Polling One's Friends: A Graph Theoretic View

Chenguang Zhu[1] and Hossein Karkeh Abadi[2] and Balaji Prabhakar[3]

Abstract— **Polling is an effective method to collect information from a population. However, pollsters are often concerned with unsatisfactory response rate. Incentive is sometimes used to boost response rate, but pollsters want to keep it low. In fact, when there is an underlying social network among responders, a pollster can elicit a high response rate while attempting to minimize polling costs via asking a friend of a user for the user's response to a poll. In this paper, we frame the problem as one that minimizes (i) monetary cost, the money it takes to elicit a response from a potential respondent, and (ii) querying cost, the number of different individuals queried, when polling users of a social network while maximizing the information collected. We consider a variety of graphs and compute the monetary and querying costs of some natural greedy polling algorithms. Our approach allows to conduct a detailed analysis and obtain, sometimes explicit, results on the cost of polling. Empirical results on both synthetic data and real social network data validate our analytic models. In particular, we find out that in the 1.46-billion-edge Twitter social graph, only 22% of the users need to be queried to get information about all the users, while monetary cost of our algorithm can be as low as 38% of that for the strategy to poll every user.**

I. INTRODUCTION

This paper is motivated by two online polls we have conducted. Commuters in the public transit system of a metropolis were asked their Chinese zodiac symbol so that, based on commuting records, one may determine how their answer informs their commuting behavior.[1] The commuters were offered points for answering the poll and the points have a monetary value. The commuters are linked to their friends and family on the online platform. In total, there are over 230,000 participants of whom more than 75% have friends on the online platform. Response rates of 25% have been recorded in these polls.

In the above, and in general, pollsters and surveyors are concerned with response rate, cost, and truthfulness of the responses. The response rate decreases with the length of surveys and the time it takes to answer questions [1]; in this sense, surveys, which tend to be detailed (e.g., customer satisfaction), fare worse than polls which are usually a single multiple choice question. Incentive (monetary or otherwise) can boost response rate as well [2], but surveyors want to keep this low.

Social networks provide another way to increase response rate: ask a friend of a user for the user's response to a poll. Indeed, this feature has been observed and studied in previous work [3], [4], [5]. In [3], the authors study a binary response polling model and a user is assumed to know the preferences of all her friends and express the aggregate response (including her own) as a weighted sum of the individual responses. The goal is to estimate the overall ratio of positive responses by sampling a small number of nodes for their aggregate response and bounding its error. In [5], the authors investigate a random walk model for choosing which nodes to sample so as to reduce the number of samples needed for a given accuracy of prediction; [4], proposes sampling from m independent random walks to lower the estimation errors. In [6], the social context of users is employed indirectly in that voters were asked for their opinions on "who they expect will win an election" as opposed to "who they will vote for", and argued that the first method generates more reasonable estimates in practice. In [7], [8], the effect of social influence from friends on individual users' opinion is studied.

Our motivation and work differs from previous literature in several ways. First, we focus on polls which concern less sensitive information (e.g., the Chinese zodiac poll) or which merely ask a potential respondent for their preference for red cars over blue cars or for roses over tulips. Such questions can have a higher response rate than more sensitive questions; for example, a question about a potential respondent's belief ("Is behavior X ethical?"). More importantly, in the former type of question, a respondent is more likely to know and be willing to truthfully share the response of their friends. Second, we are interested in minimizing polling cost, where cost is defined in two ways: (i) Monetary cost: This is the amount of money it takes to elicit a response from a potential respondent, and this varies across respondents. This cost can be determined empirically, e.g., by A/B testing. In our experiments for transit commuters, there is a lot of side information available on the number of points a commuter needs to be given to perform a particular behavior (gathered over several months of giving behavior incentives). This information can be very helpful in estimating the monetary cost of the users. (ii) Querying cost: This is the number of different potential respondents queried or sampled. This cost matters significantly in mail-in, telephone and direct polls. It also matters in online polls where a user is subject to multiple mini-polls and the goal is to minimize fatigue. Social polling can also help validate the truthfulness of a user's response;

[1]Chenguang Zhu is with Computer Science Department, Stanford University, Stanford, CA, 94305, USA cgzhu@stanford.edu

[2]Hossein Karkeh Abadi is with Electrical Engineering Department, Stanford University, Stanford, CA, 94305, USA hosseink@stanford.edu

[3]Balaji Prabhakar is with Electrical Engineering Department, Stanford University, Stanford, CA, 94305, USA balaji@stanford.edu

[1]The details of the poll are withheld to preserve anonymity. Note that there are 12 Chinese zodiac symbols, determined by the year of one's birth. Thus, depending on their response, respondents can be spread out in their ages and this can affect commuting behavior.

for example, a user's response can be validated by comparing the responses given by a number of the user's friends on the user's behalf. We do not explore this issue here.

We model a social network as a graph with each node representing a user and each edge representing a friendship. We make the following assumptions:

(i) User i has a cost of c_i for answering a poll, where the c_i's are i.i.d. across users and also independent of the graph structure.

(ii) User i knows the responses of all her friends and, if asked, is willing to provide the response of each friend at a charge of c_i.

Under these assumptions, our goals are to theoretically study the cost of eliciting high response rates, essentially the cost of eliciting every user's response. We place no restrictions on the responses and we are interested in minimizing both the monetary and querying costs and study algorithms for each purpose. It turns out that minimizing the querying cost involves determining the minimum set of users so that every other user is a friend of one of these users; in other words, the minimum dominating set (MDS) of the graph. In general, determining the MDS is NP-hard and an obvious greedy algorithm is well-known to achieve the best approximation ratio [9]. We use the method of Wormald [10] to determine the size of dominating set found by the greedy algorithm for a variety of random graphs.

Our contributions. We analyze a natural greedy algorithm, Π_m^*, that minimizes the monetary cost of social polling. We explicitly compute the querying cost of this optimal algorithm. We then analyze a well-known greedy approximation algorithm, Π_q^*, for determining the minimum dominating set in random regular graphs and degree bounded graphs. We obtain an explicit solution for directed random r-regular graphs, improving on previous work (reviewed in Section IV) on this problem. Experiments on both synthetic and real social network data validate our analysis and indicate great potential in carrying out the social polling scheme.

The rest of the paper is organized as follows: Section II frames social polling as a formal graph theoretic problem and defines incurred monetary and querying costs; Section III and IV discuss the analytic results of calculating and minimizing monetary and querying costs; Section V presents empirical results on synthetic graphs and social network data; Section VI summarizes the paper.

II. SOCIAL POLLING

Social polling aims to learn a user's preferences or other information either from herself or from one of her friends. Assume there is only one question in the poll. Denote the friendship graph by $G = (V, E), |V| = n$; $(i, j) \in E$ if and only if user i and j are mutual friends (G is an undirected graph), or user j is a friend of user i (G is a directed graph). Let $\mathcal{N}(i)$ be the set $\{i\} \cup \{j : (i, j) \in E\}$, which is the "neighbourhood of i". User i has a cost of c_i for answering the poll question either about herself or about *one* of her friends. Thus, if user i answers the questions of all nodes in $\mathcal{N}(i)$, the pollster pays her $|\mathcal{N}(i)|c_i$ dollars. We assume that the c_i's are i.i.d with some distribution $F_c(\cdot)$.

Let Π denote a social polling policy. That is, given a graph G, Π determines a subset $D_\Pi \subset V$ to poll, and for each user $i \in V$, Π specifies user $q_i \in D_\Pi$ to answer user i's question ($i \in \mathcal{N}(q_i)$). Then, the total *monetary cost* of Π equals

$$C_m(\Pi) = \sum_{i=1}^{n} c_{q_i}.$$

The *querying cost* of Π is the number of distinct users who are asked questions under Π and equals

$$C_q(\Pi) = |D_\Pi|.$$

C_q represents the "effort" involved in contacting users and soliciting their answers. Pollsters aim to minimize both C_m and C_q; however, to do this simultaneously can be hard. We shall investigate policies that aim to minimize one of the costs while keeping the other as low as possible.

Next, we will introduce and analyze policies that minimize the monetary and querying costs in various types of graphs.

III. MINIMIZING MONETARY COST

For a graph G and any monetary costs $\{c_i\}_{i=1}^n$ drawn from $F_c(\cdot)$, observe that the following greedy algorithm, Π_m^* (Algorithm 1), achieves the minimum monetary cost.

Algorithm 1 Π_m^*: minimizing monetary cost in social polling

Require: $G = (V, E)$, monetary cost $\{c_i\}_{i=1}^n$
  $n = |V|$
  $Q_0 = \emptyset, A_0 = \emptyset$
  $t = 0$
  while $A_t \neq V$ **do**
    $t \leftarrow t + 1$
    $i_t = \text{argmin}_{p \in Q_{t-1}^c}\{c_p | \mathcal{N}(p) \cap A_{t-1}^c \neq \emptyset\}$
    Query i_t for answers of all nodes in $\mathcal{N}(i_t) \cap A_{t-1}^c$
    $Q_t \leftarrow Q_{t-1} \cup \{i_t\}$
    $A_t \leftarrow A_{t-1} \cup \mathcal{N}(i_t)$
  end while
  $D_{\Pi_m^*} = Q_t$

In Π_m^*, let $Q_t \subset V$ be the set of nodes who have been queried and let $A_t \subset V$ be the set of nodes whose answers have been obtained after the t^{th} query. In each round, Π_m^* picks the cheapest node i_t which has not been queried before ($i_t \in Q_{t-1}^c$) but can answer the poll for at least one node whose answer has not been obtained ($\mathcal{N}(i_t) \cap A_{t-1}^c \neq \emptyset$). Then, i_t answers the poll for all nodes in $\mathcal{N}(i_t) \cap A_{t-1}^c$.

It is worth noting that Π_m^* optimally minimizes the monetary cost for every instance in the sample space and, hence, in expectation. The expected monetary cost of Π_m^* can be directly computed:

$$E(C_m(\Pi_m^*)) = \sum_{i=1}^{n} E(c_{q_i^*}), \tag{1}$$

where $q_i^* = \text{argmin}_j\{c_j : i \in \mathcal{N}(j)\}$.

Thus, for example, in an undirected r-regular graph of n nodes, the expected minimum monetary cost is $E(C_m(\Pi_m^*)) = nT$, where T is the expected value of the minimum over $r+1$ values i.i.d. from the cost distribution $F_c(\cdot)$.

The expected querying cost of Π_m^* can also be explicitly computed, and is *independent* of the distribution of monetary cost $F_c(\cdot)$. We show an example of undirected random r-regular graphs in the following subsection.

A. The querying cost of Π_m^* for undirected random r-regular graphs

To obtain the expected querying cost of Π_m^*, we determine $p_i = P(\text{node } i \text{ is queried under } \Pi_m^*)$ and note that

$$E(C_q(\Pi_m^*)) = E(|D_{\Pi_m^*}|) = \sum_{i=1}^{n} p_i. \tag{2}$$

To evaluate p_i, it suffices to look at the monetary costs of nodes in $\mathcal{N} = \mathcal{N}(i) \cup \mathcal{N}(\mathcal{N}(i))$, i.e., nodes that are less than or equal to 2 hops from node i.

To proceed, we make two main observations:

1) For a fixed r, a random r-regular graph is locally tree-like in the limit as $n \to \infty$ [11]. In particular, G restricted to the nodes in $\mathcal{N}$ is a depth-two tree $\mathcal{T}_i$ with i at the root.[2]

2) Crucially, note that under Π_m^* the set $D_{\Pi_m^*}$ is determined completely by the rank order of the costs c_j of all the nodes in G. That is, let π be a permutation of $1, ..., n$ such that $c_{\pi(1)} < c_{\pi(2)} < ... < c_{\pi(n)}$. Then, whether $i \in D_{\Pi_m^*}$ depends only on the relative ranking of nodes in $\mathcal{T}_i$. Since $\{c_i\}_{i=1}^n$ are i.i.d from $F_c(\cdot)$, this further implies that $P(i \in D_{\Pi_m^*})$ does not depend on $F_c(\cdot)$.[3]

Assuming that $F_c(\cdot)$ is $U[0,1]$. Let $j_1, ..., j_r$ denote i's neighbors. Define,

$$h = |\{k : c_{j_k} > c_i, 1 \le i \le r\}|.$$

Note that a node i is queried under Π_m^* if and only if (i) $h = r$: i is the cheapest node among all its neighbors, or (ii) $h < r$ and i is the cheapest node among j's neighbors for some neighbor j with $c_j > c_i$. We evaluate the probability in each of these cases.

(i) This occurs with probability $\frac{1}{r+1}$.

(ii) In this case, given h and $c_i = x$, note that

$$P(i \in D_{\Pi_m^*} | c_i = x, h) = 1 - (1 - (1-x)^{r-1})^h. \tag{3}$$

[2]More generally, it has been shown in [11] that random graphs having degree distribution P_k with finite second moment are locally tree-like.

[3]When $F_c(\cdot)$ is a continuous distribution, with probability 1, tie-breaking is not needed. When $F_c(\cdot)$ is a discrete distribution so that there are ties, the nodes can be ordered according to non-decreasing costs and Π_m^* applied to nodes in this order will also give the least monetary cost. So, without loss of generality, we consider continuous costs in the remainder.

It follows that

$$P(i \in D_{\Pi_m^*}) = \frac{1}{r+1} + \sum_{h=0}^{r-1} \binom{r}{h} \int_0^1 l(x)dx, \tag{4}$$

$$l(x) = (1-x)^h x^{r-h} (1 - (1 - (1-x)^{r-1})^h) \tag{5}$$

and $\hat{E}[C_q(\Pi_m^*)] = nP[i \in D_{\Pi_m^*}]$. Here, we use $\hat{E}[C_q(\Pi_m^*)]$ because it assumes that the all nodes in the graph are locally tree-like at depth-two.

In the following theorem, we shall show that the estimation $\hat{E}[C_q(\Pi_m^*)]$ is asymptotically accurate, with an relative error $o(1)$.

Theorem 1. *A node P locally tree-like at depth-two if:*
1. $\nexists Q, R \in V, Q \neq R, \{(P,Q),(P,R),(Q,R)\} \subseteq E$
2. $\nexists Q, R, S \in V, Q \neq R, R \neq S, S \neq Q, \{(P,Q),(P,R),(Q,S),(R,S)\} \subseteq E$

*Define T_n as the number of nodes in an undirected random r-regular graph of n nodes that are **not** locally tree-like at depth-two.*

Assume the monetary costs c_i's are i.i.d. from a distribution $F_c(\cdot)$. If T_n has a finite mean and variance when n goes to infinity, then, the expected relative error of $\hat{E}[C_q(\Pi_m^*)]$ over all undirected random r-regular graphs is $o(1)$:

$$E[\frac{|\hat{E}[C_q(\Pi_m^*)] - E[C_q(\Pi_m^*)]|}{E[C_q(\Pi_m^*)]}] = o(1)$$

Proof: When $T_n = t$, for the $n-t$ nodes which are locally tree-like at depth-two, each of them has the same probability of being selected by Π_m^*: $p = P[i \in D_{\Pi_m^*}]$. $\hat{E}[C_q(\Pi_m^*)]$ correctly calculates the expected number of nodes queried from these $n-t$ nodes. It follows that $E[C_q(\Pi_m^*)] \in [(n-t)p, (n-t)p+t]$, and apparently, $E[C_q(\Pi_m^*)] \ge 1$.

$$E[\frac{|\hat{E}[C_q(\Pi_m^*)] - E[C_q(\Pi_m^*)]|}{E[C_q(\Pi_m^*)]}] \le E[\frac{T_n}{\max\{(n-T_n)p, 1\}}]$$

Now, we use the fact that T_n has a finite mean and variance when n goes to infinity. Then, according to Chebyshev's Inequality, $P(T_n - E[T_n] \ge n^{\frac{2}{3}}) \le \frac{\mathrm{Var}(T_n)}{n^{\frac{4}{3}}}$. Define $a = \lceil E[T_n] + n^{\frac{2}{3}} \rceil$ and $f(n) = E[\frac{T_n}{\max\{(n-T_n)p,1\}}]$. Then,

$$\begin{aligned} f(n) &\le \sum_{t=0}^{a-1} \frac{P(T_n = t)t}{(n-t)p} + \sum_{t=a}^{n} \frac{P(T_n = t)t}{\max\{(n-t)p, 1\}} \\ &\le \frac{E[T_n]}{(n-a)p} + P(T_n \ge a)\frac{n}{p} \\ &\sim O(\frac{1}{n}) + O(\frac{1}{n^{\frac{1}{3}}}) \end{aligned}$$

Theorem 1 can be applied to undirected random regular graphs for the following reason. Corollary 2.19 in [12] states that for an undirected random r-regular graph, when n goes to infinity, the number of loops of length L is asymptotically a Poisson random variable with finite mean: $\frac{(r-1)^L}{2L}$. As any node which is *not* locally tree-like at depth-two must lie on a loop of length $L = 3$ or $L = 4$, T_n has a finite mean and finite variance when n goes to infinity.

IV. MINIMIZING QUERYING COST

For any polling strategy Π, since the question of every node i is answered by some $q_i \in D_\Pi$, and $i \in \mathcal{N}(q_i)$, D_Π forms a dominating set in G. It follows that minimizing the querying cost is equivalent to finding the minimum dominating set. Although the minimum dominating set (MDS) problem is NP-complete [13], a greedy approximation algorithm can achieve an approximation ratio of $O(log\Delta)$ (Δ is the maximum node-degree in the graph); moreover, no algorithm can achieve an approximation ratio less than $O((1-o(1))log\Delta)$ unless NP problems can be solved deterministically in time $n^{O(loglogn)}$ [9]. Next, we shall describe and analyze this greedy algorithm, denoted by Π_q^*. In Π_q^*, every node is uncovered initially. Then, in each step, Π_q^* adds a node i into $D_{\Pi_q^*}$ so that i covers the most number of uncovered nodes. Here, i can cover j if and only if $j \in \mathcal{N}(i)$.

In the remainder of this section we present results for random graphs. We begin with random regular graphs and describe extensions to degree-bounded random graphs. We apply the differential equation method of Wormald [10]. In particular, for directed random regular graphs, we obtain an explicit closed-form solution to the resultant differential equations describing the evolution of the dominating set determined by Π_q^*.

Related work. Previous literature on dominating sets focuses on bounding the size of the minimum dominating set in a variety of graph types (see, for example, [14], [15]). Parekh [16] uses an algebraic method to derive an upper bound on the size of the dominating set given by the greedy algorithm in undirected graphs. In this paper, we use the techniques in Wormald [10] to obtain the *exact size* of dominating set generated by the greedy algorithm for degree-bounded random graphs in the limit as $n \to \infty$. In [10], the differential equation method is developed and used to analyze greedy algorithms for the independent set problem in undirected random regular graphs. The method is employed for the *independent* dominating set problem in random regular graphs [17] and the *connected* dominating set problem in random regular graphs [18]. Howe [19] applies the method to dominating set problem in directed regular graphs. Howe's analysis obtains a complex system of $O(r^2)$ differential equations for r-regular graphs, making it difficult to solve explicitly. Since we evolve the dominating set (more specifically, the greedy algorithm for finding the dominating set) in a simpler manner, our method only has $O(r)$ equations. This allows us to compute the size of the dominating set explicitly for directed r-regular graphs, and also extend to irregular (degree-bounded) graphs.

A. Directed Random Regular Graphs

1) Model: Consider a directed random r-regular graph on n nodes; this is a simple directed graph in which each node has both in-degree and out-degree r. Such a graph can be generated through the pairing model [10], [19]: initially assign r incoming half-edges and r outgoing half-edges to each node and iteratively match half-edges as follows. For each unmatched outgoing half-edge, choose an unmatched incoming half-edge uniformly at random (u.a.r.) and independently of all else and match the two half-edges. With probability $\exp(-\frac{r^2+1}{2}+O(\frac{1}{n}))$, this process generates a random r-regular simple directed graph and any property that holds asymptotically almost surely (a.a.s) for the graph generated by this process also holds a.a.s for r-regular simple directed random graph [19].

Now we combine two processes: (i) generating the directed random r-regular graph, and (ii) determining the dominating set. The directed edges (and half-edges) play a crucial role in executing these two steps. Once the dominating set $D_{\Pi_q^*}$ is found, questions are answered by nodes in the dominating set by taking into account the *monetary cost* of the nodes. The details are as follows.

Random graph generation and dominating set determination using the greedy algorithm Π_q^*.

Time, by $t = 0, 1, 2, ...$, marks the progress of the greedy algorithm Π_q^*: at each time t, one node is added to the dominating set $D_{\Pi_q^*}$. Let $G(t)$ denote the graph at the end of t^{th} time step: $G(t)$ includes all the n nodes, the unmatched half-edges and the matched edges at the end of time t. Let $D(t) \subseteq D_{\Pi_q^*}$ be the set of nodes that have been determined to be in the dominating set according to Π_q^* at the end of time t. Let $\mathcal{U}(t)$ and $\mathcal{C}(t)$ be, respectively, the set of "uncovered" and "covered" nodes by time t; that is, a node $j \in \mathcal{U}(t)$ is "uncovered" if it is not yet in the dominating set, $D(t)$, nor has an *incoming* edge from a node in $D(t)$ ($\mathcal{C}(t)$ is the complement of $\mathcal{U}(t)$). Let $\mathcal{U}_i(t), i = 0, ..., r$ and $\mathcal{C}_i(t), i = 0, ..., r$ be the set of uncovered and covered nodes, respectively, at the end of time t which have precisely i matched outgoing half-edges. Note that $\mathcal{U}(t)$ ($\mathcal{C}(t)$) is the disjoint union of the $\mathcal{U}_i(t)$ (respectively, the $\mathcal{C}_i(t)$). Table I summarizes the notations.

Initially, $\mathcal{U}(0) = \mathcal{U}_0(0) = V$ and $G(0)$ is the graph with n nodes each with r incoming and outgoing half-edges.

$\mathbf{t = 0}$: Π_q^* picks a node k randomly from $\mathcal{U}_0(0)$ and matches each of its r outgoing half-edges to an incoming half-edge chosen u.a.r. from the existing set of incoming half-edges. Denote by $\mathcal{N}_c(k)$ the nodes newly covered by k as a result of the previous step. Further, Π_q^* matches each incoming edge of each node in $\{k\} \cup \mathcal{N}_c(k)$ to an outgoing half-edge chosen u.a.r. from the existing set of outgoing half-edges. Sets are updated as follows:

$D(1) \leftarrow D(0) \cup \{k\}$; k is moved from $\mathcal{U}_0(0)$ to $\mathcal{C}_r(1)$; $G(1)$ and $\mathcal{U}_j(1)$ and $\mathcal{C}_j(1), 0 \le j \le r$ are updated appropriately.

General t: Place $\mathcal{U}_j(t)$ and $\mathcal{C}_j(t)$ in the following order: $\mathcal{U}_0(t), \mathcal{C}_0(t), ..., \mathcal{U}_r(t), \mathcal{C}_r(t)$. Let S be the first nonempty set in this order, and note that S is the set of nodes that have the largest number of unmatched *outgoing* half-edges—the correct set of candidate nodes for Π_q^* to choose from. Say that Π_q^* picks a node $k \in S$.

Case 1. $k \in \mathcal{U}_h(t)$ for some h. Each of the $r-h$ unmatched outgoing half-edges is connected u.a.r. to an incoming half-edge in $G(t)$, and k is moved from $\mathcal{U}_h(t)$ to $\mathcal{C}_r(t+1)$. Each incoming half-edge of each node in $\{k\} \cup \mathcal{N}_c(k)$ is matched u.a.r. to an outgoing half-edge from the set of available outgoing half-edges. All sets are updated appropriately.

TABLE I: Notations in analyzing Π_q^* for directed random regular graphs

t	Time step (by time t, t nodes have been selected into $D_{\Pi_q^*}$)
$G(t)$	The graph at the end of t^{th} time step including unmatched half-edges and matched edges
$D(t)$	The set of t nodes selected into $D_{\Pi_q^*}$ by time t
$\mathcal{U}(t)$	Uncovered nodes by time t (nodes that are not in $D(t)$ nor has an incoming edge from any node in $D(t)$)
$\mathcal{C}(t)$	Covered nodes by time t (nodes that are in $D(t)$ or has an incoming edge from any node in $D(t)$)
$\mathcal{U}_i(t)$	Uncovered nodes by time t which have precisely i matched outgoing half-edges
$\mathcal{C}_i(t)$	Covered nodes by time t which have precisely i matched outgoing half-edges

Case 2: $k \in \mathcal{C}_h(t)$ for some h. Similar operations as above, except that as k has already been covered, all of its incoming half-edges have been connected.

The algorithm terminates at the first time T such that $\mathcal{C}_r(T) = V$. And $D_{\Pi_q^*} = D(T)$.

Remark 1. The graph at the termination of the algorithm is a directed r-regular random graph since all the half-edges have been matched independently and u.a.r.

Remark 2. The dominating set $D_{\Pi_q^*}$ is indeed the result of the greedy algorithm because Π_q^* picked nodes that had the largest number of available outgoing half-edges and connected them to nodes that were previously uncovered (recall that a covered node has all of its incoming edges matched). Hence, Π_q^* covered the largest number of uncovered nodes at each step.

2) Querying cost of Π_q^*: estimating the size of $D_{\Pi_q^*}$: We now analyze the evolution of the size of sets $\mathcal{U}_j(t)$ and $\mathcal{C}_j(t)$ in the greedy algorithm Π_q^*. The algorithm proceeds in phases of time; in each phase it picks nodes from one of the sets in the following ranked order $(\mathcal{U}_0(t), \mathcal{C}_0(t), ..., \mathcal{U}_r(t), \mathcal{C}_r(t))$. Say that in Phase i(a) Π_q^* picks nodes from $\mathcal{U}_i(t)$ and in Phase i(b) it picks nodes from $\mathcal{C}_i(t)$. We characterize the dynamics of each phase. Let $U(t) = |\mathcal{U}(t)|$, $C(t) = |\mathcal{C}(t)|$, $U_i(t) = |\mathcal{U}_i(t)|$ and $C_i(t) = |\mathcal{C}_i(t)|$ be the sizes of the corresponding sets. Define $V(t) = \sum_{j=0}^{r}[(r-j)(U_j(t) + C_j(t))]$.

Proposition 1. *Let $\Delta_{U_k}(t) = E[U_k(t+1) - U_k(t)|G(t)]$ and $\Delta_{C_k}(t) = E[C_k(t+1) - C_k(t)|G(t)]$ be the expected change of $U_k(t)$ and $C_k(t)$, $i \le k \le r$, conditioned on $G(t)$ during Phase i(a), then*

$$\Delta_{U_k}(t) = -\delta_{k=i} - (r-i)\frac{U_k(t)}{U(t)} - (r^2 - ir + i)Q(t) + o(1),$$

$$\Delta_{C_k}(t) = \delta_{k=r} + (r-i)\frac{U_k(t)}{U(t)} - (r^2 - ir + i)W(t) + o(1),$$

$$Q(t) = (r-k)\frac{U_k(t)}{V(t)} - (r-k+1)\delta_{k>0}\frac{U_{k-1}(t)}{V(t)},$$

$$W(t) = (r-k)\frac{C_k(t)}{V(t)} - (r-k+1)\delta_{k>0}\frac{C_{k-1}(t)}{V(t)}.$$

Proof: At time $t+1$ in Phase i(a), say that node P is selected from $\mathcal{U}_i(t)$. Since P has $r-i$ outgoing half-edges and all uncovered nodes have r incoming half-edges, it follows that in $\mathcal{U}_k(t)$, the expected number of nodes chosen as one of P's $r-i$ outgoing neighbors is $(r-i)\frac{U_k(t)}{U(t)}$.

Next, Π_q^* matches the $r + (r-i)(r-1) = r^2 - ir + i$ incoming half-edges of the above $r-i+1$ newly covered nodes u.a.r. to existing outgoing half-edges in $G(t)$. The probability that a matching outgoing half-edge e comes from a node S in $\mathcal{U}_k(t)$ and $\mathcal{C}_k(t)$ is $\frac{(r-k)U_k(t)}{V(t)}$ and $\frac{(r-k)C_k(t)}{V(t)}$, respectively. S moves to $\mathcal{U}_{k+1}(t+1)$ if $S \in \mathcal{U}_k(t)$, and moves to $\mathcal{C}_{k+1}(t+1)$ if $S \in \mathcal{C}_k(t)$. Putting the above together completes the proof.[4]

Setting $x = t/n$, $u_k(x) = U_k(t)/n$, $c_k(x) = C_k(t)/n$, $i \le k \le r$, $u(x) = U(t)/n$, $v(x) = V(t)/n$, $q(x) = Q(t)/n$, $w(x) = W(t)/n$, and letting $n \to \infty$, we obtain

$$\frac{du_k(x)}{dx} = -\delta_{k=i} - (r-i)\frac{u_k(x)}{u(x)} - (r^2 - ir + i)q(x), \tag{6}$$

$$\frac{dc_k(x)}{dx} = \delta_{k=r} + (r-i)\frac{u_k(x)}{u(x)} - (r^2 - ir + i)w(x). \tag{7}$$

Phase i(b). Π_q^* picks a node P from $\mathcal{C}_i(t)$. Similar to phase i(a), we obtain the following equations:

$$\frac{du_k(x)}{dx} = -(r-i)\frac{u_k(x)}{u(x)} - (r-i)(r-1)q(x), \tag{8}$$

$$\frac{dc_k(x)}{dx} = \delta_{k=r} - \delta_{k=i} + (r-i)\frac{u_k(x)}{u(x)} - (r-i)(r-1)w(x). \tag{9}$$

Interestingly, the differential equations in Phases i(a) and i(b), $0 \le i < r$, admit explicit solutions:[5]

For $s \in \{a, b\}$,

$$u_k(x) = C_k^{u,i(s)}u(x) + \sum_{j=i}^{k} A_{j,k}^{u,i(s)}u(x)^{W_j^{u,i(s)}},$$

$$c_k(x) = C_k^{c,i(s)}u(x) + \sum_{j=i}^{k}(A_{j,k}^{c,i(s)}u(x)^{W_j^{c,i(s)}} + B_{j,k}^{u,i(s)}u(x)^{W_j^{u,i(s)}}),$$

where $u(x) = u(x_{i(s)}) - (r - i + \delta_{s=a})(x - x_{i(s)})$. $x_{i(s)}$ is the value of x at the start of Phase i(s).

$W_k^{u,i(s)}, W_k^{c,i(s)}, C_k^{u,i(s)}, A_{j,k}^{u,i(s)}, B_{j,k}^{u,i(s)}, C_k^{c,i(s)}, A_{j,k}^{c,i(s)}$ are all constants dependent on r, i, j, k and initial values of u_j's and c_j's at $x = x_{i(s)}$.

Remark.

1. In Phase r(a), $du_i(x)/dx = -1$, so the length of Phase

[4]The $o(1)$ terms are due to the fact that the probability of choosing a half-edge changes only by $o(1)$ after each node is selected as is the probability that a particular node is chosen more than one time in a time step.

[5]The only exception to the solution is for $c_0(x)$ in Phase 0(b) when $r = 2$, where it is a linear combination of $u(x)$ and $u(x)log(u(x))$.

r(a) is trivially $u_r(x_{r(a)})$.
2. The time at which Phase i(a) ends is given by $x^* = \inf\{x|u(x) = 0$ or $C_i^{u,i(a)} + A_{i,i}^{u,i(a)} u(x)^{W_i^{u,i(a)}-1} = 0, x \geq x_{i(a)}\}$. Since $u(x)$ is linear in x, we can compute x^* analytically. The length for Phase i(b) is calculated in a similar fashion.

Extension to degree-bounded random graphs. For degree-bounded random graphs, we can still apply the aforementioned differential equation system, with the following modification: let $U_{i,j}(t)(C_{i,j}(t))$ be the number of uncovered (covered) nodes with precisely i outgoing and j incoming half-edges after step t. The initial conditions are: $C_{i,j}(0) = 0, U_{i,j}(0) = d_{i,j}, 0 \leq i, j \leq r$, where $d_{i,j}$ is the number of nodes in the random graph with out-degree i and in-degree j. The order of sets from which Π_q^* picks nodes is $(\mathcal{U}_{r,r}(t), \mathcal{U}_{r,r-1}(t)$, ..., $\mathcal{U}_{r,0}(t), \mathcal{C}_{r,r}(t)$, ..., $\mathcal{C}_{r,0}(t), \mathcal{U}_{r-1,r}(t)$, ..., $\mathcal{U}_{0,r}(t)$, ..., $\mathcal{U}_{0,0}(t))$.

Concentration of results. One essential question is whether the obtained solution to the differential equation system is close to the actual size of dominating set given by Π_q^*. We apply Theorem 1 in Wormald [10] to prove that the differential equation system has a unique solution and the solution is within $o(1)$ of the actual answer almost surely.

Theorem 2. *The differential equation system for Π_q^* has a unique solution. And almost surely, $x^* = \frac{|D_{\Pi_q^*}|}{n} + o(1)$, where $x^* = \inf\{x > 0|c_r(x) = 1\}$. $|D_{\Pi_q^*}|$ is the size of the dominating set found by Π_q^* in a degree-bounded directed random graph of n nodes.*

3) Monetary cost of Π_q^*: Having found the dominating set, Π_q^* queries nodes in $D_{\Pi_q^*}$ to answer polls for all nodes in G. When more than one node in $D_{\Pi_q^*}$ can answer polls for a node, we choose the one with the least monetary cost. However, it is possible that the number of nodes queried by Π_q^* is smaller than the size of $D_{\Pi_q^*}$. In experiments, we observe that the discrepancy is mostly 0 and always less than 0.17% of the size of $D_{\Pi_q^*}$ and decreases rapidly as the graph becomes more dense. We ignore this discrepancy and think of Π_q^* as querying all the nodes in $D_{\Pi_q^*}$.

The expected monetary cost of Π_q^* is given by:

$$E(C_m(\Pi_q^*)) = \sum_{i=1}^{|D_{\Pi_q^*}|} E[|\{j : |\mathcal{I}(j) \cap D_{\Pi_q^*}| = i\}|] \times M(i),$$

where $\mathcal{I}(j)$ is the set of nodes that can answer the poll question for node j, i.e. $\mathcal{I}(j) = \{k|j \in \mathcal{N}(k)\}$, and $M(i)$ is the expected value of the minimum among i numbers i.i.d from $F_c(\cdot)$.

The differential equation system can be employed to calculate $E[|\{j : |\mathcal{I}(j) \cap D_{\Pi_q^*}| = i\}|]$. Let $\mathcal{U}_{i,j}(t)(\mathcal{C}_{i,j}(t))$ be the set of uncovered (covered) nodes with i matched outgoing half-edges and j incoming neighbors (possibly including itself) in $D(t)$ after step t. The update of $\mathcal{U}_{i,j}(t)(\mathcal{C}_{i,j}(t))$ is done appropriately. At the end of the algorithm at time T,

$$E[C_{r,i}(T)] = E[|\{j : |\mathcal{I}(j) \cap D_{\Pi_q^*}| = i\}|] \qquad (10)$$

V. EXPERIMENTS

A. Synthetic Data

To evaluate the social polling strategies Π_m^* and Π_q^*, we conduct experiments on the following types of graphs:

- Directed r-regular random graph of $n = 100,000$ nodes
- Undirected random graph of $n = 100,000$ nodes with power-law degree distribution and maximum degree bounded at 20

As isolated nodes must be queried, we do not generate nodes with degree 0 in the graphs above. Each empirical result is averaged over 100 runs.[6] Runge-Kutta method is used for solving the differential equation system with step size equal to 10^{-5} for regular graphs and 5×10^{-7} for other graphs. The monetary cost c_i is i.i.d and uniformly randomly drawn from $\{1, 5, 10, 50, 100\}$ for regular graphs and from $[0, 1]$ for other graphs. Thus, for the baseline strategy to poll each node for its own information, the average monetary cost per node is $(1 + 5 + 10 + 50 + 100)/5 = 33.2$ for regular graphs and 0.5 for other graphs. The experimental results are shown in Table II and III.

First of all, the theoretical models give pretty close results compared with empirical ones in all cases. This verifies the effectiveness of our models in analyzing the performance of algorithms in different types of large networks.

Secondly, each strategy is effective in minimizing its objective. In r-regular graphs, the monetary cost of Π_m^* can be as small as $1/20$ of that of Π_q^*; while the querying cost for Π_q^* is around half of that of Π_m^* when r is large. It is worth noting that as r increases, the querying cost drops to 8.8% of the size of graph, while the optimum average monetary cost is only 1.04 per node, much lower than the baseline (33.2). In power-law graphs, only 35% to 40% of the nodes need to be queried, while minimum monetary cost is around 0.3 per node. This indicates that Π_m^* and Π_q^* are effective in lowering the costs in social polling.

Thirdly, we also evaluate the actual querying cost of Π_q^* since it may be smaller than the size of dominating set found by the greedy algorithm. It turns out that in the experiments, the discrepancy is mostly 0 and is always less than 0.17% of the size of dominating set.

B. Social Network Data

To evaluate the effectiveness of our algorithms and the potential of utilizing social network information in polling, we carry out experiments on real world social network data. We use anonymous data from publicly available data sources ([20], [21], [22]), which crawled the friend information among users in social networking site, blogging site, video-sharing site, etc. We introduce the basic information of the social network data in Table IV. These social graphs contain 1.1 million to 41.7 million nodes with 4.9 million to 1.46 billion friend links. All of the friend links are directed. In case a site allows undirected friendship, there will be two directed edges in the dataset describing each pair of friends.

[6] In all experiments, the standard deviation is very small, hence not reported.

TABLE II: Empirical and theoretical results for monetary and querying costs of Π_m^* and Π_q^*. Each empirical result is averaged over 100 runs on directed r-regular random graphs with $100,000$ nodes and on monetary cost uniformly randomly drawn from $\{1, 5, 10, 50, 100\}$.

	Minimum monetary cost strategy Π_m^*				Minimum querying cost strategy Π_q^*			
r	Monetary cost		Querying cost		Monetary cost		Querying cost	
	Empirical	Theoretical	Empirical	Theoretical	Empirical	Theoretical	Empirical	Theoretical
1	1,374,076	1,376,000	66,647	66,666	3,056,203	3,056,959	56,768	56,766
2	707,979	708,800	54,269	54,285	2,886,870	2,887,135	41,076	41,079
3	438,880	439,040	46,595	46,592	2,771,784	2,768,628	32,633	32,636
4	312,939	312,512	41,175	41,161	2,679,948	2,676,590	27,311	27,307
5	245,011	244,890	37,069	37,058	2,603,890	2,600,786	23,620	23,617
10	136,377	136,342	25,513	25,514	2,352,726	2,342,149	14,636	14,631
15	111,383	111,402	19,907	19,905	2,186,065	2,177,103	10,918	10,913
20	103,685	103,700	16,499	16,499	2,066,483	2,056,879	8,835	8,830

TABLE III: Empirical and theoretical results for monetary and querying costs of Π_m^* and Π_q^*. Each empirical result is averaged over 100 runs on undirected random graphs with power-law degree distribution ($P[degree = k] \propto k^{-\alpha}$) and maximum degree bounded at 20. The graph has $100,000$ nodes and the monetary cost is uniformly randomly drawn from $[0, 1]$.

	Minimum monetary cost strategy Π_m^*				Minimum querying cost strategy Π_q^*			
α	Monetary cost		Querying cost		Monetary cost		Querying cost	
	Empirical	Theoretical	Empirical	Theoretical	Empirical	Theoretical	Empirical	Theoretical
2.1	28,516	28,501	52,334	52,332	43,747	44,282	34,737	34,605
2.3	29,440	29,418	52,544	52,549	45,106	45,579	36,240	36,136
2.5	30,185	30,156	52,576	52,583	46,235	46,674	37,766	37,717
2.7	30,773	30,746	52,485	52,495	47,135	47,548	39,256	39,246
2.9	31,249	31,217	52,335	52,333	47,855	48,217	40,648	40,648

TABLE IV: Statistics of the social network data in the experiments

Dataset	Type	No. users	No. friend links
Livejournal	Blogging	4.8M	68.5M
Pokec	Social Network	1.6M	30.6M
Flickr	Photo-sharing	1.7M	22.6M
Orkut	Social Network	3.1M	223.5M
Youtube	Video-sharing	1.1M	4.9M
Twitter	Microblogging	41.7M	1.46B

Each experiment is run 100 times with i.i.d. monetary cost uniformly random from $[0, 1]$ and the average is reported.

Apart from Π_m^* and Π_q^*, we compare with the performance of a baseline algorithm, RANDOMSELECT. RANDOMSELECT assigns each node i a node q_i, chosen uniformly randomly from the set $\{j | i \in \mathcal{N}(j)\}$. In other words, each node that can answer the poll for i is selected with equal probability. Then, the question of i is answered by q_i.

Figure 1 shows the monetary and querying costs in logarithmic scale for the three algorithms on social network datasets. Table V presents the average monetary / querying cost per node for these algorithms. We summarize the findings as follows.

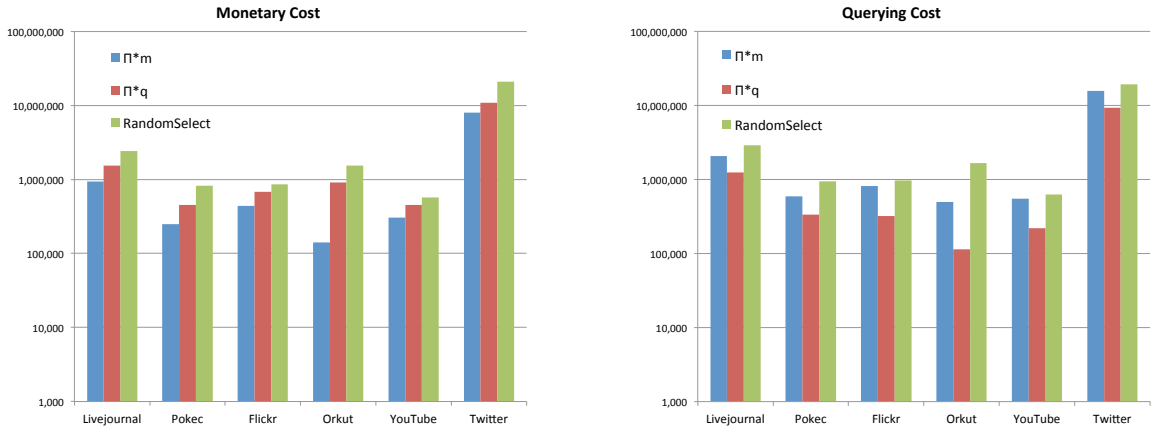
Firstly, while the expected monetary cost is 0.5 per node if we poll each node for its own information, the optimal monetary cost given by Π_m^* is much lower: the average monetary cost per node is around 0.2 and can be as low as 0.05. Similarly, under Π_q^*, only around 20% of the nodes need to be polled for the information over the entire graph. This manifests the great potential of social polling.

Secondly, the baseline algorithm, RANDOMSELECT, has both higher monetary and querying costs than that of Π_m^* and Π_q^*. This is interesting since Π_m^* and Π_q^* are designed to minimize their own objective cost, but they also keep the other cost low. Specially, RANDOMSELECT incurs 3.9 times the monetary cost of that for Π_m^* and 4.6 times the querying cost of that for Π_q^* on average. It indicates that Π_m^* and Π_q^* are effective in minimizing the costs in social polling.

Lastly, for the largest dataset in our experiment, Twitter, Π_q^* only queries 22% of the 41.7M nodes; while Π_m^* has an average monetary cost of 0.19 per node, which is 38% of 0.5 if every node is polled for its own information.

VI. CONCLUSIONS

In this paper, we studied social polling from a graph-theoretic view. We considered the monetary and querying costs of social polling and analyzed algorithms for minimizing these costs. The main contributions are an exact analysis of these costs in a variety of graphs and an explicit solution for directed random r-regular graphs. The experiments validate our theoretical models in various kinds of graphs with different monetary cost distributions. Here, the main findings are that by exploiting the social structure and providing monetary incentives, a pollster can significantly increase the amount of information gathered while keeping costs very low. Further work is needed to explore other issues such as the truthfulness of responses where, again, a social network can help (e.g., a user's response can be validated by comparing the responses given by a number of the user's friends on the user's behalf). A practical trial, which we plan to conduct, will also be highly informative.



(a) Monetary Cost

(b) Querying Cost

Fig. 1: Monetary and querying costs of Π_m^*, Π_q^* and RANDOMSELECT on social network data. Results are in logarithmic scale. Monetary cost of each node is i.i.d. uniform from $[0, 1]$. Each experiment is run for 100 times and the average cost is reported.

TABLE V: Average monetary and querying cost of Π_m^*, Π_q^* and RANDOMSELECT per node. Minimum monetary/querying costs are in bold.

	Livejournal	**Pokec**	**Flickr**	**Orkut**	**Youtube**	**Twitter**
Algorithm	**Average monetary cost per node**					
Π_m^*	**0.19**	**0.15**	**0.25**	**0.05**	**0.27**	**0.19**
Π_q^*	0.32	0.28	0.39	0.30	0.40	0.26
RANDOMSELECT	0.50	0.50	0.50	0.50	0.50	0.50
Algorithm	**Average querying cost per node**					
Π_m^*	0.42	0.36	0.47	0.16	0.48	0.37
Π_q^*	**0.26**	**0.20**	**0.19**	**0.04**	**0.19**	**0.22**
RANDOMSELECT	0.60	0.57	0.57	0.54	0.55	0.46

REFERENCES

[1] B. C, "Does adding one more question impact survey completion rate?" https://www.surveymonkey.com/blog/en/blog/2010/12/08/survey_questions_and_completion_rates/, 2010.

[2] H. Johnson, "10 tips to improve your online surveys," https://www.surveymonkey.com/blog/en/blog/2012/04/13/10-online-survey-tips/, 2012.

[3] A. Dasgupta, R. Kumar, and D. Sivakumar, "Social sampling," *Proc. 2012 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'12)*, pp. 235–243.

[4] B. Ribeiro and D. Towsley, "Estimating and sampling graphs with multidimensional random walks," *Proc. 2010 ACM SIGCOMM (SIGCOMM'10)*, pp. 390–403.

[5] P. Wang, B. Ribeiro, J. Zhao, J. C. Lui, D. Towsley, and X. Guan, "Practical characterization of large networks using neighborhood information," *arXiv:abs/1311.3037*, 2013.

[6] D. Rothschild and J. Wolfers, "Forecasting elections: Voter intentions versus expectations," *Available at SSRN 1884644*, 2011.

[7] A. Das, S. Gollapudi, R. Panigrahy, and M. Salek, "Debiasing social wisdom," *Proc. 2013 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'13)*, pp. 500–508.

[8] T. Wang, D. Wang, and F. Wang, "Quantifying herding effects in crowd wisdom," *Proc. 2014 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'14)*.

[9] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[10] N. C. Wormald, "Differential equations for random processes and random graphs," *The Annals of Applied Probability*, pp. 1217–1235, 1995.

[11] A. Dembo and A. Montanari, "Ising models on locally tree-like graphs," *The Annals of Applied Probability*, vol. 20, no. 2, pp. 565–592, 2010.

[12] B. Bollobas, "Random graphs," *Cambridge University Press*, 2001.

[13] M. R. Gary and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," *WH Freeman and Co*, 1979.

[14] C. Cooper, R. Klasing, and M. Zito, "Lower bounds and algorithms for dominating sets in web graphs," *Internet Mathematics*, vol. 2, no. 3, pp. 275–300, 2005.

[15] B. Wieland and A. P. Godbole, "On the domination number of a random graph," *Journal of Combinatorics*, vol. 8, no. 1, pp. R37–R37, 2001.

[16] A. K. Parekh, "Analysis of a greedy heuristic for finding small dominating sets in graphs," *Information Processing Letters*, vol. 39, no. 5, pp. 237–240, 1991.

[17] W. Duckworth and N. C. Wormald, "On the independent domination number of random regular graphs," *Combinatorics, Probability and Computing*, vol. 15, no. 4, pp. 513–522, 2006.

[18] W. Duckworth and B. Mans, "Connected domination of regular graphs," *Discrete Mathematics*, vol. 309, no. 8, pp. 2305–2322, 2009.

[19] S. Howe, "The deprioritised approach to prioritised algorithms," *Ph.D. Thesis, The University of New South Wales*, 2008.

[20] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" *Proc. 2010 International World Wide Web Conference (WWW'10)*.

[21] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, June 2014.

[22] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," in *Proc. 2007 ACM/Usenix Internet Measurement Conference (IMC'07)*.