Blind Network Interdiction Strategies - A Learning Approach

SaiDhiraj Amuru, R. Michael Buehrer, and Mihaela van der Schaar

Abstract*—Network interdiction refers to disrupting a network in an attempt to either analyze the network's vulnerabilities or to undermine a network's communication capabilities. A vast majority of the works that have studied network interdiction assume *a priori* knowledge of the network topology. However, such knowledge may not be available in real-time settings. For instance, in practical electronic warfare-type settings, an attacker that intends to disrupt communication in the network may not know the topology *a priori*. Hence, it is necessary to develop online learning strategies that enable the attacker to interdict communication in the underlying network in real-time. In this paper, we develop several learning techniques that enable the attacker to learn the best network interdiction strategies (in terms of the best nodes to attack to maximally disrupt communication in the network) and also discuss the potential limitations that the attacker faces in such blind scenarios. We consider settings where a) only one node can be attacked and b) where multiple nodes can be attacked in the network. In addition to the single-attacker setting, we also discuss learning strategies when multiple attackers attack the network and discuss the limitations they face in real-time settings. Several different network topologies are considered in this study using which we show that under the blind settings considered in this paper, except for some simple network topologies, the attacker cannot optimally (measured in terms of the number of flows stopped) attack the network.

***Keywords*—Networks, graph, interdiction, attack, betweenness, centrality, blind, learning, bandits.**

I. Introduction

Network-centric architectures are increasingly gaining prominence, be it social networks or wireless networks, as they allow for decentralized operation among various nodes without the need for a central entity to control their communication. With the widespread deployment of such architectures, the security aspects of the underlying networks is now a major concern. Security-related studies not only enable us to understand the vulnerabilities of the underlying network architecture, but also shed light on how to best attack that network. The ability to undermine a malicious network's communication capabilities is crucial for ensuring security in sensitive environments. In this paper, we particularly focus on attacks against networks when their topology is unknown *a priori*.

Most studies that are related to attacking communicating nodes only consider the presence of a single node (source-destination pair) and develop optimal strategies, either at the physical layer [1]-[3] or the MAC layer [4] or the network layer (denial of service, or spoofing) [5], [6] in order to disrupt the communication capabilities of this node. Various formulations, ranging from optimization, game theory, information theory and machine learning, see [1]-[6], have been used to attack this node depending on the amount of knowledge that is available to the attacker. However, as mentioned earlier, with the rapid deployment of network centric architectures, it is now crucial to understand attacks against *networks*.

Network interdiction, as it is popularly known, has predominantly been studied by assuming that the network topology is known *a priori* [7]-[18]. Optimization-based network interdiction formulations were presented in [7]-[10] and game theoretic formulations were considered in [11], [12]. In the context of wireless networks, a flow-based formulation for jamming (the popular term used for attacks in the wireless communications literature) was discussed in [13] where an optimization problem was formulated to identify the best jammer-to-flow association that will maximally disrupt the network. In [14]-[18] and references therein, the behavior and robustness of various network topologies were studied by attacking different nodes or edges-based on several graph-theoretic metrics (i.e., the network is modeled as a graph). For instance, some of the commonly used graph-theoretic metrics are degree centrality, betweenness centrality, min-cut etc. [14]-[22]. However, these metrics can only be exploited when the attacker has *a priori* knowledge of the network topology. Further, attack strategies that are developed for one type of a network topology such as random Erdös-Rényi networks [21] may not always be capable of efficiently attacking other network topologies such as scale-free networks [22]. Hence, in order to blindly attack networks, online learning strategies that determine the best nodes to attack and disrupt communication in these networks are necessary.

Learning techniques have been explored for communication networks in the past. However, most of these works are concerned with learning the influential node in the network, see [23]-[27] and references therein. In contrast, in this paper, we are interested in learning the node(s) that is most important to attack in a network to minimize the number of messages being successfully exchanged. While reinforcement learning-based algorithms such as Q-learning are one strong possibility for such a problem, they do not provide any finite-time guarantees on the learning performance of the attacker [3]. Hence, in this paper, we develop several multi-armed bandit (MAB) algorithms that provide finite-time guarantees for the attackers performance when attacking a network blindly.

We assume that a) the attacker is aware of the total number of nodes in the network and b) is capable of identifying

SA and RMB are affiliated with Wireless@VT, Department of Electrical and Computer Engineering, Virginia Tech. MvdS is affiliated with the Department of Electrical and Computer Engineering, UCLA. Email: {adhiraj,rbuehrer}@vt.edu, mihaela@ee.ucla.edu.

the total number of successful and unsuccessful flows in this network by observing the network traffic. By flow, we refer to a message that is exchanged between different nodes in the network. For instance, in the case of communication networks, acknowledgement (ACK) and no-acknowledgement (NACK) packets that are exchanged between the various nodes in this network [4] indicate the total number of successful and unsuccessful message exchanges. Using such information, we present MAB algorithms with provable regret[1] guarantees that indicate the learning performance of the attacker in comparison with the omniscient (optimal) attacker (i.e., one that has complete knowledge about the flows and topology of the network).

The following are the main contributions of this paper—

1) We establish the connection between the metrics observable to the attacker and the well-known betweenness centrality metric [19] that is popularly used to attack networks when their topology is known *a priori* [14]-[18];
2) We develop MAB-based learning strategies applicable when (a) a single attacker attacks a single node, (b) a single attacker attacks multiple nodes and (c) multiple attackers attack multiple nodes in a network;
3) We provide insight into various network topologies ranging from those where the attacker can easily learn the optimal nodes to attack, such as star networks to those where the attacker has great difficulty in learning the optimal node to attack in a blind scenario, such as random networks;
4) We uncover and discuss the various limitations, in terms of the learning performance, that the attacker(s) will face when attacking networks blindly.

Note that such blind network attack strategies have not been studied before in the literature. We also do not claim optimal performance for the attacker (optimality in terms of the total number of flows stopped in the network), because, as we will discuss in detail in this paper, it is not possible for the attacker to guarantee optimality when the topology is unknown and especially when the attacker has only limited observables to provide feedback about its attack performance.

The rest of the paper is organized as follows. In Section II, we discuss the system model which is comprised of the victim network, the parameters of the attacker and the various observable metrics that are available to the attacker. Here, we also formally describe the network interdiction problem. In Section III, we present existing attack strategies for benchmarking and metrics that can be used to compare and analyze the performance of the attacker in blind settings (i.e., when the network topology is unknown). In Section IV, we discuss learning strategies and limitations when a single attacker intends to attack the single best node (in terms of flow minimization) in this network. The associated numerical results are presented in Section V for the cases where the flows are fixed for a certain period of time and when they change randomly. This is extended to consider cases where multiple nodes can be attacked, either by a single attacker or by multiple attackers, in Section VI. Finally, conclusions are presented in Section VII. Due to space limitations, the proofs of the Theorems, discussion on related work and some special network settings are relegated to the Appendices.

[1]Regret is defined as the difference between the cumulative reward of the optimal (for example, a strategy that minimizes the total throughput of the network) attack strategy when there is complete network topology knowledge, and the cumulative reward achieved by the proposed learning algorithm.

II. System Model and Problem Formulation

A. Victim Network Model

An undirected network is modeled as a connected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. The network topology is represented by the $|V| \times |V|$ adjacency matrix $\mathbf{A}$, where $|V|$ indicates the total number of nodes in the network G. The ijth entry i.e., $\mathbf{A}_{ij} = 1$ if nodes i and j are connected (i.e., an edge exists between nodes i and j) and is 0 otherwise. It is assumed that the network is undirected i.e., $\mathbf{A}_{ij} = 1$ implies that $\mathbf{A}_{ji} = 1$ (nodes i and j can communicate with each other). The analysis in this paper can be easily extended to directed networks and to networks with edge weights (say that correspond to data rate supported on an edge).

Any generic network topology can be analyzed using the framework in this paper. For ease of exposition, we focus on three different network topologies that are commonly used to depict networks and also indicate different levels of difficulty faced by the attacker in terms of learning the best nodes to attack— a) Erdös-Rényi (ER) random network [21] with the connection probability denoted by p, b) a Barabási-Albert (BA) network with a given degree parameter d [22] and c) a star network, with one root node and $|V| - 1$ leaf nodes. We also show the attacker performance against practical wireless networks that are increasingly being modeled according to a Poisson point process with a given number of nearest neighbor connections [31].

In the ER network, a node i connects to node j with probability p. Such a random network provides various insights about the attacker's learning performance as it poses a strict challenge to the attacker as opposed to most practical networks where an inherent structure exists. The BA model generates a scale-free network using the preferential-attachment model and is a widely used network model to describe social networks. In such networks, the number of connections depend on the popularity of the node and there is commonly one most popular node to which a large fraction of nodes are connected. Based on this description, it is clear that there is a preferred node that the attacker must attack in order to stop the flows in this network model. The star network is the simplest network model that can be attacked as there is clearly only one optimal choice i.e., the root node which disconnects the network when it is attacked and thereby stops all the flows. The attackers learning performance matches that of the optimal attacker in the star network once it identifies the root node. A PPP-based network model is one where the nodes are distributed according to a Poisson point process and each node connects to a given number of its nearest neighbors. Such a model is increasingly being used in present day wireless networks [31].

In addition to these networks, we give examples using some simple network topologies such as path, cycle etc wherever necessary. Further, in all these network models it is ensured that the network is connected. In other words, a path exists between every node i and any other node $j \neq i$.

B. Flow model

At an epoch t, any node can be a source node, i.e., it has information to send to some other node i.e., a destination node. Let S_t and D_t indicate the sets of source and destination nodes at epoch t where $|S_t| = |D_t| \leq |V|/2$. This assumption indicates the fact that a node in the network can act either only as a source or only as a destination at any time instant.[2] It is also assumed that the nodes in G are aware of the shortest path between any pair of nodes i and j over which messages are exchanged. For convenience, it is assumed that the source and destination nodes communicate only along the shortest path between them (cases where nodes use paths other than the shortest path can be considered by using the analysis in this paper in a straightforward manner by changing the description of the metrics considered). A *flow* $f_t^{s,d} \in \{0,1\}$ constitutes a source $s \in S_t$, a destination $d \in D_t$ and the set of nodes that lie along the shortest path between s and d. Let $F_G^t = \sum_{s \neq d} f_t^{s,d}$ indicate the total number of flows in the network G at time t. Note that $|S_t| = |D_t| = F_G^t$.

The epoch t constitutes several time slots and is taken to be much greater than the length of the time required to traverse the maximum shortest path between the source nodes S_t and their corresponding destination nodes D_t. If more than one flow passes through a common node in the network (i.e., the shortest paths between two different source-destination node pairs pass through a common node), it is assumed that this common node delivers information for these destination nodes one after the other, i.e., scheduling takes care of such issues (which is commonly the case). For ease of exposition, these fine temporal aspects are not considered in this analysis. Hence, in this paper it is assumed that every flow (starting from the source node and ending at the destination node) in the network is completed within one time epoch and we are not concerned about the time slots required to finish these flows. This does not take away any merits of the proposed algorithm, as we do this only for simplifying the analysis in this paper. For an example on how temporal aspects of the network can be included, please see [4].

C. Attack Model

By attack, we refer to the following– in the case of communication networks, an attack refers to corrupting the information that is received at a node which thereby cannot be decoded into any valid information; in the case of social or economic networks, an attack either refers to physically removing a network connection or a node or passing wrong information to an individual that is part of these networks and thereby causing mis-communication and a breach of privacy. For more details regarding types of attacks on social networks, please see [29] and references therein.

The attacker can attack k nodes in the network during any epoch to stop flows in G. When a node is attacked, all the edges incident (both incoming and outgoing) upon this node are blocked indicating that all flows that need this node are stopped. Let $F_{G\setminus A_k}^t$ indicate the total number of remaining flows in the network after k nodes of G (denoted by the set A_k) are attacked. Therefore the attacker wishes to solve the following optimization problem for epoch t

$$\max_{A_k} \; F_G^t - F_{G\setminus A_k}^t$$
$$\text{s.t. } \mathbf{c}\mathbf{x}^t \leq k \text{ and } \{x_i\}_{i=1}^{|V|} \in \{0,1\}, \qquad (1)$$

where $\mathbf{c}$ is a $1 \times |V|$ vector of all 1's and $\mathbf{x}^t$ is a $|V| \times 1$ vector which has entries equal to either 0 or 1 depending on whether a node is attacked or not at time t. Notice that $\max_{A_k} \; F_G^t - F_{G\setminus A_k}^t$ is equivalent to $\min_{A_k} \; F_{G\setminus A_k}^t$, but we use the formulation (1) to explicitly denote the fact that the attacker is aware of F_G^t i.e., the total number of flows in the network. The set A_k^t is dependent on the set of non-zero elements in $\mathbf{x}^t$. The first constraint in (1) indicates that the attacker can attack a maximum of k nodes. By varying the definition of $\mathbf{c}$, this constraint can also be used to represent the cost c_i incurred by attacking the ith node and that the attacker is limited by a total cost constraint/budget B where the condition changes to $\mathbf{c}\mathbf{x}^t \leq B$. In this paper, we assume that the cost of attacking any node in the network is 1 and the budget for the attacker only indicates the total number of nodes that it can attack at any given time.

Remark 1. *The binary integer linear optimization problem in* (1) *is NP-complete [7]. However, the network topology is necessary in order to solve* (1) *i.e., to optimally attack the network.*

Remark 2. *The degree of a node i (i.e., $\sum_{j \neq i} \mathbf{A}_{ij}$) that can be attacked must be at least 2 in order to disrupt more than one flow in the network. Otherwise, the node is a leaf node and attacking it can only stop one flow in the network at a maximum.*

Remark 3. *The optimal nodes to attack in a network are more dependent on the network structure and the flows occurring in the network than the size of the network.*

For example, in the case of a star network and a cycle network, both with the same number of nodes, the optimal node to attack is different. In the case of the star network, attacking the root node stops all the flows in the network, where as in the cycle network the optimal choice depends on the flows in the network at a given time t.

As another example, consider a tree network (tree is a network without cycles). For the ease of analysis assume $|V|$ to be odd. Then $(|V|-1)/2$ nodes are on either side of the root node of the tree. When there are $(|V|-1)/2$ total flows in the

[2]With the advent of full-duplex technology, nodes can act both as a source and a destination simultaneously. Under such cases, both the incoming and outgoing flows can both be stopped when a node is attacked. Irrespective of whether the nodes are full-duplex or half-duplex, the learning algorithms (for the attacker) proposed in this paper can still be used as they only need to know the total number of flows stopped when a node is attacked and not the exact number of flows it transmits or receives.

network, then with probability $1 - \frac{\binom{\frac{|V|-1}{2}}{\frac{|V|-1}{4}}}{\binom{|V|-1}{\frac{|V|-1}{2}}}$ *attacking the root node is the optimal strategy when a tree network is attacked. Here,* $\binom{a}{b}$ *indicates the binomial coefficient* $\frac{a!}{b!(a-b)!}$*.*

Hence, when the network topology and the flows in the network are known, optimal attacks are possible and also enable us to bound the attacker's performance.

Remark 4. *The worst case complexity of finding a node that stops the maximum number of flows in the network is* $O(|V|^3)$*. This is also the computational complexity involved in estimating the various network centrality metrics [19]. More details about the attack complexity will be discussed in Sections III, IV.*

In the analysis that follows, we use the solution for the optimization problem (1) as a bound with which we compare the various network attack strategies. We first discuss the benchmark attack strategies (that are not blind) that are commonly used to address network interdiction and then discuss the learning strategies that can be used under blind scenarios.

III. Single-Node Attack – Benchmark Strategies and Performance Analysis

We first intend to find the *single best node* that can be attacked in order to disrupt the flows in the network. In this paper, we only focus on finding the node that enables the attacker to stop the maximum number of flows and the specific details regarding the attack procedures (either at the physical layer or MAC layer) can be found for example in [3], [4]. Similar details about attacks in social networks can be found in [29]. We first focus on the case where flows (the number as well as the source-destination pairs) are fixed over the time period of interest T and later we will consider cases where the flows change randomly.

In this section, we discuss a) existing attack strategies (which are not blind) that are used for benchmarking the attacker performance and b) performance metrics that enable for network interdiction performance comparison and to compare the gap in the learning performance of the attacker under blind scenarios.

A. Omniscient attacker

This is a know-all attack strategy where the attacker is exactly aware of the network topology and the flows in the network at any time t. Based on this knowledge, the attacker attacks a single node through which the maximum number of these flows are passing. In other words, the omniscient attacker solves the optimization problem in (1). This strategy therefore acts as the upper bound for comparison of the various network interdiction strategies proposed in this paper.

B. Node Betweenness-based attack

Betweenness centrality is a popularly used network centrality metric to develop network attack strategies [14]-[18]. Formally, betweenness centrality measures the fraction of the shortest paths passing through a node relative to the total number of shortest paths in the network. In other words, betweenness centrality is a measure to quantify the number of times a node acts as a bridge between two different nodes when they communicate along their shortest path. Mathematically, the metric for the node v is given by

$$C_B(v) = \frac{1}{(|V|-1)(|V|-2)} \sum_{s \neq v \neq d \in V} \frac{\sigma_{sd}(v)}{\sigma_{sd}}, \qquad (2)$$

where σ_{sd} indicates the total number of shortest paths between source node s and destination node d and $\sigma_{sd}(v)$ indicates the number of these shortest paths between s and d that pass through v. By $s \neq v \neq d$ we refer to the following $s \neq v$, $s \neq d$, and $v \neq d$. The scaling factor indicates the total number of possible shortest paths that can include v [19]. Notice that, with high probability, the node with the highest betweenness centrality metric supports a large number of flows that may arise in the network. The relation between this metric and the attacker performance is studied in detail in Section III-C.

When the attacker is aware of the network topology, it can pre-evaluate this metric for all nodes in the network and thereby attack the node with the maximum value. In other words, the node attacked in this strategy is given by

$$A_1^t = \arg\max_{v \in V} C_B(v) \;\forall t, \qquad (3)$$

where A_1^t indicates the single best node attacked by using this strategy at epoch t. For example, in the case of a star network, $C_B(v) = 1$ for the root node and is 0 for all other nodes and hence the root node is attacked by using this strategy, which is the optimal strategy for the star network. This is not always the case for other networks as will be discussed in detail shortly.

The betweenness metric-based attacks have been shown to be more effective than attacks-based on other centrality metrics that can be evaluated for any given network [14]-[18]. To consider networks with edge weights and/or different costs for attacking each node, weighted betweenness centrality metrics can be used [30]. However, to keep the analysis simple we consider un-weighted networks in this paper. The complexity in estimating the betweenness centrality metric is $O(|V||E|)$ which is approximately $O(|V|^3)$ [19]. Next, we will use this to compare the learning performance of the attacker.

C. Notes on attacker performance

Define $\bar{C}_B(G)$ as the average betweenness metric for the network G given by

$$\bar{C}_B(G) = \frac{1}{|V|} \sum_{v \in V} C_B(v). \qquad (4)$$

This indicates the average number of shortest paths through each node in the network. $\bar{C}_B(G)$ can be related to the average shortest path length in the network ($\bar{l}(G)$) as [42]

$$\bar{C}_B(G) = (|V|-1)(\bar{l}(G)-1) \qquad (5)$$

where $\bar{l}(G) = \frac{\sum_{u,v \in V} d(u,v)}{|V|(|V|-1)}$ and $d(u,v)$ indicates the length of the shortest path between nodes u and v.

At every time epoch, a new subgraph G' can be formed by only using the shortest paths corresponding to the flows

$\{f_t^{s,d}\}_{s\neq d\in V}$ i.e., G' has $|V|$ nodes, but only has the edges of G that are necessary for the flows $\{f_t^{s,d}\}_{s\neq d\in V}$. Notice that the average shortest path length of this new subgraph G' i.e., $\bar{l}(G') \geq \bar{l}(G)$, because the inactive edges (edges that are not used in the communication of $\{f_t^{s,d}\}_{s\neq d\in V}$) are removed from G. More specifically, when edges are removed from G, some nodes in G' *may be* disconnected from the rest of the network and hence the shortest path length between these disconnected nodes is ∞. Notice that not all networks get disconnected when some edges are removed and hence $\bar{l}(G') \in [\bar{l}(G), \infty]$. This is explained below with an example.

Example: In a fully-connected network G, every node is connected to every other node. Hence, the shortest path length is 1 *between any pair of nodes and therefore* $\bar{l}(G) = 1$. *If one edge is removed in G to form* G', *then the shortest path length between the pair of nodes connected to this edge now increases to* 2 *while the remaining shortest path lengths continue to be* 1. *Therefore* $\bar{l}(G')$ *is given by*

$$\bar{l}(G') = \frac{1}{|V|(|V|-1)}\left(\Big\{|V|(|V|-1)-1\Big\}\times 1 + 1\times 2\right)$$
$$= 1 + \frac{1}{|V|(|V|-1)}, \tag{6}$$

which indicates that $\bar{l}(G')$ *increases by a factor of* $\frac{1}{|V|(|V|-1)}$ *when compared to* $\bar{l}(G)$.

Since the average betweenness metric of a graph is related to the shortest path length as shown in (5), we now have that $\bar{C}_B(G') \geq \bar{C}_B(G)$, because $\bar{l}(G') \geq \bar{l}(G)$. Recall, that the average number of flows stopped by attacking node v is proportional to $C_B(v)$. Therefore, the total number of flows that will be stopped on average in G is proportional to $\bar{C}_B(G)F_G^t$ and to $\bar{C}_B(G')F_G^t$ in the subgraph G'. Therefore, it is clear that the total number of flows that can be stopped by the attacker in G' is higher than the ones that can be stopped in G when using the betweenness-based attack strategy. This is exactly the procedure followed by the omniscient attacker in (1) because it is completely aware of the network topology and hence can attack the best node in G' (which is not necessarily the best node to attack when G is considered because of different $C_B(v)$) such that the maximum number of flows are stopped. Lemma 1, which will be presented shortly, will formalize this argument and discuss the connection between the optimal attack strategy and the betweenness metric of the nodes in the network.

Remark 5. *In a fully connected network* $\bar{C}_B(G) = 0$ *because the shortest path between any two nodes is the edge connecting them. Hence, when the attacker attacks node v, except for the flow that is intended for node v, none of the other flows are stopped in the network.*

Example: For the case of ER-networks, it is shown in [43], [44] that $\bar{l}(G) \approx \frac{\log|V|}{\log(p|V|)}$, *where p is the connection probability of the ER network. When* $p = 1$, *this value is equal to* 1 *as expected. Similarly, it can be shown that when* ϵ_f *fraction of the edges are removed in a ER network G to form* G' *(where* ϵ_f *is a function of the total number of flows* F_G^t *and G), then we have* $\bar{l}(G') \approx \left(\frac{\log|V|}{\log(p(1-\epsilon_f)|V|)}\right)$ *which indicates that the average number of flows stopped by using the knowledge of* G' *is greater than the ones that can be stopped by just knowing G.*

We mentioned earlier that the number of flows stopped when a node is attacked is approximately proportional to the betweenness metric of that node in G. More specifically, it can be shown that the probability that a flow is stopped when node v is attacked is given by

$$p_v = \frac{\sum_{s\neq d\neq v}\sigma_{sd}(v) + 2(|V|-1)}{|V|(|V|-1)}, \tag{7}$$

where $\sum_{s\neq d\neq v}\sigma_{sd}(v)$ indicates the total number of shortest paths (that originate at source s and end at destination node d) that pass through node v. The second term in the numerator accounts for all those flows that are stopped when node v is the source or destination. The denominator indicates that the total number of paths (from any node i to any other node j) in the network is $|V|(|V|-1)$. See that the betweenness measure defined in (2) is also dependent on $\sum_{s\neq d\neq v}\sigma_{sd}(v)$. When $\sigma_{sd} = 1$ for any $s, d \in V$ i.e., there is only one shortest path between s and d nodes, then the betweenness measure of the node is directly proportional to p_v. For example, any tree network satisfies this relationship.

Furthermore, via simulations we observed that p_v and $C_B(v)$ are very similar and this is what we referred to as the reward of the attacker being proportional to the betweenness metric. To obtain an exact relationship, we can rewrite (2) as $C_B(v) = \frac{\sum_{s\neq v\neq d\in V}\sigma_{sd}(v)}{(|V|-1)(|V|-2)}$ in which case we have $p_v = \frac{(|V-2|)C_B(v)+2}{|V|}$.

Remark 6. *Based on the discussion above, it is clear that the attack performance against any network will be lower-bounded by the performance against a fully connected network where only one flow can be stopped by attacking a single node. Furthermore, the attack performance will be upper-bounded by the performance against a star network where all the flows will be stopped by attacking the root node.*

Lemma 1. *Given a network G with flows* $\{f_t^{s,d}\}_{s\neq d\in V}$, *the optimal node to attack i.e., the solution to* (1) *is the node with the maximum betweenness metric in* G'.

Proof: The proof follows by using the definitions of the optimal attack strategy in (1), the betweenness metric defined in (2) and the definition of the network G'. Specifically, since G' only consists of the nodes and edges that are necessary for $\{f_t^{s,d}\}_{s\neq d\in V}$, the optimal node to attack is the node through which maximum number of flows pass; which is nothing but the node with the maximum betweenness metric over this sub-network G'.

This lemma will later be used to discuss the performance of the attacker under blind settings. Furthermore, the attack strategies and the metrics presented in this section will enable us to understand the differences (gap) in the attacker performance and the limitations faced in blind settings when compared to its performance with *a priori* network topology knowledge.

IV. Single-Node Attack – Blind Strategies and Performance Analysis

In this section we discuss network interdiction strategies that the attacker can use under blind settings i.e., when the network topology is unknown *a priori*. These attack strategies will be compared with the benchmark non-blind attack strategies discussed in the previous section.

A. Random attacks

In this strategy, the attacker randomly attacks a node during every epoch. As expected this strategy performs worse than the other strategies discussed in this paper. This is the best it can do in completely blind settings when there is no feedback regarding its attack performance and serves as a lower bound to compare the various network attack strategies.

B. Learning strategies against fixed flow scenarios

We present multi-armed-bandit-based learning strategies where the attacker can learn the best node to attack in a real-time manner without knowledge of the network topology. For reasons that will be explained soon, except for some specific network topologies (such as tree, star), these learning algorithms cannot achieve performance close to the omniscient attack. It turns out that this is an inherent limitation in blind settings i.e., when the network topology is unknown.

All learning algorithms are presented for the case when the network flows remain fixed for T epochs. Later, we discuss the performance of these algorithms when the flows change randomly at every epoch. For all these learning algorithms, the feedback for the attacker is the fraction of the total number of flows stopped at any time instant i.e., $\frac{F_G^t - F_{G\setminus A_k}^t}{F_G^t}$, where the set A_k^t indicates the nodes attacked at epoch t.

1) Notes on feedback used by the attacker for learning: Remember that the attacker is a) aware of the total number of nodes ($|V|$) in the network, and b) can infer the total number of successful and unsuccessful flows in the network. For instance, in the case of communication networks (such as the ones based on the TCP protocol [32]), each destination node sends an acknowledgement (ACK) or a no-acknowledgement (NACK) packet directly to the source node. Since the ACK/NACK packets are not encrypted, they can be easily identified in any open protocol such as WiFi [4].[3] These packets enable the attacker to identify the throughput allowed in the network [3], [4]. In the case of economic networks, the total number of successful and failed transactions can be used to evaluate the performance of the attack strategy. In the case of social networks, the number of lost connections can be used as feedback by the attacker. Specifically, in the case of communication networks, when a flow is attacked no ACK/NACK packet is sent from the destination node because the flow never reached the destination node (the same is assumed to be the case even when the destination node is attacked). Hence, the number of flows attacked can be identified by counting the total number of flows and the ACK/NACK packets exchanged in the network.

[3]Alternate metrics such as power levels, number of re-transmissions etc. could also be used by the attacker as feedback for its actions.

Remark 7. *The assumption that the number of nodes is known is realistic because nodes can be easily detected whenever they transmit data (assuming they are within the range of the attacker) [33], [34]. However, if the attacker doesn't know the correct number of nodes, that means that either (a) the attacker assumes that there are nodes that don't actually exist (unlikely) or (b) the attacker is unaware of certain nodes that actually exist in the network (more likely). In the latter case, the attacker would not attack the unknown nodes since their existence (for example, say due to the location of the attacker in the network) is not known. When this unknown node is not the optimal node to attack, the attacker learns faster since the attacker does not spend resources by attacking a non-optimal node. However, if the missing node is in fact the optimal one to attack, then the attacker will never learn the optimal attack choice and the loss in performance will depend on how much difference there is between the optimal node and the best node that the attacker is aware of.*

Furthermore, when a single attacker is unable to listen to the feedback from all nodes in the network/ unable to attack all nodes in the network, then multiple attackers, that coordinate and share information to attack the network, will be necessary. The case of multiple attackers attacking a network is studied in Section VI.

2) Notation: Let $A_1^t = v_t$ indicate the node attacked during epoch t and $r(v_t) = \frac{\max(\min(F_G^t - F_{G\setminus v_t}^t + \epsilon, F_G^t), 0)}{F_G^t}$ indicate the reward (fraction of flows stopped) obtained by the attacker when this node is attacked. ϵ is a uniform random variable in the set $\left\{ \left\lfloor \frac{-F_G^t}{2} \right\rfloor, \ldots, \left\lfloor \frac{F_G^t}{2} \right\rfloor \right\}$ that indicates the uncertainty in the feedback received (for instance errors in the estimation of the ACK/ NACK packets in a communication setting).[4] Along similar lines, $r(v^*)$ indicates the reward obtained by node v^* which is the best node in hindsight that should have been attacked to maximize the reward. Then we define the cumulative regret incurred by the attacker as

$$R(T) = \mathbf{E}\left[\sum_{t=1}^{T} \left[r(v^*) - r(v_t)\right]\right] = Tr(v^*) - \mathbf{E}\left[\sum_{t=1}^{T} r(v_t)\right], \tag{8}$$

which indicates the expected difference in the total reward between the strategies chosen by the proposed algorithm and the optimal choice in hindsight. In the above equation, the expectation is over all the possible nodes that may be attacked when the attacker uses a learning algorithm. The goal of the attacker is to minimize this regret so as to learn the best single node to attack.

[4]In cases where the attacker is unable to listen to the ACKs/NACKs from all nodes in the network (for example, due to its location in the network), then different distributions such as impulsive/ non-Gaussian can be used to model ϵ and capture this unknown feedback. This is because non-Gaussian distributions can be used to capture rare events. Also see that the $\max$ and $\min$ constraints in the reward definitions will ensure that the reward/feedback used by the attacker in its learning algorithm is still meaningful.

Let $\Delta_{\min}$ indicate the minimum gap in the reward obtained when v^* is attacked and when the second best node is attacked. Because the topology is unknown *a priori*, the attacker can only obtain bounds on $\Delta_{\min}$. For instance, we have that

$$\Delta_{\min} \geq \frac{1}{F_G^t} \geq \frac{1}{\frac{|V|}{2}}, \tag{9}$$

which indicates the fact that by attacking the second best node, the attacker can potentially stop at most $F_G^t - 1$ flows, whereas F_G^t flows could potentially be stopped by attacking the optimal node v^*. The second inequality is obtained by noting the fact the maximum number of possible flows in this network is $\frac{|V|}{2}$. Along similar lines let $\Delta_{\max}$ indicate the maximum gap in the reward obtained by the attacker. It is easy to see that $\Delta_{\max} \leq 1$. $\Delta_{\min}$ and $\Delta_{\max}$ will be used to provide regret guarantees for the attacker's performance.

Different algorithms have been proposed in the literature to address the regret minimization problem [35]-[39]. Below, we first present two popular bandit-based learning algorithms, namely upper confidence bound (UCB) learning [35] and contextual zooming, a version of the contextual bandit algorithm [39]. Later, we present a novel *slotted exploration exploitation*-based MAB algorithm which performs better than these existing learning algorithms. As will be discussed in Section VI, this algorithm can also be extended to consider time-varying and multiple-attacker settings.

1) UCB-based Learning [35]: This is the most common MAB algorithm used in online learning settings. It maintains mean average reward values obtained by attacking a particular node v at each epoch and thereafter evaluates indices known as *ucb-indices* in order decide the node to attack in the next epoch [35]. The regret of this algorithm grows as $O\left(\frac{|V|\log(T)}{\Delta_{\min}^2}\right)$ which is linear in $|V|$ and is inversely proportional to $\Delta_{\min}$. If $\Delta_{\min}$ is large, then it is easy to identify the best node to attack.

Furthermore, as will be explained shortly in Theorem 2, when the network topology is unknown, it is not possible to achieve a learning rate better than $O(|V|)$ (computational complexity as a function of the network parameters) indicating the fact that all the nodes in the network must be attacked at least once in order to learn the best node to attack in a real-time setting.

2) Contextual Bandit-based learning [39]: A context refers to side information that is available to the learning agent (attacker in this case) [39]. More often than not, assigning a context to the learned information enables the agent to achieve better results than acting blindly and ignoring the side information. This side information can be exploited using contextual bandit algorithms [39], where the best action given the context can be learned online. These methods utilize a context-dependent history of past observations in order to estimate its context-dependent performance [39].

For this algorithm, we assume that the attacker is also aware of the source node and the destination node for all the flows in the network at epoch t, in addition to all the variables that can be accessed by the UCB algorithm (note that a flow is defined not only by the source and the destination nodes, but also all the nodes that lie on the shortest path between them, hence in this learning algorithm, we only assume partial information is available to the attacker). Therefore, the source-destination pairs are taken to be the context information. The regret in this case is defined as

$$R(T) = \mathbf{E}\left[\sum_{t=1}^{T}[r(\mathbf{s}(t), \mathbf{d}(t), v^*) - r(\mathbf{s}(t), \mathbf{d}(t), v_t)]\right], \tag{10}$$

where $\mathbf{s}(t), \mathbf{d}(t)$ indicate the $|V| \times 1$ vectors of 1's and 0's that indicate the active sources and destinations at every time instant. Since, the dimension of the context space is $2|V|$ (as any node can be a source and any other node can be a destination), the overall regret for this learning algorithm grows sub-linearly in time as $O(T^{\frac{2|V|+1}{2|V|+2}})$ [39].

3) Slotted Exploration-Exploitation Algorithm: Here, we propose a novel algorithm which performs exploration and exploitation in a slotted manner. The proposed slotted exploration-exploitation MAB algorithm is shown in Alg. 1. In this algorithm, during the exploration phases, each node in the network is attacked once every epoch and during the exploitation phase, it chooses to attack the node with the best average reward obtained during the exploration phases. The average reward values are only estimated during the exploration phases and the attacker sticks to the node with the best average reward during the exploitation phase. While the length of the exploration phase remains constant (here it is equal to the number of nodes in the network), the length of the exploitation phase increases exponentially after every exploration phase. Here, $\gamma(t)$ and $E(t)$ are two counters that keep track of these exploration and exploitation phases. The counter $\gamma(t)$ indicates the total number of exploration phases until time t. The transition between the exploration and exploitation phases will be determined by the condition $\gamma(t) < A\log(t)$ which also controls the exponential increase in the length of the exploitation phases. Specifically, when the counter $E(t) = 0$, the current slot will be an exploration slot if $\gamma(t) < A\log(t)$, else it will be an exploitation slot. However when $E(t) > 0$, the current slot is an exploration slot and the value of $E(t)$ indicates the arm (or the node) being explored (attacked in the current context) in the current exploration slot.

Theorem 1. *The regret bound for Alg. 1 is*

$$R(T) \leq A|V|\Delta_{\max}\log(T) + |V|\Delta_{\max} + \sum_{t}^{\infty} 2|V|\Delta_{\max} t^{-\frac{A}{2}\Delta_{\min}^2}, \tag{11}$$

where $A > \frac{2}{\Delta_{\min}^2}$.

Proof: See Appendix I in the extended version of this paper [28].[5]

Remark 8. *In an online setting, the exact values of the* $\Delta_{\max}$ *and* $\Delta_{\min}$ *may not be known perfectly. By using bounds on these values (presented earlier), the attacker can evaluate the regret bounds incurred during this learning process. By*

[5]A generic slotted exploration-exploitation algorithm with adjustable exploration-exploitation schedule and the proof for its associated regret bound is in Appendix I in the extended version of this paper [28].

Algorithm 1 Slotted Explore-Exploit

```
    Initialization: t← 1, $E(t) = 0$, $\gamma(t) = 0$
 1: while $t \leq T$ do
 2:     if $E(t) > 0$ then Explore
 3:     else if $\gamma(t) < A\log(t)$ then Explore
 4:     else Exploit
 5:     end if
 6: end while

    Explore
 7: $E(t+1) = \mod (E(t)+1, |V|+1)$
 8: if $E(t+1) = 0$ then $\gamma(t+1) = \gamma(t)+1$
 9: else $t = t+1$ and choose arm $v_t = E(t+1)$
10: Get the feedback and update average reward $r(v_t)$ of the chosen arm
11: end if

    Exploit
12: Choose the arm $v_t^*$ with maximum average reward $v_t^* = \arg\max_{v_t} r(v_t)$
13: $t = t+1$
```

setting $A = \frac{4}{\Delta_{\min}^2}$, the second and third terms in (11) *i.e., $|V|\Delta_{\max} + \sum_t^{\infty} 2|V|\Delta_{\max} t^{-\frac{A}{2}\Delta_{\min}^2}$ are just constants proportional to $|V|$. This explicitly shows that the regret is logarithmically dependent on T and linearly dependent on $|V|$. This linear dependency on $|V|$ will be revisited in Theorem 2.*

Remark 9. *When the network is skewed, i.e., there is a small set of popular nodes which are part of several flows, then it is easy to identify the best node that the must be attacked because $\Delta_{\min}$ is larger. Thereby, in these cases the regret is smaller. For example, this behavior will be shown in the case of BA networks and star networks in Section V.*

C. Random flow scenarios

All the bandit algorithms proposed above are valid only when the flows are fixed over a given period of time during which the attacker learns the best node to attack. These algorithms cannot accommodate for the time varying nature of the "arms" (which in the current case are the nodes attacked) involved in the learning process [35], [40]. Here we consider cases where the number of flows, as well as the source-destination pairs randomly change at every time instant. When the flows are changing, using the rewards from earlier epochs in order to estimate the expected rewards in future epochs will be inaccurate. Hence, sliding window-type techniques [3] need to be used where the attacker only uses the recently learned reward estimates to attack a node. For time-varying regret bounds using the UCB algorithm see [41] and for the case of the contextual bandit algorithm see [40]. Below, we present a time-varying regret bound for Alg. 1. For generic regret bounds for any learning algorithm, please see [28].

1) ***Time varying regret bound for the slotted explore-exploit algorithm:*** See that in the regret bound of the slotted explore-exploit algorithm in (11), we have assumed that the number of flows is fixed over a given time and that the attacker learns the best node to attack in this network. However, the analysis of the logarithmic regret presented earlier is no longer valid for the case when the flows change at every time instant. Hence, we need to consider the notion of time-averaged regret denoted by

$$R^{ave}(T) = \frac{1}{T}\left(\sum_{t=1}^{T} r(v_t^*) - \mathbf{E}(r(v_t))\right), \tag{12}$$

which indicates the difference in the rewards obtained by attacking v_t^* (optimal node to attack at time t) and the node v_t that is attacked at time t. Moreover, $\Delta_{\max}$ and $\Delta_{\min}$ will also be time varying and can be denoted by $\Delta_{\max}(t)$ and $\Delta_{\min}(t)$ respectively. Let $\bar{\Delta}_{\max}$ and $\underline{\Delta}_{\min}$ be the upper and lower bounds on these values respectively.

The slotted explore-exploit algorithm in Alg. 1 can be used in such time varying scenarios by making one modification– the exploitation time duration is no longer exponential. It is now a constant which is independent of time i.e., instead of checking for the condition $A\log(t)$ in Alg. 1, the Alg. 1 now performs alternate exploration and exploitation schedules where the exploration schedule is of length $|V|$ and the exploitation schedule is also a constant A. This constant A is chosen based on the knowledge of the time window W over which the number of flows remain constant. For instance, an obvious choice is that $A < W$. Because of these changes, notice that the regret bound is $O(|V|T)$ (linear in time), as the attacker must track the changes in the reward by continuously attacking various nodes at a constant rate.

Remark 10. *If additionally, the time varying reward satisfies $|r(v_w^*) - r(v_{w-1}^*)| \leq \epsilon$, where $r(v_w^*)$ indicates the optimal reward obtained during the wth time window and $r(v_{w-1}^*)$ indicates the optimal reward during the $w-1$th window, each of duration W time slots with $\epsilon > 0$ being a positive constant that is known to the attacker, then regret bounds that are still linear in time, but with a smaller leading constant can be achieved. For more details on this see [40].*

D. Learning rates in blind scenarios

The time aspect of the attacker's learning performance is captured by the regret bounds of the various learning algorithms discussed earlier. Here, we are interested in the scaling factor for the regret bound which depends on the network under consideration i.e., we are specifically interested in the dependency of the learning rates on $|V|$.

Theorem 2. *When the network topology is unknown, it is not possible for the attacker to achieve a learning rate better than $O(|V|)$.*

Proof: Since it is now known from Lemma 1 that the optimal attack choice depends on the betweenness metric of the network and that the reward obtained by attacking a node is proportional to its betweenness metric, we use the betweenness metric to prove Theorem 1.

We first prove that the attacker cannot learn in better than $O(|V|)$ even when the flows are fixed over a given time. We prove this by contradiction. Assume that it is indeed possible to learn the best node to attack by only attacking $|V| - 1$

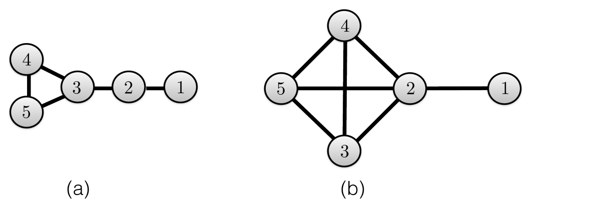

Fig. 1. Betweenness metrics for nodes in network (a) $\frac{1}{12}[0,6,8,0,0]$ and network (b) $\frac{1}{12}[0,6,0,0,0]$.

nodes. If the learning procedure can be achieved in better than $O(|V|)$, for example say $O(|V|-1)$, then it suggests that the betweenness metric of the $|V|-1$ nodes should tell us about the betweenness metric of the $|V|$th node. We show below that this is not true.

Consider a network of 5 nodes where the attacker has estimated the betweenness metric of four nodes as $\frac{1}{12}[0,6,0,0]$. Now if the attacker were capable of inferring the network topology by only using these four nodes, then it should be able to identify the betweenness metric of the remaining node. However, this is not the case. To see this, consider the two connected networks (each has 5 nodes) shown in Fig. 1. In both these networks, the betweenness metric for nodes $1,2,4,5$ is $\frac{1}{12}[0,6,0,0]$. However, the betweenness metric for node 3 is 8 in the case of the first network and 0 in the case of the second network. Thus in the case of the first network, if the attacker did not wait to learn the metric of node 3, then it would have attacked a sub-optimal node whereas in the case of the second network the attacker could have ignored learning the metric for node 3 as its value is less than the metric for the 2nd node which was already learned to be $6/12$. However, since by assumption the network topology is not known, the attacker requires all the betweenness metrics to determine the optimal node to attack.

For the case where flows change rapidly, then this result is straightforward. Specifically, in this case, the learning rate is $O(|V|^3)$, which is the computational complexity of solving (1) by using the knowledge of the network topology [19].

Thus, it is clear that it is not possible to learn the best node to attack in a blind scenario without attacking all $|V|$ nodes. In other words, when the network topology is unknown, it is not possible to achieve a learning rate better than $O(|V|)$. This completes the proof.

V. Results - Single Node Attack Scenario

The performance of various attack strategies is discussed under two cases below - a) when the flows are fixed over T epochs and b) when the flows randomly change every epoch. These are the two extreme cases that the attacker faces when attacking a network. When only a fraction of the flows change over time, then the behavior of the attacker is expected to be between the performance of these two cases. Due to space limitations, we consider only these extreme cases in this paper. In both cases, we average the performance of the proposed algorithms over 100 network instantiations each with 50 nodes and present the network interdiction performance over $T=5000$ epochs by plotting the cumulative proportion of the flows blocked at every epoch i.e., $\sum_t \frac{F_G^t - F_{G\setminus A_1^t}^t}{F_G^t}$, where A_1^t indicates the node attacked during epoch t. Note that F_G^t is a random value between 1 and $|V|/2$. The parameter A in Alg. 1 is taken to be 1.

A. Fixed flow scenarios

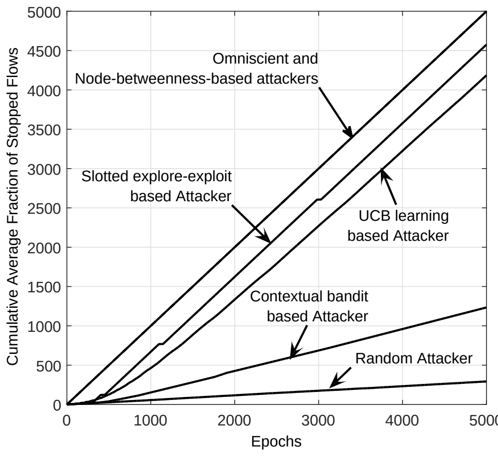

Fig. 2. Network attack performance against fixed flows in a star network, number of nodes=50.

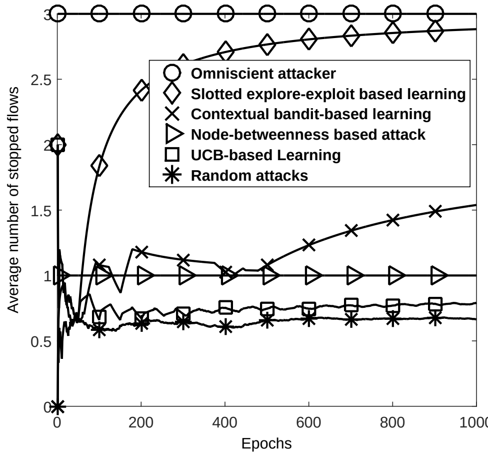

Fig. 3. Network attack performance against an Erdös-Rényi random network, connection probability $(p)=0.8$, number of nodes = 50. The average number of flows stopped in one network instantiation of the ER network is shown.

Fig. 2 shows the performance of the various attack strategies against a star network. Since the root node has the maximum betweenness metric and is also the optimal attack choice (for all the network instantiations) for the omniscient attacker in this network, the performance of both these strategies is the same. Since all flows are stopped by attacking the root node, the slope of the omniscient attacker performance is 1. Notice that after the initial learning period, the slope of the rewards attained by the slotted explore-exploit learning algorithm is also 1 during the exploitation phases. This indicates the fact that the slotted explore-exploit algorithm has learned the optimal node to attack i.e., the root node. Further, the steps (flat regions) seen in the performance of this algorithm are due to the exploration phases that are performed at exponentially increasing intervals. Since there are 50 nodes, the length of each exploration phase is also 50 epochs. While the slope of the rewards achieved by the UCB algorithm is also 1, the learning performance is slower in comparison with the slotted explore-exploit algorithm due to the continuous exploration-exploitation performed by the UCB algorithm.

The contextual bandit algorithm performs significantly worse than the slotted explore-exploit algorithm because the context provided to the contextual bandit algorithm is not sufficient to have any advantages over the other algorithms. Specifically, this is due to the high dimensional context vector and also the fact that just relying on the source-destination node pair (and not the entire flow path) does not reveal any information about the shortest path between them (which is necessary to attack the network). Therefore, in this case, the context provided confuses the attacker and thereby degrades its attack performance. As expected the random attacker's performance is worse than all the other approaches as it does not use any feedback and only serves as a lower bound for the attacker's performance.

Fig. 3 shows the average number of flows stopped $\left[\frac{1}{t}\sum_t \left[F^t_G - F^t_{G\setminus A^t_1}\right]\right]$ in an ER network by the attacker. Specifically, it shows the performance of the various attack algorithms against *one* network instance. The poor performance of the node-betweenness-based attack strategy (relative to the omniscient attacker) is due to the fact that when a fixed number of random flows is considered, there may or may not be any of these flows that pass through the node with the maximum betweenness metric value of the network G. Therefore, just relying on the overall network metric and not utilizing any information from the flows in the network is not beneficial to the attacker. As we will show shortly, the node betweenness metric of G should be used for network attacks only when the flows are randomly changing in the network. This is because the betweenness metric indicates the *average* number of flows that pass through a node and hence it may or may not work in instantaneous cases where a fixed number of flows are present in the network. This behavior indicates that even in this simple case i.e., when the flows are fixed, the popularly used attack strategies fail to perform well. This emphasizes the need for online learning techniques to be employed when networks are attacked blindly.

In Fig. 3, it is seen that the slotted explore-exploit algorithm performs very well in comparison to the other algorithms and also converges to the performance of the omniscient attack strategy. This indicates that it has learned the optimal node to attack which is the node with the maximum betweenness in G' as shown in Lemma 1. The UCB algorithm is seen to perform poorly in the case of the ER network as it continuously explores and exploits and therefore is not able to learn the best node to attack. This behavior is especially seen when there is not much difference between the rewards obtained by attacking various nodes. Under such cases, the UCB indices force the UCB algorithm to explore continuously [35]. This issue does not arise in the case of a star network because there is a significant difference in the rewards obtained by attacking the root node (the optimal node to attack) and any other node. In Fig. 3, the contextual-bandit and the random attack algorithms are seen to perform worse than the other algorithms due to the same reasons that were mentioned earlier in the context of a star network.

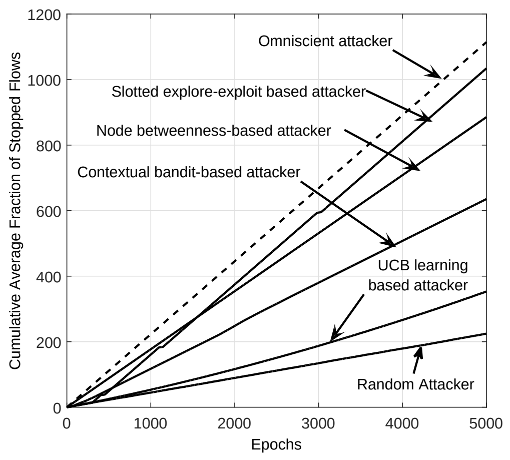

Fig. 4. Network attack performance against fixed flows in an ER network, $p = 0.8$, number of nodes = 50.

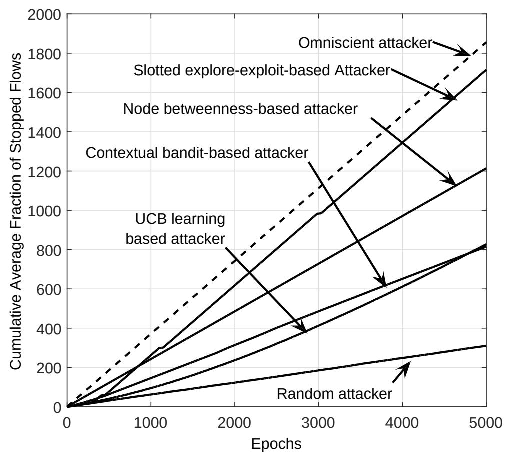

Fig. 5. Network attack performance against fixed flows in a BA network, connection degree = 5, number of nodes = 50.

Figs. 4 and 5 show the attacker's performance, averaged over 100 network instances, against ER and BA networks respectively, when the flows are fixed. As compared to the star network, the node-betweenness-based attack strategy fails to perform well in both these networks. This is because the betweenness metric is useful on average and not in instantaneous scenarios where the flows are fixed. Also notice that in both these networks, the slotted explore-exploit algorithm can perform better than the node-betweenness-based attack strategy and that the slope of the attacker's performance matches that of the omniscient attacker. This indicates that the slotted explore-exploit algorithm has learned the best node to attack. The jumps in the performance of the slotted explore-exploit algorithm again indicate the epochs where the exploration phases of the algorithm are finished and the exploitation starts. The UCB, contextual bandits and the random attack algorithm perform poorly against both the ER and BA networks due to continuous exploration-exploitation, no useful context information and a lack of feedback respectively.

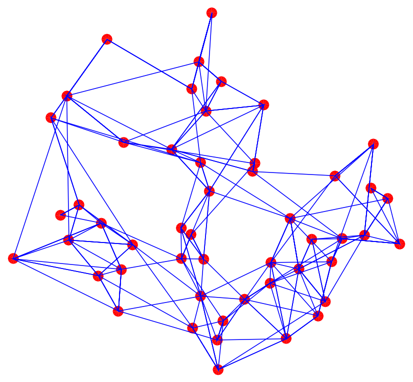

Fig. 6. PPP-based network model, with nearest neighbor connections. The red dots indicate the various network nodes and the blue lines indicate the network connections.

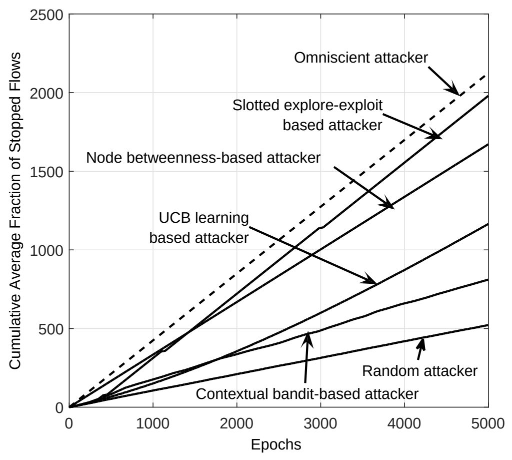

Fig. 7. Network attack performance against fixed flows in a PPP-based network, number of nearest neighbor connections = 5, number of nodes = 50.

In Fig. 6, we simulate a network with 50 nodes that are distributed in a plane according to a Poisson point process (PPP) model [31] and each node connects to its 5 nearest neighbors. Fig. 7, shows the network attack performance against this PPP network and even in this case it is seen that when the flows are fixed the slotted explore-exploit learning algorithm can match the omniscient attacker's performance after its exploration phases. The performance trends of the other attack strategies is similar as seen in other network topologies.

The main takeaways from the attackers performance against a network with fixed flows are-

1) The optimal node to attack in fixed flow scenarios is the node with the maximum betweenness metric over G' (see Lemma 1).
2) The slope of the rewards achieved by the slotted explore-exploit algorithm matches that of the omniscient attacker after the exploration phases i.e., it learns the optimal node to attack.
3) While the slope of the rewards achieved by the UCB algorithm matches the omniscient attacker in a star network, the learning procedure is slow in comparison with the slotted explore-exploit algorithm. However, due to its continuous exploration-exploitation, the UCB algorithm is seen to not perform well against ER and BA networks.
4) The source-destination node pair which is the context for the contextual bandit algorithm is not sufficient to learn the best nodes to attack and thereby degrades the attackers performance.
5) The betweenness-based attack is not ideal in fixed-flow scenarios as it only indicates the average number of flows that pass through a node and hence cannot be used to address such instantaneous scenarios.
6) The performance trends of the various algorithms in a fixed flow scenario is seen to be similar in all the networks considered.
7) The performance results show that information about flows in the network is more important to attack a network than the network topology itself in a fixed flow scenario. But notice that the network topology does provide some information as the performance of the various attack strategies is better than that of the random attacker.
8) Overall, it is important to realize that network topology and flows play a more crucial role in determining the attacker performance rather than just the number of nodes of the network. For example, even within the class of ER networks that have the same number of nodes, $\Delta_{\min}$ varies based on the connection probability p. This affects the learning rates of the algorithms proposed in this paper.

B. *Random flow scenarios*

Figs. 8-10 show the performance of the attack strategies when the flows randomly change (both the number of flows and also the source-destination node pairs) at every epoch. Since the flows change every epoch, W is taken to be 1.[6]

[6]If the attacker is aware of the time period over which flows remain fixed before changing, then W can be chosen appropriately.

The difficulty in learning over various networks is clearly seen in the results shown. The star network is the easiest to learn and hence even in this case, the performance of the slotted explore-exploit algorithm matches that of the optimal attacker after the initial learning phase. In fact it is the only network where it is possible to learn the optimal node since the optimal node does not change regardless of the flows. However, this is not the case in Figs. 9 and 10 (ER and BA networks) where there is significant difference in the performance of the optimal attacker and the slotted explore-exploit algorithm. This behavior is expected because the learning algorithms cannot learn the optimal node choice in such a random flow scenario, which is why the regret (gap between the learning performance and the optimal performance) incurred in these cases is linear in time (as mentioned in Section IV-C). In Figs. 9, 10 notice that the attacker's performance is better, in terms of the rewards achieved, in the case of BA networks as compared to the ER networks. This is because there exists a popular node through which several flows pass in the BA networks as mentioned earlier.

It is interesting to notice that in the random-flow scenario, the slope of the rewards achieved by the slotted explore-exploit algorithm matches that of the betweenness-based attack strategy after the learning phase. This indicates that the slotted explore-exploit algorithm learns to attack the node with the maximum betweenness metric in G. Since the flows are changing randomly, the betweenness metric indeed represents the average number of flows stopped by attacking a node. Hence, on average a large fraction of the flows indeed pass through the node that has the maximum betweenness metric. The performance of the slotted explore-exploit algorithm indicates that it does not learn about the flows in the network at any given epoch because it cannot match the performance of the omniscient attacker. But instead, it implicitly learns the network topology by learning the node-betweenness metric.

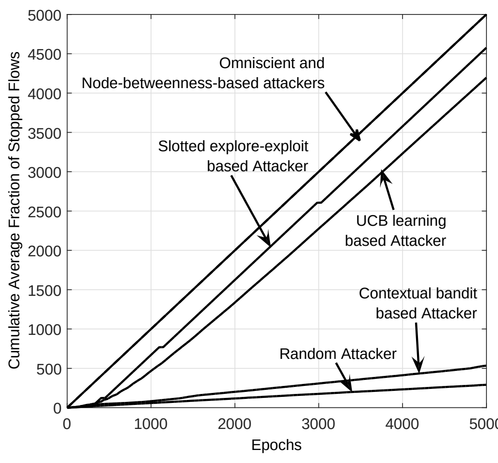

Fig. 8. Network attack performance against random flows in a Star network, number of nodes = 50.

The UCB algorithm learns to attack the optimal node only in the case of a star network albeit more slowly than the slotted explore-exploit algorithm. However, due to its constant exploration-exploitation schedule and the fact that the optimal attack choice changes every time instant, its performance is worse in the cases of ER and BA networks as seen in Figs. 9 and 10. Therefore it cannot keep up with the randomly changing flows. As mentioned in the fixed flow case, just having the source-destination node pair information as context is not helpful for the contextual bandit algorithm. This can be seen from Figs. 8-10 where its performance is close to that of a random attacker. This behavior of the contextual bandit algorithm suggests that in order to harness the power of these algorithms, it is appropriate for the attacker to be able to observe more parameters about the victim and not just the source-destination node pair.

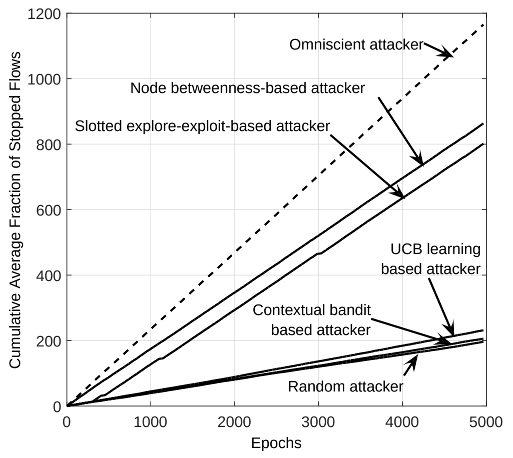

Fig. 9. Network attack performance against random flows in an ER random network, $p = 0.8$, number of nodes = 50.

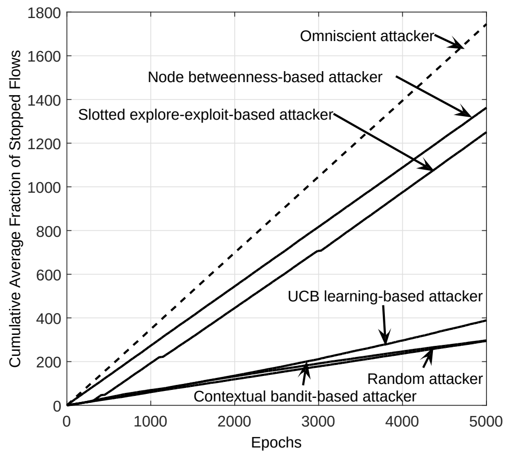

Fig. 10. Network attack performance against random flows in a BA network, number of nodes = 50, connection degree = 5.

The main takeaways from the attackers performance against a network with random flows are-

1) The star network is the easiest to learn irrespective of

whether the flows are fixed or not.

2) When the flows change randomly, the attacker can only learn about the average behavior of the flows. Specifically, the attacker learns the node with the maximum betweenness metric which indicates the average number of flows that pass through a node in the network.
3) Although the learning algorithms cannot learn the optimal attack strategies, the performance results show that learning is still important as it enables the attacker to implicitly learn the network topology by learning the node with the maximum betweenness centrality metric.

VI. Multiple node attack strategies

Here, we discuss scenarios where the attacker(s) can attack multiple nodes in the network at the same time. For instance, such scenarios occur in wireless networks where the attacker's transmit power is high enough to attack several nodes in its vicinity or can use beam forming to selectively attack different nodes in the network. The max-flow min-cut theorem states that the maximum amount of flows in a network pass through the set of edges that form the min-cut of G' (recall that G' is the sub network formed from G with only the set of nodes and edges that are needed for the flows $\{f^t_{s,d}\}_{s\neq d\in V}$). Therefore, when multiple nodes can be attacked, the attacker must attack the nodes that form this min-cut to stop a maximum number of flows. However, this min-cut can be evaluated only when the network topology is known *a priori*. However, this is not the case for the blind (topology unknown) scenarios studied in this paper, and hence the attacker faces the same dilemma as discussed previously. Therefore, we present learning algorithms that can be used to disrupt a network by attacking multiple nodes. Specifically, we study two scenarios– a) a single attacker can attack multiple nodes and b) multiple attackers can together attack multiple nodes, with each attacker capable of attacking only one node.

A. Single attacker

When any $k < |V|$ nodes can be attacked simultaneously, then there are $\binom{|V|}{k}$ possibilities that must be tried by the attacker in order to identify the best set of nodes that should be attacked. The total number of actions in such cases can be exponential depending on the network size and the number of nodes k that can be attacked. Thus, classical MAB algorithms (such as UCB) will need an exponential number of time steps to learn the best set of nodes that should be attacked. However, the attacker may not have sufficient time and such flexibility in real-time settings. Combinatorial bandit approaches [45], [46] have been used to address learning in such scenarios. In these algorithms, a set of arms (nodes in the current context) are played (attacked in the current context) together at any given time instant in an attempt to learn the best set of arms that minimizes the regret (or maximizes the reward). For example, these algorithms have been used to address social influence maximization over a directed graph in [47], to learn which popular files to be cached in a wireless network in [46] and several others.

In this paper, we employ the CUCB1 algorithm in [45] to learn the best set of k nodes to attack in order disrupt a maximum number of flows in the network under consideration. Due to space limitations, the implementation aspects of CUCB1 modified for the network interdiction setting can be found in Appendix IV in the extended version of this paper [28]. The attacker performance is shown in Fig. 11 and will be discussed shortly by comparing it with the cases where multiple attackers can simultaneously attack the network.

B. Multiple attackers

We next consider scenarios where multiple attackers can simultaneously attack the network by attacking one node each. In such scenarios, we show that it is not possible for the attackers to learn (their respective best nodes to attack) independently of each other and that they have to collaborate at every time instant to attack the network effectively.

Let $J = \{1, 2, \ldots, J\}$ indicate the set of attackers. Each attacker's decisions correspond to attacking one node in the network: the decision variable for each attacker is denoted by $\mathbf{x}^t_j$ for the jth attacker and $\mathbf{x}^t_{-j}$ indicates the decision made by the other attackers. $\mathbf{x}^t_j$ and $\mathbf{x}^t_{-j}$ are $|V| \times 1$ vectors with either a 0 or a 1 in each element indicating the node attacked. The attackers are all indirectly interacting with each other because the decisions taken by each attacker impacts the total number of flows stopped in the network. In addition, each attacker may also face a budget constraint denoted by $\mathbf{c}_j$ that limits the attack resources. In this paper, the budget constraint indicates the number of nodes that can be attacked by each attacker.

For the network interdiction problem considered in this paper, the jth attacker intends to solve the following problem

$$\begin{aligned}&\max_{\mathbf{x}^t_j}\ F^t_G - F_{G\setminus\{\mathbf{x}^t_j,\mathbf{x}^t_{-j}\}}\\ &\qquad \text{s.t. } \mathbf{c}_j\mathbf{x}^t_j \leq k \text{ and } \{x_{j_i}\}_{i=1}^{|V|} \in \{0,1\},\end{aligned} \tag{13}$$

where x_{j_i} is the ith element in the vector $\mathbf{x}_j$. Similar problems are solved by each attacker to find the best node they can attack. However, notice that each of these problems depends on the nodes attacked by the other attackers. When such knowledge is not available i.e., the attackers cannot cooperate with each other, then (13) is shown to be equivalent to a $\max$-$\max$ decentralized game [12]. However, if there is a centralized controller that coordinates the actions of all the attackers, then the network interdiction problem can be jointly written as

$$\begin{aligned}&\max_{\mathbf{x}^t}\ F^t_G - F_{G\setminus\{\mathbf{x}^t\}}\\ &\quad \text{s.t. } \mathbf{c}_j\mathbf{x}^t_j \leq k, \text{ and } \{x_{j_i}\}_{i=1}^{|V|} \in \{0,1\}\ \forall j = 1, 2, \ldots, J,\end{aligned} \tag{14}$$

where $\mathbf{x}^t = \{\mathbf{x}^t_j, \mathbf{x}^t_{-j}\}$.

The decentralized optimization in (13) is inefficient when compared to the centralized optimization in (14). This arises due to the fact that each attacker must optimize its decision without knowledge of the decisions (nodes attacked) taken by the other attackers. This inefficiency was also characterized by using a metric known as the *price of anarchy* in [12], where it is shown that the inefficiency increases as the size of the network increases. Therefore, the attackers must collaborate with each other to optimally attack a network.

Theorem 3. *Multiple attackers cannot learn the optimal network interdiction strategy independently.*

Proof: See the extended version of this paper [28].

In order for the attackers to learn the optimal attack choice in a real time setting, the slotted explore-exploit algorithm in Alg. 1 can be modified easily. Specifically, during the exploration phases in Alg. 1 all possible joint actions (Cartesian product of the individual attackers' action spaces) must be tried by the attackers together. Therefore, during the exploration phase, instead of $|V|$ time instants in Alg. 1, the attackers now spend $\prod_{i=1}^{N_J} |V_i|$ time instants where N_J is the total number of attackers and V_i indicates the set of nodes that can be attacked by the ith attacker. The exploitation phase and the transition between the exploration and exploitation phases remains the same as in Alg. 1. For this setup, the regret bound can be shown to be

$$\begin{aligned} R(T) \leq A \prod_{i=1}^{N_J} |V_i| \Delta_{\max} \log(T) + \prod_{i=1}^{N_J} |V_i| \Delta_{\max} \\ + \sum_{t}^{\infty} 2 \prod_{i=1}^{N_J} |V_i| \Delta_{\max} t^{-\frac{A}{2}\Delta_{\min}^2}, \end{aligned} \tag{15}$$

The proof is straightforward using the proof of Theorem 1 and is hence skipped.

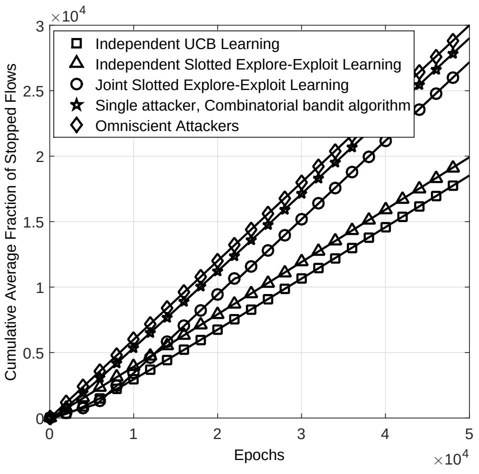

Fig. 11. Network attack performance against fixed flows in a ER network, with 25 nodes and $p = 0.8$, when two nodes can be attacked simultaneously by the attackers.

Fig. 11 shows the performance of the proposed (joint-learning) algorithm in comparison to independent learning by the attackers and the omniscient attack strategy that disrupts multiple nodes (as many as the number of attackers). Also seen is the performance of a single attacker that can attack multiple nodes. Firstly, notice that the slope of the rewards achieved by the single attacker and the multiple attackers with coordination matches that of the omniscient attacker after the initial learning phase. This indicates that in both the cases, the attackers learned the best nodes to attack. Multiple attackers with coordination learn more slowly than the single attacker because of the longer exploration phases (of duration $|V|^2$) as mentioned above. Therefore, this performance (both the learning rate and the final attack performance achieved) indicates that it is beneficial for multiple attackers to behave as a single attacker in practical real-time settings.

In Fig. 11, it is also seen that the joint slotted explore-exploit learning algorithm (where multiple attacker cooperate) performs significantly better than the independent learning techniques (both independent slotted explore-exploit and independent UCB) where the multiple attackers learn independently of the other attackers. The joint learning algorithm initially performs poorly in comparison with the independent learning algorithms because of the longer exploration phase of duration $|V|^2$ as opposed to $|V|$ in the case of independent learning (again recall that in the case of independent learning, each attacker independently uses Alg. 1 or UCB). When attackers are learning independently, they can learn the best nodes to attack if and only if all possible combinations of the attackers' choices are encountered during the respective exploration phases. But this can be ensured only when there is cooperation between the multiple attackers or if the exploration phases (i.e., which nodes to attack at any given time instant) are decided *a priori* before the learning starts. Therefore, this reinforces the fact that cooperation is necessary when multiple attackers exist and this results in significant performance gains over naive independent learning.

VII. Conclusions

In this paper, we studied blind network interdiction strategies i.e., attacking networks when their topology is unknown. Several learning algorithms have been proposed that attempt to learn the best node to attack in order to disrupt the network. We considered cases where a single attacker or multiple attackers can attack either a single node or multiple nodes in the network. We showed that (a) relying on well-known graph metrics, such as betweenness centrality, to attack a network works only in the case of star networks and does not necessarily work for all network topologies unless the flows are random, (b) under blind scenarios, the learning rates cannot be improved beyond $O(|V|)$ where $|V|$ is the number of nodes in the network, (c) in fixed-flow scenarios, the proposed slotted explore-exploit learning algorithm learns the optimal node to attack and in random flow scenarios it learns the node with the maximum betweenness metric in the underlying network, (d) multiple attackers have to collaborate at every time instant in order to learn the best set of nodes to attack in the network when the topology is unknown and (e) the learning performance, be it a single attacker or multiple attackers, depends on the network structure and not just the number of nodes in the network. We also showed that the proposed slotted explore-exploit learning always performs better, in terms of the flows stopped, than the popular learning algorithms such as UCB and contextual bandits. It remains to answer what additional observable parameters can help the attacker improve its learning rates i.e., better than $O(|V|)$.

References

[1] S. Amuru and R. M. Buehrer, "Optimal jamming in digital communication - impact of modulation," in *Proc. Global Commun. Conf.*, Austin, TX, Dec. 2014.

[2] S. Amuru and R. M. Buehrer, "Optimal jamming against digital modulation," *IEEE Trans. Inf. Forensics and Security*, vol. 10, no. 10, pp. 2212-2224, Oct. 2015.

[3] S. Amuru *et al.*, "Jamming bandits," in *arXiv:1411.3652*, Nov. 2014.

[4] S. Amuru and R. M. Buehrer, "Optimal jamming using delayed learning," in *Proc. Military Commun. Conf.*, Baltimore, MD, Oct. 2014, pp. 1528-1533.

[5] M. Litchman *et. al.,* "A communications jamming taxonomy," *IEEE Security and Privacy,* to appear, 2015.

[6] A. Proano and L. Lazos, "Selective jamming attacks in wireless networks," in Proc. *Intern. Conf. Commun.,* (ICC), Cape Town, South Africa, May 2010, pp. 1-6.

[7] R. K. Wood, "Deterministic network interdiction," *Mathl. Comput. Modelling*, vol. 17, no. 2, U.K., 1993, pp. 1-18.

[8] E. Israeli and R. K. Wood, "Shortest-path network interdiction," *Netw.* vol. 40, no. 2, pp. 97-111., 2002.

[9] D. S. Altner, O. Ergun, and N. A. Uhan, "The maximum flow network interdiction problem: valid inequalities, integrality gaps, and approximability," *Oper. Res. Lett.,* vol. 38, pp. 33-38, 2010.

[10] K. Cormican, D. Morton, and K. Wood, "Stochastic network interdiction," *Oper. Res.,* vol. 46, pp. 184-197, 1998.

[11] A. Washburn and K. Wood, "Two-person zero sum games for network interdiction," *Oper. Res.,* vol. 43, no. 2, Mar. 1995, pp. 243-251.

[12] H. Sreekumaran *et. al.,* "Multi-agent decentralized network interdiction games," in *arXiv:1503.01100*, Mar. 2015.

[13] P. Tague *et. al.,* "Linear programming models for jamming attacks on network traffic flows," in Proc. *Intl. Symp. Modeling and Opt. Mobile, Ad Hoc, and Wireless Netw.,* (WiOpt), Berlin, Germany, Apr. 2008, pp. 207-216.

[14] R. Albert *et. al.,* "Error and attack tolerance of complex networks," *Nature,* vol. 406, pp. 378-482, 2000.

[15] D. Magoni, "Tearing down the internet," *IEEE J. Sel. Areas Commun.,* vol. 21, no. 6, pp. 949-960, Aug. 2003.

[16] D. Zhang, E. K. Çetinkaya, and J. P. G. Sterbenz, "Robustness of mobile ad hoc networks under centrality-based attacks," in *Proc. Reliable Netw. Design Modeling,* (RNDM), Almaty, KZ, Sept. 2013, pp.1-7.

[17] B. R. da Cunha *et. al.,* "Complex networks vulnerability to module-based attacks," in *arXiv:1502.00353*, Feb. 2015.

[18] P.-Y. Chen and A. O. Hero, "Assessing and safeguarding network resilience to nodal attacks," *IEEE Commun. Mag.,* vol. 52, no. 11, pp. 138-143, Nov. 2014.

[19] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry,* vol. 40, no. 1, pp. 35-41, Mar. 1977.

[20] U. Brandes, "A faster algorithm for betweenness centrality," *J. Math. Soc.,* vol. 25, no. 2, pp. 163-177, 2002.

[21] P. Erdös and A. Rényi (1959). "On random graphs," *Publ. Math. Debrecen,* vol. 6, pp. 290-297, 1959.

[22] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509-512, 1999

[23] D. Acemoglu, M. Dahleh, I. Lobel, A. Ozdaglar, "Bayesian learning in social networks," *Review of Economic Studies,* vol. 78, pp. 1201-1236, Mar. 2011.

[24] D. Acemoglu, A. Ozdaglar, A. ParandehGheibi, "Spread of (mis)information in social networks," *Games and Economic Behavior,* vol. 70, no. 2, pp. 194-227, Feb. 2010.

[25] D. Acemoglu, A. Ozdaglar, A. Tahbaz-Salehi, "Cascades in networks and aggregate volatility," working paper, MIT, 2010.

[26] M. Valko, R. Munos, B. Kveton, and T. Kocak, "Spectral bandits for smooth graph functions," in *Proc. Intern. Conf. Mach. Learning* (ICML), Beijing, China, Jun. 2014, pp. 46-54.

[27] M. K. Hanawal and V. Saligrama, "Efficient detection and localization on graph structured data," in *Proc. Intern. Conf. Acoust. Speech Signal Process.* (ICASSP), Brisbane, Australia, 2015.

[28] S. Amuru, R. M. Buehrer, and M. van der Schaar, "Blind Network Interdiction Strategies - A Learning Approach" available at http://www.buehrer.ece.vt.edu/papers/BlindNetworkInterdiction.pdf

[29] B. Zhou and J. Pei, "Preserving privacy in social networks against neighborhood attacks," in *Proc. Intern. Conf. Data Engg.* (ICDE), Cancun, Apr. 2008, pp. 506-515.

[30] H. Wang, J. M. Hernandez and P. Van Mieghem, "Betweenness centrality in a weighted network," *Phys. Rev. E*, vol. 77, pp. 0461051-04610510, Apr. 2008.

[31] H. S. Dhillon, R. K. Ganti, F. Baccelli and J. G. Andrews, "Modeling and analysis of K-tier downlink heterogeneous cellular networks," *IEEE J. Sel. Areas Commun.,* vol. 30, no. 3, pp. 550-560, Apr. 2012.

[32] R. W. Stevens, *TCP/IP Illustrated, Vol. 1: The protocols*. Addison-Wesley, 2011.

[33] S. Chen, Estimating the number of nodes in a mobile wireless network, in *Proc. Global Commun. Conf.*, Miami, FL, Dec. 2010, pp. 1-5.

[34] Z. Yang *et. al.*, Determining the number of nodes for wireless sensor networks, in *Proc. Symp. on Emerging Tech. Mobile and Wireless Comm.,* Jun. 2004, pp. 501-504.

[35] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multi-armed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235-256, May 2002.

[36] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multi-armed bandit problem," *SIAM J. Comput.,* vol. 32, no. 1, pp. 48-77, Jan. 2002.

[37] R. Kleinberg, "Nearly tight bounds for the continuum-armed bandit problem," in *Proc. Neural Inf. Proc. Syst.,* 2004.

[38] S. Magureanu et al, "Lipschtiz bandits: regret lower bounds and optimal algorithms," in *Proc. Conf. Learning Theory,* (COLT), 2014.

[39] A. Slivkins, "Contextual bandits with similarity information," in *Proc. Conf. Learning Theory,* (COLT), 2011.

[40] C. Tekin, L. Canzian, and M. van der Schaar, "Context adaptive big data stream mining," in *Proc. Allerton Conf. Commun. Control and Comput.,* Monticello, IL, Oct. 2014.

[41] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *Proc. Conf. Alg. Learning Theory* (ALT), Espoo, Finland, Oct. 2011, pp. 174-188.

[42] F. Comellas and S. Gago, "Spectral bounds for the betweeness of a graph," *Linear Algebr. Appl.*, vol. 423, no. 1, pp. 74-80, 2007.

[43] A. Fronczak, P. Fronczak, and J. Holyst, "Average path length in random networks," *J. Phys. Rev. E,* vol. 70, no. 5, pp. 0561101-0561107, Nov. 2004.

[44] V. Blondel *et. al.,* "Distance distribution in random graphs and application to network exploration," *J. Phys. Rev. E*, vol. 76, no. 6, pp. 0661011-0661018, Dec. 2007

[45] B. Kveton *et. al.,* "Tight regret bounds for stochastic combinatorial semi-bandits," in *Proc. Artificial Intell. Stat.* (AISTATS), Jul. 2015.

[46] A. Sengupta, *et. al.,* "Learning distributed caching strategies in small cell networks," in Proc. *Intern. Symp. Wireless Commun. Syst.* (ISWCS), Barcelona, Spain, Aug. 2014, pp. 917-921.

[47] W. Chen, *et. al.,* "Combinatorial Multi-Armed Bandit: General Framework, Results and Applications," in Proc. *Intern. Conf. Mach. Learning*, Atlanta, GA, Jun. 2013, pp. 1-9.