Robustness of Random Walk on a Graph against Adversary Attacks

***Abstract*—Random walk-based algorithms are frequently utilized to target node search and graph exploration in unknown graph structures. Unlike deterministic algorithms such as breadth-first search and depth-first search, target node search and graph exploration with random walk algorithms are expected to exhibit robustness against adversarial attacks because of their probabilistic nature. The characteristics of random walks when adversaries change the topology of the graph, known as adversarial attacks on random walks, have been just recently received attention. These attacks have been shown to significantly degrade the efficiency of target node search and graph exploration with random walks. However, questions regarding how robust random walks are against more realistic attacks persist. In this paper, we investigate adversarial attacks in the form of rewiring a limited number of links during target node search and graph exploration, particularly in scenarios where mobile agents employ random walk algorithms. The goal is to quantitatively determine how robust or vulnerable the conventional random walk algorithms are against link rewiring attacks. We consider three types of link rewiring attacks (centrality method, clustering method, and starting node method) and evaluate how they affect target node search (hitting time) and graph exploration time (cover time) of five major random walk algorithms on a graph — Simple Random Walk (SRW), Non-Backtracking Random Walk (NBRW), k-History random walk (k-History), Biased Random Walk (BiasedRW), and Vicinity Avoidance Random Walk (VARW) — in five graphs through simulation experiments.**

***Index Terms*—random walk, adversarial attack, hitting time, cover time, link rewiring, robustness, target node search, graph exploration**

I. Introduction

Graphs, commonly referred to as networks, can be derived from the relationships among diverse entities in the real world. Various types of graphs are commonly employed to define relationships such as social networks, biological networks, patent networks, transportation networks, citation networks and communication networks [1-4]. In the context of target node search and graph exploration on unknown graphs, random walk-based algorithms have gained widespread popularity.

A random walk is a probabilistic process that defines a trajectory through a sequence of random steps in a mathematical space. In a mathematical context, a simple random walk model entails traversing a regular lattice, where, at each step, the agent at a point may move to another point based on a specific probability distribution. When applied to a specific graph, such as a network, the probability of transition between nodes is directly proportional to the strength of their connection. The more robust the connection between nodes, the greater the likelihood of transition from one to another. After a sufficient number of steps, a random path can be generated, effectively representing the underlying structure of the graph [5, 6].

Understanding and enhancing random walks on a graph holds significant importance in the development of high-quality and reliable protocols, controls, applications, and services within extensive communication networks. While conventional deterministic algorithms may suffice for small-scale and static communication networks, the landscape changes in large-scale and dynamic networks [7]. In such scenarios, entities within the network lack access to global knowledge of the entire network, limiting their information to localized, partial knowledge.

For instance, in dynamic large-scale networks, algorithms tailored for static graphs like Breadth-First Search (BFS) and Depth-First Search (DFS) prove ineffective due to the ever-changing nature of the network. Consequently, in dynamic large-scale networks, a specific class of algorithms based on random walks, which relies solely on the local information available to an agent (i.e., a random walker), often represents the sole viable solution [8].

Random walks on a graph form the basis of numerous critical techniques aimed at enhancing the Quality of Service (QoS) and Quality of Experience (QoE) within large-scale, high-performance communication networks [9, 10]. Applications of random walks in the network layer of complex large-scale networks encompass network topology discovery, network sampling, message routing, message diffusion, network resource discovery, node clustering, and network tomography [5, 11, 12].

Moreover, in the application layer, these random walks find utilities in various domains, such as large-scale content networks (e.g., the Web) and social networks. In these contexts, random walks can be utilized for network sampling, node ranking, node classification, node matching, node identification, and node embedding [12, 13]. It is worth noting that simple random walks on a graph, while straightforward and tractable, often prove inefficient for practical applications. Consequently, a range of advanced random walk-based algorithms, including those incorporating historical data and intelligent decision-making, are utilized to achieve significantly improved efficiency and reliability in these contexts [5, 12].

Over the years, several types of random walks have been proposed, including simple random walk (SRW) [5, 6], non-backtracking random walk (NBRW) [14], k-history random walk (k-History RW) [15], biased random walk (BiasedRW) [16], and vicinity avoidance random walk (VARW) [17]. These algorithms have different characteristics and can address particular problems. The probabilistic nature of random walks allows an agent to visit diverse nodes, facilitating, for instance,

comprehensive information sampling in an unknown graph.

Random walk-based algorithms are expected to be relatively more robust against adversarial attacks than deterministic algorithms due to their probabilistic nature [18], even in cases where an agent may be disrupted by potential adversarial attacks such as edge rewiring, node addition/deletion and node/edge attribute forging.

The characteristics of random walk algorithms (e.g., target node search and graph exploration) when attackers dynamically reconfigure the entire topology of a graph have been analyzed [19]. Such attacks are known to significantly impair the efficiency of the target node search and the graph exploration with random walks [20]. However, the robustness of target node search and graph exploration on an unknown graph with random walk-based algorithms under more realistic attacks has not been well understood.

In this paper, we address the following research questions.

- To what extent are target node search and graph exploration with random walks on a graph vulnerable to link rewiring attacks?
- Among several random walk algorithms on graphs with different properties, which ones exhibit severe degradation in its efficiency against link rewiring attacks?
- How does the vulnerability of random walks on a unknown graph vary based on the structure and properties of the graph?

In the context of adversarial attacks, we focus on the scenario where a small number of links are rewired by the attacker during target node search or graph exploration by an agent that obeys a given random walk algorithm. The objective of this paper is to quantitatively assess the robustness (or vulnerability) of existing major random walk algorithms (i.e., SRW, NBRW, k-history RW, BiasedRW, and VARW) against link rewiring attacks.

To address the research questions, we consider three types of link rewiring attacks called the *centrality method*, the *clustering method*, and the *starting node method*. Let S_t denote the subgraph comprising the set of nodes visited by the mobile agent until time t. We consider three methods for rewiring a link connected to the node visited by the mobile agent.

- Centrality method
  One of links connected to the current visiting node is rewired to the node with the highest degree centrality in S_t.
- Clustering method
  One of links connected to the current visiting node is rewired to a randomly-chosen node in the largest cluster of S_t.
- Starting node method
  One of links connected to the current visiting node is always rewired to the starting node v_0 of the random walk.

Through experiments, we measure the target node search time (hitting time) and the graph exploration time (cover time) for several major random walk algorithms under three link rewiring attacks. The main contributions of this paper can be summarized as follows:

- We quantitatively reveal the vulnerability of random walks on a graph, which form the foundation for many engineering applications, to different types of link rewiring attacks.
- We reveal that a (memoryless) simple random walk is approximately two to three times more robust (in terms of search and exploration efficiency) against a small number of link rewiring attacks compared to memory-enabled random walk algorithms.
- We conduct a comparative analysis on the impact of three link rewiring attacks (centrality, clustering, and starting node methods). The results reveal that random walk algorithms are particularly vulnerable to the starting node method, irrespective of the topological structure of graphs.

The paper is organized as follows. In Section II, we provide an overview of related work on existing random walk algorithms on a graph. In Section III, we formulate the random walk attack problem and introduce three link rewiring attacks that are investigated throughout this paper. In Section IV, we conduct experiments to investigate the robustness of random walks against three link rewiring attacks. Finally, in Section VI, we summarize this paper and discuss future research directions.

II. Random Walk Algorithms on a Graph

This section briefly introduces the existing random walk algorithm variants including SRW, NBRW, k-History, BiasedRW, and VARW on a unknown graph.

Simple Random Walk (SRW) is a basic random walk method in which an agent moves randomly from node to node, without constraints on memory or node revisitation [6, 5]. At each step, the agent randomly selects a node with equal probability from adjacent nodes and moves to that node. Therefore, the agent may revisit the same node and intersect its own path.

Non-Backtracking Random Walk (NBRW) is essentially the same method as SRW, but it includes an additional condition that forbids revisiting nodes within the last n $\leq$ 1 steps [14]. This approach, which utilizes short-term memory to avoid returning to nodes visited in the recent past, is known to exhibit superior characteristics compared to SRW.

k-History random walk (k-History) is a random walk algorithm where the agent possesses a memory of size k, aiming to explore a graph while avoiding revisiting previously visited nodes by as much as possible [15]. When the agent visits a node, it first records the identifier of that node in its memory according to the recording method. Then, when choosing the next node to transition to, the agent randomly selects a node from the adjacent nodes, excluding those recorded in the memory. However, if all adjacent nodes are already recorded in the memory, the agent randomly selects the next node from the adjacent nodes.

Biased Random Walk (BiasedRW) is a random walk in which the probability of moving from a given node to one of its neighbors, with a degree of k, is proportional to k^{α}, where α is a tuning parameter [16]. Depending on the sign of the bias parameter α, the walkers preferentially move either towards hubs or towards poorly connected nodes.

Vicinity Avoidance Random Walk (VARW) [17] is similar to NBRW and Self Avoidance Random Walk (which records all visited nodes and attempts to avoid revisiting them), where the mobile agent holds a memory of previously visited nodes. The difference from SARW is that VARW does not record all visited nodes. It aims to prevent the mobile agent from revisiting nodes within the past t time, as well as its neighboring nodes, as much as possible.

III. Adversarial Attack on Random Walk

A. Problem formulation

The random walk attack problem addressed in this paper involves attackers rewiring links in a graph $G = (V, E)$ to hinder (delay) the progress of mobile agents, which utilize one of the random walk algorithms, during target node search or graph exploration on unknown graphs. The attacker can observe and record the movement of the agent at each step, allowing the attacker to obtain a sequence of nodes by recording it, and rewire links at a certain frequency. The specific setup for the random walk attack problem is as follows. The sequence of nodes visited by the agent from time 0 to time t is denoted as $P(t) = \{v_0, v_1, ..., v_t\}$, and the frequency of link rewiring by the attacker is denoted as A. The attacker rewires any link on the graph $G = (V, E)$ every A steps after the agent starts moving. However, to avoid detection, link rewiring that would result in the graph becoming disconnected is not allowed i.e., the graph must be kept connected after every link rewiring.

B. Overview of three attack methods

In the following, let S_t denote the subgraph consisting of the set of nodes visited by the moving agent up to time t. We consider three methods for the moving agent to rewire a link connected to nodes is visiting: (1) a method that rewires a link to the node with the the highest degree in S_t (centrality method), (2) a method that rewires a link to the largest cluster in S_t (cluster method), and (3) a method that rewires a link to the starting node v_0 (starting method).

In each proposed attack method, the link (v_t, u) connected to node v_t, where the number of hops from node v_{t-1} to node u $(u \notin P(t), (v_t, u) \in E)$ is maximized, is selected. In each method, the link between node v_t and u is deleted, and a new link is added between node v_t and u'.

C. Centrality method

This method (Algorithm 1) aims to increase the likelihood of agents revisiting visited nodes by connecting them to the nodes with the highest degree of centrality among the visited nodes. In the centrality method, the node with the highest degree in S_t is selected as the rewiring node u' (Fig. 1). In other words, when the degree of node v is denoted as $d(v)$, select node u' as

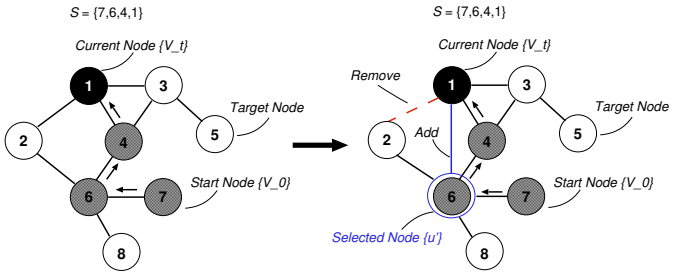


Fig. 1. Centrality method

$$\arg\max_{u' \in P(t), (v_t, u') \notin E} d(u') \tag{1}$$

Algorithm 1 Centrality method

Input: S_t set of nodes visited by the agent up to time t,
v_t current visit node, E set of edges in graph G
Step 1:
$u' \leftarrow$ a node with highest degree in S_t
if $(u', v_t) \in E$ **then**
    repeat step 1 with the next highest degree node

Step 2:
for each $n \in \mathrm{N}(v_t)$ **do**
    $h \leftarrow$ number of hops between n and v_{t-1}
$u \leftarrow$ select n where h is maximum
$g \leftarrow$ copy of G
delete $(u, v_t) \in E$ in g
if h is still connected graph **then**
    delete $(u, v_t) \in E$ in G
    add (u', v_t) in G
    return

D. Clustering method

This method (Algorithm 2) aims to encourage agents to wander within visited clusters by connecting them to nodes that constitute the largest cluster among the visited nodes. In the clustering method, first, S_t is clustered using the Louvain algorithm. Let C be the set of generated clusters and c be a cluster belonging to C. Then, define the set of nodes belonging to cluster c as N_c, and randomly select a node from N_c as the rewiring node u' (Fig. 2). In other words, when the number of nodes in cluster c is denoted as $L(c)$, select node u' as

$$\arg\max_{c \in C, u' \in N_c, (v_t, u') \notin E} L(c') \tag{2}$$

E. Starting node method

This method (Algorithm 3) aims to increase exploration time by connecting nodes back to the starting node, thereby

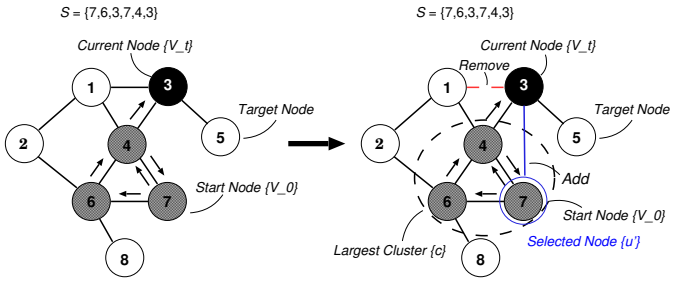


Fig. 2. Clustering method

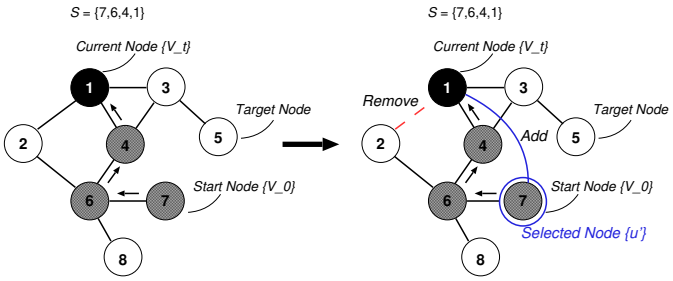


Fig. 3. Starting node method

Algorithm 2 Clustering method

Input: S_t set of nodes visited by the agent up to time t,
v_t current visit node, E set of edges in graph G,
$C = \{C_1, C_2, ..., C_n\}$ set of clusters generated by clustering S_t
Step 1:
for each $l \in C(= \{C_1, C_2, ..., C_n\})$ **do**
    $n \leftarrow$ number of nodes in l
$c \leftarrow$ select l where n is maximum
$N_c \leftarrow$ set of nodes belonging to c

Step 2:
$u' \leftarrow$ a node with randomly selected in N_c
if $(u', v_t) \in E$ **then**
    repeat step 2 with the other node in c

Step 3:
for each $n \in \mathrm{N}(v_t)$ **do**
    $h \leftarrow$ number of hops between n and v_{t-1}
$u \leftarrow$ select n where h is maximum
$g \leftarrow$ copy of G
delete $(u, v_t) \in E$ in g
if h is still connected graph **then**
    delete $(u, v_t) \in E$ in G
    add (u', v_t) in G
    return

Algorithm 3 Starting node method

Input: S_t set of nodes visited by the agent up to time t,
v_t current visit node, E set of edges in graph G
Step 1:
$u' \leftarrow$ a node visited earliest in S_t
if $(u', v_t) \in E$ **then**
    repeat step 1 with next earliest node

Step 2:
for each $n \in \mathrm{N}(v_t)$ **do**
    $h \leftarrow$ number of hops between n and v_{t-1}
$u \leftarrow$ select n where h is maximum
$g \leftarrow$ copy of G
delete $(u, v_t) \in E$ in g
if h is still connected graph **then**
    delete $(u, v_t) \in E$ in G
    add (u', v_t) in G
    return

returning to the initial position. In the starting node method, the node visited earliest by the agent is selected as the rewiring node u' (Fig. 3). In other words, when the time t at which node v was visited is denoted as $H_t(v)$, select node u' as

$$\underset{u' \in P(t), (v_t, u') \notin E}{\arg\min} \quad H_t(u') \tag{3}$$

IV. Experiments

A. Methodology

In this section, we present simulation results that evaluate the vulnerability, specifically in terms of target node search (hitting time) and graph exploration (cover time), of mobile agents employing random walk algorithms (such as SRW, NBRW, k-History, BiasedRW, and VARW) on an unknown graph when attacked by the three attack methods. Specifically, we conduct simulation experiments on five types of graph (Erdos and Rényi (ER) [21], Barabási-Albert (BA) [22], ring, 3-regular [23], and Voronoi graphs [24]) with the same size and the same density (i.e., average degree). These graphs are commonly used in simulations to model real-world networks. The following is an explanation of each graph.

- Erdos and Rényi (ER) [21] graph is a random graph model where nodes are connected by edges with a fixed probability. ER graphs serve as fundamental models in network theory, aiding in the analysis of real-world networks and processes.
- Barabási-Albert (BA) [22] graph is a random graph model where new nodes are added, and they preferentially attach to existing nodes based on the degree of those nodes. This results in the formation of hubs, and nodes with high degrees, and follows a power-law distribution of node degrees. BA graphs are used to model real-world networks with growth and preferential attachment characteristics.
- 3-regular graph [23] is a type of graph where each node has exactly three edges connected to it. In other words, every node in a 3-regular graph has exactly three neighboring nodes. This graph is often used in

various applications, including network modeling because it exhibits uniform connectivity and can represent certain physical or social systems efficiently.

- Ring graph is a type of graph where nodes are arranged in a circular or ring-like structure. Each node in a ring graph is connected to its two adjacent nodes, forming a closed loop. Ring graphs are commonly used to model scenarios where nodes are arranged in a cyclical pattern or when periodicity is a key characteristic of the system being studied.
- Voronoi graph [24] partitions a space into regions determined by the proximity to specific seed points. Each point within the space is assigned to the region associated with its nearest seed point. Voronoi graphs find applications in diverse fields such as spatial analysis and pattern recognition.

In this experiment, each graph was generated with 100 nodes. In each graphs, we performed link rewiring attacks once every 50 or 100 [steps] from the initiation of mobile agent movement and we measured the average hitting time taken by an agent, starting from a randomly selected initial node, to reach a randomly chosen target node. We also measured the avearage cover time taken by a mobile agent, starting from a randomly selected initial node, to visit all nodes except the starting node at least once. By conducting simulations under the same conditions for 2000 trials, we measured the average and 95% the confidence interval of the hitting time, as well as the average and 95% confidence interval of the cover time.

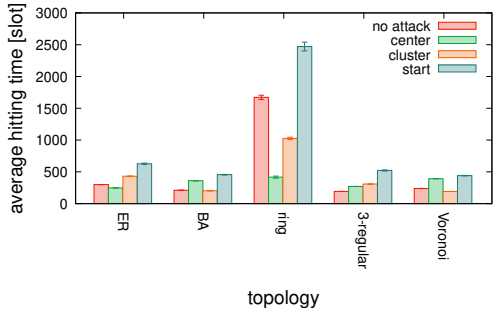
B. Results on average hitting time

The average hitting time of SRW when attacked once every $A = 50$, 100 steps are shown in Fig. 4, those of NBRW are shown in Fig. 5, those of k-History are shown in Fig. 6, those of BiasedRW are shown in Fig. 7, and those of VARW are shown in Fig. 8.

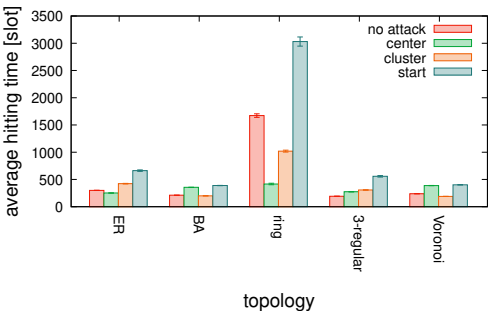
From these results, it is observed that, regardless of the graph's topology, the average hitting time of all random walks increases by approximately 2–3 times with the starting point attack method. This increase is attributed to the higher likelihood of revisiting node sequences by returning from the currently visited node to the starting node, consequently increasing the number of steps required to reach the target node.

Similarly, with the central and cluster methods, in most topologies, the average initial arrival times of random walks either remain similar or increase by a factor of 2–3. However, in the case of the ring topology, the average initial arrival time of SRW decreases by approximately 1/2–1/3. This phenomenon may be explained by the central method's tendency to return to the node with the highest degree centrality among the visited node sets, increasing the probability of returning to nodes with a high degree of centrality, which may hinder the exploration of new nodes and consequently increase the number of steps required to reach the target node.

Likewise, with the cluster method, returning to the cluster with the maximum number of nodes visited so far may result in increased time spent within that cluster, leading to an increase in the number of steps required to reach the target node. However, in a ring topology where all nodes have equal degrees, it may be challenging to execute intended attacks, potentially decreasing the number of steps required to reach

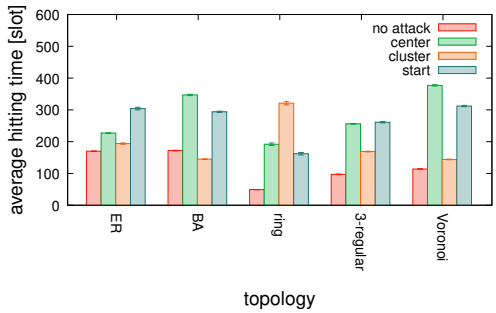


(a) attack interval of $A = 50$

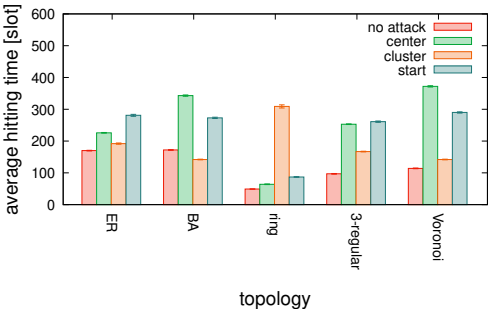


(b) attack interval $A = 100$

Fig. 4. Average hitting time with SRW

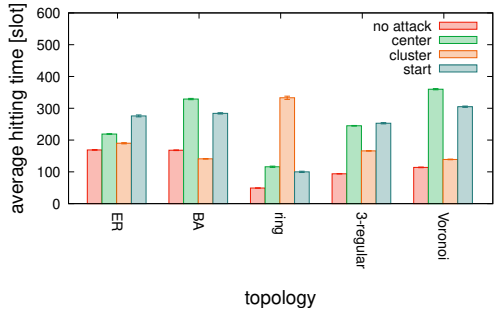


(a) attack interval of $A = 50$

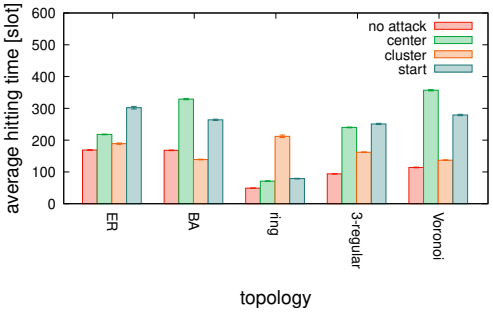


(b) attack interval $A = 100$

Fig. 5. Average hitting time with NBRW

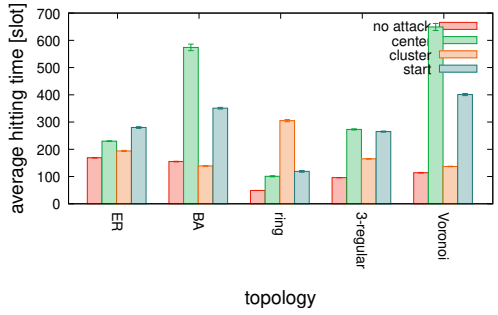


(a) attack interval of $A = 50$

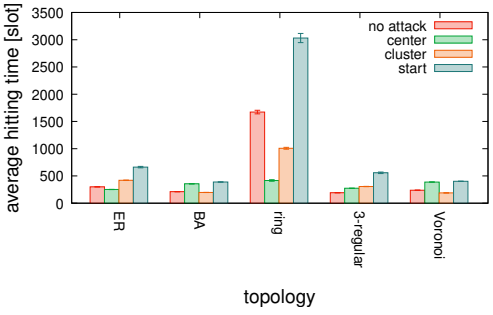


(b) attack interval $A = 100$

Fig. 6. Average hitting time with kHistory

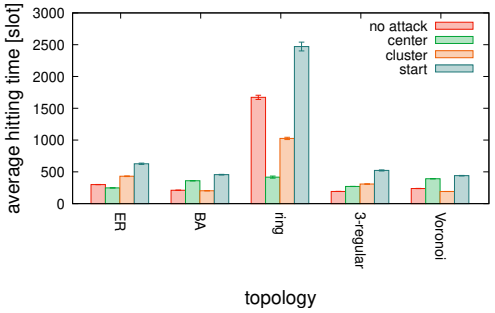


(a) attack interval of $A = 50$

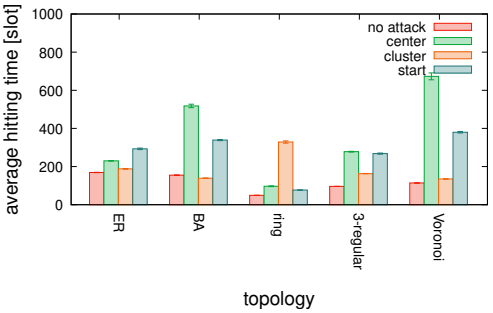


(b) attack interval $A = 100$

Fig. 7. Average hitting time with BiasedRW

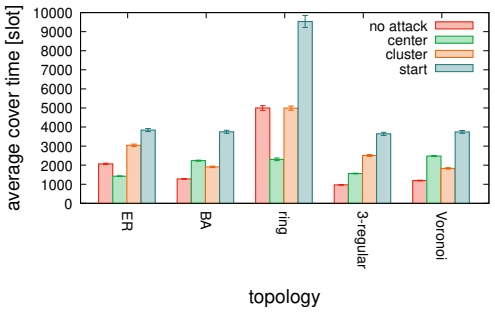


(a) attack interval of $A = 50$

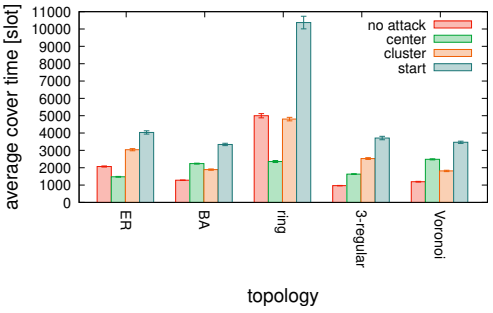


(b) attack interval $A = 100$

Fig. 8. Average hitting time with VARW



(a) attack interval of $A = 50$



(b) attack interval $A = 100$

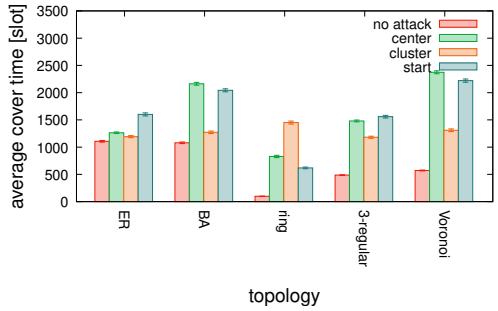
Fig. 9. Average cover time with SRW

the target node.

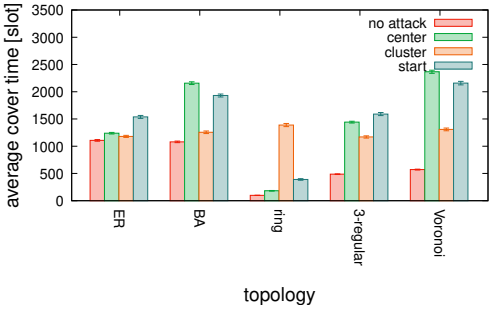
C. Results on average cover time

The average cover time of SRW when attacked once every $A = 50$, 100 steps are shown in Fig. 9, those of NBRW are shown in Fig. 10, those of k-History are shown in Fig. 11, those of BiasedRW are shown in Fig. 12, and those of VARW are shown in Fig. 13.

From these findings, it is evident that, regardless of the graph's topology, the average coverage times of all random walks increase by approximately 2–3 times with both the

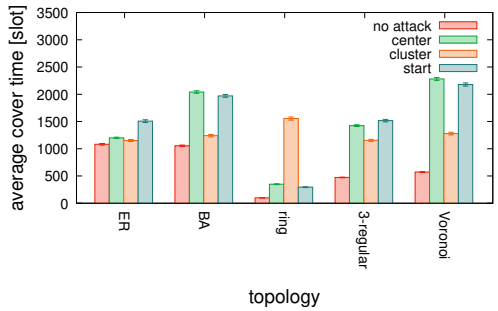


(a) attack interval of $A = 50$

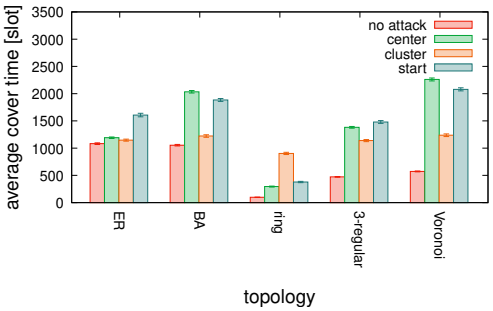


(b) attack interval $A = 100$

Fig. 10. Average cover time with NBRW

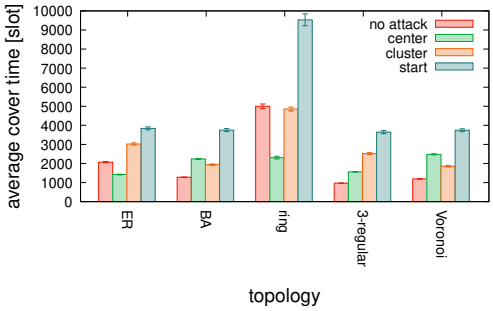


(a) attack interval of $A = 50$

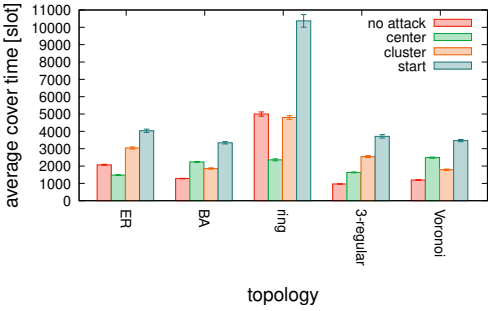


(b) attack interval $A = 100$

Fig. 11. Average cover time with kHistory

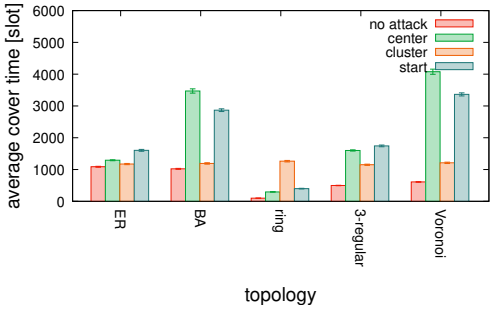


(a) attack interval of $A = 50$

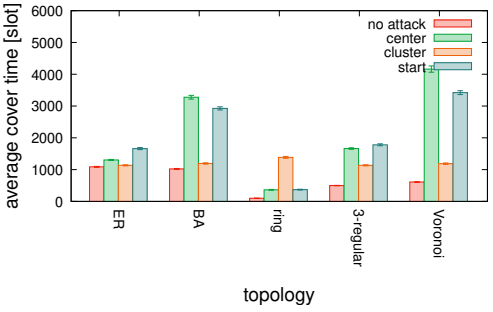


(b) attack interval $A = 100$

Fig. 12. Average cover time with BiasedRW



(a) attack interval of $A = 50$



(b) attack interval $A = 100$

Fig. 13. Average cover time with VARW

cluster and starting point methods. This increase may be attributed to the tendency of the cluster method to become trapped within the replaced cluster, thus increasing the number of steps required to explore all nodes. Similarly, the starting point method may increase the number of steps required to explore all nodes due to the higher likelihood of revisiting node sequences by returning to the starting node.

On the other hand, it is observed that coverage times decrease by approximately $1/2$ in some cases with the central method. This decrease might be attributed to the central

method's tendency to facilitate visits to nodes with a large number of hops from the currently visited node, which has not been visited before and might take longer to reach without attacks, thereby assisting in exploring nodes that have a high degree of hop count from the currently visited node and have not been visited before.

V. Discussions

Our findings can be applied in real-world scenarios. For example, when attempting to identify the source of malware on a website. Using a web crawler, links on the website are randomly followed to find the source of the malware. However, the malware distributor has the authority to rewire the links on the website, thereby hindering the crawler's exploration. In such situations, the results of this study serve as clues to make identifying the source of malware more difficult.

Furthermore, consider the case of wanting to find a specific individual person on the social network. When exploring the community to which that individual belongs, link rewiring attacks may be effective. If each node represents a friend and the links represent those friend relationships, it is possible to delay the identification of a specific individual by rewiring links.

VI. Conclusions

In this paper, we proposed adversarial attacks in the context of target node search and graph exploration using random walks. We had assumed attacks involving the rewiring of a small number of links by an adversary and had analyzed how robust or vulnerable existing random walk-based movement algorithms are to link rewiring attacks. In five types of random walks, the rewiring of a small number of links increased the average hitting time and average cover time by up to approximately 5 times. SRW had been found to be the least susceptible to the impact of attacks. Among the three attack methods, the starting node method had a significant deteriorating effect on the characteristics of random walks, while the centrality method, under certain conditions, showed an improvement in the characteristics of random walks. In this experiment, graph topology does not significantly affect the vulnerability of random walks.

Future tasks include analyzing the robustness or vulnerability of existing movement algorithms based on random walks against link rewiring attacks other than the three discussed in this paper and exploring their robustness and vulnerability against other attacks methods beyond link rewiring attacks.

References

[1] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, et al., "Graph Learning: A Survey," IEEE Transactions on Artificial Intelligence, vol. 2, pp. 109–127, 2021.

[2] E. Codling, M. Plank, and S. Benhamou, "Random walks in biology," Journal of the Royal Society, vol. 5, pp. 813–34, Apr. 2008.

[3] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," National Academy of Sciences of the United States of America, vol. 105, pp. 1118–1123, Jan. 2008.

[4] P. Sarkar and A. W. Moore, "Random Walks in Social Networks and their Applications: A Survey," Social Network Data Analytics, pp. 43–77, Jan. 2011.

[5] F. Xia, J. Liu, H. Nie, Y. Fu, et al., "Random Walks: A Review of Algorithms and Applications," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 4, pp. 95–107, Apr. 2020.

[6] L. Lszl, L. Lov, and O. P. Erdos, "Random Walks on Graphs: A Survey," Combinatorics, Paul Erdos is Eighty, pp. 1–46, Jan. 1996.

[7] M. Bui, T. Bernard, D. Sohier, and A. Bui, "Random walks in distributed computing: A survey," in Innovative Internet Community Systems: 4th International Workshop, pp. 1–14, June 2006.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '14, ACM, Aug. 2014.

[9] X. Wang, P. He, J. Zhang, and Z. Wang, "QoS Prediction of Web Services Based on Reputation-Aware Network Embedding," IEEE Access, vol. 8, pp. 161498–161508, Sept. 2020.

[10] D. Ryu, K. Lee, and J. Baik, "Location-Based Web Service QoS Prediction via Preference Propagation to Address Cold Start Problem," IEEE Transactions on Services Computing, vol. 14, pp. 736–746, Apr. 2021.

[11] M. M. Keikha, M. Rahgozar, and M. Asadpour, "Community aware random walk for network embedding," Knowledge-Based Systems, vol. 148, p. 47–54, Feb. 2018.

[12] N. Masuda, M. A. Porter, and R. Lambiotte, "Random walks and diffusion on networks," Physics Reports, vol. 716-717, pp. 1–58, July 2017.

[13] P. Sarkar and A. W. Moore, "Random walks in social networks and their applications: a survey," Social Network Data Analytics, pp. 43–77, 2011.

[14] R. Fitzner and R. van der Hofstad, "Non-backtracking Random Walk," Journal of Statistical Physics, vol. 150, pp. 264–284, Jan. 2013.

[15] R. Yoshitsugu, R. Matsuo, R. Nakamura, and H. Ohsaki, "A Study on Effect of Random Walk Storage Management Scheme on Exploration Efficiency," in Proceedings of the IEICE Society Conference (Society 2023), p. 268, Sept. 2023.

[16] M. Bonaventura, V. Nicosia, and V. Latora, "Characteristic times of biased random walks on complex networks," Phys. Rev. E, vol. 89, p. 012803, Jan. 2014.

[17] K. Kitaura, R. Matsuo, and H. Ohsaki, "Random Walk on a Graph with Vicinity Avoidance," in proceedings of the International Conference on Information Networking (ICOIN 2022), pp. 232–237, Jan. 2022.

[18] G. Ohayon, T. J. Adrai, M. Elad, and T. Michaeli, "Reasons for the Superiority of Stochastic Estimators over Deterministic Ones: Robustness, Consistency and Perceptual Quality," in International Conference on Machine Learning, pp. 26474–26494, July 2023.

[19] O. Denysyuk and L. Rodrigues, "Random Walks on Evolving Graphs with Recurring Topologies," in Proceedings of 28th International Symposium on DIStributed Computing (DISC 14), pp. 333–345, Oct. 2014.

[20] C. Avin, M. Koucký, and Z. Lotker, "How to Explore a Fast-Changing World," in ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming - Volume Part I, pp. 121–132, July 2008.

[21] P. L. Erdos and A. Rényi, "On random graphs. I.," Publicationes Mathematicae Debrecen, vol. 6, pp. 290–297, Nov. 1959.

[22] A. L. Barabasi and R. Albert, "Emergence of Scaling in Random Networks," Science, vol. 286, pp. 509–512, Oct. 1999.

[23] W. K. Chen, "Graph theory and its engineering applications," World Scientific Publishing Company, Feb. 1997.

[24] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, et al., Principles of Robot Motion: Theory, Algorithms, and Implementations. Bradford Books, May 2005.