Bandit Algorithms for Social Network Queries

Zahy Bnaya and
Rami Puzis
Information System Engineering Department and
Telekom Innovation Laboratories
Ben Gurion University, Beer Sheva, Israel
Email: {zahy,puzis}@bgu.ac.il

Roni Stern
SEAS
Harvard University
Cambridge MA, USA
Email:roni.stern@gmail.com

Ariel Felner
Information System Engineering Department
Ben Gurion University, Beer Sheva, Israel
Email: felner@bgu.ac.il

***Abstract*—In many cases the best way to find a profile or a set of profiles matching some criteria in a social network is via targeted crawling. An important challenge in targeted crawling is to choose the next profile to explore. Existing heuristics for targeted crawling are usually tailored for specific search criterion and could lead to short-sighted crawling decisions. In this paper we propose and evaluate a generic approach for guiding a social network crawler that aims to provide a proper balance between exploration and exploitation based on the recently introduced variant of the Multi-Armed Bandit problem with *volatile* arms (VMAB). Our approach is general-purpose. In addition, it provides provable performance guarantees. Experimental results indicate that our approach compares favorably with the best existing heuristics on two different domains.**

I. Introduction

Online *social networks* (SNs) such as Facebook, Twitter, Google+, or Academia.edu are a part of everyday life for many people around the world. SNs are a source of valuable information. This information can be used for tracking public opinions, political trends, disseminate news, or just socialize by finding the right group of people to communicate with. Extracting this information from SNs, referred to as *SN querying*, is widely practiced by commercial companies, government agencies and even individual people. The objective of SN queries is to pinpoint profiles that provide valuable information according to some criteria. SN queries can select subjects for surveys, research, marketing purposes etc.

Lately, Facebook realized the need for SN queries and introduced the *SN search* feature. However, many other SN services lag behind and do not provide search capabilities at all. Generally, in SN queries, a complex criteria is required to define which profiles are relevant and which are not. In case of complex SN queries, the relevant profiles are commonly pinpointed using dedicated *crawls* on the SN. We assume that during such crawls, a function for analyzing a profile and evaluating its relevance, referred to as *profile-acquisition*, is repeatedly executed in order to: (1) Extract information from the analyzed profiles and (2) Extract new profiles from their lists-of-friend (LOF) which become candidates for future *profile-acquisition*. The aim of this paper is to provide efficient policies for guiding the SN crawling process by intelligently choosing the next profile to acquire. The challenge is to focus the search at relevant areas of the network that are most likely to contain valuable profiles which will satisfy the criteria of the query.

This research was partially funded by IDF and by the Chief Scientist in the Israeli Ministry of Economy under the Magneton Program.

Previous work has shown that blind, arbitrary selection of next profiles to examine (aka. snowball crawling) is inefficient [1]. Thus, a number of intelligent selection methods for the next profile to acquire were introduced [1]–[3]. These methods of intelligent crawling usually devised task-specific heuristics, based on the homophile principle, which choose SN profiles that tend to have many properties in common such as age-group, gender, common friends, etc. However, these works overlooked an important aspect of SN querying - the *exploration vs. exploitation* tradeoff. A deeper look reveals that the effect of the profile acquisition operation is twofold. **(1) exploitation:** It may immediately contribute to the knowledge collected by the process (e.g., if the profile turns to be relevant to the search criteria and; **(2) exploration:** By exposing the LOF of this profile, the acquisition enriches available information about the SN, thus, allowing more accurate acquisition decisions in later stages. These two effects constitute a natural *exploration versus exploitation* tradeoff. Existing SN querying methods are *pure-exploitation* in the sense that a specific, task-dependent ad-hoc heuristic function is suggested which aims to predict the likelihood that the profile will meet the query criteria. Profiles that achieve the maximal heuristic value are selected for the next acquisition. None of these methods gave weight to *exploration* aspects.

In this paper we introduce a method which considers both *exploration* and *exploitation* in SN querying. A common approach in the Artificial Intelligence (AI) literature for balancing exploration and exploitation is via the formulation of Multi-armed bandit problem (MAB) [4]. We propose a generic algorithm for *SN querying* based on MAB which provides a viable balance between exploration and exploitation. Two important properties of our new approach are: **(1)** it guarantees performance bounds on the crawling process and **(2)** it does not require an ad hoc heuristic function tailored to the search query. Instead, it is based merely on past experience of profiles acquired during the search. We provide experimental results on two domains with different search queries and compare our approach to state of the art heuristics in Section IX. The first domain is the Target Oriented Network Intelligence Collection (TONIC) [1]. The second domain is community search in the DBLP network[1]. We show that our approach outperforms existing methods on the tested domains and more importantly, provides performance guarantees.

[1] http://dblp.uni-trier.de/xml/

II. Problem Description

A SN can be represented as a graph $G = (V, E)$ where V is the set of the SN profiles and E are links between them. Once analyzed, each profile $v \in V$ might reveal relevant information for the SN query. Let $\omega(v)$ denote the *utility* gained by analyzing a profile. Intuitively, *utility* indicates the degree of *relevance* of the profile to the query. We define a *profile acquisition* operation that reveals its content. In particular, *profile-acquisition* consists of two functions: **Find neighbors:** reveals all the edges and vertices connected to v; and **Calculate** $\omega(v)$**:** calculates the utility of v using the data extracted from the profile. The *profile-acquisition* operation is done via external information extraction methods which download the profile data, parse it and extract relevant information. We note that the bottleneck of the acquisition operation is the actual downloading of profile data. We thus assume that once the data was downloaded, it is possible to quickly extract any available information from this data, including the neighbors of that profile. The information extraction methods used to implement the profile acquisition action are beyond the scope of this paper and have been discussed by others in prior work [5], [6].

Due to the size of SNs, we assume that only a small part of the network is known during SN querying. Following past notation [1], [7], let $CKG = (V', E')$ represent the *currently known graph*, where $V' \subseteq V$ and $E' \subseteq E$. The CKG contains two types of nodes: **Internal nodes** - Profiles that have been acquired; and **Frontier nodes** - Profiles that have have been revealed (i.e. connected to internal nodes) but were not acquired yet. Edges incident to a node are revealed when the node is acquired. Thus, E' contains only edges connecting internal nodes to other nodes. For example, consider the CKG shown in Figure 2(a). The dark nodes are internal nodes. All edges incident with them are known. Their neighbors are frontier nodes. Acquiring a profile removes it from the set of frontier nodes and adds it to the set of internal-nodes (see Figure 2(d)). $\omega(v)$ is known for internal-nodes but not for frontier nodes. The aim of crawling is to choose the next frontier node to acquire. Many SNs limit crawling. Moreover, the stealth of the overall process is important. Thus, we assume that the SN query process can perform at most b profile acquisitions before it is forced to halt (b may be unknown).

The input to the *SN querying* problem are (a) the utility function, ω (b) a set of known profiles, known as the *seeds*. These profiles constitute the initial internal nodes and their neighbors are the initial *frontier* nodes. Let Ω be the overall accumulated utility on all acquired profiles defined as $\Omega = \sum_{v \in I} \omega(v)$ where I is the set of internal nodes in CKG. The task in SN querying is to acquire b profiles such that Ω is maximized. The major challenge is to choose which of the known profiles (from the *frontier* nodes) to acquire next.[2]

III. SN Queries as a best-first search problem

Prior work has pointed out that intelligent SN querying can be modeled and solved as a *best-first search* on a graph [1]. Best-first search maintains two lists of nodes *OPEN list* and *CLOSED list*. At every step, the *best* node (according to some cost or utility function) from OPEN is *expanded*, i.e., it is popped from OPEN and moved to CLOSED. Its children are *generated* and being inserted to OPEN. In SN querying, initially, the *seeds* are generated and added to OPEN. The internal nodes are the nodes that have been expanded (those in CLOSED) and the *frontier* nodes are those that have been generated but not yet expanded (those in OPEN).

The heart of the querying task is to choose a *frontier* node (from OPEN) to acquire next. The content of a given profile v such as its residence information, education, etc, might provide an indication about its utility $\omega(v)$. However, all of this information is available only after the profile has been acquired and its utility $\omega(v)$ has been calculated. Mostly, heuristic functions for a *frontier* node v are based on two types of information: (a) topological properties derived from v's connections in the CKG (e.g., its known degree in the CKG) and (b) the already known utility $\omega(v)$ of the internal nodes that are connected to v. We call this type of heuristic functions *topological* heuristics. Many existing heuristics for SN querying fall into this category of topological heuristic functions [1], [7]. Coming up with a heuristic function for a SN querying task, is done by carefully examining the query and understanding the topological features that can indicate relevance for that *specific* query. Moreover, such heuristic functions usually make some assumptions about the topology of the graph and the relations between neighbors.

For example, consider the *target oriented intelligence crawling* (TONIC) domain [1], where the query searches for profiles that know or has interacted with a certain specific profile (called the *target*). The utility of a profile $\omega(v)$ in this case is a boolean function indicating whether a profile contains information about the target or not. A heuristic function that proved efficient in this domain is called *known degree* (KD). This heuristics ranks high *frontier* nodes that have many connections with other nodes in CKG that were already found to have information about the *target*. KD assumes that profiles that have information about a target profile are likely to have connections to other profiles that have information about that target profile. In [1], other topological heuristics were given, all based on this principle.

A. Our approach: balance exploration and exploitation

An interesting property of topological heuristic functions is that many frontier nodes share the same topological properties and thus have the same heuristic value. We use the notion of *structural equivalence* classes (SEC) to group nodes having exactly the same set of neighbors [8]. Figure 1 demonstrates three different SECs ($E1$,$E2$,$E3$). Topological heuristics values (such as the *known degree* heuristic) are identical for all *frontier* nodes that belong to the same SEC as these are identical from the perspective of the network topology.

Let e be a SEC. Consider $\omega(e)$ to be a random variable representing the utility of a random *frontier* node that belongs to SEC e. Let $E[\omega(e)]$ be the expected utility obtained by selecting to acquire a profile that belongs to e. Obviously, if $E[\omega(e)]$ was known for all SECs, the optimal acquisition decision on any stage is to acquire a profile from the SEC that achieved the maximal expected utility. However, this is not the case, as for this to happen, we need to acquire the entire

[2] We assume that the budget limit b is not necessarily known. Thus, an *anytime* solution for the problem is to constantly try to increase Ω as fast as possible. However, we also assume that b is large enough, such that the long-term considerations for the acquisition decisions should be taken into account.

SN graph G. Therefore, typical best-first search chooses to expand a *frontier* node from the SEC with nodes having the best heuristic values. We denote such SEC as e^*.

Let $\overline{\omega}(e)$ be the average utility obtained so far by acquiring profiles from SEC e. Naturally, with sufficient acquisitions from a specific SEC, $\overline{\omega}(e)$ can be used as an estimate for $E[\omega(e)]$. We call $\overline{\omega}(e)$ the *exploitation* value of e since by deciding to acquire a profile that achieved the maximal value of $\overline{\omega}(e)$, we *exploit* the knowledge we accumulated so far on the SN graph. Obviously, as the number of acquisitions from a specific equivalence class increases, the estimation of $\overline{\omega}(e)$ becomes more accurate and can be used to perform better acquisition decisions in the future. Thus, performing acquisitions of *frontier* nodes that belong to SEC e also *explores* the potential of selecting SEC e for future acquisitions. We use the term *exploration value of e* to denote the number of profiles acquired from a given SEC e. Low exploration value indicates that the estimated value $\overline{\omega}(e)$ may not be accurate. Therefore, simply choosing to acquire a node from the SEC with maximal $\overline{\omega}(e)$ on each acquisition, without considering the exploration is not enough.

A tradeoff exists between the exploration and the exploitation values of SECs. Specifically, the dilemma in SN querying is to decide which of the following nodes to acquire: **1)** A profile that belongs to the SEC with the maximal estimate, thus maximizing the probability to obtain immediate utility. **2)** A profile from a SEC which has not been "sampled" too many times, thus exploring the topology and providing better estimates for that SEC. Our approach aims to balance exploration and exploitation during SN querying. A classical problem of balancing *exploration vs exploitation* is expressed as the *Multi-armed bandit* (MAB) problem which is discussed in the next section. We use the Multi-armed bandit problem to develop methods that balance exploration and exploitation for SN querying. In the next section, we introduce the MAB problem and then show how we adopt it to our problem.

IV. Multi-armed bandit problem

The multi-arm bandit problem (MAB) [4] assumes a set of K gambling machines (or arms). At each turn (n), a player pulls one of the arms (a_i) and receives a reward $(W_{i,n})$ drawn from unknown distribution with unknown mean value μ_i.

A policy for MAB chooses the next arm to pull based on previously observed rewards. A MAB policy can decide to "exploit" the knowledge accumulated so far by pulling the arm that was shown to produce the best sequence of rewards. Alternatively, a policy can decide to "explore" an arm that was only pulled a few times even though its reward history is less attractive than other arms, in order to get a more accurate estimation of the arm's expected reward.

MAB policies are often designed to minimize their *accumulated regret* (R_n), defined as the expected loss of rewards incurred by not always pulling the arm with the maximal expected reward (denoted as μ^*) in the turns up to n. Formally, $R_n = n \cdot \mu^* - \sum_{i=1}^{K} \mu_i \cdot E[T_i(n)]$ where $E[T_i(n)]$ is the expected number of pulls of arm i in the first n turns.

The *Upper Confidence Bound 1* policy [9] (denoted UCB1) is a common policy ensuring that $R_n = O(log(n))$. Initially, UCB1 pulls each arm once. Then, on each turn n UCB1 pulls an arm a_i that maximizes the following expression:

$$\overline{W_i} + \sqrt{\frac{2 \cdot \ln n}{T_i(n)}} \tag{1}$$

where $\overline{W_i}$ is the average reward observed so far by pulling arm a_i. MAB applications spread on many areas such as clinical trials, web search, advertising and multi-agent systems. Modeling these problems as MAB and using MAB policies for solving them has been shown to be very successful [10]–[13].

V. Modeling SN Querying as MAB

We now model the problem of SN querying as a MAB problem and develop suitable algorithms. Each SEC in the currently known graph CKG, represents an arm a_i. The expected reward μ_i of each arm a_i is the expected utility of acquiring a random frontier node that belongs to the SEC represented by a_i. Thus, $\mu_i = E[\omega(v)]$ where v is a random frontier node of the SEC represented by arm a_i. The immediate reward is received by acquiring v, observing $\omega(v)$ and adding it to Ω.

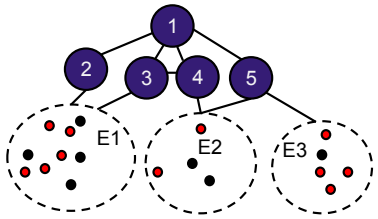


Fig. 1: Structural Equivalence classes in CKG

To demonstrate this model, consider Figure 1. The CKG consists of 5 internal nodes (numbered 1-5) and three SECS. All frontier-nodes that are connected to nodes 1 and 2 are part of SEC $E1$. SEC $E2$ contains all frontier-nodes connected to nodes 4 and 5 and SEC $E3$ contains all frontier nodes that are connected to node 5.

Assume a boolean utility function, where each frontier node v can have either $\omega(v)$ of 1 (colored red in Figure 1) or 0 (colored black). Thus, the actual expected rewards of the arms in this example are $\frac{5}{9}$, $\frac{1}{2}$ and $\frac{4}{5}$ for SECs $E1$, $E2$ and $E3$ respectively. After a profile was selected and acquired, the immediate reward is received and the average utility of the SEC is calculated. Assume further that three acquisitions were performed from SEC $E1$, two acquisitions were performed from SEC $E2$ and additional two acquisitions were performed from $E3$. All three acquisitions of frontier nodes from $E1$ produced utility of 1, both acquisitions of frontier nodes from $E2$ produced utility of 0. The acquisitions of frontier nodes from $E3$ produced utility of 1 and 0. Therefore, the average utilities of the SECs are now 1,0 and $\frac{1}{2}$, respectively. A "pure exploitation" MAB policy will choose to repeatedly acquire frontier nodes from $E1$ until its average utility degrades. A MAB policy that balanced exploration and exploitation might choose to explore $E2$ and $E3$ earlier in order to produce better estimates for these SECs.

VI. The influence of profile acquisitions on SECs

Acquiring profiles cause dynamic changes to the CKG, to the SECs and to their expected utility μ. In this section we describe these changes and show how we modify our MAB formalization accordingly.

A. Non-stationary bandits

As described earlier, when a profile v that belongs to a given arm a_i is acquired, the *frontier* node that represents this profile, changes into an *internal* node. After observing $\omega(v)$, profile v is fully known, it cannot be acquired again and it is no longer relevant for the calculation of μ_i. This is similar to a case of *sampling without replacement* [14]. Thus, the value of μ_i can potentially change after each pull of an arm. For example, in Figure 1, acquiring the single frontier node that produced utility of 0 from SEC $E3$, change μ_3 from $\frac{4}{5}$ to 1.

This phenomenon is called *non-stationary bandits* and it is discussed especially in the context of Monte-Carlo tree search (MCTS) algorithms [15], [16] where the expected rewards of branches in the search tree change as the search progresses. A simple modification to the UCB1 formula exists such that the regret bounds are still logarithmically on such cases [16]. The modified formula is shown in Equation 2:

$$\overline{W_i} + 2C_p\sqrt{\frac{\ln n}{T_i(n)}} \tag{2}$$

where C_p is an appropriate constant as defined by the *UCT* algorithm [16]. Thus, since bandits in SN querying are *non-stationary*, in our algorithms we use Equation 2 as well (instead of Equation1).

B. Dynamic changes of arms

Furthermore, besides the fact that μ can change, the structures of the SECs themselves may dynamically change too. As exploration continues and frontier nodes are being acquired, SECs (arms) can undergo three events: (1) *appear* (2) *disappear* or (3) *split* into two different arms. We define the three events, triggered by a profile acquisition action and demonstrate each event via Figure 2 where internal nodes are dark and frontier nodes are light. The CKG prior to any of the events is depicted at Figure 2(a). In this example there are two internal nodes (1 and 2) and three frontier nodes (3,4 and 5). Edges connecting nodes 3-5 to other parts of the graph are still unknown. This CKG consists of a single SEC (nodes 3,4,5).

Disappear event- Given a SEC e, when all its nodes have been acquired, the arm that represents e can no longer be selected for the next acquisition and it is said to "disappear". Figure 2(b) demonstrates this event. The frontier nodes (3,4 and 5) were acquired and no new edges had been found. Thus, the SEC defined by the internal nodes 1 and 2 "disappears".

Appear event- If a proflie v is acquired and none of the new profiles which are friends of v are already represented in CKG, a new SEC is formed and a new arm is said to "appear". This is demonstrated in Figure 2(c). By acquiring node 5, the frontier node 6 is found and added to CKG, making it the only node connected to nodes 1,2 and 5. Thus, node 6 is a sole member of a new SEC defined by connection to nodes 1,2 and 5 (marked with a red circle). Nodes 3 and 4 form their own SEC.

Split event- When a profile v is acquired and one or more of its new neighboring profiles are already represented in CKG, these profiles are no longer associated with their original SEC. The original SEC is said to "split" into two SECs. This is demonstrated in Figure 2(d) where node 5 is acquired, revealing that it is connected to node 4. Node 4 forms a new SEC (connected to 1,2 and 5) and node 3 stays in the original SEC (connected to 1 and 2).

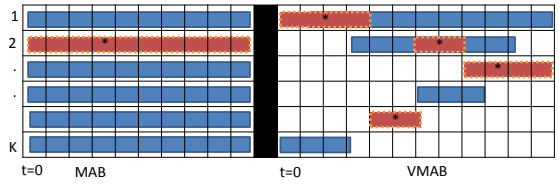


Fig. 3: Regular MAB (left) and Volatile MAB (right).

In order to deal with this dynamic behavior, we now introduce an extension of MAB called "Volatile Multi-Armed Bandits" (VMAB) where arms can "appear" or "disappear" unexpectedly. We introduce a policy for VMAB that achieves a bounded regret. We then use this policy in our algorithm. Unlike the "appear" and "disappear" events, the "split" event is not part of the definition of (VMAB). However, we describe how to deal with "split" events when we describe our SN querying algorithm in Section VIII.

VII. Volatile Multi-Armed Bandits

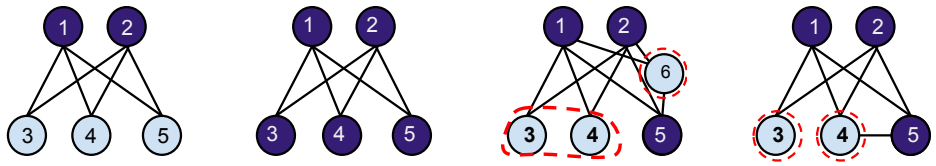
Standard MAB problems assume a constant set of K arms which are available indefinitely. We propose an extension of MAB, where new arms can "appear" or "disappear" on each turn. We call this MAB variant *Volatile-multi-Arm bandit problem* (VMAB). In VMAB, every arm a_i is associated with a *lifespan* (z_i, t_i) during which this arm is available. Figure 3 illustrates the difference between MAB and VMAB. The standard MAB (left) has a fixed set of K arms, one of them is optimal (colored red). In VMAB arms "appear" and "disappear" and the optimal arm may change over time (right). We assume that the arms' *lifespans* are unknown in advance.

There are many other cases where it is beneficial to extend MAB model to the model of VMAB, even for applications unrelated to SN queries. For example, consider an Internet advertisements matching where the task is to choose an advertisement that is most likely to be clicked by some user. On some cases, new advertisements might suddenly become available when new campaigns are approved. Alternatively, existing advertisements can become unavailable when the campaign budget runs out.

A policy for VMAB chooses on each turn n an arm a_i with a valid lifespan $z_i \leq n \leq t_i$. The expected regret of a VMAB policy (labeled R_n^v) is:

$$R_n^v = \sum_{t=1}^{n} \mu^*(t) - \mu(I(t))$$

Where $\mu^*(t)$ is the expected reward of the optimal arm at turn t and $I(t)$ is the arm selected at turn t. Obviously, the regret of VMAB depends on the number of available arms at each time step.For a set of K arms with mutual-exclusive lifespans the accumulated regret is 0 since there is only one arm to pull at each time. By contrast, if all arms have the same lifespans, the problem is identical to standard MAB.



(a) Initial (b) Disappear (c) Appear (d) Split

Fig. 2: Structural Equivalence Classes events.

A. VMAB Policy

For solving a VMAB problem, we propose the VUCB1 policy. VUCB1 selects the arm that maximizes Equation 3.

$$\overline{W_i} + \sqrt{\frac{2 \cdot \ln(n - z_i)}{T_i(n)}} \tag{3}$$

where z_i is the turn in which arm i first "appeared", n is the number of pulls performed and $T_i(n)$ is the number of times arm i had been pulled. When $T_i(n)$ is 0, the expression in Equation 3 is evaluated to positive infinity and thus, this new arm will be chosen. We define an *epoch* as an interval in which new arms are not created (arms can "disappear" in a given *epoch* but new arms cannot "appear").

Theorem 1: Let B be the number of *epochs* up to turn n. The accumulated expected *regret* R_n of VUCB1 policy is $O(B \cdot \ln(n))$.

We refer the reader to the journal version of this paper for the complete proof.

VIII. Bandit algorithm for SN Querying

We now turn to introduce our algorithm for SN querying denoted as SN-UCB1. SN-UCB1 models SN querying as a VMAB problem and use our suggested VUCB1 policy.

SN-UCB1 repeatedly performs profile acquisitions until the limit b is reached. Since SECs are dynamic, we use the VUCB1 policy and consider also the turn in which each arm first "appeared" (z_e). As explained above, to deal with non-stationary bandits we use the modified formula provided in Equation 2. Thus, we select the SEC which maximizes $\overline{\omega}(e) + C_p\sqrt{\frac{\ln(n-z_e)}{|acquisitions(e)|}}$, where $|acquisitions(e)|$ is the number of profile acquisitions of a specific SEC e). Next, one of the profiles of this SEC is randomly selected for acquisition. The information about this profile, such as the utility of acquiring it and its LOF are then added to CKG.

A. "Split" events

SECs are dynamic. Acquired profiles can migrate from one SEC to a different one when "splits" occur. Therefore, we re-calculate $\overline{\omega}(e)$ before each iteration according to current acquired profiles that belong to each SEC. On some cases, the "split" event creates a new SEC, for which the value of $\overline{\omega}$ is unknown since none of the profiles in the new SEC had been acquired. According to the VUCB1 policy, in these cases, a profile acquisition from this new SEC must immediately be performed. We would like to "reuse" information collected so far as an indication for the performance of this SEC. For this our algorithm initializes a "default" value and assigns it as the average utility for this SEC (with no prior acquisitions) instead of immediately performing acquisitions from it. Thus, the average utility of a new SEC e with no prior acquisitions is initialized according to the following formula:

$$\overline{\omega}(e) = P \cdot \overline{\omega}(e_1) + (1 - P) \cdot h(e)$$

where e_1 is a SEC from which e had "split", P is the proportion of e's frontier-leads in $e1$ and $h(e)$ is the heuristic value of e.

B. SN-UCB1 improvements

Arm tie-breaking: When a topological heuristic function exists, the heuristics values can be used to break ties among arms. It is also possible to break ties between frontier nodes in a given arm, with any other tie breaking rule.

Virtual acquisitions: When calculating the VUCB1 formula for each SEC, we add to the calculation M "virtual" acquisitions (M is an adjustable parameter). These acquisitions had not been performed in reality. We assign a default value for these acquisitions according to some heuristic function. Thus, new arms are formed with some "virtual-experience". This prevents frequent "exploration" acquisitions when the rate of appearance of new SECs is high. A similar idea was also used in [17] for solving the *Canadian Traveler Problem*.

IX. Experimental results

To demonstrate SN-UCB1 we performed experiments on two domains: the Target Oriented Network Intelligence Collection (TONIC) problem [1] on a Google+ dataset and finding research communities on a co-authorship network of computer science bibliography, constructed from DBLP [18]. In TONIC we are interested in retrieving profiles connected to a given profile of interest, denoted as the *target*. Due to privacy issues, the target SN profile is inaccessible. However, other SN profiles called *leads* contain information about the target and are accessible. TONIC tries to find as many leads as possible while minimizing the number of analyzed profiles. Several TONIC heuristics were proposed to decide which profile to acquire next in order to find more leads [1]. The best performing TONIC heuristic, called *BysP*, associates each lead with a *promising rate* (PR) and prioritized profiles by aggregating the PR values of the leads they are connected to. Our experiments for TONIC were performed on a data set obtained from the Google+ network and includes 211K profiles with 1.5M links between them. From this data set we randomly selected a set of 100 profiles having at least 30 friends. These profiles were used as the targets in our experiments. For each target we selected three random profiles as *seeds*.

We demonstrate the results of our experiments on the TONIC domain in Figure 4. The X axis describes the percentage of total acquisitions performed during the experiment.

The Y axis, describes the percentage of *leads* found from the total *leads* that exist in the network. The curves show the averages over all 100 targets. When more acquisitions are made more leads are found and in the end all nodes have been acquired and thus all leads were found. The challenge is to find as many leads as early as possible. Four curves are shown. Two benchmark polices are used. A *random* policy is used that randomly chooses an arm. We also used an *oracle* policy that chooses the arm with the actual maximal expected utility (*Optimal Arm* in Figure 4). We then implemented the state of the art topological heuristic for TONIC (BysP) and SN-UCB1. Both BysP and SN-UCB1 perform well and are close to optimal. Yet, SN-UCB1 policy slightly outperforms BysP even though BysP is a hard-tailored heuristic while SN-UCB1 is a general-purpose heuristic. Applying our algorithm on the DBLP domain produced similar results but are omitted here due to lack of space.

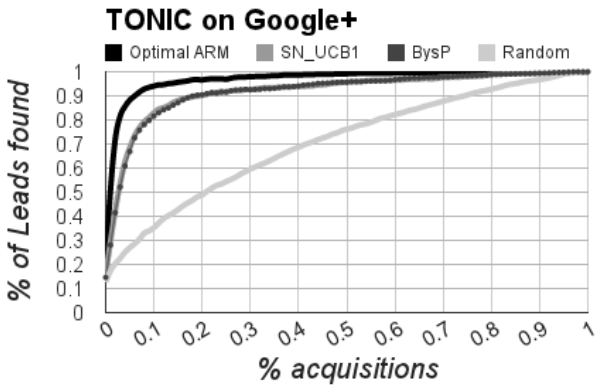


Fig. 4: The average recall obtained on Google+ TONIC

TABLE I: Average number of acquisitions required to reach specific level of recall.

Recall	DBLP		TONIC	
	BysP	SN-UCB1	BysP	SN-UCB1
0.75	1275	1200	900	900
0.80	2100	1883	1150	1150
0.85	3025	2725	1400	1375
0.90	3600	3425	2025	1900
0.95	5650	4518	3075	2825
0.96	5877	5500	3325	3175
0.97	6957	5582	3700	3575
0.98	7509	6561	4150	3975
0.99	8020	7464	4925	4600

Table I describes the average number of acquisitions required by each of the algorithms to reach specific level of recall on the DBLP and TONIC domains. On average, much less acquisitions are needed by SN-UCB1 to achieve the same level of recall compared to BysP on both domains.

X. Conclusion and summary

The proposed approach balances exploration and exploitation in the crawling process based on our new suggested variant of the Multi-Armed Bandit problem. Social network crawling was modeled as a VMAB by splitting the frontier nodes into disjoint sets of structurally equivalent nodes (SECs). Each SEC is regarded as an "arm" in our VMAB modeling. A corresponding policy, SN-UCB1, is used to choose a SEC while balancing the exploration vs. exploitation tradeoff in the most query agnostic way. Importantly, SN-UCB1 is not tailored to specific type of SN queries as other heuristics yet, it can be augmented with query specific heuristics using Virtual Acquisitions. Evaluation results show that the new policy has non negligible contribution over the state of the art saving up to 20% requests as shown in Table I. Moreover, experiments performed with an oracle that chooses the SEC having the highest expected utility demonstrate that there is much more space for improving state of the art crawling policies using our approach. In addition, unlike previous existing heuristics, SN-UCB1 has performance guarantees of being logarithmic in the accumulated regret.

References

[1] R. Stern, L. Samama, R. Puzis, A. Felner, Z. Bnaya, and T. Beja, "Target oriented network intelligence collection for the social web," in *AAAI*, 2013.

[2] M. Fire, R. Puzis, and Y. Elovici, "Organization mining using online social networks," *CoRR*, vol. abs/1303.3741, 2013.

[3] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu, "Link privacy in social networks," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 289–298.

[4] H. Robbins, "Some aspects of the sequential design of experiments," in *Herbert Robbins Selected Papers*. Springer, 1985, pp. 169–177.

[5] J. Tang, L. Yao, D. Zhang, and J. Zhang, "A combination approach to web user profiling," *ACM Trans. Knowl. Discov. Data*, vol. 5, no. 1, pp. 2:1–2:44, 2010.

[6] P. Pawlas, A. Domanski, and J. Domanska, "Universal web pages content parser," in *Computer Networks*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2012, vol. 291, pp. 130–138.

[7] R. Stern, M. Kalech, and A. Felner, "Finding patterns in an unknown graph," *AI Commun.*, vol. 25, no. 3, pp. 229–256, 2012.

[8] L. D. Sailer, "Structural equivalence: Meaning and definition, computation and application," *Social Networks*, vol. 1, no. 1, pp. 73 – 90, 1978.

[9] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2-3, pp. 235–256, May 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1013689704352

[10] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 784–791.

[11] Y. Yue and T. Joachims, "Interactively optimizing information retrieval systems as a dueling bandits problem," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1201–1208.

[12] S. Pandey and C. Olston, "Handling advertisements of unknown quality in search advertising," *Advances in Neural Information Processing Systems*, vol. 19, p. 1065, 2007.

[13] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, "A contextual-bandit algorithm for mobile context-aware recommender system," in *Neural Information Processing*. Springer, 2012, pp. 324–331.

[14] D. G. Horvitz and D. J. Thompson, "A generalization of sampling without replacement from a finite universe," *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 663–685, 1952.

[15] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 1–43, 2012.

[16] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," *Machine Learning: ECML 2006*, pp. 282–293, 2006.

[17] P. Eyerich, T. Keller, and M. Helmert, "High-quality policies for the canadian traveler's problem," in *Third Annual Symposium on Combinatorial Search*, 2010.

[18] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *CoRR*, vol. abs/1205.6233, 2012.