Abstract: Given a specific propagation speed $h$ in a social network $G(V, E)$, an influence circle (IC) of a node $s$ in time $t$ is a node set of its influenced nodes, where the distance between $s$ and its expected influenced node $w$ is less than radius $r = ht$. Different from the Influence Maximization (IM) problem which finds a set of $k$ initial seed nodes in a network so that the expected size of cascade is maximized, the aim of the proposed influence circle covering (ICCovering) problem in this work is to find a minimum number of seeds or ICs to cover the whole network. The general approach for this covering problem is greedy strategy, which iteratively selects a seed with the largest influence circle. However, the upper bound of greedy algorithms does not perform very well, and the value will increase further as the network scale expands. In this paper, we propose an $\alpha$-approximation partitioning algorithm for large-scale social networks, where $\alpha$ is the maximum number of outer edges of Voronoi cells appeared in the partition. The algorithm divides the input graph into smaller cells so that each cell can be solved separately, and a feasible solution to the input object can be constructed by combining the solutions of the smaller cells. When solving the smaller cells, we adopt the linear programming method. In order to improve its effectiveness and efficiency, we also propose two optimization algorithms. Extensive experiments on real social networks confirm the superiorities and effectiveness of our solution.

Index Terms: Social networks, influence circle, influence circle covering, approximation algorithms.

I. INTRODUCTION

Given a social network $G(V, E)$, where $V$ denotes the node set and $E$ denotes the directed edge set. Let $M$ be a probabilistic model that captures interactive behaviors of nodes in $G$. The influence maximization problem in $G$ and under the assumption of $M$, which seeks for the largest number of influenced nodes with only $k$ seeds, draws a lot of attention recently [1]–[7].

Suppose we have a seed node $s$ and one of its expected influenced node $v \in V'$, where $V'$ represents the set of nodes that may be activated by $s$. We denote the distance between $s$ and $v$ as $d(s, v)$, the value of $d(s, v)$ equals the length of a shortest $(s, v)$-path. An influence circle (IC) of a seed node $s$ in $G$ is defined as a node set where $\max d(s, v) | v \in V' \leq r$, where $r$ is the radius of the influence circle, represents influence spread in time $t$ with a specific speed $h$. Instead of the influence maximization problem, we address the influence circle covering (ICCovering) problem in this paper. That is, when $r$ is decided, how many influence circles or seed nodes are needed to cover a social network graph.

In some cases, the mission of influence spread is time-critical. On the one hand, the propagation of influence from one node to another may introduce a certain amount of time delay. The length of the propagation path will further scale the delay. Iribarren and Moro [8] and Karsai et al. [9] observed and reported the slowdown evidence of the influence spread and attributed it to the bursty nature of node interactions and the topological correlations in networks. In some other cases, the spread of influence may have diminishing property, that is, beyond a certain distance, it is no longer capable of spreading. In these scenarios, we argue that taking influence circle to model a constrained diffusion process is appropriate.

Based on the observation of the time limit, some papers have raised the issue of time-critical influence maximization, which aims to maximize influence spread within a given deadline [10]–[14]. However, their proposed models and algorithms cannot ensure that all nodes in the social network have a certain degree of opportunity to be activated within the
time limit. The IC model and ICCovering problem proposed in this paper are totally different from their works. Firstly, we can guarantee that each node in the social network has the opportunity to be activated so that they will not miss any important information. Secondly, their models generally look for a fixed number of nodes, but our algorithms are to find as few seed nodes as possible in order to save cost while ensuring full coverage of the area. Therefore, although the ICCovering problem studied in this paper is related to the IM problem, it is a new problem.

The influence circle covering problem has its applications. Two representative applications are introduced in the following:

- **Social Marketing:** For example, in a carpet cleaning marketing campaign for some service, to reach all the customers in an area within a restricted time, the company can conduct an influence circle covering search and return the nodes that satisfy the requirement.
- **Social Data Analysis:** Studying features of a social network is an important problem in data analysis. With IC covering, we can learn the minimum nodes needed to cover a network and help to set appropriate strategies for other analyses. For example, it can help to set an appropriate $k$ in IM to achieve a cost-effective solution.

Nevertheless, there exist challenges in tackling the ICCovering problem. First is the fact that interactions in social networks and influence parameters tend to be volatile. To estimate the influence circle of a node with a reasonable accuracy requires the recomputation of solutions over time. The other challenge is relevant social networks for this problem can scale to massive size, even on the order of the million-node network, there are two basic approaches. One of them is Monte Carlo method. Suppose Reverse Influence Sampling (RIS) and the other is Monte Carlo simulations. Borgs et al. [1] formulate the influence maximization problem and propose a greedy algorithm with $(1 - 1/e - \epsilon)$-approximation by utilizing Monte Carlo simulations. Borgs et al. [15] later introduces the reverse sampling technique resulting in a more efficient algorithm. The reverse sampling technique was further improved by many other works [5], [6], [16]–[18]. However, none of these approaches considers the distance limitation between the seed node and its influenced nodes when information is disseminated. To the best of our knowledge, this is the first article addressing the influence circle covering problem in the field. In summary, our contributions are as follows:

- We are the first group to utilize influence circle (IC) to model a constrained influence diffusion process. In order to achieve full IC coverage of the network, we proposed the influence circle covering (ICCovering) problem. In particular, we put forward an estimation method for IC and also provide theoretical analysis.
- We innovatively devise a partitioning algorithm based on Voronoi cells for IC covering problem, and solve it with a linear programming model in each partition. We also prove the proposed algorithm could achieve an $\alpha$-approximation, where $\alpha$ is the maximum number of outer edges of Voronoi cells. Furthermore, we propose a novel partitioning method for the large-scale network.
- To further improve the effectiveness and efficiency of the algorithm, we propose two optimization techniques, i.e., reduction optimization and combination optimization.
- We have conducted experiments on both small and large-scale networks, and the experimental results strongly corroborate the effectiveness and efficiency of our approach.

**II. RELATED WORK**

The IM attempts to find a small subset of nodes that makes the expected influence maximized. Domingos and Richardson [20] first study influence maximization from a data mining perspective. Later, Kempe et al. [1] formulate the influence maximization problem and propose a greedy algorithm with $(1 - 1/e - \epsilon)$-approximation by utilizing Monte Carlo simulations. Borgs et al. [15] later introduces the reverse sampling technique resulting in a more efficient algorithm. The reverse sampling technique was further improved by many other works [5], [6], [16]–[18]. However, none of these approaches considers the distance limitation between the seed node and its influenced nodes when information is disseminated.

In order to support time-critical applications, some researchers have further considered the IM problem with a time constraint, and put forward several models and algorithms to solve the time-constrained influence maximization (TAM) problem [10]–[14]. However, they only ask for a fixed number of seed nodes to maximize the time-critical influence, and cannot guarantee that every node in the social network has a certain degree of opportunity to be activated.

Kuhnle et al. [21] proposed the threshold activation problem (TAP): Given positive threshold $T$, find a minimum-size seed set $A$ that can trigger the expected activation of at least $T$. They proposed Scalable TAP Algorithm with Bicriteria guarantees (STAB). However, on the one hand, their work does not take into account that the spread of influence may be limited by distance. On the other hand, when the minimum number of seed nodes is selected, it cannot guarantee that all nodes in the network graph have the opportunity to be activated at a certain time.
The Influence Circle Covering Problem is essentially a Set Covering Problem (SCP), which aims at seeking a minimum number of seeds or ICs to cover the entire network. Most of the literature [22]–[24] uses the Greedy strategy to solve it, which iteratively selects the seed with the largest influence circle. Several other works [25], [26] adopt it by linear programming method. Furthermore, [26] proposes a reduction rule to improve efficiency. Nevertheless, since the IC covering problem is NP-complete [27], these methods neither guarantee an approximation, nor are capable of solving large instances.

The literature [28] proposes the Discrete Unit Disk Cover (DUDC) problem, that is, given a set of \( n \) points on a two-dimensional plane \( P \) and several unit discs, the DUDC problem aims at finding the minimum number of unit disks, so that it covers all points in \( P \). The literature [19] proves set covering problem is NP-hard, and literature [28] solve DUDC by dividing technology. If we imagine an influence circle as a unit disk, the influence circle covering problem just like graph version of the unit disk covering on 2-dimensional plane. However, there are still many differences between ICCovering and DUDC. Firstly, we deal with problems in social networks composed of nodes and edges rather than discrete points in a plane; Secondly, the influence circle is related to probability and the number of hops, which is different from the unit disk in DUDC; Thirdly, we innovatively adopt the Vorinoi-based partitioning algorithm and use linear programming to solve ICCovering.

### III. PRELIMINARIES

In this section, we briefly introduce the Independent Cascade diffusion model first, then formally define the Influence Circle Covering problem. A greedy approach is also presented in this section, and it will serve as the main comparison of our proposed method in the following sections. At the end we introduce a partition method.

#### A. THE INDEPENDENT CASCADES MODEL

We denote the social network as a graph \( G \). Graph \( G \) consists of a node set \( V \) and a directed edge set \( E \). The size of set \( V \) is \( n \), and \( m \) for \( E \). That is, we have \( |V| = n \) and \( |E| = m \). The Independent Cascade model origins from interacting particle systems [29], and first gets its name in [1]. Starting from an initial active node set \( S \), the diffusion process unfolds in discrete steps according to the following rules:

1. When a node is active in step \( i \), it has a chance to activate each currently inactive neighbor which is on the other side of an outgoing edge of node \( s \). If node \( v \) is such a node, then the activation succeeds with a probability of \( p(s, v) \in [0, 1] \). In other words, \( p \) is a propagation probability correspondence with each edge in the graph and is a parameter of the graph which is set at the beginning.
2. If node \( s \) successfully activates \( v \), node \( v \) will become active in step \( i + 1 \) and remain its state in subsequent steps.

**Definition 1 (Influence Circle):** The influence circle \( C(s) \) of node \( s \) is defined as its expected influenced node set \( V' \). Node \( v \) in set \( V' \) satisfies the following requirements:

- There is a shortest path \( p(s, v) \) origins from \( s \) and ends at \( v \), and \( v \) can be activated along this path with a probability greater than a threshold.
- If we denote the length of \( p(s, v) \) as \( d(s, v) \), then \( d(s, v) \) must be no more than \( r \), where the value \( r \) is called the radius of the influence circle.

**Definition 2 (Influence Circle Covering in a Graph):** Given a graph \( G \) with set \( V \) of nodes and radius of influence circle \( r \), for influence circles drawn from a node set \( S \subseteq V \), an Influence Circle Covering of \( G \) refers to \( \bigcup_{s \in S} C(s) \) covers a given high percentage of nodes in \( V \), denoted as \( V_r \), that is \( \bigcup_{s \in S} C(s) = V_r \). The optimal Influence Circle Covering is to select minimum cardinality subset \( S^* \subseteq V \) such that each node in \( V_r \) is covered by at least one influence circle generated by \( S^* \).

**Fig. 1.** Two solutions of influence circle covering.

Fig. 1 shows two seed selection methods to solve the influence circle covering problem in the same network topology. In this example, the solid node represents the selected seed node. The radius of the influence circle on the left is 1, and this method selects 6 seed nodes. The radius of the influence circle on the right is 2, and it only needs to choose 3 seed nodes to complete the coverage.

#### B. A GREEDY APPROACH

Obtaining an optimal solution of the influence circle covering problem is hard. In fact, the influence circle covering problem is a set-covering problem, which is known to be NP-complete [27]. To find near-optimal covers, a heuristic procedure called greedy is widely used for solving this kind of problem [1], [22], [25].

Suppose we have known influence circle of every node in \( G \), that is, \( C(v_1), C(v_2), C(v_3), \ldots, C(v_n) \). It’s obvious that \( \bigcup_{1 \leq j \leq n} C(v_j) = V \), where \( V \) is the node set of \( G \). The goal of the greedy approach is to obtain a \( S^* \subseteq V \) which covers \( V \) and its cardinality is minimal. The approach starts from an empty node set \( S^* = \emptyset \), and then iteratively adds into \( S^* \) the node \( j \) that with the largest \( |C(v_j)| \) which counts the number of nodes expected to cover by \( v_j \).
This approach is conceptually simple. Young et al. [30] establish a tight bound on the worst-case behavior of this approach. The minimal cardinality of $S^*$ returned by the algorithm is at most $H(\kappa)$ times of OPT, such that $H(\kappa) = \sum_j 1$, where $\kappa$ is the size of the largest set $C(v_j)$.

The problem of the greedy approach is the upbound deteriorates when $\kappa$ increases. The function $H(\kappa)$ actually is a harmonic progression, and its value is approximately equal to $lnk + y$ (y is Euler-Mascheroni constant). For example, when $\kappa = 10000$, the value of $H(\kappa)$ will be 9.78. But in a real evolving online social network, this number can be easily exceeded.

C. A PARTITION METHOD

If we imagine an influence circle as a unit disk, the influence circle covering problem is just like graph version of the unit disk covering on 2-dimensional plane. Given a set $P$ of $n$ points and a number of unit disks on a 2-dimensional plane, the Discrete Unit Disk Cover (DUDC) problem is to find a minimum number of unit disks such that they cover all the points in $P$. This actually is a set cover problem which is NP-hard [19]. DUDC problem can be solved with partition technique. The basic idea of partition is to divide the input 2-dimensional plane into smaller cells so that each cell has a simple solution, and a feasible solution to the input instance can be obtained by combining the solutions of the smaller cells. Following this approach, we explore influence circle covering with partition and combination methods in a graph.

IV. PROPOSED SOLUTION

In this section, we first introduce the estimation method of influence circle. Then we present a partitioning algorithm based on Voronoi cells to solve the IC covering problem, and give an approximation ratio with necessary theoretical analysis. At the same time, we propose a novel strategy of dividing a large network diagram into several Voronoi cells. Next, we introduce how to use linear programming to find the optimal solution in each cell. Finally, we give an example and analyze the complexity of the algorithm.

A. ESTIMATING THE INFLUENCE CIRCLE

Kempe et al. [1] propose two models (IC and LT models) to estimate a set of nodes’ influence having a reasonable accuracy using a Monte Carlo method, which has attracted a lot of followers [31], [32] that aim at reducing the computation overhead. However, there is no formal analysis in the literature on how many repetitions are needed to obtain an acceptable estimation. To handle the influence maximization in a million-node graph, Tang et al. [17] provided theoretical analysis on this problem for the first time. In the light of their efforts, we present an algorithm which estimates an influence circle with a $(1 - \epsilon)$-approximate solution with at least $1 - |D|^{-(l+1)}$ probability, where $D$ is the set of nodes that can be reached from $s$ within $r$ distance in $G$, and $l$ is a given parameter. Our analysis is based on the Chernoff bounds [33].

**Lemma 1**: Let $X_1, X_2, X_3, \ldots X_n$ be i.i.d. random variables sampled from a distribution on $[0,1]$. Then, for $X = \sum_{i=1}^n X_i$, mean $\mu$ and any $\delta > 0$,

$$Pr[X \geq (\mu + \delta) \cdot n\mu] < \exp(-\frac{\delta^2}{2 + \delta}n\mu).$$

$$Pr[X \leq (\mu - \delta) \cdot n\mu] < \exp(-\frac{\delta^2}{2}n\mu).$$

In addition to Lemma 1, we have following Lemma 2 that describes the connection between the influence circle of a node $s$ and influence propagation between a randomly chosen node $v$. We use set $D$ to denote the set of nodes that can be reached from $s$ within $r$ distance in an unweighted graph $G$.

**Lemma 2**: The expected influence circle of a node $s$ in a graph $g$ which is constructed from $G$ by removing each edge $e$ with $1 - p(e)$ probability, is $|D|$ times the probability that a node $v$, chosen uniformly at random, influenced by node $s$.

**Proof**: We write $g \sim G$ to mean that $g$ is drawn from the random graph distribution $G$. Let $I_a(s)$ be the probability of node $u$ that can be reached by node $s$ in short than distance $r$, $\mathbb{E}_g[I(a)]$ be the expected influence circle of node $s$, $C_g(s)$ be the set of nodes reachable from $s$, and $V_g$ be the node set in $g$. Then, we have

$$\mathbb{E}_{g \sim G}[I(s)] = \sum_{u \in V_g} Pr_{g \sim G}[u \in C_g(s)]$$

$$= |D| \cdot Pr_{g \sim G}[u \in C_g(s)]$$

$$= |D| \cdot \mathbb{E}[I_a(s)].$$

Lemma 2 implies that we can estimate $\mathbb{E}_G[I(s)]$ by estimating the probability of the event that a randomly chosen node $u$ can be reached from $s$. Let $OPT$ be the optimal spread of node $s$ in $G$ within distance $r$. Using the Chernoff bound, we show $|D| \cdot I_a(s)$ is an accurate estimator of $s$’s expected spread, when a sampling times $\beta$ is sufficiently large.

**Theorem 1**: Given a graph $G$, a source node $s$ and a randomly selected node $v$ from $D$, if there is a shortest path between node $s$ and $v$ in a graph $g$ which created from $G$ by removing each edge $e$ with $1 - p(e)$ probability, then node $v$ falls in the influence circle of $s$. Repeat this process $\beta$ times, where $\beta$ satisfies

$$\beta \geq (2 + \epsilon)|D| \cdot \frac{(1 + \sqrt{2}) \ln |D| + \ln 2}{OPT \cdot \epsilon^2}$$

The returned influence circle estimation $\mathbb{E}_G[I(s)]$ is a $(1 - \epsilon)$-approximate solution with at least $1 - |D|^{-(l+1)}$ probability.

**Proof**: By Lemma 2, we can denote

$$\rho = \mathbb{E}[I_u(s)] = \mathbb{E}[I(s)]/|D| \leq OPT/|D|$$

Then, we have

$$Pr[|I_u(s) - \mathbb{E}[I(s)]| \geq \epsilon \cdot OPT] = Pr[|\beta \cdot I_u(s) - \rho \beta| \geq \epsilon \cdot OPT]$$

$$\geq \frac{\epsilon \cdot OPT}{|D| \cdot \rho \cdot \beta}$$

VOLUME 9, 2021

146113
Using the Chernoff bounds and let \( \delta = \varepsilon \cdot OPT/(|D| \rho) \), we have

\[
    P(\{ |\beta \cdot I_u(s) - \rho \beta | \geq \delta \cdot \rho \beta \}) \leq 2\exp(-\frac{\delta^2}{2 + \delta} \cdot \rho \beta) \\
    \leq 2\exp(-\frac{\varepsilon^2 \cdot OPT^2}{2|D| \cdot OPT + |D| \cdot \rho \cdot OPT}) \\
    \leq 2\exp(-\frac{\varepsilon^2 \cdot OPT}{|D|(2 + \varepsilon) \cdot \rho \beta}) \\
    \leq |D|^{-(l+1)} \tag{7}
\]

Therefore, we have \(|D| \cdot I_u(s) - E[I(s)] \geq \varepsilon \cdot OPT \) holds with at least \( 1 - |D|^{-(l+1)} \) probability. The theory is proved.

Of course, it’s impossible to know \( OPT \), or the expected influence circle of any node. In the following algorithm, we first take the Breadth-First Search (BFS) start from node \( s \) with distance limit \( r \) in graph \( G \). It will return all possible influenced nodes by \( s \). Then we take three other Breadth-First Searches in graph \( g \) which are derived from \( G \) by removing each edge \( e \) with probability \( 1 - p(e) \). The results obtained from the above search are used to estimate \( OPT \). With the estimation of \( OPT \), we can set the sampling times \( \beta \) which are used to estimate an influence circle of node \( s \).

**Algorithm 1**

**Input:** A graph \( G = (V, E) \), vertex \( s \), strength \( h \), and time \( t \)

**Output:** An Influence Circle \( C(s) \) of \( s \)

1. Initialize a set \( C(s) = \emptyset \)
2. Set \( r = ht \)
3. Set \( D = \{ v | v \in V \& distance(s, v) \leq r \} \)
4. for \( i = 1 \) to \( 3 \) do
5. Generate a subgraph \( g \) from \( G \)
6. Set \( D_i = \{ v | v \in V_g \& distance(s, v) \leq r \} \)
7. Set \( OPT = \sum_{i=1}^{3} |D_i|/(3|D|) \)
8. Set \( \beta = (2 + \varepsilon)|D| \cdot \frac{(l+1)\ln|D|+\ln 2}{OPT + \delta^2} \)
9. for \( j = 1 \) to \( \beta \) do
10. Generate a subgraph \( g \) from \( G \)
11. Randomly select a node \( v \) from \( D \)
12. if There is a shortest path from \( s \) to \( v \) then
13. Add \( v \) to \( C(s) \)
14. return \( C(s) \)

The approach Algorithm 1 adopts is simple. Its complexity lies in lines 7 and 12. Given a specific \( \varepsilon \) and \( l \), the value of \( \beta \) depends on \(|D|\) and \( OPT \). If we denote the maximum degree in graph \( G \) with \( m_{\text{degree}} \), the upper bound of \(|D|\) will be set to \( m_{\text{degree}}^\delta \), where \( r \) is the radius of an influence circle. To decide whether there is a shortest path between \( s \) and \( v \), we use Breath-First Search. Given a graph \( G = (V, E) \), the running time of BFS is \( O(V + E) \). Considering the worst case, the number of nodes is at most \( m_{\text{degree}}^\delta \), and the number of edges is at most \( m_{\text{degree}} \cdot m_{\text{degree}}^\delta \). The upper bound of running time for line 12 is \( O((1 + m_{\text{degree}})m_{\text{degree}}^\delta) \).

**B. PARTITIONING ALGORITHM FOR IC COVERING**

In this subsection, we propose a partitioning algorithm for IC covering. The basic approach of partition is to divide the input graph into smaller cells so that each cell can be solved separately, and a feasible solution to the input object can be constructed by combining the solutions of the smaller cells. A partitioning algorithm usually will lead to a better optimal solution than a greedy one [28].

The Voronoi diagram is a data structure extensively investigated in the field of computational geometry. It is also investigated in a graph because they can efficiently serve as solutions to many network problems [34]. In a directed graph \( G = (V, E) \), with a nonadaptive partitioning approach, we can separate graph into a Voronoi diagram partition cell \( \{V_1, V_2, \ldots, V_k, U\} \) for a set of nodes \( K = \{v_1, v_2, \ldots, v_k\} \subseteq V \) so that

- each node \( v \in V_i \), \( d(v_i, v) \leq d(v_j, v) \) for all \( j \in 1, 2, \ldots, k \).
- \( U \) contains all nodes \( v \) with \( d(v_i, v) = d(v_j, v) \) for all \( i \in 1, 2, \ldots, k \).

By carefully selecting nodes set \( K \), we can control the radius and edges of each Voronoi cell in the diagram.

After dividing the graph \( G \) into a set of Voronoi cells, we solve the ICCovering problem for each cell. Finally, we take the union of the solutions of all Voronoi cells as the solution to the original graph. We have following Algorithm 2.

**Algorithm 2**

**Input:** A graph \( G = (V, E) \), integer \( \alpha \) and radius \( r \)

**Output:** A seed set \( S \)

1. Divide graph \( G \) into Voronoi cells (cell edges \( \leq \alpha \) and radius \( > r \) )
2. Set \( Cell \) be all cells in \( G \)
3. Set \( S = \emptyset \)
4. for each \( v \_cell \) in \( Cell \) do
5. Find a minimum IC covering \( S(v \_cell) \) (with linear programming)
6. \( S = S \cup S(v \_cell) \)
7. return \( S \)

In subsection IV-A, we have obtained influence circle of every node. Every node is said to be covered if it falls into a node’s influence circle. An IC covering problem now is a minimum IC set covering problem. In line 5, we solve IC covering problem with linear programming, and we will address this issue in detail in subsection IV-D.

While considering the performance of Algorithm 2, Theorem 2 shows it is an approximation solution to IC covering and will return a solution not worse than \( \alpha \) times of optimal solution, where \( \alpha \) is the maximum number of outer edges of Voronoi cells.

**Theorem 2:** Algorithm 2 is an \( \alpha \)-approximation for minimum IC covering.
Proof: Suppose we have a feasible approximate solution $S$, which is such a solution that can be represented as

$$ S = \bigcup_{v_{cell} \in \text{Cell}} S(v_{cell}) \quad (8) $$

where each $S(v_{cell})$ is an IC cover of nodes in Voronoi cell $v_{cell}$.

Also, suppose we have an optimal IC covering solution $S^*$, it can be modified to one of a feasible approximate solutions. The modification process is conducted for each IC in $S^*$ that intersects more than one Voronoi cell. By selecting an additional IC in the intersected cell which originally was part of the IC in $S$ (the part which IC intersects with the Voronoi cell), and use them to cover nodes in different Voronoi cells.

If an IC in a Voronoi cell intersects with $q$ adjacent cells, it’s obvious that $q$ is no more than $\alpha$ cells since we partition a graph into cells with edges no more that $\alpha$. That is, $2 \leq q \leq \alpha$, then we add $q-1$ additional IC to the optimal solution $S^*$.

If there are $x$ influence circles generated by $S^*$ that intersect more than one cell, then the above process adds at most $(\alpha - 1)x$ ICs to $S^*$. A solution $S$ returned by Algorithm 2 satisfies

$$ |S| \leq |S^*| + (\alpha - 1)x $$

$$ \leq |S^*| + (\alpha - 1)|S^*| $$

$$ = \alpha|S^*| $$

Hence Algorithm 2 is $\alpha$-approximation for minimum IC covering problem.

C. NETWORK PARTITION

In this subsection, we present a partition method based on the Voronoi diagram, which is the detail of Algorithm 2 (Line 1). As mentioned in Theorem 2, ICCovering is an $\alpha$-approximation algorithm, where $\alpha$ is the number of outer edges in each cell. Therefore, the core of the network partition is to divide the large network into cells that edges less than $\alpha$. However, some network partitioning algorithms, such as LPA-based [35] and game theory-based [36] partitioning algorithms, are not suitable for the network partitioning in ICCovering, because there is no guarantee that partitioned subgraphs meet the above conditions.

Inspired by [37], [38] that Delaunay triangulation of a discrete point set $P$ corresponds to the dual graph of the Voronoi diagram for $P$. As shown in Fig. 2, a vertex of a Voronoi cell belongs to three Voronoi polygons at the same time, and each Voronoi polygon corresponds to a central node. Connecting the central nodes of three Voronoi polygons with common vertices forms one Delaunay triangle.

We divide the network by controlling the number of Delaunay triangles so that the number of edges of the Voronoi cells is no more than $\alpha$. Note that triangulation satisfies the following conditions:

- A plane can be divided into many triangles.
- Any two triangles are either disjoint, or just intersect on a common edge.

More specifically, the circumcenters of Delaunay triangles are the vertices of the Voronoi diagram, and the vertex of the triangle is the center of each cell. In other words, the edge of the cell is the line segment connecting the circumcircles of two adjacent triangles.

Algorithm 3 Partition($G$, $\alpha$, $r$, $d$)

Input: A graph $G = (V, E)$, an integer $\alpha$, radius $r$ and distance $d$

Output: A Voronoi diagram partition $\{V_1, V_2, \ldots, V_k, U\}$

1: Initialize $K = \emptyset$;
2: Initialize $N = \text{All nodes in } G$;
3: Initialize $P = \text{Randomly select a node from } N$;
4: while $N \not= \emptyset$ do
5:    Initialize $Q = \emptyset$;
6:    for each $v_i \in P$ do
7:        Set $A = \{v | v \in N \& \text{distance}(v_i, v) = d\}$;
8:        $\zeta = \alpha |\text{existing triangles with } v_i \text{ as vertex}|$;
9:        $\zeta = \zeta < |A| \& \zeta : |A|$;
10:       $Q_i = \zeta \text{ nodes randomly selected in } A$;
11:       Delaunay triangulation with vertex $v_i$;
12:       Delete nodes in circumsphere of triangles from $N$;
13:       $K = K \cup Q_i$, $Q = Q \cup Q_i$;
14:       $P = Q$;
15:       Divide $G$ into $\text{Cell}$ according to $K$;
16:       return $\text{Cell}$

Algorithm 3 presents the pseudocode of network partition. First, center nodes set $K$, candidate nodes set $N$ and set $P$ are initialized with an empty set, all nodes in $G$ and a random node respectively (Lines 1-3). Next, as long as $N$ is not empty, we iterate to seek the center nodes of Voronoi cells (Line 4). For each node $v_i \in P$, we look for triangles with $v_i$ as the common vertex, and the number of these triangles does not exceed $\alpha$, we use the divide-and-conquer approach mentioned in [39] to triangulate Delaunay (Lines 7-11). Then we delete the candidate nodes in the circumsphere of these triangles (Line 12). After that, we update $K$ and $Q$, respectively (Line 13). Finally, we divide the network into several Voronoi cells according to the vertical bisector of adjacent nodes in set $K$ (Line 15). The complex of Algorithm 3 lies in line 11. Chew et al. [39] has proved that the time complexity of

FIGURE 2. Voronoi diagrams and Delaunay triangles.

- A plane can be divided into many triangles.
- Any two triangles are either disjoint, or just intersect on a common edge.

More specifically, the circumcenters of Delaunay triangles are the vertices of the Voronoi diagram, and the vertex of the triangle is the center of each cell. In other words, the edge of the cell is the line segment connecting the circumcircles of two adjacent triangles.
Delaunay triangulation is $O(n_1 \log n_1)$, where $n_1$ is the number of nodes that need to be triangulated.

**D. LINEAR PROGRAMMING**

After we estimate the influence circle of nodes in each cell, influence circle covering becomes a combinatorial optimization problem, that is, how to find a minimum number of influence circles so that every node is covered by at least one of it. Furthermore, we simulate it as a linear programming problem.

However, the complexity of linear programming will increase with the expansion of the network scale. For instance, handling a massive number of columns in a matrix consumes most of the memory resources [40]. So instead of directly apply this approach to the original graph $G$, we partition the graph first, and then calculate separately in each cell.

Suppose $J$ is the set of nodes in each cell, which is the set of demand nodes. Associated with each node $i \in I$ is a subset $C(i)$ (i.e. influence circle of $i$) of the candidate seeds $j \in J$ (where $J$ is the set of candidate seed nodes), and $C(i)$ can cover the demand node. The set $C(i)$ may also be specified in terms of binary coefficients $\delta_{ij}$ that take a value of 1 if the influence circle of candidate seed $j \in J$ can cover demand node $i \in I$; 0 otherwise. That is,

$$\delta_{ij} = \begin{cases} 
1, & \text{if candidate seed } j \in J \text{ can cover node } i \in I \\
0, & \text{if not}
\end{cases} \quad (10)$$

Let $X_j$ denote an indicator function of whether to select node $j$ as the seed node, that is,

$$X_j = \begin{cases} 
1, & \text{if node } j \in J \text{ is selected as a seed node} \\
0, & \text{if not}
\end{cases} \quad (11)$$

Then the problem of influence circle covering can be formulated as the following Integer Linear Programming (ILP) problem.

$$\min \sum_{j \in J} X_j \quad (12)$$

s.t. $\delta_{ij}X_j \geq 1, \ \forall i \in I$ \quad (13)

$$X_j \in \{0, 1\}, \ \forall j \in J$$ \quad (14)

The objective function (12) minimizes the number of seeds that are selected. The constraint (13) requires that each demand node is covered by at least one influence circle of seeds. Note that the left-hand side of (13) gives the number of seeds that can cover demand node $i \in I$. These constraints may be rewritten in terms of the set $C(i)$, as follows:

$$\sum_{j \in C(i)} X_j \geq 1, \ \forall i \in I$$ \quad (15)

where $C(i)$ is the influence circle of node $i$, obviously $C(i)$ is the set of candidate seeds $j \in J$ that can cover demand node $i \in I$. The two forms of constraints are equal. Constraints (15) are the integrality constraints.

After modeling the influence circle covering problem as a linear programming problem, we use LINGO to solve it, which is an important software to solve mathematics programming developed by USA LINDO company [41], [42].

**E. AN EXAMPLE AND COMPLEXITY ANALYSIS**

This section first presents a graphical example of the proposed ICCovering solution, and then analyzes the complexity of the proposed ICCovering approach.

To see that ICCovering(Algorithm 2) runs in polynomial time, we claim that the problem restricted to a single Voronoi cell $v_{cell}$ can be solved in time $n^{O(d^2)}$ by an exhaustive search algorithm, where $d$ is the distance between the center nodes of two adjacent Voronoi cells, which is the constant set in Algorithm 3. In order to explain the problem more clearly, we use regular polygons to explain it. The area of a regular polygon can be represented as

$$S_v = \frac{\alpha \cdot \rho \cdot a_p}{2} \quad (16)$$

where $\alpha$ is number of edges of a polygons, $\rho$ is length of one edge, and $a_p$ is the apothem.

**FIGURE 3.** An example of our proposed solution.

**FIGURE 4.** Graphics used in the argument.
We use $\theta$ to denote the angle shown in the fig. 4. Obviously, $\theta = \frac{360^\circ}{\alpha} = \frac{180^\circ}{\alpha}$. Note that an influence circle can cover a polygon (see fig. 4(a)) with area $S_1$, which satisfies
\[ S_1 = \frac{\alpha \cdot 2 \sin \theta \cdot r \cos \theta}{2} = ar^2 \sin \theta \cos \theta \]  
(17)

We use $S_2$ to represent the area of a Voronoi cell (see fig. 4(b)) with apothem of $\frac{d}{2}$. Then, we have
\[ S_2 = \frac{\alpha \cdot 2 \frac{d}{2} \tan \theta \cdot \frac{d}{2}}{2} = \frac{1}{4} \alpha \tan \theta d^2 \]  
(18)

Since a cell can be partitioned into at most $\lceil \frac{S_2}{S_1} \rceil \leq \lceil \frac{\alpha}{\alpha^2 \cos^2 \theta} \cdot d^2 \rceil$ such polygons, at most $O(d^2)$ influence circles are needed to cover all nodes in a cell.

Assume that a Voronoi cell $v_{cell}$ contains $n_e$ nodes. If there is a node in cell $v_{cell}$ having distance greater than radius $r$ from any other node, then we need to use an isolated influence circle to cover it. If a node has a distance at most $r$ from some other nodes, then we can use a circle $R_1$ to cover it with some other nodes. In this case, we can move the influence circle to a canonical position so that at least two nodes covered by $R_1$ lie on the boundary of $R_1$. For any two given nodes within distance $r$, there are at most two possible canonical positions (see fig. 4(c)). Therefore, for $n_e$ given nodes in cell $v_{cell}$, we need to consider at most $2 \left( \frac{n_e}{2} \right)$ canonical positions. Together with the earlier observation that we need at most $O(d^2)$ influence circles to cover all nodes in a cell, we see that, in the exhaustive search algorithm, we need to inspect at most $(n_e(n_e-1))O(d^2) = n_e^2O(d^2)$ possible solutions to find a minimum circle cover for cell $v_{cell}$.

Thus, over all nonempty Voronoi cells, the total time for line 4 of Algorithm 2 is
\[ \sum_{v_{cell} \in Cell} n_eO(d^2) \leq \left( \sum_{v_{cell} \in Cell} n_e \right)O(d^2) = nE O(d^2) \]  
(19)

V. OPTIMIZING ALGORITHM

In this section, we propose two optimization techniques to further improve ICCovering which make it highly efficient.

A. OPTIMIZATION WITH THE REDUCTION

As mentioned in subsection IV-D, we adopt a linear programming model to find a seed node, and it is NP-complete. If expanding constraint (15), we observe that there are $|I|$ rows on the left side of the inequality group, and the $i$-th row has $|C(i)|$ terms, which is the number of nodes that can cover node $i$. Now we optimize this model with a reduction technique to improve the effectiveness of the algorithm.

It is obvious that many influence circles are contained or overlapped by others in a social network. We eliminate the nodes that generate these influence circles before calculating the linear programming problem. That is, consider two nodes $a$ and $b$, if $C(a) \subseteq C(b)$, then we say $b$ dominates $a$ and eliminate node $a$ from candidates. So we let $X_a = 0$ in the constraint (14), in other words, eliminate the corresponding rows in constraint (15). This process is presented in Algorithm 4. We first sort nodes in descending order according to the size of each influence circle (Lines 1-3), then traverse and delete the candidate seed nodes whose influence circles are contained or overlapped by others (Lines 4-7). The candidate set $N$ returned by Algorithm 4 is a regular set, which reduces the calculation scale and improves the accuracy of the algorithm.

Algorithm 4 Optimization With Reduction

**Input:** Node set $I$

**Output:** Candidate seeds $N$

1. Initialize $N = I$;
2. Estimate influence circle $C(i)$ of each node $i \in I$;
3. Sort $N$ in descending order;
   (according to the size of each influence circle)
4. for $i = 1$ to $|I|$ do
5.   for $j = i + 1$ to $|I|$ do
6.     if $C(i) \subseteq C(j)$ then
7.       Delete $i$ from $N$;
8. return $N$

B. OPTIMIZATION WITH THE COMBINATION

If the distance from the seed node to the boundary of a Voronoi cell is less than the radius of IC, or that the IC of a seed does not all fall into a Voronoi cell, then the seed node is said to be located at the boundary of the cell.

After getting the solution in each cell, if the seed nodes are simply combined, the influence circles at some cell boundaries will overlap with those at their adjacent cell boundaries, resulting in the final number of seed nodes not being optimal.

We propose a combination-based optimization, which aims at merging all seed sets and simplify the sets at the same time. More specifically, to check seeds at the boundary of each cell in turn, and remove the seeds whose influence circle are contained or overlapped by others, thereby improving the effectiveness of the results.

Algorithm 5 Optimization With Combination

**Input:** An IC cover of nodes $S(v_{cell})$ in each cell

**Output:** Simplified seeds $S$

1. Initialize $T = \emptyset$, $S = \emptyset$;
2. Merge all the $S(v_{cell})$ into $S$;
3. for each $s \in S$ do
4.   Estimate influence circle $C(s)$;
5.   Add $C(s)$ to $T$;
6. for each $s \in S$ do
7.   if $C(s)$ appears more than once in $T$ then
8.     Delete $s$ from $S$;
9.   Update $T$;
10. return $S$

Algorithm 5 shows the complete combining optimization algorithm. First, the algorithm simply merges the seed nodes
TABLE 1. Dataset details. ($K = 10^3$, $M = 10^6$).

| Dataset  | n   | m    | Avg. degree | Density     |
|----------|-----|------|-------------|-------------|
| Wiki     | 0.8K| 2.9K | 6.55        | $7.38 \times 10^{-3}$ |
| Facebook | 5.9K| 41.7K| 14          | $2.39 \times 10^{-3}$ |
| HetHEPT  | 12.0K| 118.5K| 26.99      | $1.87 \times 10^{-3}$ |
| Epinions | 27K | 100K | 7.53        | $2.83 \times 10^{-4}$ |
| GEMSEC   | 41.8K| 125.8K| 6          | $1.44 \times 10^{-4}$ |
| DBLP     | 0.31M| 1M   | 6.62        | $2.08 \times 10^{-5}$ |

of each cell together (Line 2). Then it estimates the influence circle for each seed and records the times that each node appearing in all influence circles (Lines 3-5). At last, it deletes redundant seeds whose influence circles appear more than once, and the optimized seed set $S$ is returned at the end (Lines 6-10).

VI. EXPERIMENTS
In this section, we evaluate the performance of our proposed algorithms through extensive experiments on six real-world datasets. We measure the effectiveness and efficiency of our algorithms in terms of the quality of seed sets and running time. We also design experiments to evaluate the effectiveness of partitioning method, reduction optimization, and combination optimization. All of our experiments are programmed by python 3.7, and deployed on a Linux machine with an Inter Xeon 2.6GHz CPU and 64GB RAM.

A. EXPERIMENTAL SETTING
1) DATASETS
We adopt six real-world datasets: (a) Wiki [43]: a who-votes-on-whom network, which comes from the collaboration Wikipedia voting; (b) Facebook [43]: Data is collected about Facebook pages (November 2017). Nodes represent the pages and edges are mutual likes among them. (c) HetHEPT [44]: an academic collaboration relationship on high energy physics area. (d) Epinions [43]: a who-trusts-whom social network from the Epinions consumer review site. (e) GEMSEC [45]: The data was collected from the music streaming service Deezer (November 2017). (f) DBLP [46]: a large collaboration network from the DBLP Computer Science Bibliography. The specific information of these six datasets is represented in TABLE 1. For fair comparison, we randomly generate 20 possible realizations for each dataset to test the performance of our algorithms, and the reported data are the average results on these realizations.

2) ALGORITHMS
As we discuss in Section III-B, Greedy approach is a very common method to find near-optimal covers. Therefore, we compare the proposed ICCovering algorithm with Greedy. We also compared our algorithm with the original Integer Linear Programming(ILP) algorithm without partitioning. In order to further prove the effectiveness of our algorithm, we adopt the number of seeds selected by ICCovering to the IMM-based, CELF-based, and Pagerank-based approaches respectively, and then calculate their coverage percentage as a comparison. In this comparative experiment, the IMM [16], CELF [47] and Pagerank [48] are used to select seed nodes according to the magnitude of influence, so that the IC they generate covers the network as much as possible.

We also evaluate our partitioning method and LPA-based partitioning method. LPA [35], [49] is wildly used in community detection, so we divide the network according to the detected communities, and merge small communities to improve efficiency. To demonstrate the performance of linear programming reduction optimization and combination optimization, we design two comparative experiments, which do not use these optimizations for comparison.

3) PARAMETER SETTINGS
We use the classical independent cascade model in our experiments. Following most of the prior works about influence maximization [1], [15]–[17], we set the propagation probability of each edge $(u, v)$ to $\frac{1}{d_{in}(v)}$, where $d_{in}(v)$ is the in-degree of node $v$.

For fair comparison, we vary the radius of the influence circle $r$ such that $r \in [1, 2, 3, 4]$. We set $\alpha = 5$, and $d = 2r$. Note that ILP is too slow to finish on Epinions, GEMSEC, and DBLP within a reasonable amount of time, as is the greedy algorithm when $t = 4$.

B. RESULTS AND ANALYSIS
We compare the algorithms on the quality of seed sets, the running time, and the coverage percentage. We also study the performance of two network partitioning methods, and the effectiveness of reduction optimization and combination optimization.

1) COMPARISON WITH GREEDY APPROACH
a: COMPARISON OF QUALITY OF SEED SETS
The quality of seed sets is evaluated based on the number of seeds. In this paper, the fewer the seeds, the better the quality. Fig. 5 shows the number of selected seeds against the radius of the influence circle. We can find that our proposed ICCovering is able to select seed sets of higher quality than baseline algorithm, and the optimization ratio is more than 30% on each dataset. For instance, when $r = 2$, the seeds size of our proposed algorithm has 37.8%, 45.07%, 48.7%, 31.2%, 44.29%, and 38.3% smaller than that of Greedy on Wiki, Facebook, HetHEPT, Epinions, GEMSEC, and DBLP, respectively. When $r = 3$, the optimization ratios are 58.3%, 37.7%, 53.7%, 35%, 46.46%, and 34.9% respectively. And we can observe that our algorithm is effective on both small and large-scale social networks.

b: COMPARISON OF THE RUNNING TIME
We demonstrate the running time results in Fig. 6. We can find that when the scale of the network or the radius of the influence circle is small, that is, the calculation scale is small,
the running time of our algorithm is slightly longer than that of Greedy by a few seconds. When the network becomes larger or the radius of the influence circle becomes larger, the efficiency advantage of our algorithm becomes obvious. Except for the smallest Wiki data set, the time efficiency of our algorithm is optimized by more than 35% when $r > 1$. It is easy to observe that the larger the scale of the network, the more obvious the efficiency advantage of our algorithm.
TABLE 2. Coverage percentage of IMM-based approach.

| Radius of IC | Wiki | Facebook | HetHEPT | Epinions |
|--------------|------|----------|---------|----------|
| 1            | 89.42% | 90.25% | 87.73% | 90.12% |
| 2            | 93.95% | 96.93% | 95.17% | 92.92% |
| 3            | 97.75% | 98.09% | 97.34% | 97.25% |

TABLE 3. Coverage percentage of CELF-based approach.

| Radius of IC | Wiki | Facebook | HetHEPT | Epinions |
|--------------|------|----------|---------|----------|
| 1            | 85.82% | 83.12% | 85.32% | 87.68% |
| 2            | 92.23% | 86.13% | 87.25% | 90.11% |
| 3            | 91.78% | 93.94% | 94.14% | 94.79% |

TABLE 4. Coverage percentage of Pagerank-based approach.

| Radius of IC | Wiki | Facebook | HetHEPT | Epinions |
|--------------|------|----------|---------|----------|
| 1            | 88.97% | 90.95% | 83.63% | 90.08% |
| 2            | 95.17% | 92.65% | 89.07% | 92.14% |
| 3            | 97.86% | 97.24% | 95.67% | 97.14% |

2) COMPARISON WITH ILP APPROACH
Fig. 5 and fig. 6 also show a comparison between ICCovering and the original linear programming method without partitioning. The results show that our proposed algorithm is better than the ILP approach, especially when \( r \geq 2 \), the number of seed nodes is optimized by at least 10%. The advantage in running time is even more obvious, with at least 80% optimized on the three data sets.

3) COMPARISON WITH THREE IM ALGORITHMS
ICCovering can guarantee that the IC achieves full coverage of the network while selecting as few seeds as possible, as a comparison, we calculated the coverage percentage selected by the IMM-based, CELF-based, and Pagerank-based approaches under the same number of seeds. Note that the coverage percentage of ICCovering is 100%, so only the results of the comparative experiments are listed.

a: COMPARISON WITH IMM-BASED APPROACH
TABLE 2 shows the coverage percentage generated by the seeds which selected by the IMM-based algorithm. We can find that when \( r = 1 \), the coverage rate of the IC generated by the seeds which selected by the IMM-based approach is only about 90%. Although this percentage becomes slightly higher as \( r \) increases, it is still impossible to guarantee full coverage of nodes in the network.

b: COMPARISON WITH CELF-BASED APPROACH
TABLE 3 shows the results of the CELF-based algorithm. We can observe that the coverage percentages on the four data sets are all less than 95%.

c: COMPARISON WITH PAGERANK-BASED APPROACH
TABLE 4 shows that the same number of seeds selected by the Pagerank-based approach algorithm cannot fully cover the network. When \( r = 1 \), it is even only 83.63% on the HetHEPT dataset.

4) COMPARISON OF PARTITIONING ALGORITHMS
a: COMPARISON OF NETWORK PARTITIONING METHODS
Fig. 7 shows the performance comparison of two different division methods. Obviously, from the histogram, the effects of two partitioning methods are very close. In fact, when \( r = 1 \) and \( r = 2 \), partition based on Voronoi is better than partition based on LPA by less than 3%.

But when it comes to the running time, as is shown in fig. 8, our proposed division method has huge advantages. For example, partition based on Voronoi is more than 40% faster than partition based on LPA on all datasets when \( r = 2 \). To explain that, the size of the sub-networks obtained by the label propagation algorithm varies greatly, and there may be many large-scale sub-networks, which increases the running time. On the contrary, the size of the Voronoi cells is relatively uniform.

5) THE EFFECT OF TWO OPTIMIZATION ALGORITHMS
a: EFFECTS OF REDUCTION OPTIMIZATION
Recall that model reduction optimization aims to reduce constraints in linear programming, we show its effectiveness in fig. 9 and fig.10. In fig. 9, we compare the size of seed sets with and without Reduction Optimization. It is clear to find that there are fewer seeds on each dataset with the optimization turned on, especially on Wiki, Facebook and HetHEPT.
the improvement is around 10%. This is because, with reduction optimization, many non-optimal nodes whose influence circles are covered by other nodes have been removed before calculation. It is for this reason that the scale of the problem becomes smaller, and fig. 10 shows that the running time is less with reduction optimization.

**b: EFFECTS OF COMBINATION OPTIMIZATION**

Fig. 11 and 12 illustrate the effectiveness of Combination Optimization on four datasets. We can observe that for different values of $r$, the number of seeds on each data set has a certain degree of reduction with Combination Optimization, especially when $r = 2$ and $r = 3$. When $r = 2$ with Combination Optimization turned on, the size of seed sets is 20.6%, 20.7%, 22.5%, 16.3%, 12.5%, and 17.1% smaller on Wiki, Facebook, HetHEPT, Epinions, GEMSEC, and DBLP, respectively. Correspondingly, the improvement is 16.7%, 28.1%, 34.8%, 36%, 28.3%, and 26.9% when $r = 3$. The reason is that the influence circles generated by some selected seed nodes overlap at the boundaries of some Voronoi cells after partitioning the network graph. Combination Optimization is to remove these redundant selected seeds. Of course, this process consumes a bit of time, as shown in fig. 12.

**VII. CONCLUSION**

This paper puts forward the concept of influence circle (IC) to model a constrained influence diffusion process, and studies the problem of IC covering problem which aims to identify a minimum number of seeds or influence circles to cover a high percentage of nodes in the network. Our partitioning algorithm is to divide the network into smaller parts so that each part can be solved by linear programming separately, and then combine the solutions of every smaller part. It has a provable expected approximation guarantee. We propose two optimization algorithms to improve the effectiveness and efficiency, one is optimization with reduction, and the other is combination optimization. We have also conducted extensive experiments on real social networks to test the performance, and the experimental results strongly corroborate the superiorities and effectiveness of our approach.

**REFERENCES**

[1] D. Kempe, J. Kleinberg, and E. Tardos, “Maximizing the spread of influence through a social network,” in Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2003, pp. 137–146.

[2] K. Huang, J. Tang, K. Han, X. Xiao, W. Chen, A. Sun, X. Tang, and A. Lim, “Efficient approximation algorithms for adaptive influence maximization,” VLDB J., vol. 29, pp. 1385–1406, Nov. 2020.

[3] K. Huang, J. Tang, X. Xiao, A. Sun, and A. Lim, “Efficient approximation algorithms for adaptive target profit maximization,” in Proc. IEEE 36th Int. Conf. Data Eng. (ICDE), Apr. 2020, pp. 649–660.

[4] S. Li, F. Kong, K. Tang, Q. Li, and W. Chen, “Online influence maximization under linear threshold model,” 2020, arXiv:2011.06578.

[5] J. Guo and W. Wu, “Influence maximization: Seeding based on community structure,” ACM Trans. Knowl. Discovery From Data, vol. 14, no. 6, pp. 1–22, Oct. 2020.

[6] S. Bian, Q. Guo, S. Wang, and J. X. Yu, “Efficient algorithms for budgeted influence maximization on massive social networks,” Proc. VLDB Endowment, vol. 13, no. 9, pp. 1498–1510, May 2020.

[7] W. Ju, L. Chen, B. Li, W. Liu, J. Sheng, and Y. Wang, “A new algorithm for positive influence maximization in signed networks,” Inf. Sci., vol. 512, pp. 1571–1591, Feb. 2020.

[8] J. L. Iribarren and E. Moro, “Impact of human activity patterns on the dynamics of information diffusion,” Phys. Rev. Lett., vol. 103, no. 3, 2009, Art. no. 038702.

[9] M. Karsai, M. Kivelä, R. K. Pan, K. Kaski, J. Kertész, A.-L. Barabási, and J. Saramäki, “Small but slow world: How network topology and burstiness slow down spreading,” Phys. Rev. E Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 83, no. 2, Feb. 2011, Art. no. 025102.

[10] M. G. Rodriguez, D. Balduzzi, and B. Schölkopf, “Uncovering the temporal dynamics of diffusion networks,” 2011, arXiv:1105.0697.

[11] M. Gomez-Rodriguez, L. Song, N. Du, H. Zha, and B. Schölkopf, “Influence estimation and maximization in continuous-time diffusion networks,” ACM Trans. Inf. Syst., vol. 34, no. 2, pp. 1–33, Apr. 2016.
[12] W. Chen, W. Lu, and N. Zhang, “Time-critical influence maximization in social networks with time-delayed diffusion process,” in Proc. 26th AAAI Conf. Artif. Intell., 2012, pp. 1–7.

[13] B. Liu, G. Cong, D. Xu, and Y. Zeng, “Time constrained influence maximization in social networks,” in Proc. IEEE 12th Int. Conf. Data Mining, Dec. 2012, pp. 439–448.

[14] G. Tong, R. Wang, Z. Dong, and X. Li, “Time-constrained adaptive influence maximization,” IEEE Trans. Comput. Syst., vol. 8, no. 1 pp. 33–44, Feb. 2021.

[15] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, “Maximizing social influence in nearly optimal time,” in Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms, Philadelphia, PA, USA: SIAM, 2014, pp. 946–957.

[16] Y. Tang, Y. Shi, and X. Xiao, “Influence maximization in near-linear time: A martingale approach,” in Proc. ACM SIGMOD Conf. Manage. Data, May 2015, pp. 1539–1554.

[17] Y. Tang, X. Xiao, and Y. Shi, “Influence maximization: Near-optimal time complexity meets practical efficiency,” in Proc. ACM SIGMOD Conf. Manage. Data, Jun. 2014, pp. 75–86.

[18] H. T. Nguyen, M. T. Thai, and T. N. Dinh, “Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks,” in Proc. Int. Conf. Manage. Data, Jun. 2016, pp. 695–710.

[19] D. S. Johnson, “The NP-completeness column: An ongoing guide,” J. Algorithms, vol. 6, no. 3, pp. 434–451, Sep. 1985.

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

[21] A. Kuhnle, T. Pan, M. A. Alim, and M. T. Thai, “Scalable bicriteria algorithms for the threshold activation problem in online social networks,” in Proc. IEEE INFOCOM Conf. Comput. Commun., May 2017, pp. 1–9.

[22] V. Chvátal, “A greedy heuristic for the set-covering problem,” Math. Oper. Res., vol. 4, no. 3, pp. 233–235, 1979.

[23] G. Cormode, H. Karloff, and A. Wirth, “Set cover algorithms for very large datasets,” in Proc. 19th ACM Int. Conf. Inf. Knowl. Manage. (CIKM), 2010, pp. 479–488.

[24] G. Lin, “An implementation of harmony search algorithm to the set covering problem,” in Proc. 11th Int. Conf. Natural Comput. (ICNC), Aug. 2015, pp. 200–204.

[25] B. Wang, “Coverage problems in sensor networks: A survey,” ACM Comput. Surv., vol. 43, no. 4, pp. 1–53, 2011.

[26] M. S. Daskin, Network and Discrete Location: Models, Algorithms, and Applications, Hoboken, NJ, USA: Wiley, 2011.

[27] R. M. Karp, “Reducibility among combinatorial problems,” in Complexity of Computer Computations. Boston, MA, USA: Springer, 1972, pp. 85–103.

[28] D. Z. Du, K. I. Ko, and X. Hu, Design and Analysis of Approximation Algorithms, New York, NY, USA: Springer, 2012.

[29] T. M. Liggett, Interacting Particle Systems, vol. 276. New York, NY, USA: Springer, 2012.

[30] N. E. Young, “Greedy set-cover algorithms (1974–1979),” Chvátal, Johnson, Lovász, Stein,” Encyclopedia Algorithms, pp. 379–381, 2008.

[31] K. Jung, W. Heo, and W. Chen, “IRIE: Scalable and robust influence maximization in social networks,” in Proc. IEEE 12th Int. Conf. Data Mining, Dec. 2012, pp. 918–923.

[32] C. Wang, W. Chen, and Y. Wang, “Scalable influence maximization for independent cascade model in large-scale social networks,” Data Mining Knowl. Discovery, vol. 25, no. 3, pp. 545–576, Nov. 2012.

[33] R. Motwani and P. Raghavan, Randomized Algorithms, vol. 68. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[34] M. Ewig, “The graph Voronoi diagram with applications,” Networks, vol. 36, no. 3, pp. 156–163, Oct. 2000.

[35] M. Wang, C. Wang, J. X. Yu, and J. Zhang, “Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework,” Proc. VLDB Endowment, vol. 8, no. 10, pp. 998–1009, 2015.

[36] V. Moscato, A. Picariello, and G. Spierli, “Community detection based on game theory,” Eng. Appl. Artif. Intell., vol. 85, pp. 773–782, Oct. 2019.

[37] L. J. Guibas, D. E. Knuth, and M. Sharir, “Randomized incremental construction of Delaunay and Voronoi diagrams,” Algorithmica, vol. 7, nos. 1–6, pp. 381–413, Jun. 1992.