2017 IEEE 37th International Conference on Distributed Computing Systems

Adaptive Reconnaissance Attacks with Near-Optimal Parallel Batching

Xiang Li, J. David Smith, and My T. Thai
Computer and Information Science and Engineering Department
University of Florida
Gainesville, FL, 32611
Email: {xixiang, jdsmith mythai}@cise.ufl.edu

***Abstract*—In assessing privacy on online social networks, it is important to investigate their vulnerability to reconnaissance strategies, in which attackers lure targets into being their friends by exploiting the social graph in order to extract victims' sensitive information. As the network topology is only partially revealed after each successful friend request, attackers need to employ an adaptive strategy. Existing work only considered a simple strategy in which attackers sequentially acquire one friend at a time, which causes tremendous delay in waiting for responses before sending the next request, and which lack the ability to retry failed requests after the network has changed.**

In contrast, we investigate an adaptive and parallel strategy, of which attackers can simultaneously send multiple friend requests in batch and recover from failed requests by retrying after topology changes, thereby significantly reducing the time to reach the targets and greatly improving robustness. We cast this approach as an optimization problem, Max-Crawling, and show it inapproximable within $(1 - 1/e + \epsilon)$. We first design our core algorithm $\mathbf{P}_M$-AReST which has an approximation ratio of $(1 - e^{-(1-1/e)})$ using adaptive monotonic submodular properties. We next tighten our algorithm to provide a near-optimal solution, i.e. having a ratio of $(1 - 1/e)$, via a two-stage stochastic programming approach. We further establish the gap bound of $(1 - e^{-(1-1/e)^2})$ between batch strategies versus the optimal sequential one. We experimentally validate our theoretical results, finding that our algorithm performs near-optimally in practice and that this is robust under a variety of problem settings.

***Index Terms*—Adaptive Approximation Algorithms; Adaptive Stochastic Optimization; Target Reconnaissance Attacks; Social Networks Analysis.**

I. Introduction

Online Social Networks (OSNs) have become quite vulnerable to privacy breaches due to automated social engineering attacks from socialbots. These *reconnaissance attacks* commonly employ the tactic of befriending target users to extract sensitive information [1], [2]. Attackers often seek to improve the odds of befriending their targets by first befriending multiple users in the target's social circle. Only when attackers acquire enough mutual friends with the targets, will they attempt lure the target into becoming their "friend". Once successful, attackers can exploit the sensitive information obtained for a number of purposes, including spearphishing and account compromise [2]. Therefore, studying the vulnerability of OSNs to these threats is of great importance.

Understanding the friending strategies of reconnaissance attacks is one of the ways to quantify the privacy vulnerability, as recently introduced by Li *et al.* [3]. The intuition is that once we identify the set of vulnerable users that attackers need to befriend, we can raise awareness and protect these users, thereby limiting the threat to user privacy. Furthermore, the size of this set compared to the amount of information collected in the attack can shed the light on how vulnerable a network is. If this ratio is small, one can conclude that the network is vulnerable. On the other hand, a large ratio indicates greater robustness against reconnaissance attacks.

The realization of this study, however, encounters several challenges. First of all, the variety of social responses to friend requests make it difficult to design an efficient friending strategy. Users tend to accept friend requests from people with whom they share many friends. Users' attributes, such as location, gender, hometown, employer, and so forth, also play an important role in making a decision. The greater the similarity between the friend-er and friend-ee's attributes, the higher the chance that the friend request will be accepted. Last – but perhaps most importantly – the topological information of social networks is generally unavailable to attackers. Since only two-hop topology is available by default in closed OSNs like Facebook, the users connections are gradually revealed to attackers when acquiring new friends. This poses a critical question: *How frequently should attackers send their requests and observe the network topology?*

Unfortunately, the study of this problem is still in infancy. Initial existing work [3], [4] suggested a simple adaptive strategy to combat the incomplete information. In their solutions, attackers *sequentially* acquire one friend per time, wait for a response (accept or reject), observe the network topology, and continue sending the next request. This approach may obtain the best possible solution as network topology is revealed frequently. However, it causes tremendous delay in waiting for responses before sending the next request. Existing solutions further disallow retrying failed friend requests, which may decrease the performance as it may eliminate important users due to a prior rejection.

To overcome the above shortcomings, we investigate an adaptive and parallel strategy in which attackers can simultaneously send multiple friend requests as a batch and recover from failed requests by retrying in later batches, thereby significantly reducing the time of reaching the targets. Specifically, we propose to study the following optimization problem, namely *Adaptive Maximum Benefit Crawling (Max-Crawling)*, which asks us to find a batch attacking strategy so as to maximize the information collected within the budget K. Solving this problem introduces two more challenges: 1)

1063-6927/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCS.2017.130

Efficiently selecting best nodes within each batch without observing the network topology, and 2) Deciding the size of each batch.

Our contributions are summarized as follows:

- We model reconnaissance attacks as a Max-Crawling problem, taking into account the uncertainty of friend acceptance rate, based on the common neighbors and attributes. We prove Max-Crawling cannot be approximated within the ratio of $(1 - 1/e + \epsilon)$ unless $P = NP$ even when the whole network topology is available.
- We develop an adaptive and parallel algorithm, named P_M-AReST, for Max-Crawling. P_M-AReST provides a batch strategy that enables attackers to send many friend requests simultaneously in uniformly-sized batches. Using adaptive stochastic optimization, we show that P_M-AReST has a performance ratio of $(1 - e^{-(1-\frac{1}{e})})$. We further develop a two-stage stochastic program to tighten the performance bound to $(1 - 1/e)$.
- We bound the gap between batch strategies versus sequential strategies, which we then use to devise an attack with varying batch sizes. In this attack, the attackers can send $k_i \ll K$ friend requests in each batch for differing k_i. The variation is an important consideration, as it facilitates the evasion of OSN defenses.
- We experimentally show that P_M-AReST performs only marginally worse than AReST, a sequential attacking strategy developed by Li et al. [3], and that this difference all but vanishes when failed requests may be retried.
- We show that under realistic assumptions of user response times, P_M-AReST significantly outperforms AReST in terms of reconnaissance speed – and therefore that networks are significantly more vulnerable to P_M-AReST than AReST.

Related Work. Reconnaissance attacks on social networks have shown to be very crucial to the privacy vulnerability of OSNs [5]. Ryan & Mauch showed that fake profiles can be effectively used to befriend members of the NSA, military intelligence agencies, and Global 500 corporations [6]. These fake profiles can be automated to form socialbots [2], which can then be used to automatically infiltrate organizations [3], [4], [7]. Furthermore, reconnaissance attacks preclude the use of any defending mechanism against Sybil attacks, on which there has been significant study [8] (and references therein). Therefore, it urgently calls for radically new models and analytical techniques to assess the vulnerability of OSNs to these attacks.

When considering offensive methods, two works that are relevant the most to our paper are in [3], [4]. However, these papers only provided a sequential attacking strategy, which may be impractical to execute in the real world setting. In addition, the network models are simplified. They do not account for the changing acceptance rate nor allowing attackers to retry failed requests.

On the defensive front, Paradise *et al.* suggested monitoring a subset of users by simulating an attacker on topologies taken from social networks [9]. However, this and much of the related work along this direction are based on heuristics built on known sociological properties. Furthermore, they did not provide a theoretical analysis of identifying the set of vulnerable users. Additionally, in the literature the attacker is usually assumed to have knowledge of the network topology.

Further, assessing network vulnerability to infiltration and reconnaissance attacks is a new metric [3]. The vulnerability of networks due to malicious actors or external disasters has been characterized in a number of ways [10]–[16] (and references therein). However, the differences between these destructive attacks and reconnaissance attacks precludes applying these vulnerability measurements directly. Where prior work is concerned with the ability of an attacker to disrupt a network in term of network connectivity for example, we instead look at the ability of an attacker to extract information from it, thereby forming a new research direction.

Organization. The rest of the paper is organized as follows. Section II presents the social network models, our problem definition, and its inapproximablity. The P_M-AReST algorithm and all the theoretical analysis are introduced in Sections III and IV respectively. Section V presents our experimental evaluation and Section VI concludes the paper.

II. Problem Setting and Inapproximability

We begin by modeling the problem space, then formally define the problem and prove its inapproximability.

A. Network Models

We abstract an OSN as a directed graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n, s\}$ is the set of n users and attacker s, who initially has no connections to other users. E is the set of m directed edges where each edge $(u, v) \in E$ represents the friendship between u and v. Note that due to privacy protections, the s only has access to incomplete friendship information (i.e. network topology). However, the likelihood of each relation can be estimated with link prediction methods [17]–[19], which may incorporate both publicly observable connections and users public profiles. We model this by letting each edge $e \in E$ exist with some probability $p_e \in [0, 1]$. Once u accepts the friend request of s and all of u's friends are revealed, fixing p_{uv} to 0 or 1 depending on whether the edge (u, v) was absent or present, respectively.

Friend Request Acceptance Model. Each node u is associated with a random variable X_u where $X_u = 1$ indicates u will accept a friend request from s, 0 otherwise. X_u follows a distribution which reflects the current topology during an attack. For example, suppose u has probability $q(u)$ of accepting s's friend request. The attacker may be able to increase this probability by first befriending some of u's friends, resulting in a probality of $q'(u) > q(u)$ later in the attack. Without loss of generality, we assume X_u follows a certain discrete distribution $q(u)$ with a domain D_u, where $d_u^i \in [0, 1]$ be the i^{th} value in D_u. We allow s to re-send friend requests to users that had previously rejected them at the later time.

B. Attacker Model

We consider an attacker seeking to collect as much information as possible from a set of users $T \subseteq V$, constrained by a budget K. The attacker knows the cost $c(u) \to \mathbb{Z}^+$ of sending a request to each user u. In the simplest case, this represents

the opportunity cost of spending time attempting to befriend user u. More complex cases, such as considering the cost of updating the bot's profile to match a user u's attributes (i.e. exploiting *homophily*), are also representable with this model. Let s be a single account on the network controlled by the adversary, with $N(s) = \emptyset$ initially.[1] This account may be a socialbot or manually-operated.

Conceptually, the reconnaissance attack works as follows. The attacker first obtains a master-list of target users T through one or more public channels, e.g., an organization's website or from the OSNs themselves. The privacy features common on OSNs prevent s from reliably gathering complete information on $t \in T$ except by becoming friends with t. Therefore, s aims to choose the best set of users to maximize the benefit gained from friends made. As a result of the acceptance model described above, it may be beneficial for s to befriend $u \notin T$ in order to boost the acceptance chance of one or more targets. Therefore, the friending set $F \not\subseteq T$. However, s cannot send too many requests or they will be detected by even a trivial network monitoring system. The best strategy for s is to mimic the behavior of normal users, e.g. sending small numbers of requests, observing the responses, and then repeating the process. Once a user v accepts s's friend request, their information will be harvested and their neighborhood $N(v)$ will be revealed. This strategy of repeatedly making decisions subject to observed results of previous steps is called an *adaptive* strategy.

Information Benefit Model. The benefit gained from a friending set F depends on both the final network topology (i.e. which users accepted or rejected s's requests) and the target set. We represent the benefit gained by each of (a) friends made, (b) friends-of-friends (i.e. $v \in N(s), u \in N(v)$ if u is a friend-of-friend of s) made, and (c) edges revealed as $B_f(u \mid T)$, $B_{fof}(u \mid T) \leq B_f(u \mid T)$, and $B_i(u, v \mid T)$, respectively. For notational clarity, we leave the T parameter implied in the remainder of this work. Each node u may only produce one kind of benefit: $B_f(u)$ or $B_{fof}(u)$. These functions allow expression of complex targeting priorities.

C. Preliminaries and Problem Definitions

Since G is partially unknown to s and friend requests sent from s to u may fail, we use adaptive stochastic optimization to tackle our problem. We begin by introducing notation. For each node $u \in V$, let $Y_u \in \{0, 1, ?\}$ denote the state of u, where 1 indicates that u accepts the friend request from s, 0 indicates that u rejects the friend request, and ? represents an unknown, i.e., s has not sent a request to u yet. Initially, the states of all u should be ?. Likewise, for each edge $(u, v) \in E$, define $Y_{uv} \in \{0, 1, ?\}$. 1 means the edge (u, v) exists (revealed when s friends with u and v is u's friend), 0 indicates edge (u, v) is not present (revealed when s friends with u and we learn for certain that v is not a friend of u), and ? means unknown, i.e. u rejects the friend request from s, or s has not sent a friend request to u yet, or u has their privacy level set to self-only. Let Ω be the set of all possible states of G and $\phi = \{Y_v\}_{v \in V} \cup \{Y_{uv}\}_{(u,v) \in E} \rightarrow \Omega$ be a possible state, called a *(full) realization*. We call $\phi(u, e)$ the state of node u and edge e under realization ϕ. Without abusing the notation, we use $\phi(u)$ and $\phi(e)$ to denote the state of node u and edge e under ϕ, respectively. Clearly there are many possible realizations which follow a probability distribution $P[\phi]$. We represent this with the random variable Φ, with $P[\phi] = P[\Phi = \phi]$ over all realizations.

We consider a batch strategy π in which s sends friend requests to a batch F', observes the state of all $u \in F'$ and all edges incident to u, picks the next set of batch to be friend with, sees their state, and so on. Therefore, our observation on networks after sending each batch of friend requests can be represented as a *partial realization* ω. We use $dom(\omega)$ to refer to the domain of ω, ie., the set of nodes and edges observed in ω. A partial realization ω is *consistent* with a realization ϕ if they are equal everywhere in the domain of ω, written as $\phi \sim \omega$. If ω and ω' are both consistent with some ϕ and $dom(\omega) \subseteq dom(\omega')$, we say ω is a *subrealization* of ω'. We use the notation $F(\pi, \phi)$ (sometimes just F) to represent the set of node selected by strategy π under realization ϕ.

The total benefit gain from strategy π under realization ϕ can be written as follows.

$$f(\pi, \phi) = \sum_{u \in N_f(\pi,\phi)} B_f(u) + \sum_{v \in N_{fof}(\pi,\phi)} B_{fof}(v) + \sum_{(u,v) \in N^i(\pi,\phi)} B_i(u, v) \quad (1)$$

where

$$N_f(\pi, \phi) = \{u | u \in F(\pi, \phi), \phi(u) = 1\}$$
$$N_{fof}(\pi, \phi) = \{v | \exists u \in F(\pi, \phi) : \phi(u, v) = 1\} \setminus N_f(\pi, \phi)$$
$$N_i(\pi, \phi) = \{(u, v) | u \in N_f(\pi, \phi), \phi(u, v) = 1\}$$

We are now ready to formally define our problem as follows:

Definition 1 (Adaptive Maximum Benefit Crawling (Max-Crawling))**.** *Given a social network $G = (V, E)$ where V is the set of user accounts, E is a set of possible friendships between users, a budget $K \in \mathbb{Z}^+$, and all associated functions discussed above, find a batch attacking strategy π with $c(F(\pi, \phi)) \leq K$ where $c(F) = \sum_{u \in F} c(u)$ for all ϕ so as to maximize the expected benefit $f(\pi) = \mathbb{E}[f(F(\pi, \Phi), \Phi)]$, where the expectation is taken w.r.t $P(\phi)$.*

D. Inapproximability

We prove the following inapproximability result, bounding the approximation guarantee that any algorithm can provide.

Theorem 1. *The Max-Crawling problem cannot be approximated within a factor $(1 - 1/e + \epsilon)$ unless $P = NP$*

Proof. We will reduce from the Max-Cover problem, which is defined as follows: given a collection of possibly overlapping sets $\mathcal{S} = \{S_1, S_2, ..., S_{m'}\}$ and integer k', find a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that $|\mathcal{S}'| \leq k'$ to maximize $|\cup_{S_i \in \mathcal{S}'} S_i|$.

Given an instance of Max-Cover, we reduce it to an instance of Max-Crawling as follows: Let $V = \cup_{S_i \in \mathcal{S}} S_i$, then for each element $e_j \in V$, we create a node v_j in the Max-Crawling

[1] For simplicity, we consider only one attacker, although our solutions are readily extended to the case of multiple attackers.

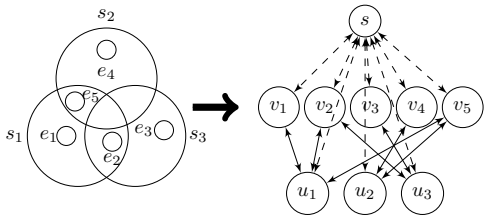


Fig. 1: Reduction from Max-Cover to Max-Crawling

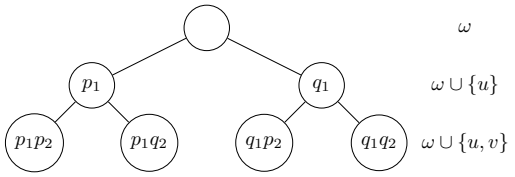


Fig. 2: The tree of realizations for a batch of size $k = 2$, where $q_i = 1 - p_i$.

instance. For each set $S_i \in \mathcal{S}$, we also create a node u_i. If $e_j \in S_i$, create a directed edge (u_i, v_j) with $p_{u_i v_j} = 1$. Additionally, fix $q(u) = 1, \forall u \in V$, $B_f(u_i) = 0$, $B_f(v_j) = 1$, $B_{fof}(u_i) = 0$, $B_{fof}(v_j) = 1$ and $K = k'$. The reduction is illustrated in Fig 1. This reduction can clearly be done in polynomial time.

If we have a Max-Crawling solution that has benefit D' then there must exist a solution $\tilde{D} \geq D'$ when we send all friends request to u_i. Notice that the substitution of a v_j with a u_i could only increase the gain. By the reduction we obtain a Max-Cover solution with coverage $\tilde{D}$ by choosing the sets S_i corresponding to selected u_i. Clearly, if we have an α-approximation algorithm for Max-Crawling, then we also have an α- approximation algorithm for the Max-Cover problem. Thus due to the inapproximability of Max-Cover [20], the Max-Crawling problem cannot be approximated within a factor $(1 - 1/e + \epsilon)$ unless $P = NP$ □

III. Adaptive and Parallel Attacking Strategies

In this section, we present our algorithm, the Parallel and Adaptive Reconnaissance Strategy (P_M-AReST), for solving Max-Crawling. For a clearer presentation, we first introduce our solution with all nodes having uniform cost, i.e. $c(u) = 1$, a fixed friend request acceptance probability $q(u)$, and a fixed batch size k. We will discuss our solution for a general case later in section IV-C. For simplicity, we assume that K is divisible by k, i.e $K = mk$ for $m \in \mathbb{Z}^+$.

A. Algorithm Overview

P_M-AReST, detailed in Alg. 1, has two main phases: *Batch Selection* and *Observation*. In the *Batch Selection* phase, P_M-AReST will call BatchSelect (line 3) to select the best possible set $F' = \{u_1, \ldots, u_k\}$ to friend in parallel. Upon sending the friend requests, P_M-AReST executes the *Observation* phase, which establishes "accept" or "reject" status of u_j. If $u_j \in F'$ accepted the friend request from s, the partial realization ω will be updated with exact information on p_{uv} for all $v \in N(u)$ and $q(u)$. The expected benefit Q is also updated with the expected marginal gain of u conditioned on partial realization ω (line 7), which is defined as follows:

$$\Delta_f(u|\omega) = \mathbb{E}[f(dom(\omega) \cup \{u\}, \Phi) - f(dom(\omega), \Phi) | \Phi \sim \omega]$$

These two phases will be iteratively executed until the total number of friend requests reached to K.

The most crucial step of P_M-AReST lies in the Batch Selection phase. Within each batch, the nodes u_j are selected without observing the results of selecting the previous $j - 1$ nodes in F'. P_M-AReST addresses this challenge by considering the expected marginal gain of a node, taking into account all observations made so far and all nodes that have already been selected within the batch (but not yet observed).

Algorithm 1: P_M-AReST Algorithm

Input: Graph $G = (V, E, p, B, q)$, $K, k, z \in \mathbb{Z}^+, K = zk$
Output: An ordered set of sets of nodes $F \subset 2^V$ for s to send a friend request. Q the benefit gained by friending all batch $F' \in F$.

1 $F \leftarrow \emptyset; \omega \leftarrow \emptyset; Q \leftarrow 0$
2 **for** $i = 1..m$ **do**
3 $F' \leftarrow$ BatchSelect(ω, k)
4 Send friend request to all $u_j \in F'$ in parallel.
5 **for** $u_j \in F'$ **do**
6 **if** u_j *accepts* **then**
7 $Q \leftarrow Q + \Delta_f(u \mid \omega)$
8 $\omega \leftarrow \omega \cup \{(u_j, \text{accept})\} \cup \text{observe}(G, u_j)$
9 **else**
10 $\omega \leftarrow \omega \cup \{(u_j, \text{reject})\}$
11 $F \leftarrow F \cup \{F'\}$
12 Return F, Q

B. Batch Selection Procedure

The BatchSelect procedure takes the current partial realization ω as an input, and returns the set F' of k nodes that s should send its friend requests. The goal of BatchSelect is to greedily choose the best set F' that can gain the most possible expected benefit upon sending friend requests to F'. Such computation is non-trivial, since choosing an additional node for a batch F' requires considering all combinations of success and failure for each node chosen in F' thus far. Fig. 2 illustrates the tree formed by this process. Each node in the tree is directly computed as if the sequence of requests for it had been done (note, however, that no requests are sent or observations made until after the batch is complete).

Therefore, BatchSelect must carry forward additional data that is not part of the partial realization ω, as ω is not updated during the batch process. Specifically, three pieces of additional information are needed: the set of all edges that would be revealed in a given branch of the expectation tree (R_E), $U[v]$ the *un*likelihood of a node v being added as a friend-of-friend during another request in the batch, and γ the probability of the given branch being taken when the requests are sent. We denote each branch state $\beta = (\gamma, R_E, U)$ (the

root having $\beta_0 = (1, \emptyset, \emptyset)$). Two further states are computed for each state in the current row, denoted accept(β, u) and reject(β, u), which represent the branches in which friending u succeeds and fails, respectively. These are defined as:

$$\text{accept}(\beta, u) = (\gamma q(u \mid \omega), R_E \cup E(u), U \cup \text{up}(U, u))$$
$$\text{reject}(\beta, u) = (\gamma(1 - q(u \mid \omega)), R_E, U)$$

where $\text{up}(U, v) = \{v \to (1 - p_{uv})U[v] \mid v \in N(u)\}$ and $v \notin U \implies U[v] = 1$. We further define the batch marginal gain Δ_b as follows:

$$\Delta_b(u \mid \psi, R_E, U) = q(u)\Bigg(B_f(u) + \sum_{v \in N'(u)} p_{uv} U[v] B_{fof}(v) + \sum_{e \in N''(u) \setminus R_E} B_i(e)\Bigg)$$

where $N'(u) = N(u) \setminus N_{fof}(\pi, \psi) \setminus N_f(\pi, \psi)$ and $N''(u) = \{(u, v) \mid v \in N(u), (u, v) \notin N_i(\pi, \psi)\}$

We use the unlikelihood of v being a friend-of-friend because that allows us to efficiently compute the joint probability on-line as the product of $1 - p_{xv}$ values. During computation of batch marginal gain (Δ_b), this probability is used to weight $B_{fof}(v)$ by the likelihood that (a) after friending u, v is a friend-of-friend, and (b) that it has not become a friend-of-friend already from one of the other nodes in the batch.

Note that each row of the expectation tree depends only on the previous row and the values within each row are independent. Knowing this enables us to compute the branches in each row in a massively parallel fashion without any reduction in solution quality. It is only necessary to update the Δ_b and p values of the 2-hop neighborhood of each selected node. Therefore, a cache is used to avoid re-computing these values unnecessarily (as shown in lines 8-11 of Alg. 2).

Algorithm 2: BATCHSELECT Procedure

Input: Partial Realization ω, batch size k
Output: F' a set of friend requests to send

1 $F' \leftarrow \emptyset$
2 $A \leftarrow [\beta_0]$
3 $C_\Delta = \emptyset$ **for** $i = 1 \dots k$ **do**
4 $u \leftarrow \max_u \Gamma(u \mid A)$
5 where $\Gamma = \sum_{\beta \in A} \gamma C_\Delta[u]$
6 with $C_\Delta[u] = \Delta_b(u \mid \omega, R_E, U)$ computed lazily
7 $A \leftarrow [\text{accept}(\beta, u),\ \text{reject}(\beta, u) \forall \beta \in A$ in parallel$]$
8 **for** $v \in N(u)$ **do**
9 **for** $w \in N(v)$ **do**
10 Erase w from cache C_Δ
11 Erase v from cache C_Δ
12 $F' = F' \cup \{u\}$
13 Return F'

IV. THEORETICAL PERFORMANCE ANALYSIS

Before analyzing the approximation ratios of P$_M$-AReST, we formally present two properties of adaptive monotonicity and submodularity [21] as follows:

Definition 2 (Adaptive Monotonicity). *A function $f(.)$ is adaptive monotone w.r.t the distribution $P(\phi)$ if for all ω with $P[\Phi \sim \omega] > 0$ and all $v \in V$, we have: $\Delta(v \mid \omega) \geq 0$*

Definition 3 (Adaptive Submodularity). *A function $f(.)$ is adaptive submodular w.r.t the distribution $P[\phi]$ of all realizations if for all ω and ω' such that $\omega \subseteq \omega'$ and for all $v \in V \setminus dom(\omega')$, we have:*

$$\Delta(v|\omega) \geq \Delta(v|\omega') \tag{2}$$

A. Analysis of P_M-AReST

Lemma 1. *For the batch selection at iteration i, the* BATCHSELECT *procedure greedily selects a node u such that*

$$u = \arg\max_{u \in V \setminus F} \Delta_f(u|F', \omega)$$

where

$$\Delta_f(u|F', \omega) = \mathbb{E}[f(dom(\omega) \cup \{u\} \cup \{F'\}, \Phi) - f(dom(\omega) \cup \{F'\}, \Phi) | \Phi \sim \omega]$$

F denotes the set of selected nodes up to iteration $i-1$, F' is the current batch, and ω is a partial realization at iteration $i-1$.

Proof. To begin, we remark that the closed form of $\Delta_f(u \mid \omega)$ can be written as:

$$\text{accept}(u)\left(B_f(u) + \sum_{v \in N'(u)} p_{uv} B_{fof}(v) + \sum_{e \in N''(u)} B_i(e)\right)$$

Next, observe that at state β_0, $\Delta_b = \Delta_f$ since $U[v]$ is identically 1, $R_E = \emptyset$.

Now, suppose that l requests have been added to the batch. Let $\mathbf{\Omega_l}$ be a random variable representing the possible partial realizations after those l requests have been sent. This gives

$$\mathbb{E}\left[\Delta_b(u \mid \omega, R_E, U) \mid \mathbf{\Omega_l} = \omega'\right] = \mathbb{E}\left[\Delta_f(u \mid \omega') \mid \mathbf{\Omega_l} = \omega'\right]$$

where R_E and U are taken from the branch corresponding to ω'. Then, by observing that $\Pr[\mathbf{\Omega_l} = \omega'] = \gamma$ we directly obtain $\Gamma(u \mid A) = \mathbb{E}\left[\Delta_f(u \mid \omega') \mid \mathbf{\Omega_l} = \omega'\right] = \Delta_f(u \mid F', \omega)$, where F' is the set of those l requested users. □

Now let us consider the following sub-problem, **Finding Optimal Batch (FOB)**. Define the function $g : 2^V \times 2^{V \times \Omega} \to \mathbb{R}^+$. For $F' \subseteq V$, and for any partial realization $\omega \subseteq V \times \Omega$, $g(F', \omega) = \mathbb{E}[f(dom(\omega) \cup F') - f(dom(\omega))]$. Given a graph G and a fixed ω, FOB asks us to find a subset $F' \subseteq V$ of size k such that $g(F', \omega)$ is maximum.

Lemma 2. *Let ω be a fixed partial realization and OPT_b be the optimal solution to FOB. Then $g(F', \omega) \geq (1 - 1/e)OPT_b$.*

Proof. It is easy to see that g is monotonic increasing for any ω. We now show that g is also submodular for any ω. Since a non-negative linear combination of submodular function is submodular, instead of proving g submodular over all realization of G (expected value), we will prove g submodular under a random realization $\Phi \sim \omega$. This reduces to prove f is submodular under Φ. Formally, consider two subsets F_1 and F_2 where $dom(\omega) \subseteq F_1 \subseteq F_2$, and $u \notin F_2$, we need to prove:

$$f(F_2 \cup \{u\}) - f(F_2) \leq f(F_1 \cup \{u\}) - f(F_1)$$

Let $\Delta_u f(F) = f(F \cup \{u\}) - f(F)$. We consider the possibilities for u by cases:

1) If $u \notin N(v) \forall v \in F_2$, then $\Delta_u f(F_2) = \Delta_u f(F_1)$;
2) If $u \in N(v)$ for some $v \in F_1$ and $u \notin N(v) \forall v \in F_2 \setminus F_1$, then $\Delta_u f(F_2) = \Delta_u f(F_1)$;
3) If $u \in N(v)$ for some $v \in F_2 \setminus F_1$ and $u \notin N(v) \forall v \in F_1$, then

$$\Delta_u f(F_1) - \Delta_u f(F_2) = B_{fof}(u) + \sum_{v \in F_2 \setminus F_1} B_{fof}(v)$$
$$\implies \Delta_u f(F_2) \leq \Delta_u f(F_1)$$

Thus we have g is submodular. Since g is monotonic submodular, and BATCHSELECT greedily selects each node u as shown in Lemma 1, it will return a solution at least $(1 - 1/e)OPT_b$ □

Now let us consider the following mapping. Given a graph $G = (V, E)$, construct a graph $G' = (V', E')$ as follows. For each subset $F_i = \{v_{i1}, v_{i2}, \ldots, v_{ik}\} \subseteq V$, create a node $v'_i \in V'$ and its associated set F_i (sometimes denoted as $F(v'_i)$ for clarity). For each edge $(u, v) \in E$, create an edge $(v'_i, v'_j) \in E'$ if $u \in F_i$ and and $v \in F_j$. Each new edge (v'_i, v'_j) has the same probability p_{uv} and $B_i(u, v)$ with edge (u, v). Therefore, there are multiples edges between a pair of nodes in G'. For each node $v'_i \in V$, set $q(v'_i) = 1$.

Define function h as follow. For a subset $D \subseteq V'$, for any pair $(v'_i, v'_j) \in D$, $F(v'_i) \cap F(v'_j) = \emptyset$, we have $h(D) = f(\cup_{v'_i \in D} F_i)$.

Lemma 3. *Function h is monotonic submodular*

Proof. It is easy to see that h is monotone increasing. Now we prove that h is submodular. Again, since a non-negative linear combination of submodular function is submodular, we only prove the following for one fixed realization. Consider two subsets $D_1 \subseteq D_2 \subseteq V'$, and $v' \notin D_2$, we need to prove that:

$$h(D_2 \cup \{v'\}) - h(D_2) \leq h(D_1 \cup \{v'\}) - h(D_1)$$

We have:

$$\begin{aligned}\Delta_{v'} h(D_2) &= h(D_2 \cup \{v'\}) - h(D_2) \\ &= f(\cup_{v'_i \in D_2} F_i \cup F(v')) - f(\cup_{v'_i \in D_2} F_i) \\ &\leq f(\cup_{v'_j \in D_1} F_j \cup F(v')) - f(\cup_{v'_j \in D_1} F_j) \\ &= \Delta_{v'} h(D_1)\end{aligned}$$

The inequality is due to f is submodular (Lemma 2) and the fact that $\forall v'_i \in D_2$, $F_i \cap F(v') = \emptyset$. □

Lemma 4. *(h, Z) is adaptive monotonic and adaptive submodular where $Z(\phi')$ is the distribution induced by $P(\phi)$, followed by the above mapping.*

Proof. Adaptive monotonicity is immediate due to the definition of h. For the proof of adaptive submodularity, consider two fixed partial realizations ω'_1 and ω'_2 of G' where $\omega'_1 \subseteq \omega'_2$ and a node $v' \in V' \backslash dom(\omega'_2)$, we need to prove that $\Delta_h(v'|\omega'_1) \geq \Delta_h(v'|\omega'_2)$.

Let ω_1 and ω_2 denote the two corresponding partial realization of ω'_1 and ω'_2 on G through the above mapping. Let Φ' denote a random realization on G'. We have:

$$\begin{aligned}\Delta_h(v'|\omega'_1) &= \mathbb{E}[h(dom(\omega'_1) \cup \{v'\}, \Phi') - h(dom(\omega'_1), \Phi')|\Phi' \sim \omega'_1] \\ &= \mathbb{E}[f(dom(\omega_1) \cup F(v'), \Phi) - f(dom(\omega_1), \Phi)|\Phi \sim \omega_1] \\ &\geq \mathbb{E}[f(dom(\omega_2) \cup F(v'), \Phi) - f(dom(\omega_2), \Phi)|\Phi \sim \omega_1] \\ &= \Delta_h(v'|\omega'_2)\end{aligned}$$

The inequality follows due to the fact that (f, P) is adaptive monotonic and submodular [3] and $dom(\omega_2) \cap F(v') = \emptyset$. □

Theorem 2. *The P_M-AReST algorithm has an approximation ratio of $\left(1 - e^{-(1-\frac{1}{e})}\right)$.*

Proof. Let OPT_M be the optimal solution to the Max-Crawling problem, and SOL_M be the solution obtained by Alg. 1. Based on our mapping construction, it is easy to verify that there is a 1-1 correspondence between finding the batch strategy (each batch of size k) for (f, P) and sequentially selecting one node at a time for (h, Z). Thus we will bound the ratio based on (h, Z).

As shown in Lemma 2, at each iteration i of Alg. 1, the gain BATCHSELECT obtained compared with OPT_b is at least $(1 - 1/e)$ with respect to function g. In term of h, let v'_i represent a batch F'_i selected at iteration i, then we have

$$\Delta_h(v'_i \mid \omega_{i-1}) \geq (1 - 1/e) \max_{v'} \Delta_h(v' \mid \omega_{i-1})$$

As shown in Lemma 4, h is adaptive monotone and submodular, thus by Thm. 5.2 [21], we obtain:

$$SOL_M \geq \left(1 - e^{-(1-\frac{1}{e})}\right) OPT_M$$

□

B. Two-stage Stochastic Linear Program

In this section, we present a two-stage stochastic programming based approach to exactly solve the FOB problem. We first use binary decision variables x_u to represent whether or not user u is selected as a target to friend, i.e, $x_u = 1$ if user u is selected, and 0, otherwise.

We impose on $\mathbf{x}$ the batch size constraint $\sum_{u \in \mathcal{A}(s)} x_u = k$. Variables $\mathbf{x}$ are known as first stage variables. The values of $\mathbf{x}$ are to be decided before the actual realization of the uncertain parameters in G. For each realization $\phi \sim \omega$, we denote the neighborhood of u as $N^\phi(u)$, and the set of nodes that accept the requests as $\mathcal{A}^\phi(s)$. Let $B(x, y, \phi)$ represent the benefit gain of friend requests $\mathbf{x}$ under a realization ϕ, $F(s)$ represent the set of friends of s, $FoF(s)$ represent the set of friend of friend of s, $\mathcal{C}(s) = V \setminus F(s) \setminus FoF(s)$, and $y_v = 1$ when v is a new friend of friend of s, and 0 otherwise. We have:

$$\begin{aligned}B(x, y, \phi) = \max \sum_{u \in \mathcal{A}^\phi(s)} x_u (B_f(u) + \sum_{v \in N^\Phi(u)} B_i(u, v)) \\ + \sum_{u \in \mathcal{C}(s)} B_{fof}(u) y_u^\phi\end{aligned}$$

For each realization ϕ, $B(x, y, \phi)$ can be computed using a second stage integer programming:

$$\max \quad B(x, y, \phi) \tag{3}$$

$$\text{s. t. } y_v^\phi \leq \sum_{v \in N^\phi(u)} x_u, \forall v \in \mathcal{C}(s) \tag{4}$$

$$y_v^\phi + x_v \leq 1, \forall v \in \mathcal{A}^\phi(s) \tag{5}$$

$$y_v^\phi \in \{0, 1\}, \forall v \in \mathcal{C}(s) \tag{6}$$

The two-stage stochastic integer formulation for the Max-Crawling problem is as follows. We denote Φ as a random realization that is consistent with ω.

$$\max_{x \in \{0,1\}^n} \quad \mathbb{E}\left[B(x, y, \Phi)\right] \tag{7}$$

$$\text{s. t. } \sum_{u \in V \setminus F(s)} x_u = k \tag{8}$$

$$x_u \in \{0, 1\}, \forall u \in V \setminus F(s) \tag{9}$$

The objective is to maximize the expected gain $\mathbb{E}\left[B(x, y, \Phi)\right]$. This stochastic programming problem is, however, not yet ready to be solved with a linear algebra solver.

1) Discretization: To solve a two-stage stochastic problem, one often need to discretize the problem into a single (very large) linear programming problem. That is we need to consider all possible realizations $\Phi \sim \omega$ and their probability masses $P(\Phi = \phi)$. Since the objective involves only the *expected gain* of the *second stage* variables y_v^ϕ, the two-stage stochastic program can be discretized into a mixed integer programming, as follows.

$$\max \sum_{\Phi \sim \omega} P(\Phi = \phi) B(x, y, \phi) \tag{10}$$

$$\text{s.t.} \sum_{u \in u \in V \setminus F(s)} x_u = k, \tag{11}$$

$$x_u \in \{0, 1\}, \forall u \in V \setminus F(s) \tag{12}$$

$$y_v^\phi \leq \sum_{v \in N^\Phi(u)} x_u, \forall v \in \mathcal{C}(s), \forall \phi \sim \omega \tag{13}$$

$$y_v^\phi + x_v \leq 1, \forall v \in \mathcal{A}^\phi(s), \forall \phi \sim \omega \tag{14}$$

$$y_v^\phi \in \{0, 1\}, \forall v \in \mathcal{C}(s), \forall \phi \sim \omega \tag{15}$$

The objective is to maximize the expected gain. In (10), $P(\Phi)$ denotes the probability of having the realization ϕ, and the remainder of the objective computes the benefit gained from the batch of friend requests under realization ϕ. The first pair of constraints limit the size of the batch to k. The remaining constraints pertain to benefit calculation. As the benefits gained from each ϕ are different, we calculate them separately. Within each realization ϕ, all the edges and acceptances are revealed and thus we can neglect the probabilities. We ensure that each benefit is counted only once by constraining the x's and y's to $\{0, 1\}$, and that no node has both B_f and B_{fof} counted for it (constraint (14)). In constraint (13), we ensure that node v becomes a friend of friend of s only if $N(v) \cap N(s) \neq \emptyset$.

2) Realization Reduction: An approach to reduce the number of realizations is to apply the Sample Average Approximation (SAA) method. We generate T independent samples $\phi^1, \phi^2, \cdots, \phi^T$ using Monte Carlo simulation (i.e. to generate sets $N^\phi(u)$ and $A^\phi(s)$). The objective $\mathbb{E}[B(x, y, \Phi)]$ is approximated by the sample average. The new formulation is

$$\max \quad \frac{1}{T} \sum_{\phi \in \phi^1, \cdots, \phi^T} B(x, y, \phi)$$

As $T \to \infty$, the objective function converges to $\mathbb{E}[B(x, y, \Phi)]$ with probability 1. Moreover, an optimal solution of the sample average approximation provides an optimal solution of the stochastic program with probability approaching 1 exponentially fast w.r.t. T. Formally, denote by x^* and $\hat{x}$ the optimal solution of the stochastic programming and the sample average approximation, respectively. For any $\epsilon > 0$, by [22] we have

$$\begin{aligned} &\mathsf{Pr}\left[\mathbb{E}\left[B(\hat{x}, y, \Phi)\right] - \mathbb{E}\left[B(x^*, y, \Phi)\right] > \epsilon\right] \\ &\qquad \leq \binom{n}{k} \exp\left(-T \frac{\epsilon^2}{\delta_{max}^2}\right) \end{aligned} \tag{16}$$

where δ_{max}^2 is upper bounded by the maximum variance for the random variable $B(x, y, \Phi)$. As the benefits are constant and B_f, B_{fof} are usually larger than B_i, we have $B(x, y, \Phi) \sim \Omega(n)$ and $\delta_{max}^2 \leq n^2$. Therefore, if $T \geq \frac{n^2}{\epsilon^2}(k \log n - \log \alpha)$, then $\mathsf{Pr}\left[\mathbb{E}\left[B(\hat{x}, y, \Phi)\right] - \mathbb{E}\left[B(x^*, y, \Phi)\right] < \epsilon\right] > 1 - \alpha$ for any $\alpha \in (0, 1)$.

As discussed in Theorem 2, we immediately have the following theorem.

Theorem 3. *If Alg. 1 used the above IP instead of* BATCHSELECT*, the total benefit gained from Alg. 1 is at least* $(1 - 1/e)$ *from* OPT_M.

C. Generalization

We now detail the theory enabling the generalized cost function, acceptance model, and batch selection.

Cost Function. When $c(u)$ is no longer a uniform cost, we slightly modify BATCHSELECT to greedily select u such that:

$$u = \arg\max_{u \in V \setminus F} \frac{\Delta_f(u | F', \omega)}{c(u)}$$

As g is still monotonic submodular, Lemma 2 is still held.

Retrying Failed Requests. An unlikely rejection early in a batch may greatly hurt the performance of P_M-AReST, especially as the batch size increases. Allowing rejected requests to be retried in later batches is not only more realistic, but allows s to recover some of this lost performance. Alg. 1 is slightly modified to put rejected nodes back and update the cache.

Theoretical Analysis. As $K = mk$, each node u may be being requested at most m times. To represent these repeated request, we construct an auxiliary graph $G_a = (V_a, E_a)$ as shown in Fig. 3. For each $u_i \in V$, add u_{i0} and m nodes u_{ij} for $j = 1...m$ to V_a. Also add a directed edge (u_{ij}, u_{i0}) to E_a. For any $(u_i, u_j) \in E$, add (u_{i0}, u_{j0}) to E_a. Let $V_a = V_n \cup V_o$ where $V_n = \{u_{ij} \mid i = 1...|V|, j = 1...m\}$ and $V_o = \{u_{i0} \mid i = 1...|V|\}$. Likewise, let $E_a = E_n \cup E_o$ where $E_n = \{(u_{ij}, u_{i0}) \mid i = 1...|V|, j = 1...m\}$ and $E_o = \{(u_{i0}, u_{j0}) \mid i = 1...|V|, j = 1...m\}$. Each edge

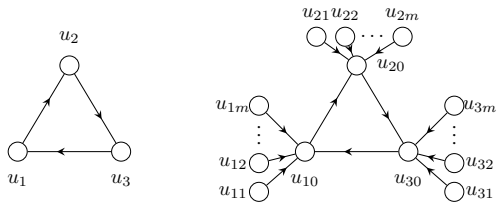


Fig. 3: An auxiliary graph to represent the repeated friend requests. The figure on the left is the original graph G and the one on the right is the auxiliary graph G_a.

$(u_{i0}, u_{j0}) \in E_o$ have the same probability with p_{u_i,u_j} in E. Each edge $(u_{ij}, u_{i0}) \in E_n$ will has a probability randomly draw from probability distribution D_{u_i}.

The realization in G_a is now defined on edge only. For each edge $e = (u_{ij}, u_{0i}) \in E_n$, $Y_e \in \{0, 1, ?\}$ where 0 indicates node u_i rejected the friend request at time j, 1 indicates accept, and ? refers not being requested yet. For each edge $e = (u_{i0}, u_{j0}) \in E_o$, $Y_e \in \{0, 1, ?\}$ where 0 indicates $u_j \notin N(u_i)$, 1 indicates $u_j \in N(u_i)$, and ? refers the unknown. For each i, only one edge in the set of $\{(u_{ij}, u_{i0})\}$ can be 1.

Given a realization ω of G_a, a node $u_i \in V$ is a friend of s if u_{i0} is one hop away from any node in V_n via an edge e with $Y_e = 1$, called live edge. u_i is a friend of friend of s if u_{i0} is two hops away from any node in V_n via a live path (of length 2). All of the following analysis will be on G_a.

Lemma 5. BATCHSELECT *returns a batch with a benefit gain at least* $(1 - 1/e)$ *from* OPT_b

Proof. Note that during the execution of BATCHSELECT, no observation has been done. Thus it is straightforward to prove this lemma using similar arguments as in Lemma 2. □

Consider a decision tree of an attacking strategy π where each branch from level i to level $i + 1$ corresponds to ω_i after round i under π. Let $\Lambda_i = \{\omega_i^1, \ldots, \omega_i^{|\Lambda_i|}\}$ be the set of all partial realization from level i to $i + 1$. ($|\Lambda_i|$ = # of the branches.) Each node in this tree represents a *batch* selected.

Let $F = \{w_1, \ldots, w_m\}$ be the set of batches selected by the modified Alg 1 in that order. Each w_i denote one batch. Let $F^* = \{c_1, \ldots, c_m\}$ be the set of batches selected by the optimal solution in that order. And finally let $F_i = \{w_1, \ldots, w_i\}$. Likewise for F^*. We have the following lemma

Lemma 6.

$$h(F^*) \leq h(F_i) + m\Delta_i \text{ for } 0 \leq i \leq m-1 \qquad (17)$$

where $\Delta_i = h(F_{i+1}) - h(F_i)$ *and* $h(F_i) = \sum_{j=1}^{|\Lambda_i|} \sum_{\phi \sim \omega_i^j} P[\phi \mid \omega_i^j] h(F_i, \phi)$

Proof. Consider any fixed iteration i, for $1 \leq z \leq m$, we have:

$$\begin{aligned} h(F_z^* \cup F_i) - h(F_{z-1}^* \cup F_i) &\leq h(F_i \cup \{c_z\}) - h(F_i) \\ &\leq h(F_{i+1}) - h(F_i) = \Delta_i \end{aligned}$$

Therefore,

$$\begin{aligned} &\sum_{z=1}^{m} \big(h(F_z^* \cup F_i) - h(F_{z-1}^* \cup F_i)\big) \\ \leq &h(F^* \cup F_i) - h(F_i) \leq m\Delta_i \end{aligned}$$

□

Theorem 4. *The modified Alg. 1 to this general setting has an approximation ratio of* $(1 - e^{-(1-\frac{1}{e})})$.

Proof. We first show that $(1 - 1/e)h(F^*) \leq h(F)$. From Lemma 6, we have:

$$h(F^*) \leq h(F_i) + m\Delta_i = \Delta_0 + \cdots + \Delta_{i-1} + m\Delta_i \qquad (18)$$

For the second term of Eq. (18), by the definition of Δ_i, it is easy to verify that $\Delta_0 + \cdots + \Delta_{i-1} = h(F_i)$.

Multiplying both sides of (18) by $(1 - \frac{1}{m})^{(m-1-i)}$, and adding them up for $0 \leq i \leq m - 1$, we have:

$$\begin{aligned} m(1 - (1 - \frac{1}{m})^m)h(F^*) &\leq m(\Delta_0 + \cdots + \Delta_{m-1}) \\ &= mh(F) \end{aligned} \qquad (19)$$

Since $(1 - 1/m)^m \to 1/e$ when $m \to \infty$, we have $h(F) \geq (1 - 1/e)h(F^*)$. Combining with Lemma 5, we complete the proof. □

Remarks. Even though the theoretical performance bound is the same for both non-repeat and repeat failed requests, experiments show that re-sending friend requests obtain more benefit. This approach allows s having more attempts to be friend with some critical and important users.

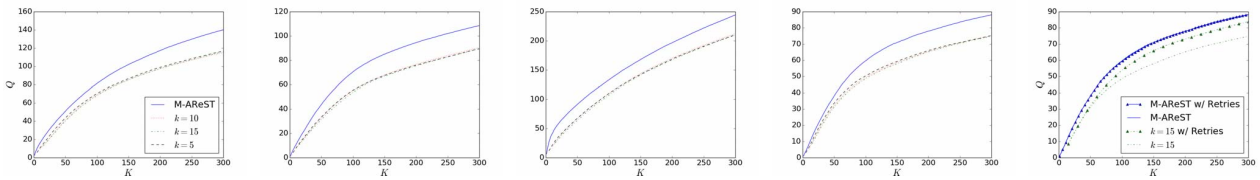
Varying Batch Sizes. It is suspicious for s to keep sending the same number of requests simultaneously. Thus s may want to vary the batch sizes. One simple way is that for each batch F' obtained by Alg. 1, s can break F' into several subsets with different sizes and send the requests to users in each subset simultaneously, without observation. Alternatively, Alg.1 can be modified to execute BATCHSELECT with different input k, randomly pick in a range of $[k_{min}, k_{max}]$. Thus for the same set of F obtained by Alg. 1, we can have different patterns for attack. Therefore, we are interested in analyzing the relationship between different patterns. Due to the space limit, we omit the following proof.

Theorem 5. *Let* π_s^* *be an optimal adaptive sequential strategy with length* K*, and* π *the adaptive batch strategy selecting* m *batches of varying sizes, we have:*

$$f(\pi) \geq (1 - e^{-(1-1/e)^2}) f(\pi_s^*)$$

V. EXPERIMENTAL EVALUATIONS

Prior work has established the utility of AReST as a tool for measuring the vulnerability of OSNs to reconnaissance attacks [3]. We extend AReST to the Max-Crawling problem, terming this variant M-AReST, to enable direct comparison to P_M-AReST. In this section, we show that in realistic settings attacks conducted with P_M-AReST are significantly more dangerous. To further support this, we explore the practical performance of our batch algorithm and show that it scales



(a) Enron Email (b) Slashdot (c) Facebook (d) Twitter (e) Twitter, retries allowed.

Fig. 4: Benefit Q as a function of friend requests sent K.

Network	Nodes	Edges
US Pol. Books [24]	105	441
Facebook	4k	88k
Enron Email	37k	184k
Slashdot	77k	905k
Twitter	81k	1.77M

TABLE I: Networks used in simulations. All networks are from SNAP [25] unless otherwise noted.

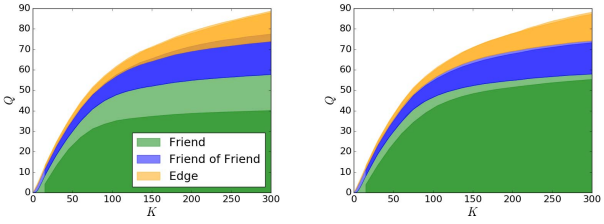
very well with both graph size and available compute power. Further, we experiment with an extended model that allows recovery from failed friend requests and show that this closes the gap between M-AReST and P$_M$-AReST. To confirm that our results generalize, we run on a variety of networks (see Table I).

To decide the value of k, we note that the threat of detection forms a natural limit on the batch size used. Boshmaf et al. [2] elected to submit no more than 25 friend requests per day on Facebook after observing the detection patterns in a pilot trial. Later, Yang et al. [23] found that "accounts sending more than 20 invites per [hour] are Sybils". Their data additionally shows that the 95th percentile normal user sends fewer than 5 invites per hour. These lead us to a firm upper limit of 25 on the batch size, assuming a time interval of one day. With shorter intervals, smaller batch sizes will be necessary. Even with the one-day interval, the advances in detection since [2] would seem to encourage smaller batch sizes in general. We therefore focus our experiments on the $k = 1$ to $k = 15$ range.

P$_M$-AReST is implemented in Rust[2] and run, with M-AReST, on a server that can support up to 36 simultaneous threads. To focus on the general performance of P$_M$-AReST, we fix the costs at uniform values. To ensure reliable comparison to M-AReST, we adopt the same benefit model: $B_f(u) = 1$ if $u \in T$, 0 otherwise. $B_{fof}(u) = 0.5$ if $u \in T$, 0 otherwise. $B_i(u, v) = 2^{|\{u,v\} \cap T|}/M$, where M is the maximum expected degree of any node in the network. Unless stated otherwise, each simulation is repeated 100 times and the results averaged.

A. Algorithm Performance

To begin, we show that P$_M$-AReST performs similarly to M-AReST on Max-Crawling. Fig. 4 shows that although there is a gap in performance between M-AReST and P$_M$-AReST, the latter remains competitive, even when the batch size is allowed to vary (Fig. 7). From Fig. 5, it is clear that this performance gap is largely due to the difference in friend benefit between M-AReST and P$_M$-AReST. This is somewhat stymied by P$_M$-AReST's larger FoF benefit, though the gap is still significant.



(a) $k = 15$ (b) $k = 15$, retries allowed

Fig. 5: Breakdown of benefit by source on Twitter. Lightly-shaded regions indicate that M-AReST reached greater benefit than P$_M$-AReST. Brown regions indicate the opposite.

The question remains: is this a result of the BATCHSELECT algorithm or an inherent property of any batch-based solution? We run M-AReST, P$_M$-AReST with BATCHSELECT, and P$_M$-AReST with the Discretized MIP from sec. IV-B. A comparison is shown in Fig. 6. The small US Political Books network is used due to the exorbitant running time and memory consumption of solving the MIP, which is compounded by the fact that sampling must be repeated before each batch to collect only samples that are consistent with the current partial realization. The MIP is implemented and solved with CPLEX. In the end, the optimal batch selection routine does only marginally better than BATCHSELECT, leading us to conclude that P$_M$-AReST is a near-optimal adaptive batch algorithm.

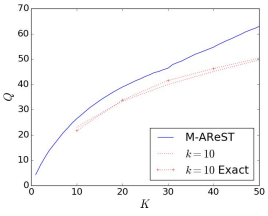


Fig. 6: BATCHSELECT vs. Exact MIP Solution (1000 samples for each batch) on the US Pol. Books dataset.

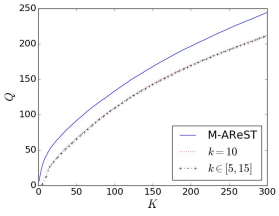


Fig. 7: Performance on Facebook when k is allowed to vary each step on $[5, 15]$.

[2]https://www.rust-lang.org

Threads	Enron Email	Facebook	Slashdot	Twitter
5	**0.92**	0.89	**0.93**	**0.98**
10	0.86	0.80	0.86	**0.97**
15	0.79	0.70	0.80	**0.95**
20	0.73	0.61	0.74	**0.94**
25	0.70	0.57	0.71	**0.93**
30	0.67	0.52	0.68	**0.93**

TABLE II: Mean fraction of available compute power utilized by P_M-AReST with $K = 300$ and $k = 15$.

When we enable retrying rejected requests, the gap between M-AReST and P_M-AReST is dramatically reduced, as shown in Fig. 4e. Note that M-AReST is treated as having a batch size of 1 for this process, and that it performed no better than shown in Fig. 4d. From Figs. 4e & 5b, we can reasonably ascertain that the loss of benefit relative to M-AReST arises primarily from ambitious node selection within a batch, and that the simple act of allowing recovery from these failed requests nearly eliminates the problem.

B. Efficiency & Scalability of P_M-AReST

Batch Size	Enron Email	Facebook	Slashdot	Twitter
M-AReST	11.90	3.46	16.80	477.73
5	30.22	8.67	35.22	900.13
10	114.93	30.70	133.48	2069.02
15	1678.79	391.95	1876.44	8629.69

TABLE III: Mean compute time in seconds across all threads for P_M-AReST with $K = 300$.

While the practical limitations of a real-world socialbot constrain the cost of computing the batches to reasonable levels, larger batches still come at a noticeable cost. Table III shows the mean compute time necessary to complete a simulation of P_M-AReST with $K = 300$ (chosen as each $k \in \{5, 10, 15\}$ evenly divides 300).

However, these values are tempered by the fact that P_M-AReST parallelizes extremely well, as shown in Table II. On smaller networks (e.g. Facebook), the CPU utilization drops towards 50% as available compute power outstrips demand. However, on even a medium-sized network such as Twitter, the utilization remains above 90%.

C. Real-Time Network Vulnerability

Li et al. introduced the *Reconnaissance Resistance Score (RRS)*, the expected number of friend requests required to reach a given benefit threshold Q, as a metric of network vulnerability to the reconnaissance attacks. The intuition is that the number of friend requests corresponds roughly to the amount of time a socialbot employing AReST would take to infiltrate a network. However, when requests are made sequentially as in AReST the delay introduced by waiting for user responses will dominate the computation time in real-world scenarios. The P_M-AReST algorithm addresses this problem by sending requests in batches.

To illustrate the difference between these approaches, we extend RRS from the expected number of friend requests to the expected time (RT-RRS). Table IV shows the values of both metrics. From this, it is clear that the RRS values are similar between M-AReST and P_M-AReST. However, from the RT-RRS values we can conclude that even when the expected response time for a friend request is a mere 5 minutes, P_M-AReST would infiltrate the network an order of magnitude faster than M-AReST.

No Delay	Enron Email	Facebook	Slashdot	Twitter
M-AReST	$\mathbf{1.0 \times 10^{-1}}$	$\mathbf{2.0 \times 10^{-2}}$	$\mathbf{1.6 \times 10^{-1}}$	**2.2**
$k = 5$	3.4×10^{-1}	5.8×10^{-2}	5.5×10^{-1}	6.3
$k = 10$	9.0×10^{-1}	1.5×10^{-1}	1.5	1.2×10^{1}
$k = 15$	1.2×10^{1}	1.6	1.9×10^{1}	6.5×10^{1}
5 minutes	Enron Email	Facebook	Slashdot	Twitter
M-AReST	6.4×10^{2}	3.7×10^{2}	8.3×10^{2}	1.0×10^{3}
$k = 5$	1.5×10^{2}	8.5×10^{1}	2.0×10^{2}	2.4×10^{2}
$k = 10$	7.7×10^{1}	4.1×10^{1}	9.8×10^{1}	$\mathbf{1.3 \times 10^{2}}$
$k = 15$	$\mathbf{6.1 \times 10^{1}}$	$\mathbf{2.9 \times 10^{1}}$	$\mathbf{8.3 \times 10^{1}}$	1.4×10^{2}
1 hour	Enron Email	Facebook	Slashdot	Twitter
M-AReST	7.7×10^{3}	4.4×10^{3}	9.9×10^{3}	1.2×10^{4}
$k = 5$	1.8×10^{3}	1.0×10^{3}	2.4×10^{3}	2.8×10^{3}
$k = 10$	9.1×10^{2}	4.9×10^{2}	1.2×10^{3}	1.4×10^{3}
$k = 15$	$\mathbf{6.0 \times 10^{2}}$	$\mathbf{3.3 \times 10^{2}}$	$\mathbf{7.8 \times 10^{2}}$	$\mathbf{9.8 \times 10^{2}}$
1 day	Enron Email	Facebook	Slashdot	Twitter
M-AReST	1.8×10^{5}	1.1×10^{5}	2.4×10^{5}	2.9×10^{5}
$k = 5$	4.4×10^{4}	2.4×10^{4}	5.7×10^{4}	6.8×10^{4}
$k = 10$	2.2×10^{4}	1.2×10^{4}	2.8×10^{4}	3.3×10^{4}
$k = 15$	$\mathbf{1.4 \times 10^{4}}$	$\mathbf{7.8 \times 10^{3}}$	$\mathbf{1.8 \times 10^{4}}$	$\mathbf{2.2 \times 10^{4}}$

TABLE IV: Exepected *Real-Time Reconnaissance Resistance Scores* (in seconds-per-benefit) under increasing user response delays. These values are computed by adding the delay d between each logged batch step to take into account the increasing running time required to compute larger batches.

VI. Conclusion

In this paper, we investigate an adaptive and parallel strategy, of which attackers can simultaneously send multiple friend requests in batch and recover from failed requests by retrying in later batches. We introduce Max-Crawling, and show its inapproximability of $(1-1/e+\epsilon)$. We first design our core algorithm P_M-AReST which has an approximation ratio of $(1 - e^{-(1-1/e)})$ and next provide a near-optimal solution with an approximation ratio of $(1-1/e)$. We also establish the gap bound of $(1-e^{-(1-1/e)^2})$ between batch strategies versus the optimal sequential one. Extensive experiments not only confirm the performance of our algorithm, but also provide new insights towards privacy protection under reconnaissance attacks.

Acknowledgment

This work is supported in part by the NSF grant #CCF-1422116 and DTRA HDTRA1-14-1-0055.

References

[1] E. Novak and Q. Li, "A survey of security and privacy in online social networks," *College of William and Mary Computer Science Technical Report*, 2012.

[2] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The Socialbot Network: When Bots Socialize for Fame and Money," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. ACM, 2011, pp. 93–102.

[3] X. Li, J. D. Smith, T. N. Dinh, and M. T. Thai, "Privacy issues in light of reconnaissance attacks with incomplete information," in *Proceedings of the 2016 IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE/WIC/ACM, 2016.

[4] H. T. Nguyen and T. N. Dinh, "Targeted Cyber-attacks: Unveiling Target Reconnaissance Strategy via Social Networks," in *Proceedings of the IEEE Int Conf. on Computer Com., Security and Privacy in BigData Workshop*, ser. INFOCOM BigSecurity 2016, 2016.

[5] I. Jeun, Y. Lee, and D. Won, "A Practical Study on Advanced Persistent Threats," in *Computer Applications for Security, Control and System Engineering*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2012, no. 339, pp. 144–152.

[6] T. Ryan and G. Mauch, "Getting in bed with robin sage," in *Black Hat Conference*, 2010.

[7] A. Elyashar, M. Fire, D. Kagan, and Y. Elovici, "Homing socialbots: intrusion on a specific organization's employee using socialbots," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 2013, pp. 1358–1365.

[8] Y. Boshmaf, K. Beznosov, and M. Ripeanu, "Graph-based sybil detection in social and information systems," in *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*. IEEE, 2013, pp. 466–473.

[9] A. Paradise, A. Shabtai, and R. Puzis, "Hunting Organization-Targeted Socialbots," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ser. ASONAM '15. New York, NY, USA: ACM, 2015, pp. 537–540.

[10] T. N. Dinh, Y. Xuan, M. T. Thai, P. M. Pardalos, and T. Znati, "On New Approaches of Assessing Network Vulnerability: Hardness and Approximation," *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 609–619, 2012.

[11] M. A. Alim, N. P. Nguyen, T. N. Dinh, and M. T. Thai, "Structural Vulnerability Analysis of Overlapping Communities in Complex Networks," in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01*, ser. WI-IAT '14. IEEE Computer Society, 2014, pp. 5–12.

[12] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, "Assessing the vulnerability of the fiber infrastructure to disasters," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 6, pp. 1610–1623, 2011.

[13] N. P. Nguyen, T. N. Dinh, Y. Shen, and M. T. Thai, "Dynamic social community detection and its applications," *PloS one*, vol. 9, no. 4, p. e91431, 2014.

[14] Y. Shen, N. P. Nguyen, Y. Xuan, and M. T. Thai, "On the discovery of critical links and nodes for assessing network vulnerability," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 3, pp. 963–973, 2013.

[15] N. P. Nguyen, Y. Xuan, and M. T. Thai, "A novel method for worm containment on dynamic social networks," in *Military Communications Conference, 2010-MILCOM 2010*. IEEE, 2010, pp. 2180–2185.

[16] T. N. Dinh and M. T. Thai, "Network under joint node and link attacks: Vulnerability assessment methods and analysis," *IEEE/ACM Transactions on Networking*, vol. 23, no. 3, pp. 1001–1011, 2015.

[17] M. Fire, L. Tenenboim, O. Lesser, R. Puzis, L. Rokach, and Y. Elovici, "Link prediction in social networks using computationally efficient topological features," in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 73–80.

[18] M. Fire, R. Puzis, and Y. Elovici, "Link prediction in highly fractional data sets," in *Handbook of computational approaches to counterterrorism*. Springer, 2013, pp. 283–300.

[19] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 635–644.

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

[21] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, pp. 427–486, 2011.

[22] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002.

[23] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 1, p. 2, 2014.

[24] V. Krebs, "Books about US politics," unpublished, compiled by Mark Newman. Retrieved from http://www-personal.umich.edu/ mejn/netdata/.

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