Interactive Activities Initiation through Retrieving Hidden Social Information Networks

Yulong Song[1], Bin Fu[2], Jianxiong Guo[3], and Xiaofeng Gao[1,*]
[1]MoE Key Lab of Artificial Intelligence, Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, China
[2]Department of Computer Science, University of Texas Rio Grande Valley, Texas, USA
[3]Advanced Institute of Natural Sciences, Beijing Normal University, Zhuhai, China
sylacd@sjtu.edu.cn, bin.fu@utrgv.edu, jianxiongguo@bnu.edu.cn, gao-xf@cs.sjtu.edu.cn

2023 IEEE International Conference on Data Mining (ICDM) | 979-8-3503-0788-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDM58522.2023.00063

***Abstract*—The rise of social platforms based on online social networks has greatly enriched people's lives, resulting in various applications. Traditional research mainly focuses on users but pays less attention to the edges between users, and they all assume the topology of social networks is known in advance. Indeed, obtaining the network topology is challenging because of privacy protection and business competition. In this paper, we propose an activity initiation problem inspired by real business applications, such as Pinduoduo and Tencent, where each edge can be abstracted as an activity in which both ends (users) of the edge participate together, and the edge can be initiated by one of them. At this time, we hope to select as few users as possible to initiate activities and make the users of the whole network participate together. This problem can be reduced to the classic vertex cover problem, but the network information is hidden by social platforms as much as possible. To address this challenge, we put forward a solver-detector model. In each round of interaction, the solver uses a detector to obtain a small amount of edge information and achieves vertex coverage. This is a model in which a solver has limited access to input, but still gives a 2-approximation that is as good as the conventional model. Finally, we can cover the whole network with very few edge samples, which is a brand-new research perspective.**

***Index Terms*—Social Network, Vertex Cover, Interactive Computation Model, Approximation Algorithm**

I. Introduction & Related Work

Online social platforms, such as Twitter, WeChat, Facebook, and LinkedIn, have been developed quickly in recent years and have gradually become a mainstream way to communicate and make friends. These social platforms rely on user connections, which can be abstracted as a social graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. In this graph, each node $u \in V$ represents a user, and each edge $\{u, v\} \in E$ represents the friend relationship between user u and user v. In traditional social applications, they mainly considered maximizing a target function by measuring nodes, such as influence maximization (IM) [1]. It selects a subset of nodes as the seed set that maximizes the follow-up influence spread. Based on the IM problem, various research works discuss the influence spread in social networks [2]–[9], and social recommendations [10]–[13]. All these works focus on nodes in the graph to measure the objective performance. In fact, it is also meaningful to measure the edge for many kinds of targets inspired by real applications.

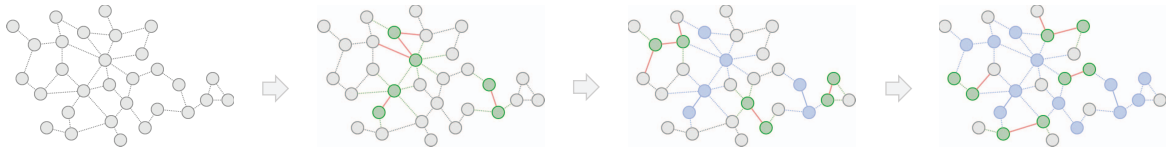
Let us consider the following scenario. As an organization, we hope that users in the whole network discuss a common topic. At this point, this topic can be associated with edges in the graph. Two users can participate in discussing this topic based on their friend relations. Thus, if an edge $\{u, v\} \in E$ exists, user u can initiate a discussion with user v, and vice versa. We can further abstract the discussion into a more general concept: an activity, and formulate the *activity initiation* (AIN) problem. One wants to select as few users as possible to initiate an activity, so that all users on the network can participate, and each user can initiate an activity with her friends. Slightly different from the IM scenario, informing the activity information to the whole network is not enough, the organizer want users to participate with their friends as many couple-times as possible, which may be corresponding with the flow heat and the profit of the organizer. A representative example is Pinduoduo's "mass bargain campaigns" [1], which starts with the least number of users and hopes that the whole network will participate in this activity.

Obviously, our AIN problem is not an instance of the IM problem because we are concerned about the interaction between users instead of the diffusion process, which is an instance of the well-studied vertex cover (VC) problem. The VC problem [14] is to identify the smallest subset $S \subseteq V$ such that each edge is adjacent to at least one of the vertex in S, which is NP-hard and one of Karp's twenty-one NP-complete problems. We can achieve a 2-approximation ratio by using a greedy algorithm [15], and 2-approximation is essentially the best-possible polynomial time approximation algorithm [16], [17]. In addition, the VC problem has also been used to model the problems in social networks, such as crowd-sensing [18], positive influence dominating set [19], [20], competitive influence [21], and so on [22]–[25].

This work was supported by the National Key R&D Program of China [2020YFB1707900], the National Natural Science Foundation of China [62272302, 62172276, 62202055], the Shanghai Municipal Science and Technology Major Project [2021SHZDZX0102]. *Xiaofeng Gao is the corresponding author.

[1]Pinduoduo's "mass bargain campaigns": https://finance.yahoo.com/news/chinas-lower-tier-markets-slow-093000386.html

979-8-3503-0788-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDM58522.2023.00063



(a) The Original Graph (b) Round 1 (c) Round 2 (d) Round 3

Fig. 1. An Example of Solving the interactive VC Problem.

However, not only for the classic VC problem but also for the applications mentioned above, there is a basic assumption that the underlying graph is known. Unfortunately, because of privacy protection, business competition, and the requirement of data freshness, the network topology owned by social platforms is difficult to obtain in advance. When the network topology is unknown, how should we deal with our AIN problem? We hope to achieve our goal with the least cost. Few works paid attention to solving combinatorial optimization problems with an unknown graph topology. In [26], [27], researchers estimated the average degree and degree distribution through small and random sampling in large-scale networks. In [28], they studied how to approximate the size of an unknown set by using group queries and conditional sampling. Recently in 2022, Behnezhad [29] argued that we can estimate the size of the minimum VC in sublinear time. To the best of our knowledge, this is the closest work to ours, but the goal is significantly different, and no literature has considered the VC problem with hidden topology.

Under the situation with unknown network topology, in this paper, we use a "traffic detector" to infer which user pairs may have friend relationships by monitoring their network traffic. If there is frequent data packet forwarding between two users, we can think that they are friends, and thus an edge may exist between these two users. Random traffic detection can obtain a certain number of unknown friend relationships. Then, a natural question is how to use as little detection as possible to achieve our activity initiation. Motivated by this, we propose an interactive activity initiation (InterAIN) problem in which we can constantly interact with the detector on hidden social networks.

To solve our InterAIN problem, we design an interactive algorithm where the computation model is based on interactions between the VC solver and the traffic detector round by round. In each round, the VC solver selects some potential nodes for maximizing coverage according to randomly sampled edges returned by the traffic detector. It terminates when the detector cannot return any new edges that have not been covered. For illumination, let us continue the example of Pinduoduo's "mass bargain", as shown in Figure 1. The social network is controlled by Tencent, who is unwilling and unable to provide fresh user information to its competitor. At this time, Pinduoduo gradually excavates network information through detectors. Figure 1(a) shows a graph in which the topology is not known to the solver. In each round of the algorithm, the solver asks the detector to find 5 new (uncovered) edges, denoted by the red edges in these figures, and generates a temporary VC node set, which is marked in green. The temporary result is able to cover much more potential edges (shown in blue edges) than those the solver knows, and the detector will omit such edges that have been covered. As Figure 1(d) shows, the solver generates a proper solution for the whole graph after three rounds. Thus, it is unnecessary to know the network topology to achieve a vertex cover. In fact, knowing a small part of edge information can cover the whole graph, which will be argued by our subsequent theoretical analysis and experiments.

Back to our InterAIN problem, in this paper, we propose two computation models to model the interaction process. Firstly, we assume that the platform is willing to disclose the scale of the network (i.e. the probable number of online users). Secondly, we know nothing about the potential social graph, including nodes and edges. The solver gives an approximation for the problem when it only has partial information about the problem input. The number of rounds tells us the depth of computation, and the number of random samples in each round tells us what information a solver needs to know. This interactive algorithm cannot be a sublinear time as it needs to check whether the chosen nodes cover all edges of a graph. In our theoretical analysis, we show that it has a 2-approximation ratio and takes $O(1/\epsilon)$ rounds and $O(n^{1+\epsilon})$ samples in each round if the number of online users is known. Besides, we also prove a lower bound $\Omega(1/\epsilon)$ of the number of rounds if it calls for $n^{1+\epsilon}$ samples in each round. For the scenario that we know nothing about the network, a 2-approximate algorithm can solve the problem with $O(g(n)\log\log n)$ rounds and $n^{1+O(1/g(n))}$ samples in each round, where $g(n)$ satisfies $g(n^2) \leq cg(n)$ for a constant c.

In all, these computation models and solutions can not only solve our InterAIN problem, but also deal with any real applications that can be reduced to the VC problem where it is impractical to get the entire graph but possible to collect randomly sampled edges. This is a rigorous theoretical model that needs investigation and has found some interesting properties.

II. Computation Model

The InterAIN problem is a specific instance of the general model for hiding input computing, and we construct our algorithms based on a solver-detector model, as shown in Figure 2.

Definition 1 (Solver-Detector Model)**.** *Let $G = (V, E)$ be a network that is unknown to the solver S. The S can only*

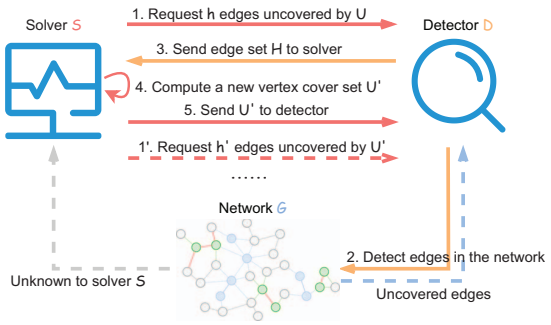


Fig. 2. The running process of the solver-detector model.

partially access G via the detection of a detector D. In round i, we denote the message from detector D to solver S as $\gamma_i^{D\to S}$ (detected edge information), and the message from solver S to detector D as $\gamma_i^{S\to D}$ (temporary result for the VC). Let Γ_i be $((\gamma_1^{D\to S},\gamma_1^{S\to D}),\cdots,(\gamma_i^{D\to S},\gamma_i^{S\to D}))$ that contains all the messages of communication between solver S and detector D. After the conduction of round i, a reserving network $G_i = res(G,\Gamma_i)$ contains all edges uncovered by S yet. Let $G_0 = G$, the function res is monotonic because G_i is a subgraph of G_{i-1} for $i \geq 1$. $G_i = \emptyset$ when detector D is not able to detect any uncovered edge. Then, the solving process terminates and it is supposed that we achieve a vertex cover for network G.

Definition 2 (Algorithm Process)**.** *For a hidden graph $G = (V,E)$, a randomized algorithm for vertex cover supports the following operations: (1) Use function $R_{G,V'}$ to get r different random edges from a detector, which is not covered by node set V' in G; (2) The algorithm conducts in rounds. In each round, the algorithm can select some nodes and put them into set V' according to the newly known edges. In the beginning, V' is empty; and (3) When the detector returns no uncovered edge, the algorithm terminates and returns V' as an (approximate) VC solution finally.*

In addition, the complexity model can be shown in Definition 3, which describes the complexity in the subsequent analysis.

Definition 3. *Let $A(\cdot,\cdot)$ be a randomized algorithm for the VC problem in the model given by Definition 2. We use the tuple $(T(\cdot),R(\cdot))$ to characterize the computational complexity, where we have: (1) $A(L,p)$ is a deterministic computation at a path p; (2) $T(\cdot)$ is a function for the number of rounds such that each round allows to generate at most $R(\cdot)$ random samples from those uncovered elements; (3) $R(\cdot)$ is a function to be the upper bound the number of random samples each round via the random source $R_{L',\cdot}$; and (4) The algorithm $A(L,\cdot)$ gives $f(n)$-approximation if it returns a sublist L' of sets from L in at least $3/4$ paths p such that $|L'| \leq f(n)\cdot|opt(L)|$, where $opt(L)$ is a sublist of sets in an optimal solution.*

III. Preliminaries

To support the proofs of our algorithm in the following sections, we introduce several lemmas and definitions here.

Lemma 1 (Chernoff Bound [30])**.** *Let $X_1,\cdots,X_s$ be s independent random 0-1 variables and $X = \sum_{i=1}^{s} X_i$. If X_i takes 1 with probability at least p for $i = 1,\cdots,s$, then for any $\delta > 0$, we have*

$$\Pr[X < (1-\delta)\cdot ps] < e^{-\frac{1}{2}\cdot\delta^2 ps}$$

$$\Pr[X > (1+\delta)\cdot ps] < \left[e^{\delta}/(1+\delta)^{1+\delta}\right]^{ps}$$

The Chernoff Bound shows how the number of samples in each round determines the accuracy of the approximation.

A well-known fact in probability theory is the union bound that is represented by the inequality $\Pr(E_1\cup E_2\cup\cdots\cup E_m) \leq \Pr(E_1)+\Pr(E_2)+\cdots+\Pr(E_m)$, where $E_1,E_2,\cdots,E_m$ are m events that may not be independent. In the analysis of our randomized algorithm, there are multiple events such that failure from any of them may fail the entire algorithm. We often characterize the failure probability of each of those events, and use the above inequality to show that the whole algorithm has a small chance to fail, after showing that each of them has a small chance to fail.

There exists a relationship between the number of random edges and the size of an edge set that we need to cover those random edges. For convenience, we define a weight for an edge set F:

Definition 4. *For an edge set $F \subseteq E$, let $V(F)$ be the set of edge e such that e is adjacent to at least one of the edges in F. Then, we define the weight of F as $W(F) = |V(F)|$.*

We are going to prove that there is a chance to expand a set of independent edges if their sum of weight is small.

Lemma 2. *Let $f(n)$ be a function that satisfies $f(n) = o(\log n)$ and $g(n): N \to N$ be a non-decreasing function. Then among $\frac{1}{2}n^{1+1/g(n)}$ random edges, a set of independent edges F with $W(F) \geq m\cdot\left(1-n^{-\frac{1}{2g(n)}}\right)$ can be found with probability at least $1-n\cdot(1/2)^{n^{\frac{1}{2g(n)}}}$, where m is the number of edges in graph G.*

Proof. We use a set A of $\frac{1}{2}n^{1+1/g(n)}$ random edges to generate a subset of independent edges, i.e. edges are disjoint with each other. Partition A into $A_1,\cdots,A_{n/2}$ such that $|A_i| = n^{1/g(n)}$. Let $F_i = \{e_1,\cdots,e_k\}$ be a set of independent edges after processing $A_1,\cdots,A_i$, we prove that each A_{i+1} will bring at least one more independent edge for F if $W(F)$ is not large enough. Now we assume that $W(F_i) < m\cdot\left(1-n^{-\frac{1}{2g(n)}}\right)$. When a random edge (u,v) is generated, (u,v) is not independent of F_j with probability at most $\left(1-n^{-\frac{1}{2g(n)}}\right)$. Thus, with probability at most $\left(1-n^{-\frac{1}{2g(n)}}\right)^{n^{\frac{1}{g(n)}}} = \left(1-n^{-\frac{1}{2g(n)}}\right)^{n^{\frac{1}{2g(n)}}\cdot n^{\frac{1}{2g(n)}}} \leq \left(\frac{1}{2}\right)^{n^{\frac{1}{2g(n)}}}$ (recalling that the function $(1-1/k)^k$ is increasing and has a limit $\lim_{k\to+\infty}(1-1/k)^k = \frac{1}{e} \leq \frac{1}{2}$ $(e \approx 2.7183)$), no edge in A_{i+1} is independent of F_i. If A_{i+1} contains independent edges F' that are also independent of the edges

in F_i, $F_{i+1} = F_i \cup F'$. A set F can contain at most $\frac{n}{2}$ independent edges since there are n vertices in graph G. Thus, after $\frac{1}{2}n^{1+1/g(n)}$ random edges, by union bound, with probability at most $n \cdot (1/2)^{n^{\frac{1}{2g(n)}}}$, each F_i $(1 \le i < \frac{n}{2})$ has weight $W(F_i) < m \cdot \left(1 - n^{-\frac{1}{2g(n)}}\right)$. Thus, with probability at least $1 - n \cdot (1/2)^{n^{\frac{1}{2g(n)}}}$, some F_i $(1 \le i < \frac{n}{2})$ has weight $W(F_i) \ge m \cdot \left(1 - n^{-\frac{1}{2g(n)}}\right)$. □

IV. Algorithm with known number of nodes

Back to our InterAIN problem, we consider the following two cases: (1) the number of nodes is known; and (2) the number of nodes and the number of edges is unknown. We discuss the first case in this section, and discuss the second case in Section V.

Algorithm 1: VC with known nodes

Input: A number n_1 such that $c_1|V| \le n_1 \le c_2|V|$ and $\epsilon \in (0, 1)$.
Output: A vertex cover U of G.

1 $U \leftarrow \emptyset$, $h \leftarrow \left(\frac{n_1}{c_1}\right)^{1+\epsilon}$;
2 **while** *the detector can find uncovered edges in* G **do**
3 Achieve a set H of h uncovered random edges via a detector, where $H \leftarrow Request(U, h)$;
4 Generate a maximal matching P for H by Algorithm 2;
5 $U \leftarrow U \cup P$;
6 **return** U.

For the case that the solver and the detector can get an estimation of the number of nodes, we propose Algorithm 1 to solve the InterAIN problem. Let n_1 be an integer such that $c_1|V| \le n_1 \le c_2|V|$ for two positive real numbers c_1 and c_2. In Algorithm 1, h denotes the number of detected uncovered edges in each round, and ϵ is a parameter of the algorithm to control the number of rounds. The temporal result for the VC problem is stored in a node set U, and it stores the final result when the algorithm terminates. Actually, in line 5 of Algorithm 1, we want to generate a vertex cover for H (a set of uncovered edges), which can be achieved approximately by maximal matching. Thus, "Generate a maximal matching P for H" can be implemented by Algorithm 2 via finding the maximum independent set.

In the theorem below we give out the analysis about the time complexity (in the form shown in Definition 3) of Algorithm 1.

Theorem 1. *For a fixed $\epsilon \in (0, 1)$, there is an $\left(O\left(1/\epsilon\right), n^{1+\epsilon}\right)$ interactive algorithm for the vertex cover with 2-approximation ratio.*

Proof. Let $G_0 = G(V, E)$ be the input graph with m edge. Let $m_1 = m$ and $E_1 = E$. Let E_i be the set of uncovered edges in the beginning of round i. Let $m_i = |E_i|$ be the number

Algorithm 2: VC(H)

Input: A known graph H (a set of edges).
Output: A vertex cover U of H.

1 $U \leftarrow \emptyset$, $H' \leftarrow H$;
2 **while** *there exist edges in* H' **do**
3 Find an arbitrary edge $e = \{u, v\}$ in H';
4 $U \leftarrow U \cup \{u\} \cup \{v\}$;
5 Remove from H' every edge incident on either u or v;
6 **return** U.

of uncovered edge before round i. Let $G_i = (V, E_i)$ be the graph with uncovered edges and the same set of vertices from the input graph. Let $g(n) = \frac{2}{\epsilon}$. Each round the server receives $h \ge n^{1+\epsilon} \ge n^{1+1/g(n)}$ uncovered edges from the client. By Lemma 2, with probability at least $1 - n \cdot (1/2)^{n^{\frac{1}{2g(n)}}}$, we have $m_{i+1} \le m_i \cdot n^{-\frac{1}{2g(n)}}$ for each i with $m_i \ge 1$. Assuming $m_{i+1} \le m_i \cdot n^{-\frac{1}{2g(n)}}$ for each i with $m_i \ge 1$. Let k be the largest integer with $m_k \ge 1$. After $k = 2g(n) = O(1/\epsilon)$ rounds, we have $m_{k+1} \le m_k \cdot n^{-\frac{1}{2g(n)}} \le m_{k-1} \cdot n^{-\frac{2}{2g(n)}} \le \cdots \le m_1 \cdot n^{-\frac{k-1}{2g(n)}}$. Since $m_1 \le n^2$, we have $k \le 4g(n)+2 = O\left(1/\epsilon\right)$. Among the first $4g(n) + 2$ rounds, by union bound, with probability at most $(4(g(n)+2)n \cdot (1/2)^{n^{\frac{1}{2g(n)}}} \le \frac{1}{4}$ for a large n, there is $i \le 4g(n)+2$ such that $m_{i+1} > m_i \cdot n^{-\frac{1}{2g(n)}}$ for some i with $m_i \ge 1$. With probability at least $1 - \frac{1}{4} = \frac{3}{4}$, we get an approximation solution, which is the set of vertices over a subset of disjoint edges in the input graph, for the vertex cover problem. As the output of the approximate solution is derived from the set of nodes in set U of independent edges, it is 2-approximation for the vertex cover since every optimal solution must have at least one vertex from each edge in U. □

Definition 5. *For an undirected graph $G(V, E)$, and two subsets of edges $S_1, S_2 \subseteq E$, define $I(S_1, S_2)$ to be the subset of edges (u, v) in S_1 such that neither vertex u nor vertex v belongs to any edge in S_2. In other words, $I(S_1, S_2)$ is the set of edges in S_1 that are independent of all edges in S_2.*

Then, we derive a lower bound for the number of rounds in Theorem 2, which almost matches the upper bound. We assume that each path only picks the nodes that are from the edges. This is because the solver does not know the name of the nodes. It is natural since the vertices for the VC are from the edges provided by the detector. This model covers the model in Section III.

In Theorem 2, we split the graph $G = (V, E)$ into subgraphs $G_1(V_1, E_1), \cdots, G_k(V_k, E_k)$ with $|V_1| = \cdots = |V_{c+1}|$, and the degree of a node in G_i is $n^{2(c+2-i)\epsilon}$, where c is $\Omega(1/\epsilon)$, and used to be a lower bound for the number of rounds. We will prove that in the round i, the edges in G_j with $i < j$ have a small portion to be sampled if the number of uncovered random edges is bounded by $n^{1+\epsilon}$. Each computation path

only output vertex cover with vertices from the edges sampled. This will imply that some edges are not covered by the vertices being selected.

Lemma 3. *Let n and h be two positive integers with $h < \frac{n}{2}$. Then there is an undirected graph $G = (V, E)$ such that $|V| = n$ and each vertex v of G satisfies $degree(v) \in [h, 2h]$.*

Proof. Let G be n vertices $v_0, v_1, \cdots, v_{n-1}$. For i from 0 to $n-1$, if v_i has less than $t = degree(v) < h$ edges, add $(h-t)$ edges (v_i, v_j) for $j = (i+1)(\mathrm{mod}\ n), (i+2)(\mathrm{mod}\ n), \cdots, (i+(h-t))(\mathrm{mod}\ n)$. It is easy to see that the degree of each vertex is in the range $[h, 2h]$ since each vertex v_i only connects to at most $2h$ vertices $v_{(i+j)(\mathrm{mod}\ n)}$ with $j \in [-h, h]$. □

Theorem 2. *For each $\epsilon \in (0, 1)$, it needs $\Omega(1/\epsilon)$ rounds if there are $n^{1+\epsilon}$ uncovered random edges being used in each round.*

Proof. To simplify the discussion at follows, we denote $\Phi = \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}$. Note that $\Phi(\delta)$ is always strictly less than 1 for all $\delta > 0$. This can be verified by checking that the function $f(x) = (1+x)\ln(1+x) - x$ is increasing and $f(0) = 0$. This is because $f'(x) = \ln(1+x)$ which is strictly greater than 0 for all $x > 0$. Let $c = \lceil \frac{1}{100\epsilon} \rceil$. We are going to prove it needs at least c rounds. Let $G = (V, E)$ and $n = (c+1)n'$ for some integer n' large enough such that

$$64(c+2)^2 \leq n^{\epsilon} \tag{1}$$

$$\Phi(1)^{\sqrt{n}} < 1/\left[100(c+1)^2\right] \tag{2}$$

Partition V into $V_1, \cdots, V_k$ with $k = c+1$, and $|V_1| = \cdots = |V_k| = n'$. Each vertex $v \in V_i$ has a degree $d(v) \in [n^{2(c+2-i)\epsilon}, 2n^{2(c+2-i)\epsilon}]$, and is adjacent to the vertices in V_i, which has been proven in Lemma 3. Let $G_i(V_i, E_i)$ be the subgraph of G with vertices in V_i.

Let F_i be the set of sampled edges from round 1 to round i in the algorithm. Let U_i be the set of vertices to form a partial vertex cover solution after round i, and is the set U right after round i in the algorithm 1. Let $d_i = \frac{(c+2)-i}{2(c+2)}$. Assume that after round $i-1$ ($i \geq 1$), there are at least $d_i n^{(c+2-j)\epsilon} n'$ edges in subgrap G_j to be independent of the sampled edges for all $j \geq i$. In other words, $|I(E_j, F_{i-1})| \geq d_i n^{(c+2-j)\epsilon} n'$ (see Definition 5) for all $j \geq i$. After round $i-1$, a partial vertex cover U_i only contains the vertices from the edges in F_{i-1}. The set $I(E, F_{i-1})$ is a subset of uncovered edges in G. In round i, the probability for getting the edges from V_j with $i < j$ is at most $P_1 \leq \frac{|E_j|}{|I(E_i, F_{i-1})|} \leq \frac{2n^{2(c+2-j)\epsilon} n'}{2d_i n^{2(c+2-i)\epsilon} n'} = \frac{1}{d_i n^{2(j-i)\epsilon}}$. It is trivial for $i = 1$ since no edge has been randomly sampled in round 0. By Lemma 1, with $n^{1+\epsilon}$ random edges selected at round i, there are more than $2n^{1+\epsilon}P_1 \leq \frac{2n^{1+\epsilon}}{d_i n^{2(j-i)\epsilon}}$ to be selected from V_j ($j > i$) with probability at most

$$\begin{aligned}
\Phi(1)^{n^{1+\epsilon} \cdot P_1} &\leq \Phi(1)^{n^{1+\epsilon} \cdot \frac{1}{d_i n^{2(i-j)\epsilon}}} \leq \Phi(1)^{\frac{n^{1-(2(j-i)-1)\epsilon}}{d_i}} \\
&\leq \Phi(1)^{\frac{n^{1-(2(c+1)-1)\epsilon}}{2}} \leq \Phi(1)^{n^{\frac{1}{2}}} < \Phi(1)^{n^{\frac{1}{2}}} \\
&< 1/\left[100(c+1)^2\right] \quad \text{(by Ineqn. 2).}
\end{aligned}$$

Assume that there is no more than $\frac{n^{1+\epsilon}}{d_i n^{2(j-i)\epsilon}}$ be selected from V_j in the round i. Each edge can touch at most $4n^{2(c+2-j)\epsilon}$ edges. Thus, the total number edges in V_j touched with $i < j$ by those picked edges from G_i is at most

$$\begin{aligned}
4n^{2(c+2-j)\epsilon} \cdot \frac{n^{1+\epsilon}}{d_i n^{2(j-i)\epsilon}} &\leq 4n^{2(c+2-j)\epsilon} \cdot \frac{n^{1-2(j-i)\epsilon+\epsilon}}{d_i} \\
&\leq 4n^{2(c+2-j)\epsilon} \cdot \frac{n^{1-\epsilon}}{d_i} \\
&\leq \frac{1}{2(c+2)} n^{2(c+2-j)\epsilon} n'
\end{aligned}$$

edges, which is based on Ineqn. 1. After round i, V_j still has at least $2d_i n^{2(c+2-j)\epsilon} n' - \frac{1}{2(c+2)} n^{2(c+2-j)\epsilon} n' \geq 2d_{i+1} n^{(c+2-j)\epsilon} n'$ edges definitely.

Therefore, the chance is small to get the edges in V_j, then it needs at least c rounds. We can assume that only a small fraction of nodes at V_j are picked before round j. By Ineqn. 2 and union bound, with probability at most $\frac{1}{100(c+1)^2} < \frac{1}{100}$, after round c, we still have a lot of edges uncovered at G_{c+1}. Thus, the randomized algorithm cannot output a vertex cover with probability at least $1 - \frac{1}{100}$. This is a contradiction. □

V. Algorithm with unknown number of nodes and edges

In this section, we propose an algorithm for the case that the solver and the detector cannot get the number of nodes and edges of the network. Since we lack the knowledge of edges in the network, the first step of the algorithm is to estimate the number of edges in the network, as shown in section V-A. Then, the algorithm is conducted with the result of an estimation, as shown in section V-B.

A. Edge Number Estimation

When interacting with a large graph with unknown parameters (such as the scale of the graph, the average degree of nodes in the graph, and so on), researchers design efficient algorithms to estimate those parameters first, and one of the design ideas is random sampling based on the Birthday Paradox [27], [28]. Similarly, we first estimate the number of edges via random sampling. For a finite set H, generator R_H is a random source to generate elements in H. Actually, R_H is also realized by a round of detection to the network when we use it in estimating the number of edges. Based on the Birthday Paradox, we can compare the value of $\sqrt{|H|}$ with a number p, which is shown in Algorithm 3.

To approximate the value of $\sqrt{|H|}$ based on Algorithm 3, we propose two search algorithms for different search options. When the search starts from a number smaller than $\sqrt{|H|}$, Algorithm 4 is proposed to get the approximation of $\sqrt{|H|}$. During the search, the number goes up via a factor determined by function f each time.

On the other hand, when the search starts from a number larger than $\sqrt{|H|}$, Algorithm 5 is able to estimate the value of it, and function f is utilized to control the search stride. Lemma 4 gives out a bound of the estimation error.

Algorithm 3: Test (R_H, p)

Input: A random source R_H of set H and a number p.
Output: A prediction of whether p is larger than $\sqrt{|H|}$.

Generate a list $L = a_1 \cdots a_p$ of p elements in H by calling R_H;
if *L contains two identical elements* **then**
  return $True$;
return $False$.

Algorithm 4: CountingUp (R_H, p_{start}, f)

Input: A random source R_H of set H, a number p_{start}, and a function $f : N \to N$.
Output: A number p to approximate $\sqrt{|H|}$.

$p_1 \leftarrow p_{start}$, $i \leftarrow 1$;
while *Test (R_H,p_i) is* $False$ **do**
  $p_{i+1} \leftarrow p_i \cdot f(p_i)$, $i \leftarrow i + 1$;
return p_i.

Lemma 4. *Given a non-decreasing function f with $f(k) \geq 2$ for any integer k, we have the following facts about the performance of the algorithms (set $\sqrt{|H|} = m$): (1) If $p_{start} \leq \sqrt{m}f(m)$, then with probability at most $e^{-f(m)} + \frac{1}{4f(m)^2}$, CountingUp ($R_H$,$p_{start}$,$f$) returns an $m' \notin [\sqrt{m}/f(m), \sqrt{m}f(m)]$; and (2) If $p_{start} \geq \sqrt{m}/f(m)$, then with probability at most $e^{-f(m)} + \frac{1}{4f(m)^2}$, CountingDown ($R_H$,$p_{start}$,$f$) returns an $m' \notin [\sqrt{m}/f(m), \sqrt{m}f(m)]$.*

Proof. For $a_1, \cdots, a_k$ be k elements independently generated by R_H, we first show that with probability at most $e^{-k(k-1)/m}$, they are all different. The probability that all k elements are different is

$$\begin{aligned} Q_k &= \frac{m(n-1)\cdots(m-k+1)}{m^k} \\ &= 1 \cdot \left(1 - \tfrac{1}{m}\right)\left(1 - \tfrac{2}{m}\right)\cdots\left(1 - \tfrac{k-1}{m}\right) \\ &\leq e^{-\frac{1}{m} - \frac{2}{m} - \cdots - \frac{k-1}{m}} = e^{-\frac{k(k-1)}{m}} \end{aligned}$$

The probability that at least two of k elements are the same is

$$\begin{aligned} P_k &= 1 - \frac{m(m-1)\cdots(m-k+1)}{m^k} \\ &= 1 - 1 \cdot \left(1 - \tfrac{1}{m}\right)\left(1 - \tfrac{2}{m}\right)\cdots\left(1 - \tfrac{k-1}{m}\right) \\ &< 1 - \left(1 - \tfrac{k}{m}\right)^k \end{aligned}$$

By Taylor expansion, we have $\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \cdots - \frac{x^k}{k} - \cdots$. Then, $(1-k/m)^k = e^{k\ln(1-k/m)} = e^{k\left(-k/m - \frac{1}{2}(k/m)^2 - \cdots\right)} = e^{(k^2/m)\left(-1 - \frac{1}{2}(k/m) - \cdots\right)} > e^{-\frac{k^2}{2m}} > 1 - \frac{k^2}{4m}$. Thus, $P_k < \frac{k^2}{4m}$. Let k_0 be the largest with $m_k \leq \sqrt{m}/f(m)$ and k_1 be the least with $m_{k_1} \geq \sqrt{m}f(m)$. The probability that it returns an $m' < \sqrt{m}/f(m)$ is at most $\sum P_k < \sum \frac{k^2}{4m} \leq \sum P_{k_0} < \frac{k_0^2}{4m}\sum \frac{1}{2^i} \leq \frac{k_0^2}{2m}$. In the case that $k < \sqrt{m}/f(m)$, we have $\sum P_k < \frac{1}{4f(m)^2}$. The probability it returns an $m' > \sqrt{m}f(m)$ is at most $e^{-k(k-1)/m}$. Let k_1 be the least with $k_1 \geq \sqrt{m}f(m)$. In the case that $k \geq \sqrt{m}f(m)$, we have $P_{k_1} < e^{-f(m)}$. Then, the probability that it returns an $m' \geq \sqrt{m}f(m)$ is at most $\sum Q_k < \sum e^{-k(k-1)/m} \leq 2e^{-k_1(k_1-1)/m}$. In the case that $k_1 \geq \sqrt{m}f(m)$, we have $\sum Q_k < 2e^{-f(m)^2/2}$. Thus, the probability that it returns an $m' < \sqrt{m}/f(m)$ is at most $P_{k_0} < \frac{k_0^2}{4m}$. □

Algorithm 5: CountingDown (R_H, p_{start}, f)

Input: A random source R_H of set H, a number p_{start}, and a function $f : N \to N$.
Output: A number p to approximate $\sqrt{|H|}$.

$p_1 \leftarrow p_{start}$, $i \leftarrow 1$;
while *Test (R_H,p_i) is* $True$ **do**
  $p_{i+1} \leftarrow p_i/f(p_i)$, $i \leftarrow i + 1$;
return p_i.

Through the above method, we can approximate the size of a set by random sampling.

B. Algorithm with Edge Number Estimation

Firstly, we discuss some proprieties of the functions used in the algorithm.

Definition 6. *Let $g(n), f(n) : N \to N$ be two functions. We define $f^{(1)}(m) = f(m)$ and $f^{(i+1)}(m) = f(f^{(i)}(m))$. If $g(n)$ is a non-decreasing function, $g(n) \geq 1$, and $g(n^2) \leq cg(n)$ for all $n \in N$ for some fixed $c > 0$, then $g(n)$ is slow.*

According to Definition 6, the following three statements are easy to be verified:

1) Function $\log n$ is a slow function.
2) If $g_1(\cdot)$ and $g_2(n)$ are slow functions, then so is $g_1(g_2(n))$.
3) If $g(n) = \max\left(1, \log^{(k)} n\right)$ for some integer k and $f(n) = n^{1/g(n)}$, then $f(n^2)^{c_1} \leq f(n) \leq f(\sqrt{n})^{c_2}$ for some fixed positive c_1 and c_2.

To control the number of sampled edges, we define the following functions according to Definition 6 with constant parameters d_1, and d_2:

$$d_1 = c/2,\ d_2 = c^2/4 \tag{3}$$

$$g(n) = \log\log\log n \tag{4}$$

$$f(n) = \max\left(2, n^{1/g(n)}\right) \text{ and } f_1(n) = (f(n))^{d_2} \tag{5}$$

Now, the algorithm for the VC problem with unknown nodes and edges has been formulated, which is shown in Algorithm 6.

In Algorithm 6, m_i is an estimation of the number of edges in the reserving graph $G_i = (V_i, E_i)$ (as defined in Definition

Algorithm 6: VC with unknown nodes and edges

Input: A random source $R_{G,U}$ for uncovered edges in G by U.

Output: A vertex cover U of G.

1 $U \leftarrow \emptyset$, $i \leftarrow 0$, $q \leftarrow 0$;
2 $m_0 \leftarrow$ CountingUp ($R_{G,\emptyset}$, 1, f_1);
3 **while** *the detector can find uncovered edges in* G **do**
4 $\quad h \leftarrow m_i f(m_i)^q$;
5 $\quad$ Achieve a set H of h uncovered random edges by $R_{G,U}$ via a detector;
6 $\quad$ Generate a maximal matching P for H by Algorithm 2;
7 $\quad U \leftarrow U \cup P$;
8 $\quad m_{i+1} \leftarrow$ CountingDown ($R_{G,\emptyset}$, m_i, f_1);
9 $\quad$ **if** $(m_{i+1} f(m_i)^{d_1} \geq m_i)$ **then**
10 $\quad\quad q \leftarrow q + 1$;
11 $\quad i \leftarrow i + 1$;
12 **return** U.

1). $G_0 = G$ represents the whole network and m_0 denotes the estimation of the number of edges in G. Different from the strategy in Algorithm 1, here the parameter h (used to control the number of requested edges in each round) is calculated by m_i, which is corresponding to the size of the reserving graph. Since the number of edges covered in each round is much more than that of requested edges, the strategy in Algorithm 6 is able to reduce the total number of edges achieved from the detector. On the other hand, we propose a self-increasing policy for h in case the algorithm may struggle in a loop instead of terminating properly, as shown in line 9 and 10 in Algorithm 6. Each time the condition denoted in line 9 is triggered, the counter q increases itself by 1, and the next round h will upgrade with a qth power of the self-increasing function f.

Lemma 5. *Let $f(n)$ and $g(n)$ be defined as Eqn. 5 and Eqn. 4. Let n be a number and $a_1, a_2, \cdots, a_k, \cdots$ be a series of number with $a_{i+1} = f(a_i)a_i$. Then, we have $n \leq a_k$ with $k = O(g(n) \log\log n)$.*

Proof. Assuming that a_i is in the range $\left[2^{2^j}, 2^{2^{j+1}}\right)$, for a number m, it is easy to see that $(f(m)^{g(m)}) \geq m$ by the definition of $f(\cdot)$ and $g(\cdot)$. Since $g(\cdot)$ is not decreasing, we have

$$f(m) \cdot f(f(m)) \cdots f^{(g(m))}(m) \geq m \tag{6}$$

It takes at most $g(2^{2^j})$ steps, thus we have $a_{i+j} \geq 2^{2^{j+1}}$. Therefore, if n is in the range $\left[2^{2^t}, 2^{2^{t+1}}\right)$, we have $a_k \geq n$ for $k = \sum_{j=1}^{t} g(2^{2^j}) \leq g(n)(\log\log n)$. □

Theorem 3. *Let $f(n)$ and $g(n)$ be defined as Eqn. 5 and Eqn. 4, and $g(\cdot)$ be slow. There is a $\left(O(g(n)\log\log n), n^{1+O(1/g(n))}\right)$ interactive algorithm for the vertex cover with 2-approximation ratio with unknown number of nodes and edges.*

Proof. The function $f(\cdot)$ satisfies the property that $f(m^2)^{c_1} \leq f(m) \leq f(\sqrt{m})^{c_2}$ for some fixed positive c_1 and c_2 since $g(\cdot)$ is a slow function. Let $f(n)$ and $g(n)$ be defined as Eqn. 5 and Eqn. 4. The number of uncovered random edges is controlled by the variable h that is not decreasing. The approximation for the number of edges follows from Lemma 4. Let $G_i(V_i, E_i)$ and $G_{i+1}(V_{i+1}, E_{i+1})$ be two graphs such that G_{i+1} be the one after covering the edges of G_i with U in the algorithm.

Let $I_n = \left[n^{1+1/g(n)}, n^{1+100/g(n)}\right]$. When $h \in I_n$, we have a small error that $|E_i|\left(1 - n^{-\frac{1}{2g(n)}}\right)$ edges are not removed after the new vertex cover released. The number of uncovered edges is $|E_{i+1}| \leq |E_i| \cdot n^{-\frac{1}{2g(n)}}$. If $|E_{i+1}| \leq |E_i| \cdot n^{-\frac{1}{2g(n)}}$, then m_{i+1} is an approximation to $\sqrt{|E_{i+1}|}$ with $m_{i+1} \in \left[\frac{\sqrt{|E_{i+1}|}}{f_1(\sqrt{|E_{i+1}|})}, \sqrt{|E_{i+1}|} f_1(\sqrt{|E_{i+1}|})\right]$. Since $|E_{i+1}| \leq |E_i|$, we have $f(\sqrt{|E_{i+1}|}) \leq f(\sqrt{|E_i|})$. Note $|E_i| \leq n^2$ as n is the number of vertices. Thus, we have

$$\begin{aligned} m_{i+1} &\leq \sqrt{|E_{i+1}|} f_1(\sqrt{|E_{i+1}|}) \leq \sqrt{|E_i|} \cdot n^{-\frac{1}{4g(n)}} \cdot f_1(\sqrt{|E_i|}) \\ &\leq \frac{\sqrt{|E_i|}^{\frac{1}{4g(\sqrt{|E_i|})}}}{\sqrt{|E_i|}} \cdot |E_i|^{\frac{1}{16g(\sqrt{|E_i|})}} \leq \sqrt{|E_i|} \cdot |E_i|^{-\frac{1}{8g(\sqrt{|E_i|})}}. \end{aligned}$$

Now, we assume that $m_{i+1} \cdot f(m_i)^{d_1} \leq m_i$. We are able to show that there is a considerable decreasing for the number of uncovered edges after adding the last U to the set of vertex cover. Based on the above analysis, we can assume that $m_i \in \left[\frac{\sqrt{|E_i|}}{f_1(\sqrt{|E_i|})}, \sqrt{|E_i|} f_1(\sqrt{|E_i|})\right]$ and $m_{i+1} \in \left[\frac{\sqrt{|E_{i+1}|}}{f_1(\sqrt{|E_{i+1}|})}, \sqrt{|E_{i+1}|} f_1(\sqrt{|E_{i+1}|})\right]$, then we can get that $\frac{\sqrt{|E_{i+1}|}}{f_1(\sqrt{|E_{i+1}|})} f\left(\frac{\sqrt{|E_i|}}{f_1(\sqrt{|E_i|})}\right)^{d_1} \leq \sqrt{|E_i|} f_1(\sqrt{|E_i|})$. We note $f_1(n) \leq \sqrt{n}$, thus we have $|E_i|^{\frac{1}{4}} \leq \frac{\sqrt{|E_i|}}{f_1(\sqrt{|E_i|})}$ and $f(|E_i|^{\frac{1}{4}})^{d_1} \leq f\left(\frac{\sqrt{|E_i|}}{f_1(\sqrt{|E_i|})}\right)^{d_1}$. Therefore, we have

$$\begin{aligned} \sqrt{|E_{i+1}|} &\leq \frac{\sqrt{|E_i|} f_1(\sqrt{|E_i|}) \cdot f_1(\sqrt{|E_{i+1}|})}{f(|E_i|^{1/4})^{d_1}} \\ &\leq \frac{\sqrt{|E_i|} f_1^2(\sqrt{|E_i|})}{f(\sqrt{|E_i|})^{\frac{d_1}{4}}} \leq \frac{\sqrt{|E_i|}}{f(\sqrt{|E_i|})^{\frac{d_1}{4} - 2d_2}}. \end{aligned}$$

The number of failure times for the edge reduction is at most $O(g(n))$. Each failure will cause h to be increased by a factor of $f(m_0)$ according to line 8 in the Algorithm 6. After $O(g(n))$ times, h will get to the range in I_n. By Lemma 2, the failure probability is at most $P_1 = n \cdot (1/2)^{n^{\frac{1}{2g(n)}}}$, and the total failure probability is at most $100g(n)P_1 \leq 100g(n)n \cdot (1/2)^{n^{\frac{1}{2g(n)}}}$. Before h enters I_n, there are $O(g(n))$ times to call CountingDown $(\cdot)$ that does not return $m_i \leq \frac{m_{-1}}{f(m_{i-1})}$. If it does not return a $m_i \leq \frac{m_{-1}}{f(m_{i-1})}$, the function Test $(\cdot)$ is called at most one time. Therefore, there are $O(g(n))$ times to call Test $(\cdot)$ before h enters I_n. If $h \in I_n$, $|E_{i+1}| \leq |E_i| \cdot n^{-\frac{1}{2g(n)}}$

with failure probability at most P_1. There are at most $O(g(n))$ times calling CountingDown $(\cdot)$ at line 11 in the Algorithm 6 does not return $m_i \leq \frac{m_{-1}}{f(m_{i-1})}$. After h enters I_n, CountingDown $(\cdot)$ does not return $m_i \leq \frac{m_{-1}}{f(m_{i-1})}$ each time P_1. Thus, the total failure probability is at most $100g(n)P_1 \leq 100g(n)n \cdot (1/2)^{n^{\frac{1}{2g(n)}}}$. There are $O(g(n)\log\log n)$ times to return $m_i \leq \frac{m_{i-1}}{f(m_{i-1})}$. Therefore, the number of rounds is also $O(g(n)\log\log n)$ by Lemma 5. □

VI. Experiments

In this section, we evaluate our algorithms via a series of evaluations on four real-world datasets.

A. Experiment Settings

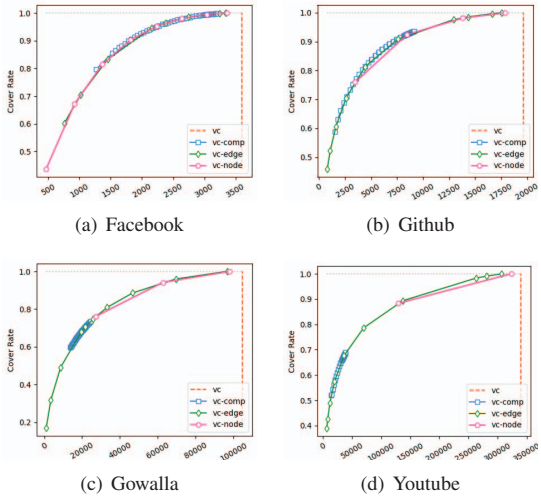
We choose four real-world datasets to evaluate our algorithms, which include graph datasets from Facebook [31] , Github [32], Gowalla [33], and Youtube [34]. All datasets are available at the SNAP website [2]. The statistics of the datasets are shown in Table I. Facebook is a social media site, and the dataset collects the "circles", or "friend lists", of the users from Facebook. Github is a sharing site for computer program developers, where nodes are developers who have shared at least 10 repositories and edges are mutual follower relationships between them. Gowalla is a location-based social networking website, of which the dataset contains the friendship network. Youtube is a video-sharing website that includes a social network, of which the dataset contains the friendship of members on the website.

TABLE I
The Statistics of the Data Sets

Data set	Node num.	Edge num.	Average degree
Facebook	4,039	88,234	21.8
Github	37,700	289,003	7.7
Gowalla	196,951	950,327	4.8
Youtube	1,134,890	2,987,624	2.6

We evaluate the two main algorithms of this paper. The first algorithm is for the case that we have an approximation of the number of nodes, as shown in Algorithm 1. We refer to it as "vc-node" algorithm and set $h = 0.05n^{1.2}$. The second algorithm is for the case that we know neither the number of nodes nor the number of edges, as shown in Algorithm 6. We refer to it as "vc-edge" algorithm. For the parameters and functions of the algorithm, we set the functions corresponding to the self-increasing policy as $g(x) = 5\log\log\log(x)$ and $f(x) = n^{1/g(x)}$. Both algorithms are compared to the result of the 2-approximation vertex cover algorithm shown in Algorithm 2 (for which the topology of the graph is known). In addition, to evaluate the effectiveness of the self-increasing policy in vc-edge algorithm, we compare it to an algorithm that discards the policy denoted in line 9 and 10 of Algorithm 6. We refer to it as "vc-comp" algorithm. The evaluation results of those algorithms in real-world datasets are as follows.

[2]Stanford Network Analysis Project (SNAP): http://snap.stanford.edu



(a) Facebook (b) Github

(c) Gowalla (d) Youtube

Fig. 3. Number of Nodes Chosen in Each Round

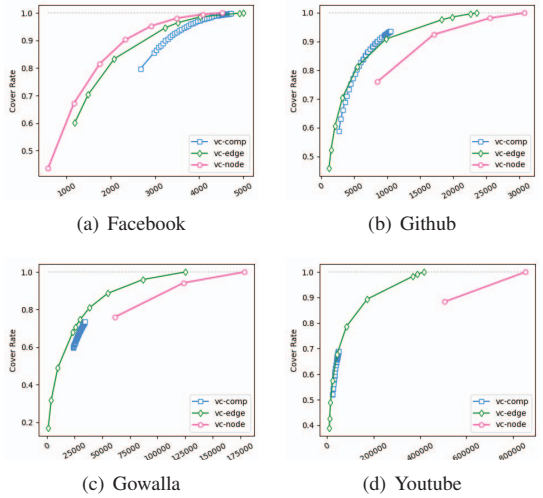
B. Evaluation Results in Real-world Datasets

Evaluation results in the datasets are shown in Table II. In particular, the algorithm without the self-increasing policy (i.e. vc-comp algorithm) loops instead of outputting the final solution, so we run the algorithm with 50 rounds and show the final cover rate in the table. As shown in Table II, except vc-comp algorithm, all of the algorithms accomplish generating a vertex set to cover the whole graph, of which the number of chosen nodes is also similar. We have proposed that both vc-node algorithm and vc-edge algorithm generate a 2-approximation result (in Section IV, V), and here we can find that the result generated by those algorithms are not worse than that of the original vertex cover algorithm, which corresponds to the theoretical analysis. From Table II we can also find that both algorithms only need a small proportion of edges (which are provided by the detector randomly) while returning a vertex cover of the whole graph. That is to say, in situations similar to that in real-world social media, our algorithms are able to hide most of the topology information.

TABLE II
Results of Algorithms in Real-world Datasets

Data set	Algorithm	Node chosen	Edge requested
Facebook	vc	3,714	(All)
	vc-node	3,596	3,105
	vc-edge	3,594	3,347
	vc-comp (99.9%)	3,456	2,405
Github	vc	19,667	(All)
	vc-node	17,882	30,817
	vc-edge	17,540	23,582
	vc-comp (93.6%)	9,135	10,492
Gowalla	vc	104,761	(All)
	vc-node	97,821	178,750
	vc-edge	96,731	125,146
	vc-comp (73.5%)	24,441	34,308
Youtube	vc	339,992	(All)
	vc-node	323,358	854,893
	vc-edge	306,904	416,509
	vc-comp (68.9%)	39,021	48,402

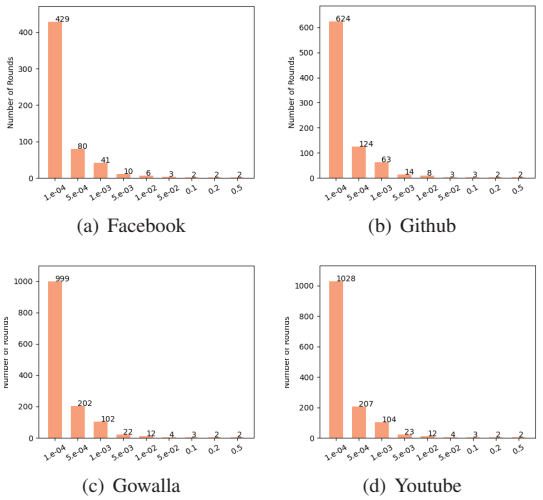
Figure 3 shows the trend between the cover rate and the number of vertices chosen in the VC set. From the figures, we can find that the same number of nodes corresponds to



(a) Facebook (b) Github

(c) Gowalla (d) Youtube

Fig. 4. Number of Requested Edges in Each Round

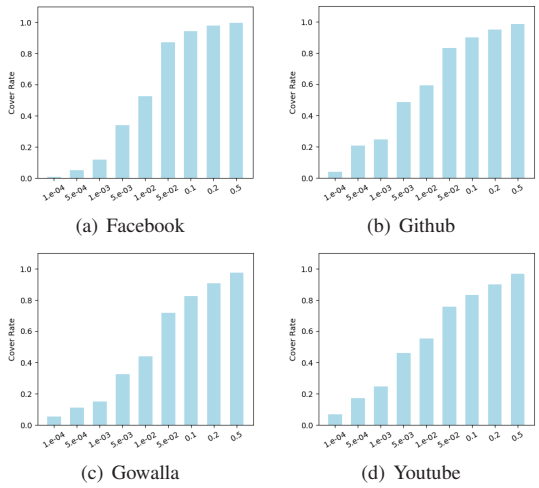


(a) Facebook (b) Github

(c) Gowalla (d) Youtube

Fig. 5. Number of Rounds with Different Proportion of Requested Edge

nearly the same cover rate in all three algorithms. Since in each round of each algorithm, we use the 2-approximation VC algorithm to generate a VC set, the correspondence reflects that the approximation ratio holds after the result of each round aggregates. The figures also denote that vc-node algorithm takes fewer rounds than vc-edge algorithm. The reason lies in that in vc-edge algorithm, we choose the number of requested edges mainly according to the estimation of the number of edges in the reserving graph. In each round, the number of edges covered by the vertex set generated by the algorithm is much larger than the number of edges requested in that round, and the number of requested edges in the next round is readjusted to reduce the total number of requested edges. However, in vc-node algorithm, we cannot know the information of the reserving graph during the conducting of the algorithm. Therefore, there is no readjustment to the number of requested edges. And as a result, the number of requested edges is larger than what the algorithm truly needs, and it takes a bigger step in each round, which causes a decrease in



(a) Facebook (b) Github

(c) Gowalla (d) Youtube

Fig. 6. Cover Rate in the First Round with Different Proportion of Requested Edge

the number of rounds.

Figure 4 shows the trend between the cover rate and the number of requested edges in each round. As shown in the figures, between algorithms that have a similar number of total requested edges, the algorithm that has run more rounds gets a better cover rate. The reason is that in the original graph, edges detected by the detector are likely connected to others, and the size of the independent edge set is small. After several rounds, however, the edges connected to others are more likely to be covered, while edges independent of others remain. Thus, there will be a larger set of independent edges in a set of edges requested in more rounds, and that will generate a larger vertex set of vertex cover and get a higher cover rate.

In order to evaluate the effectiveness of the self-increasing policy, we construct vc-comp algorithm and run the algorithm for 50 rounds (since it will fall into loops). Comparing vc-edge algorithm with vc-comp algorithm, we can find that it is quite frequent for vc-edge algorithm to trigger the self-increasing condition and increase its number of requested edges in the next round. By contrast, vc-comp algorithm becomes nearly standstill in the last several rounds.

In addition, we fix the proportion of requested edges in each round (which is different from the three algorithms above), and discuss the number of rounds conducted in the algorithm. The results are shown in Figure 5. From the figures, we can find that the number of rounds that the algorithm needs before it generates a solution to the whole graph decreases quickly as the proportion of requested edges increases. What is more, the algorithm needs only about 100 rounds to get the final solution when the number of requested edges is larger than 0.1% of that of the total edges.

Vertices with a large degree are more likely to be covered in the early stage, and here we display the cover rate of the network after the first round of the algorithm with a fixed proportion of requested edges, as shown in Figure 6. It indicates that, in a network similar to real-world social media, we can cover over half of the edges in the network with only

about 1% random edges known. With 5% random edges, we are able to cover over 80% edges of the network in a round.

VII. Conclusion & Future Work

Considering real applications in social networks and network topology is hard to obtain, we formulate the InterAIN problem in this paper. To address it, we develop a random hidden input computation model and its corresponding approximation algorithms for the interactive VC problem with an unknown network topology. Based on the real applications in social networks, we categorize them into two different cases: an approximate number of nodes is known and nothing is known. For each case, we conduct a detailed theoretical analysis and give a 2-approximation in the bounded number of rounds and bounded number of random sampling in each round. Through a large number of experiments, a small number of edge sampling can achieve effective vertex coverage, which is an amazing discovery.

In the future, we can further extend it to a larger scope by allowing interaction between a solver and detector that save the whole input and only exposes a small part of the network to the solver. There are many other applications that should be studied along with the concept of hiding input computation. For example, we can consider the problem of the possibility of reconstructing the graph from sampled edges and how much of the graph can be recovered. This kind of solution can be designed in the same line as this paper.

References

[1] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2003, pp. 137–146.

[2] C. Budak, D. Agrawal, and A. El Abbadi, "Limiting the spread of misinformation in social networks," in *International Conference on World Wide Web (WWW)*, 2011, pp. 665–674.

[3] G. Tong, W. Wu, L. Guo, D. Li, C. Liu, B. Liu, and D.-Z. Du, "An efficient randomized algorithm for rumor blocking in online social networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 845–854, 2017.

[4] W. Lu, W. Chen, and L. V. Lakshmanan, "From competition to complementarity: Comparative influence diffusion and maximization," *The VLDB Endowment*, vol. 9, no. 2, 2015.

[5] X. Li, J. D. Smith, T. N. Dinh, and M. T. Thai, "Tiptop:(almost) exact solutions for influence maximization in billion-scale networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 649–661, 2019.

[6] J. Guo and W. Wu, "A novel scene of viral marketing for complementary products," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 4, pp. 797–808, 2019.

[7] J. Guo, T. Chen, and W. Wu, "A multi-feature diffusion model: Rumor blocking in social networks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 386–397, 2020.

[8] G. Tong, R. Wang, Z. Dong, and X. Li, "Time-constrained adaptive influence maximization," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 1, pp. 33–44, 2020.

[9] J. Cheng, K. Yang, Z. Yang, H. Zhang, W. Zhang, and X. Chen, "Influence maximization based on community structure and second-hop neighborhoods," *Applied Intelligence*, pp. 1–16, 2022.

[10] R. Wang, Z. Huang, S. Liu, H. Shao, D. Liu, J. Li, T. Wang, D. Sun, S. Yao, and T. Abdelzaher, "Dydiff-vae: A dynamic variational framework for information diffusion prediction," in *International Conference on Research and Development in Information Retrieval (SIGIR)*, 2021, pp. 163–172.

[11] W. Wang, H. Yin, X. Du, W. Hua, Y. Li, and Q. V. H. Nguyen, "Online user representation learning across heterogeneous social networks," in *International Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2019, pp. 545–554.

[12] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation," in *International Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2019, pp. 235–244.

[13] C. Chen, M. Zhang, C. Wang, W. Ma, M. Li, Y. Liu, and S. Ma, "An efficient adaptive transfer neural network for social-aware recommendation," in *International Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2019, pp. 225–234.

[14] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Springer, 1972, pp. 85–103.

[15] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[16] I. Dinur and S. Safra, "On the hardness of approximating minimum vertex cover," *Annals of Mathematics*, pp. 439–485, 2005.

[17] S. Khot and O. Regev, "Vertex cover might be hard to approximate to within 2- ε," *Journal of Computer and System Sciences*, vol. 74, no. 3, pp. 335–349, 2008.

[18] P. Nguyen and K. Nahrstedt, "Context-aware crowd-sensing in opportunistic mobile social networks," in *International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, 2015, pp. 477–478.

[19] F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, and S. Shan, "On positive influence dominating sets in social networks," *Theoretical Computer Science*, vol. 412, no. 3, pp. 265–269, 2011.

[20] X. Zhu, J. Yu, W. Lee, D. Kim, S. Shan, and D.-Z. Du, "New dominating sets in social networks," *Journal of Global Optimization*, vol. 48, no. 4, pp. 633–642, 2010.

[21] A. Borodin, Y. Filmus, and J. Oren, "Threshold models for competitive influence in social networks," in *International Workshop on Internet and Network Economics*. Springer, 2010, pp. 539–550.

[22] J. Zheng and L. Pan, "Least cost rumor community blocking optimization in social networks," in *International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*. IEEE, 2018, pp. 1–5.

[23] M. Wagner, T. Friedrich, and M. Lindauer, "Improving local search in a minimum vertex cover solver for classes of networks," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 1704–1711.

[24] T. Y. Berger-Wolf and J. Saia, "A framework for analysis of dynamic social networks," in *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2006, pp. 523–528.

[25] S. Çınaroğlu and S. Bodur, "A new hybrid approach based on genetic algorithm for minimum vertex cover," in *Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2018, pp. 1–5.

[26] A. Dasgupta, R. Kumar, and T. Sarlos, "On estimating the average degree," in *International Conference on World Wide Web (WWW)*, 2014, pp. 795–806.

[27] T. Eden, S. Jain, A. Pinar, D. Ron, and C. Seshadhri, "Provable and practical approximations for the degree distribution using sublinear graph samples," in *International Conference on World Wide Web (WWW)*, 2018, pp. 449–458.

[28] D. Ron and G. Tsur, "The power of an example: Hidden set size approximation using group queries and conditional sampling," *ACM Transactions on Computation Theory (TOCT)*, vol. 8, no. 4, pp. 1–19, 2016.

[29] S. Behnezhad, "Time-optimal sublinear algorithms for matching and vertex cover," in *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022, pp. 873–884.

[30] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge university press, 1995.

[31] J. J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2012, pp. 548–556.

[32] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *Journal of Complex Networks*, vol. 9, no. 2, 2021.

[33] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2011, pp. 1082–1090.

[34] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2012, pp. 1–8.