problem is to seek a minimum sized D such that every node in a graph has a fraction ρ of neighbors in D. They proved a minimized TPIDS cannot be approximated with factor $1 - \epsilon \ln \max\{\delta, V^{\frac{1}{2}}\}$ for general graphs where δ is the maximum degree in a graph. A simple proof was provided to show the problem can be approximated within factor $\ln \delta + O(1)$. For a power law graph, a constant factor approximation algorithm was proposed [22]. Note that, all the PIDS related problems are to seek a dominating set which can instantly affect all the nodes in a network, thus the size of a PIDS is large, which is not practical. In this paper, we will introduce a novel positive influence dominating set formulation by taking the time limit into account.

2.1 Problem Definition

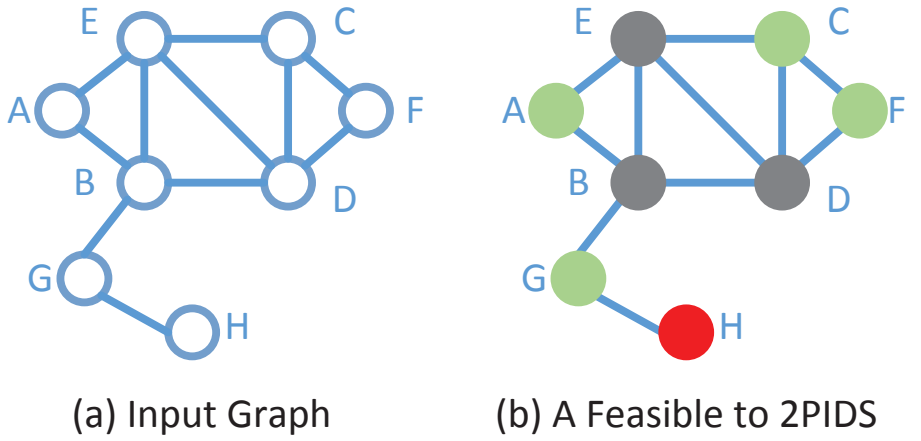
Assume a social network can be represented by an undirected graph $G = (V, E)$, where $v \in V$ is an individual in the social network and $e \in E$ represents the existence of social relationship between two individuals. Let $deg(v)$ be the number of the neighbors of v. That is, there are in total $deg(v)$ individuals who have direct social relationship with v. A set D is called a dominating set of G if each node not in D has at least one neighbor in D.

Definition 1 (Positive Influence Dominating Set (PIDS)) Given an undirected graph $G = (V, E)$, a node v is influenced by set D if at least $\lceil \frac{deg(v)}{2} \rceil$ of its neighbors is in the influence set D. A set P is called an influenced set of D if each $v \in P$ is influenced by D, defined as $P \models D$. A set D is a Positive Influence Dominating Set (PIDS) of G if $V \models D$.

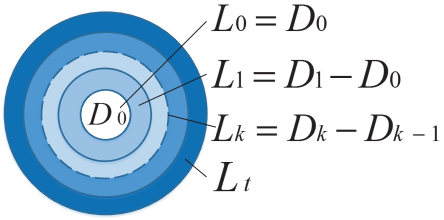
Definition 2 (Time-bounded Positive Influence Dominating Set (tPIDS)) Given an undirected graph $G = (V, E)$, a set D_0 is a t Positive Influence Dominating Set (tPIDS) of G if $V \models D_{t-1} \models \cdots D_i \cdots \models D_1 \models D_0$ where $D_i \subset V$ $(1 \leq i \leq t-1)$ and $D_i = \arg\max |P|_{P \models D_{i-1}}$. That is, D_i is the largest set among all the set influenced by D_{i-1}.

Note that in a tPIDS, all the nodes influenced by D_i will be included in D_{i+1}. That is, $D_i \subseteq D_{i+1}$.

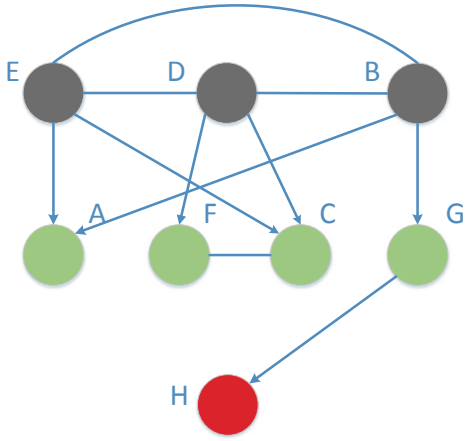
The goal of constructing a $tPIDS$ is to find a set D_0 which can positively influence the whole network within time t. For example, a commercial company wants to find a group of customers to propagate the produces such that all the customers in a social network will be positively influenced within a certain time. An intervention program wants to educate a set of binge drinkers such that positive effect of the intervention will conquer the negative effect within a certain time. Government wants to select a set of people in a social network to stop a rumor in a short time. Apparently, V is the largest tPIDS of $G = (V, E)$. However, it is costly to include all or too many individuals in a tPIDS. Our considered problem is to find a Minimum tPIDS (MtPIDS). That is, we want to use the least possible cost to positively influence the whole network in a certain time. The left graph



(a) An example of 2PIDS



(b) An example of layers



(c) An example of t-SPREAD graph

Fig. 1 Examples

in Fig.1(a) is an input graph $G = (V, E)$ where $V = \{A, B, C, D, E, F, G, H\}$. Let $D_0 = \{B, D, E\}$ which is the black node set in the right graph Fig.1(a). The green nodes A,C,F and G in the graph are influenced by D_0 and will be included in D_1. So $D_1 = \{A, B, C, D, E, F, G\}$. The red node H is influenced by D_1 and will be included in D_2. So $D_2 = \{A, B, C, D, E, F, G, H\} = V$. Thus D_0 is a feasible 2PIDS with size 3.

Theorem 1 *MtPIDS is NP-hard.*

Proof: It has been shown in [8] that finding a minimum PIDS is NP-hard. Since finding MtPIDS is a general version of finding a minimum PIDS, MtPIDS is NP-hard.

Definition 3 **(t-SPREAD)** Given an influence set $D_k \subseteq V$, D_{k+1} is the set with the maximum size such that $D_{k+1} \models D_k$. Let $L_{k+1} = D_{k+1} - D_k$,then L_{k+1} is the

set of influence nodes of D_k. Note that $L_0 = D_0$. Then the transition from D_k to D_{k+1} is called a SPREAD. Given a graph $G = (V, E)$, G is said to be t-SPREAD to D_0 if V can be obtained from an initial set D_0 ($D_0 \subseteq V$) in t times of SPREAD.

If graph G is t-SPREAD to D_0, then V can be divided into $t+1$ layers (Fig.1(b)). Let $L_0, L_1, \ldots, L_t$ be the set of nodes in each layer. For an arbitrary node u in set V, there exists an integer k ($0 \leq k \leq t$), such that u belongs to set L_k. Let $deg(u)$ be the degree of node u, and $N(u)$ be the set of neighbors of u. According to the definition of tPIDS, there exists an integer p ($p \geq \lceil \frac{deg(u)}{2} \rceil$) and a set $H(u)$ such that $H(u) \subseteq N(u)$, $H(u) \subseteq D_{k-1}$ and $|H(u)| = p$. That is to say, if u belongs to L_k, then at least half of its neighbors are in the inner layers of L_k.

Definition 4 (t-SPREAD graph) For each node u in V, draw a directed edge from each node in $H(u)$ to u, an undirected edge from each node in $N(u)$ which is in the same layer with u to u and a directed edge from u to the rest of the nodes in $N(u)$ to form a t-SPREAD graph.

An example of t-SPREAD graph based on the graph in Fig.1(a) is shown on Fig.1(c). For each node u in a t-SPREAD graph, let R_u be its in-degree, C_u be its out-degree, and P_u be the number of its neighbors in the same layer with u, then $R_u + C_u + P_u = deg(u)$. We also have

$$\begin{cases} R_u \geq C_u + P_u & \text{if } u \notin D_0 \\ P_u \geq C_u & \text{Otherwise} \end{cases} \tag{1}$$

If u belongs to D_0, $P_u \geq C_u$ means that at least half of the neighbors of u are in the same layer with u, thus u is a influence node of D_0 and would be influenced by D_0. If u is not in D_0, suppose that u belongs to L_k, $R_u \geq C_u + P_u$ means that over half of neighbors of u are in the inner layer of L_k, that is, u is a influence node of D_{k-1} and would be influenced by D_{k-1}.

Definition 5 (Redundant Node) For any node u in D_0, u is called a *redundant node* if it satisfies any of the following conditions:
1)$C_u = 0$ and for each $v \in N(u) \cap D_0$, $P_v - 1 \geq C_v + 1$.
2)For each $m \in N(u) \cap D_1$, $R_m - 1 \geq C_m + (1 + P_m)$. If $v \in N(u) \cap D_0$, $P_v - 1 \geq C_v + 1$. Suppose that v is one of the neighbors of u, which is a *redundant point* in D_0. If v satisfies the above conditions, then we say that v supports u.

If all the neighbors of u in D_0 satisfy $P-1 \geq C+1$ and all the neighbors of u in D_1 satisfy $R-1 \geq C+P+1$, after moving u down, all the neighbors of u would still satisfy Formula (1). In other word, moving u down would not break the structure of SPREAD graph. That is to say, redundant nodes are the candidate nodes to be moved down. The goal of defining redundant node is to find some nodes in D_0 to be moved down. Note that moving down these nodes should not result in other nodes dissatisfying Formula (1). For a redundant node v_i in D_0 and a node v_j in L_k, we say v_j supports v_i if v_j still satisfies Formula (1) even when moving v_i down to any layer between L_t and L_k. In the definition of redundant node, only nodes in L_0 and L_1 that support some redundant nodes in D_0 are taken into account. According to Theorem 4., moving down a redundant node will certainly influence the value of P, R and C of neighbors in L_0 and L_1. That is to say, a node must be a redundant node if all the neighbors in L_0 and L_1 support it, thus it can be

moved down so as to reduce the size of D_0.

The problem definition of MtPIDS is as follows:
Input: A given graph $G = (V, E)$ and a time limit t.
Output: A tPIDS of G with the smallest size.

3 Algorithm Description

3.1 Move-Down Algorithm

Once the t-SPREAD graph is constructed, the node set in each layer is determined. During the process of construction, if the size of D_0 is minimized, then an optimal result is obtained for the tPIDS problem. We employ a top-down approach to construct the t-SPREAD graph. To be specific, all the nodes in the initial graph G are included in D_0, and then all the redundant nodes in D_0 are moved down to a certain layer between L_1 and L_t. Once all the redundant nodes are moved down, the final result of the tPIDS problem is approximately optimal. The key point of the move-down operation is how to choose a candidate node to be moved down in each step. According to Theorem 5, when we moving down a redundant node, some other redundant nodes in D_0 might become non-redundant nodes, while some non-redundant nodes might become redundant nodes. Suppose that when moving down a redundant node in D_0, say v_i, n_i nodes in D_0 become redundant nodes and m_i nodes become non-redundant nodes. A cost function is defined as $Cost(i) = m_i - n_i$. Intuitively, the smaller the $Cost(i)$ is, the more redundant nodes can be found. In our greedy algorithm, each time we choose a redundant node v_i to minimize the cost function until there are no redundant nodes any more. Then the size of D_0 is as small as possible and D_0 is the approximate optimal solution to the tPIDS problem.

We design a practical and efficient greedy algorithm to solve the tPIDS problem

Algorithm 1 Move-Down Algorithm

Require: Graph $G = (V, E)$, t: the spreading times
Ensure: The set of initial influence nodes D_0
1: $Rdd = \varnothing$, D_0=V, $L_i = \varnothing$ $(1 \leqslant i \leqslant t)$.
2: For each node v_i in node set V, if v_i satisfies the definition of redundant node, then $Rdd = Rdd \cup v_i$.
3: For each node $v_i \in Rdd$, calculate $Cost(i)$.
4: If v_j owns the minimum cost function, move v_j to L_t, and examine whether all of the neighbors of v_j satisfy Formula (1). If not, consider $L_{t-1}, ..., L_k, ..., L_1$ in order until when moving v_j down to L_k, all the neighbors of v_j satisfy Formula (1). Then $L_k = L_k \cup v_j$, $D_0 = D_0 - v_j$.
5: Scan the node set V, add new redundant nodes into Rdd and delete non-redundant nodes from Rdd.
6: Repeat step 3-5 until $Rdd = \varnothing$.
7: Output D_0.

as shown in Algorithm 1. It takes as input the network graph $G = (V, E)$ and the spreading times t. Initially, all the nodes in V are assigned to D_0 ($i.e., D_0 = V$). Rdd is the set of the redundant nodes in V and L_i is the node set of each layer i $(1 \leqslant i \leqslant t)$. The final result is given by the subset D_0 which is the initial influence

set. Firstly, the algorithm scans all the nodes in V. For each node v_i, if it satisfies the definition of redundant node, add v_i to *Rdd* (Step 4). It initializes *Rdd*, thus we know which nodes might be moved down from D_0.

Secondly, the algorithm needs to choose the node to be moved down based on a cost function. It calculates the cost function for each node in *Rdd* and chooses the node, say v_j, to minimize the cost function (Step 5). Particularly, if more than one node share the minimum cost function, then the one with the least degree is chosen intuitively for reason that moving down a node with a smaller degree means less impact on other nodes. Then the algorithm considers the appropriate layer in descending order (from L_t to L_1) to which v_j should be moved down. In more details, first we consider the situation that we move v_j down to the outer most layer (L_t). It should be examined whether all the neighbors of v_j satisfy Formula (1). If not, v_j cannot be moved to L_t and other layers $(L_{t-1}, L_{t-2}, ..., L_k, ..., L_1)$ should be taken into account in descending order. That is, in the next step, we attempt to move v_j to L_{t-1} and examine the constraints and so on. Finally, we find that moving v_j down to L_k satisfies the constraints. Then we update the node sets L_k and D_0, that is, $L_k = L_k \cup v_j$ and $D_0 = D_0 - v_j$ (Step 6). After we moving down a redundant node, some redundant nodes might become non-redundant nodes, and vice versa, according to Theorem 5. In the last step, the algorithm updates *Rdd* and repeats the above steps until there is no redundant nodes (Steps 7-8). Whether a node is a redundant node or not depends on the property of its neighbors in L_0 and L_1. Intuitively, if we can reduce the number of the nodes in L_0 and L_1, the constraint on redundant nodes might become weak, thus the number of the redundant nodes might increase. The only way to reduce the nodes in L_0 is to move down some nodes, while the way to reduce nodes in L_1 is to move fewer nodes from L_0 to L_1. In addition, a redundant node must be moved down to L_1 if it cannot be moved to L_2, that is, the neighbors in L_2 of the redundant node do not support it. Naturally, we tend to reduce the nodes in L_2. By that analogy, each time we move down a redundant node, the target layer should be as outer as possible. That is to say, the priority of an outer layer is higher than that of an inner layer. The example of our algorithm is in technique report [23].

Performance Guarantee

According to Theorem 1, the $tPIDS$ problem is NP-Hard. Our greedy algorithm runs in $O(D^2 * |V|^2)$ where D is the maximum degree and $|V|$ is the number of the nodes in a network. The extensive evaluation results also show that the performance and efficiency of our algorithm are convincing.

3.2 The Correctness of the Move-Down Algorithm

Now we prove the result returned by our greedy algorithm is feasible. The proofs of these theorems are in our technique report[23] . Firstly, we will show as long as all the nodes in a t SPREAD graph satisfy Formula (1), D_0 is a valid $tPIDS$. This can be proven if we demonstrate that every node in L_k will be influenced at time k if it satisfies Formula (1).

Theorem 2 *A node is called an influence node iff it satisfies Formula (1).*

Proof Suppose node $v \in L_k$ and D_{k-1} is the influence set at the current stage. The number of the neighbors in D_{k-1} of v is denoted by I_v.
Necessity: If v is an influence node of D_{k-1}, then

$$I_v \geqslant \lceil \frac{deg(v)}{2} \rceil.$$

According to the definition of a t-SPREAD graph,

$$I_v = R_v. \tag{2}$$

Invoking Formula (1), Formula (3) can be rewritten as

$$R_v \geqslant \lceil \frac{R_v + C_v + P_v}{2} \rceil. \tag{3}$$

Thus,

$$R_v \geqslant C_v + P_v \tag{4}$$

which means that node v satisfies Formula (1).
Sufficiency: If node v satisfies Formula (1),

$$R_v \geqslant C_v + P_v. \tag{5}$$

According to Formula (1),

$$C_v + P_v = deg(v) - R_v. \tag{6}$$

Then we can obtain that

$$R_v \geqslant deg(v) - R_v \tag{7}$$

which can be rearranged to give the following formula

$$R_v \geqslant \lceil \frac{deg(v)}{2} \rceil. \tag{8}$$

Obviously,

$$I_v = R_v. \tag{9}$$

Thus,

$$I_v \geqslant \lceil \frac{deg(v)}{2} \rceil. \tag{10}$$

Then v is an influence node of D_{k-1}.

Theorem 3 demonstrates that all the nodes will not be affected by a move-down operation.

Theorem 3 *For a node v in L_0 or L_1, if it satisfies Formula (1) in the first place, then it still satisfies Formula (1) after we move down any node in L_0.*

Proof Apparently, for any node v which is not adjacent to u, moving down u has no influence on v. Thus, we just need to consider the nodes adjacent to u. Since L_0 is the initial influence set, any node u in L_0 must satisfy the following:

$$P_u \geqslant C_u \tag{11}$$
$$R_u = 0 \tag{12}$$

That is to say, at least half of the neighbors of u are in L_0. We know that any node needed to be moved down must be a redundant node. Suppose that u is a redundant node in layer 0, according to the definition of redundant node, for any node v in L_0 which is adjacent to u, v satisfies $P_v - 1 \geqslant C_v + 1$. Let $C_v^{'}$ and $P_v^{'}$ be the new value of C_v and P_v respectively.

$$P_v^{'} = P_v - 1 \tag{13}$$
$$C_v^{'} = C_v + 1 \tag{14}$$

Obviously, v still satisfies

$$P_v^{'} = P_v - 1 \geqslant C_v + 1 = C_v^{'}.$$

For any node v in layer 1 which is adjacent to u, there are two cases:
1) If $C_u = 0$, then moving u down will not influence the nodes in layer 1 to layer t.
2) If $C_u \neq 0$, for any node w in layer 1 which is adjacent to u, $R_w^{'}$, $C_w^{'}$ and $P_w^{'}$ will be the new value of R_w, C_w and P_w respectively.

$$R_w^{'} = R_w - 1 \tag{15}$$
$$C_w^{'} = C_w \tag{16}$$
$$P_w^{'} = P_w + 1 \tag{17}$$

According to the definition of redundant node, node w must satisfy the following relation

$$R_w - 1 \geqslant C_w + (1 + P_w). \tag{18}$$

Thus, $R_w^{'} \geqslant C_w^{'} + P_w^{'}$.

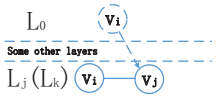
Moving down redundant nodes is an effective way to reduce the size of D_0.

(a) $j \neq 0$ (b) $j = 0$

Fig. 2 Initial status.

(a) $j \neq 0$ (b) $j = 0$

Fig. 3 Results of moving down v_i



(a) $j = k$

Fig. 4 The result of moving down v_i.

Theorem 4 *Moving down a redundant node will influence the P, R, and C values of its neighbors.*

Proof We assume that v_i is a redundant node in L_0 and v_j is its neighbor in L_j. When we moving v_i down from L_0 to L_k, there are three cases:

1. If $j > k$, that is, v_i is moved to the inner layer of L_j, then the values of P, R, and C remain the same.
2. If $j < k$, that is, v_i is moved to the outer layer of L_j, and then as shown in Fig.2(a) and Fig.2(b), two statuses should be taken into account:
   If $j = 0$, that is, v_j belongs to L_0, then

$$P_{v_j} = P_{v_j} - 1 \tag{19}$$
$$R_{v_j} = 0 \tag{20}$$
$$C_{v_j} = C_{v_j} + 1 \tag{21}$$

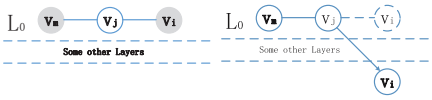
   Otherwise,

$$P_{v_j} = P_{v_j} \tag{22}$$
$$R_{v_j} = R_{v_j} - 1 \tag{23}$$
$$C_{v_j} = C_{v_j} + 1 \tag{24}$$

   The results of two statuses are shown in Fig.3(a) and Fig.3(b).
3. As shown in Fig.4, if $j = k$, that is, v_i is moved down from L_0 to the same layer as v_j, then

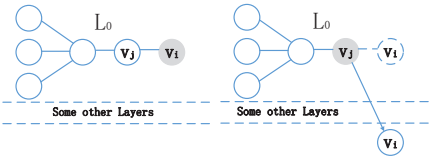
$$P_{v_j} = P_{v_j} + 1 \tag{25}$$
$$R_{v_j} = R_{v_j} - 1 \tag{26}$$
$$C_{v_j} = C_{v_j} \tag{27}$$

Theorem 5 *After we moving down a redundant node, some redundant nodes may become non-redundant nodes and some non-redundant nodes may become redundant nodes.*



(a)

Fig. 5 Example of a redundant node becoming a non-redundant node.



(a)

Fig. 6 Example of a non-redundant node becoming a redundant node.

Proof We assume that v_i is a redundant node in L_0 and v_j is its neighbor in L_j. When moving v_i down from L_0 to L_k, according to Theorem 4, the values of P, R and C of v_j may be changed. If the new values of the three variables satisfy the following condition

$$j = 0, P - C < 2$$

or

$$j = 1, R - P - C < 2$$

that is, v_j does not satisfy Formula (1) any more, then all of the redundant nodes adjacent to v_j become non-redundant.
A node is called a redundant node if and only if all of its neighbors in L_0 and L_1 satisfy Formula (1). As shown in Fig.5, assume that v_j is a neighbor in L_0 of v_i, and v_j is a non-redundant node before v_i is moved down. Furthermore, suppose that v_i not satisfying Formula (1) after moving down v_j is the exact reason why v_j is a non-redundant node. As shown in Fig.6, if v_i is moved down from L_0 to a layer between L_2 and L_t, then v_j becomes redundant because all of its neighbors in L_0 and L_1 satisfy Formula (1). In other words, moving down v_i may cause some of its neighbors in L_0, such as v_j, to become redundant.

4 Experiments

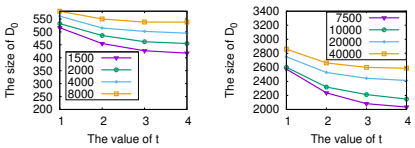
We conduct experiments on both synthetic and real-world networks to evaluate the performance and efficiency of our greedy algorithm. Our experiments aim at illustrating the effect of the properties of the given graph G, such as the number of the nodes, average degree and times of spreading, on the performance of our algorithm involving the size of D_0, the percentage of D_0 in V and the runtime of the algorithm. Moreover, we compare our algorithm with the work in [8] for that the $PIDS$ problem is a special case of our $tPIDS$ problem.

4.1 Data set

We use the SNAP tools provided by [24] to generate Erdos-Renyi network graphs, for the reason that we only consider the loosely structured networks with roughly uniform degree of nodes. As shown in Table 1, we generate a lot of networks with different numbers of nodes and average degree.

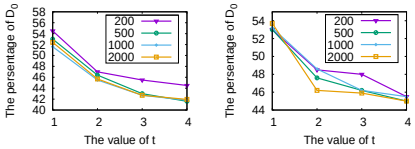
Nodes	Edges	Average Degree
200	300,400	3,4
300	450	3
400	600	3
500	750,1000,2000,4000	3,4,8,16
600	900	3
800	1200	3
1000	1500,2000,4000,8000	3,4,8,16
2000	4000	4
5000	7500,10000,15000,30000	3,4,6,12
10000	15000	3

Table 1 The information of all the networks.



(a) Network with 1000 nodes (b) Network with 5000 nodes

Fig. 7 The impact of average degree

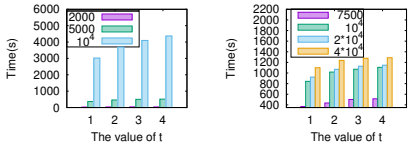


(a) Average degree is 3 (b) Average degree is 4

Fig. 8 The impact of number of nodes

4.2 The impact of average degree

To study the impact of average degree, we consider the networks with 1000 and 5000 nodes respectively, as shown in Fig.7(a) and Fig.7(b). For each network, we



(a) The impact of number of nodes (b) The impact of average degree

Fig. 9 The impact of network size and average degree

(a) Performance on real-world networks(The size of D_0)

(b) Performance on real-world networks(The running time)

Fig. 10 Performance on real-world networks

set the average degree as 3, 4, 8, and 16 and plot the relationship between the size of D_0 and the spread times t. Intuitively, the interdependence among the nodes is stronger in networks with large average degrees than those with small average degrees. We find that the size of D_0 does not continuously decrease dramatically when t is larger than 4, so we just consider the situations with t smaller than 4. Fig.7(a) and Fig.7(b) show that the size of D_0 in V decreases as t increases. On average, in the network whose average degree is 16, 2PIDS reduces by about 5.0% compared with 1PIDS, while 4PIDS decreases by about 7.0% compared with 1PIDS. However, on average, in the network whose average degree is 3, 2PIDS reduces by about 11.8% compared with 1PIDS, while 4PIDS decreases by about 18.9% compared with 1PIDS. When the average degree is much larger, the size of D_0 decreases more slowly. In a word, for the network with the same number of nodes, our greedy algorithm performs well for both large average degree and small average degree. But the result obtained from the network with small average degree is much better.

4.3 The impact of number of nodes

To study the impact of number of nodes, we consider the networks with 200, 500, 1000 and 2000 nodes respectively, as shown in Fig.8(a) and Fig.8(b). In Fig.8(a) the average degree is 3 for each network and in Fig.8(a) the average degree is 4. We plot the relationship between the percentage of D_0 in V and the spreading times t. In Fig.8(a), the x-axis represents the times of spreading and the y-axis represents the percentage of D_0 in V. Apparently, the percentage of D_0 in V decreases more or less due to the increasing spreading times t. For all of the above networks, the percentage of D_0 in V decreases to about 40% when t grows to 4.

In other words, we can choose a subset with only 40% individuals of the society rather than the whole society to influence all the other individuals in 4 steps using our algorithm. The cost of choosing initial influence set is reduced largely. The situations of the 4 different networks are similar. Our greedy algorithm performs a little better in the networks with more nodes, since the average degree is the same. For example, when the average degree is 3, the interdependence of nodes is stronger in the network with 200 nodes. Generally speaking, the number of nodes does not affect the performance of our greedy algorithm much in condition that the interdependence among nodes is weak.

4.4 The impact of network size and average degree on time efficiency

To study the impact of nodes numbers on time efficiency of our greedy algorithm, we consider the networks with 2000, 5000 and 10000 nodes respectively, as shown in Fig.9(a). The average degree of the network is set to 3. In Fig.9(a), the x-axis represents the times of spreading and the y-axis represents the running time of our greedy algorithm. The histogram shows the relationship between running time and the times of spreading for the networks with different numbers of nodes. Visually, the running time of our algorithm increases slowly as the increasing spreading times increases. For example, consider the situation when t is 3. For the network with 2000, 5000, and 10000 nodes, the cost of our algorithm is roughly 10, 500 and 4000 seconds respectively. To study the impact of average degree on time efficiency, we consider the network with 5000 nodes. The average degree is set to 3, 4, 6 and 12 respectively, as shown in Fig.9(b). Intuitively, when the average degree is 3, the running time of our algorithm is less than 500 seconds, while the running time doubles when the average degree is larger. That is to say, our algorithm performs much better for sparse networks, such as the disruption tolerant networks [25]. For real-world social networks with millions of nodes, our algorithm is also feasible since a large network with weak interdependence among nodes can be divided into a sequence of strongly connected components, which contain much fewer individuals and can be served by our algorithm in an acceptable time bound.

4.5 Comparison with the PIDS algorithm in [8]

The goal of the PIDS problem is to find a dominant set D_0 such that all the other nodes can be influenced by D_0 instantly, while the purpose of our tPIDS problem is to find a dominant set D_0 to influence other individuals in a certain bounded time period. It is clear that the PIDS problem is a particular case of our tPIDS problem when t is 1, that is, PIDS is equivalent to 1PIDS. Then our greedy algorithm is also feasible for the PIDS problem. To compare our greedy algorithm with the algorithm in [8], we consider the network with 500 and 1000 nodes respectively and set the average degree to 3. As shown in Fig.**??**, the x-axis represents the number of nodes in the graph and y-axis represents the size of the initial influence set D_0. Our result is almost the same as that of [8]. However, the time efficiency of our algorithm is much better than that of [8]. In Fig.**??**, the x-axis represents the number of nodes in the graph and y-axis represents the running time of the algorithm. In the network with 500 nodes, the running time

of our algorithm is about 1 second while the running time of the algorithm in [8] is over 22 seconds. In the network with 1000 nodes, the running time of our algorithm is about 8 seconds while the running time of the algorithm in [8] is over 180 seconds. Our algorithm is almost 20 times faster than the algorithm in [8]. **The difference between the two algorithms**.

4.6 Performance on real-world networks

In addition to simulated experiments, we conduct experiments based on realistic trace data sets provided by [10]. The given data set records the real-time longitude and latitude of 2370 taxies during half a month. Once two taxis appear in the same place at the same time, they will exchange some information. If the encounter time is greater than a constant, then we can consider that there is an edge between these two taxis. The taxi network contains 2370 nodes and 7631 edges, and the average degree is about 6.44. This taxi network is a mobile social network, which is a kind of social network and has been widely studied these years. Fig.10(a) and 10(b) shows the result of our algorithm on the given network. As shown in Fig.10(a), the percentage of 1PIDS, 2PIDS, 3PIDS and 4PIDS in the whole network is about 50.0%, 45.3%, 43.5% and 42.7% respectively. 2PIDS reduces by 9.4% and 4PIDS reduces by 14.6% compared with 1PIDS. That is, in order to send some information to all the 2370 taxies, one can choose about 42.7% of the whole individuals. After 4 times of spread, all the taxies would receive the information. As shown in Fig.10(b), the running time of 1PIDS, 2PIDS, 3PIDS and 4PIDS is 112s, 137s, 141s and 143s respectively. Evidently, the marginal utility of the running time decreases as the times of spread increases. The performance and time efficiency of our greedy algorithm are acceptable and feasible on the real-world networks.

5 Conclusion

This paper takes the first step towards the t Positive Influence Dominating Set problem in social networks. The $tPIDS$ problem takes the time factor, positive and group-to-individual interaction into account which is more practical in real social networks. We build the t SPREAD graph for a network and design a corresponding greedy algorithm. We also prove several significant theorems to support our algorithm. We conduct extensive experiments on some randomly generated graphs and a real-world mobile social network. Our greedy algorithm works much better on loosely structured networks with roughly uniform degree of nodes. The algorithm can reduce the percentage of initial nodes to about 40%. 4PIDS reduces about 15%-20% compared with 1PIDS. Moreover, the running time of our algorithm is reasonable and acceptable.The experiment results show that our algorithm is effective and efficient. To some extent, as for the PIDS problem, our greedy algorithm performs better than the algorithm in [8] especially on the aspect of effectiveness.

References

1. http://www.tns-opinion.com/sites/default/files/THINK%209.pdf

2. X. Wang, Y. Lin, S. Zhang, Z. Cai, Enterprise Information Systems pp. 1–30 (2015)
3. Y. Wang, G. Yin, Z. Cai, Y. Dong, H. Dong, Journal of Network and Computer Applications (2015)
4. Z. He, Z. Cai, X. Wang, in *The 35th IEEE International Conference on Distributed Computing Systems, Columbus, Ohio, USA* (2015)
5. Y. Wang, Z. Cai, G. Yin, Y. Gao, Q. Pan, in *Computational Social Networks* (Springer, 2015), pp. 236–247
6. J. Jaccard, H. Blanton, T. Dodge, Developmental Psychology p. 135147 (2005)
7. D. Kim, D. Li, O. Asgari, Y. Li, A.O. Tokuta, in *COCOON* (2013), pp. 841–848
8. F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, S. Shan, Theoretical Computer Science **412**(3), 265 (2011)
9. D. Kempe, J. Kleinberg, E. Tardos, in *SIGKDD* (2003), pp. 461–467
10. Suvnet-trace data. URL http:/wirelesslab.sjtu.edu.cn
11. M. Richardson, P. Domingos, in *Proceedings of the 8th ACM SIGKDD Conferenceon Knowledge Discovery and Data Mining* (2002), pp. 61–70
12. M. Kimura, K. Saito, in *Principles and Practice of Knowledge Discovery in Databases* (2006), pp. 259–271
13. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (2007), Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 420–429. 1281239
14. W. Chen, Y. Wang, S. Yang, in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 199–208
15. X. Liu, S. Li, X. Liao, L. Wang, Q. Wu, in *Proceedings of the Fifth Workshop on Social Network Systems* (2012), Proceedings of the Fifth Workshop on Social Network Systems, pp. 1–6
16. D. Sheldon, B. Dilkina, A. Elmachtoub, R. Finseth, et al., arXiv preprint arXiv:1203.3514 (2012)
17. K. Jung, W. Heo, W. Chen, arXiv preprint arXiv:1111.4795 (2011)
18. Y. Wang, G. Cong, G. Song, K. Xie, in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010), pp. 1039–1048
19. X. Gao, Y. Wang, X. Li, W. Wu, Discrete Mathematics, Algorithms and Applications **1**(1), 71 (2009)
20. W. Zhang, W. Wu, F. Wang, K. Xu, Social Network Analysis and Mining **2**(1), 31 (2011)
21. W. Chen, W. Lu, N. Zhang, (2012)
22. T. Dinh, Y. Shen, D. Nguyen, M. Thai, Journal of Combinatorics Optimization **27**(3), 487 (2014)
23. Time-bounded positive influence in social networks. URL https://www.dropbox.com/s/v68oihergag7l8h/tPIDSTR.pdf?dl=0
24. J. Leskovec, R. Sosič. SNAP: A general purpose network analysis and graph mining library in C++. http://snap.stanford.edu/snap (2014)
25. L. Zhang, X. Wang, J. Lu, M. Ren, Z. Duan, Z. Cai, Transactions on Emerging Telecommunications Technologies (2014)