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, such as 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 target node search and 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 their efficiency against link rewiring attacks?
- How does the vulnerability of random walks on an 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 attach methods for rewiring a link connected to the node visited by the mobile agent.

- Centrality method
  One of the links connected to the current visiting node is rewired to the node with the highest degree centrality in S_t. The idea behind the centrality method is to try to push the mobile agent back to the hub node in subgraph S_t for delaying the target node search and graph exploration.
- Clustering method
  One of the links connected to the current visiting node is rewired to a randomly chosen node in the largest cluster in subgraph S_t. The idea behind the clustering method is to try to confine the mobile agent within the largest cluster in a hope that the mobile agent is not likely to escape from the largest cluster.
- Starting node method
  One of the links connected to the current visiting node is always rewired to the starting node v_0 of the random walk. The idea behind the starting node method is to try to push the mobile agent back to centroid of subgraph S_t.

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 random walk algorithms with history.
- 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 random walk algorithms on a graph. In Section III, we formulate the random walk attack problem and describes 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 V, we summarize this paper and discuss future research directions.

II. Random Walk on a Graph

This section introduces the variants of random walk algorithms applied to graphs, specifically Simple Random Walk (SRW), Non-Backtracking Random Walk (NBRW), k-History Random Walk (k-History RW), Biased Random Walk (BiasedRW), and Vicinity Avoidance Random Walk (VARW).

The Simple Random Walk (SRW) involves an agent moving randomly across nodes without any constraints on memory or revisitation patterns [6, 5]. At each step, the agent selects an adjacent node at random with equal probability and moves to it, allowing for potential revisits to previously visited nodes.

Non-Backtracking Random Walk (NBRW) modifies the SRW by prohibiting the agent from revisiting the last node it came from, thereby incorporating a minimal form of memory to avoid immediate node repetition [14]. This constraint enhances the exploration efficiency by preventing going-and-returning movements.

The k-History Random Walk introduces a memory of size k, enabling the agent to avoid revisiting the last k nodes it has encountered [15]. This method allows for more extensive graph exploration by preventing the agent from revisiting recently visited nodes.

Biased Random Walk (BiasedRW) adjusts the selection probability based on the degree of adjacent nodes, with the transition probability to a neighbor node being proportional to its degree to the power of α [16]. The parameter α dictates the bias towards either highly connected nodes or less connected ones, influencing the exploration pattern.

Vicinity Avoidance Random Walk (VARW) focuses on avoiding not only the recently visited nodes but also their immediate neighbors to a certain extent [17]. This strategy aims to diversify the exploration path by minimizing the likelihood of revisiting both recent nodes and their vicinities.

III. Adversarial Attack against 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 interval 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 attack methods for rewiring a link connected to nodes that the mobile agent is currently visiting: (1) a method that rewires a link to the node with 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 all attack methods, when the mobile agent is visiting node v_t at time t, link (v_t, u) is rewired to link (v_t, u').

Node u is selected to hinder the mobile agent's visit to unvisited nodes. For this purpose, node u is chosen from among the unvisited neighboring nodes of node v_t as the node that is the most difficult to visit from the node v_{t-1}, which the moving agent had visited immediately prior. More specifically, regardless of the attack method, node u is chosen from neighboring nodes of node v_t ($(v_t, u) \in E$), and node u is the one that has not visited by the mobile agent ($u \notin P(t)$) and that the distance between the previous node v_{t-1} and u is the maximum.

The choice of node u', which is connected with node v_t after rewiring, is different among three attack methods.

Algorithm 1 Centrality method

Input: S_t set of nodes visited by the agent up to time t,
v_t currently visiting node, E set of links 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**
    $h \leftarrow$ the 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

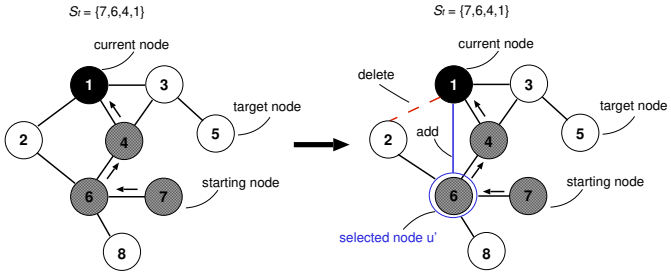


Fig. 1. Centrality method

C. Centrality method

The centrality method (Algorithm 1) is designed to maximize the chances of agent's revisiting previously visited nodes by establishing connections with the node that exhibit the highest degree of centrality within the set of visited nodes.

In the centrality method, the node possessing the maximum degree within S_t is identified and selected as the node u' for rewiring (Fig. 1).

More specifically, letting $d(v)$ be the degree of a node v, then the node u' is selected as

$$\underset{u' \in P(t), (v_t, u') \notin E}{\arg\max} \; d(u'). \tag{1}$$

D. Clustering method

The clustering method (Algorithm 2) is devised to facilitate agents' exploration within highly interconnected segments of visited nodes by establishing links with nodes that are part of the most substantial cluster identified among the visited nodes.

In the clustering method, the subgraph S_t is initially partitioned into clusters employing the Louvain algorithm. Let C represent the ensemble of clusters derived, and c denote an individual cluster within C. The nodes that are members of

Algorithm 2 Clustering method

Input: S_t set of nodes visited by the agent up to time t,
v_t currently visiting node, E set of links in graph G,
$C = \{C_1, C_2, ..., C_n\}$ set of clusters generated from S_t
Step 1:
for each $l \in C(= \{C_1, C_2, ..., C_n\})$ **do**
    $n \leftarrow$ the 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 randomly selected from N_c
if $(u', v_t) \in E$ **then**
    repeat step 2 with another in c

Step 3:
for each $n \in \mathrm{N}(v_t)$ **do**
    $h \leftarrow$ the 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

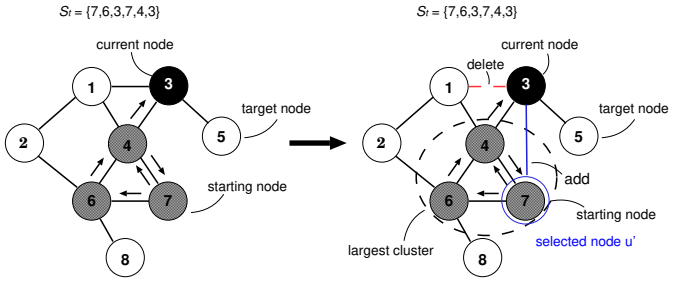


Fig. 2. Clustering method

cluster c are denoted as N_c. A node is then randomly chosen from N_c to serve as the node u' for rewiring (Fig. 2).

Specifically, letting $L(c)$ be the number of nodes within cluster c, the node u' is chosen as

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

E. Starting node method

The starting node method (Algorithm 3) is designed to prolong the exploration duration by establishing connections that lead the mobile agent back to around its starting point, facilitating a return to around its initial position.

In the starting node method, the node that was least recently visited by the mobile agent is chosen as the node u' for rewiring (Fig. 3). Among all visited nodes in subgraph S_t,

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 links 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**
    $h \leftarrow$ the 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

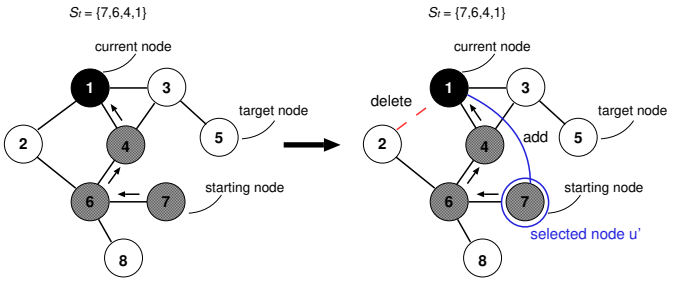


Fig. 3. Starting node method

the node with the oldest visiting record but unconnected with node v_t is selected.

Specifically, let $H_t(v)$ denote the last visited time of node v by the mobile anget, node u' is determined as criterion:

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

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.

In this section, we present simulation results assessing the susceptibility of mobile agents employing random walk

algorithms, such as SRS, NBRW, k-History, BiasedRW, and VARW, to adversarial attacks on an unknown graph. The evaluation focuses on two primary metrics: the time required for target node search (hitting time) and the efficiency of graph exploration (cover time), under the influence of three distinct attack methods.

We conducted simulation experiments across five types of graphs — Erdos and Rényi (ER) [21], Barabási-Albert (BA) [22], ring, 3-regular [23], and Voronoi graphs [24] — maintaining uniformity in size and density (i.e., average degree) across simulations. These graph models are selected for their relevance and frequency of use in simulating real-world network scenarios.

Hereinafter, we provide a brief overview of each graph type utilized in our simulations.

- Erdos and Rényi (ER) [21] graph is a random graph model where nodes are connected by links with a fixed probability. ER graphs serve as fundamental models in graph 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 (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, we generated each graph with 100 nodes. For each graph, link rewiring attacks were initiated once every $A = 50$ or $A = 100$ steps from the start of the mobile agent's movement. We measured the hitting time required for an agent, beginning from a randomly chosen initial node, to reach a randomly selected target node. Additionally, we assessed the average cover time necessary for a mobile agent, starting from a randomly chosen initial node, to visit all nodes at least once. Through simulations conducted under identical conditions across 2,000 trials, we calculated both the average and the 95% confidence interval for the hitting and cover times.

B. Results on average hitting time

Average hitting times across different types of graphs, employing major random walk algorithms under three attack methods, are depicted in Fig. 4 through Fig. 8.

From these results, we observe that irrespective of the graph's topology, the average hitting time for all random walk algorithms generally increases by approximately 2–3 times under the starting point attack method. This significant increase is due to the enhanced probability of agents revisiting sequences of nodes by rerouting from the currently visited node back to the starting node, thereby augmenting the number of steps necessary to reach the target node.

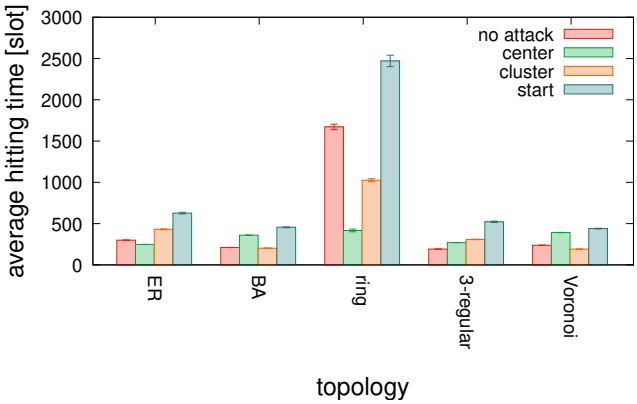
Similarly, for the centrality and cluster methods, across most graph topologies, the average hitting times for random walks either remain consistent or exhibit an increase by a factor of 2–3. Notably, within the ring topology, the average hitting time for SRW decreases by approximately 1/2–1/3. This reduction can be attributed to the centrality method's propensity to revert to nodes exhibiting the highest degree of centrality within the set of previously visited nodes. Such a pattern elevates the likelihood of revisiting nodes with significant centrality, potentially accelerating the exploration of unvisited nodes and, as a result, decrementing the steps needed to reach the designated target node.

C. Results on average cover time

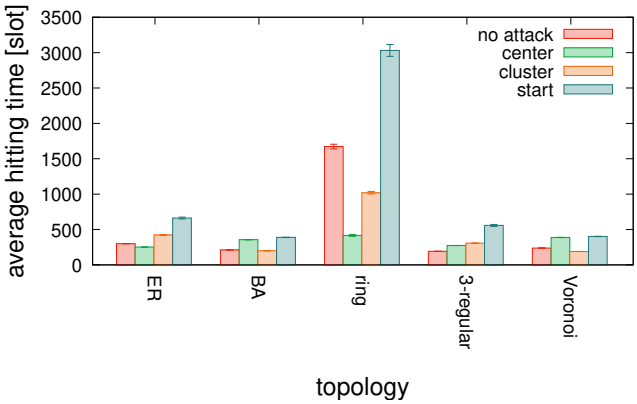
Average conver times across different types of graphs, employing major random walk algorithms under three attack methods, are depicted in Fig. 9 through Fig. 13.

From these results, it becomes clear that irrespective of the graph's topology, similarly (but slightly differently) to the cases of the average hitting time presented above, the average cover times for all random walk algorithms generally exhibit an increase of approximately 2–3 times under both the cluster and starting point attack methods. This escalation can be linked to the cluster method's propensity for causing the agent to remain within a particularly revisited cluster, thereby extending the number of steps needed to explore all nodes comprehensively. Similarly, the starting point method likely increases the step count necessary for complete exploration due to a greater tendency of revisiting sequences of nodes by rerouting back to the starting node.

Conversely, in certain instances, cover times are observed to decrease by approximately 1/2 with the centrality method. This reduction may be explained by the centrality method's inclination to encourage visits to nodes that are significantly distant from the currently visited node and have not been previously explored. This strategy potentially aids in more efficiently covering nodes that are a large number of hops away from the current location, which might otherwise be more challenging to reach in the absence of attacks.

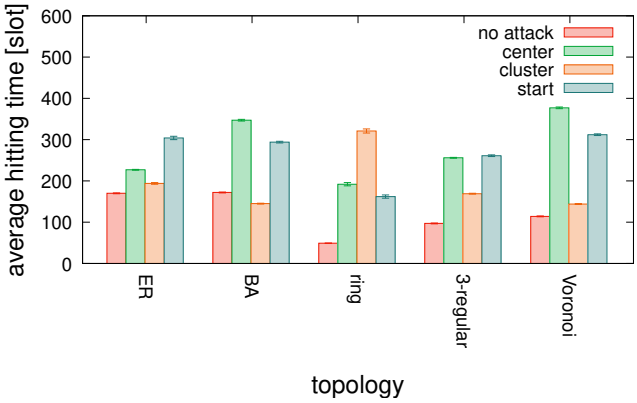


(a) attack interval of $A = 50$

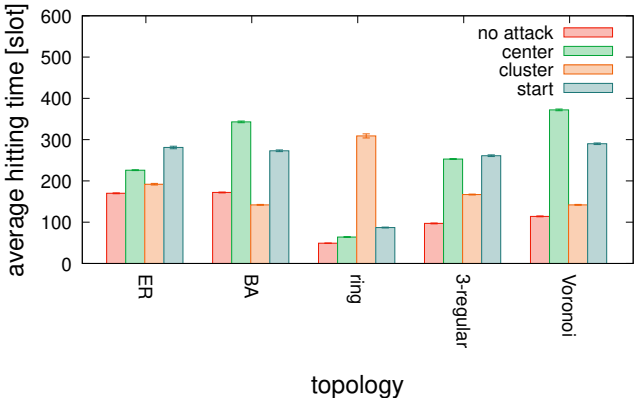


(b) attack interval $A = 100$

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

V. Conclusions

In this paper, we have explored adversarial attacks in the context of target node search and graph exploration using random walks on unknown graphs. We assumed attacks involving the rewiring of a small number of links by an adversary and analyzed the robustness or vulnerability of existing random walk algorithms 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 was found to be the least susceptible to the impact of attacks. Among the three attack methods, the starting node method significantly deteriorated the characteristics of random walks, while the centrality method, under certain conditions, unexpectedly showed an improvement in the characteristics of random walks. walks. Future tasks include analyzing the robustness or vulnerability of random walk algorithms walks against more complicated attacks than the three attack methods studied in this paper.

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.

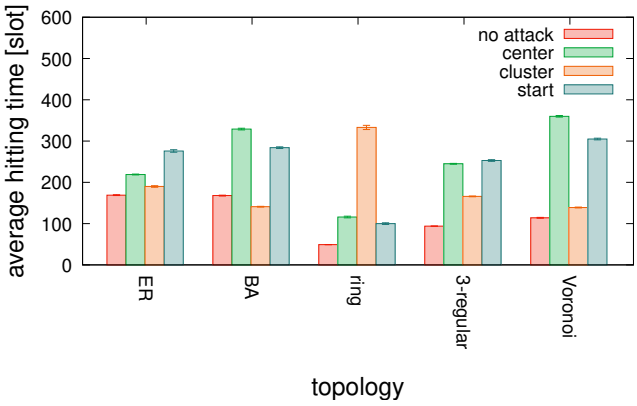
[7] M. Bui, T. Bernard, D. Sohier, and A. Bui, "Random walks in distributed computing: A survey," in *Innovative Internet Community Systems: 4th International Workshop*, pp. 1–14, June 2006.

[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.

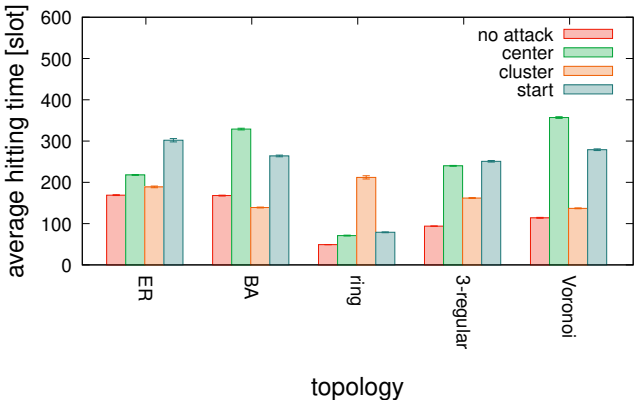
[9] X. Wang, P. He, J. Zhang, and Z. Wang, "QoS Prediction of Web Services Based on Reputation-Aware Network Embedding," *IEEE Access*, vol. 8, pp. 161498–161508, Sept. 2020.

[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,

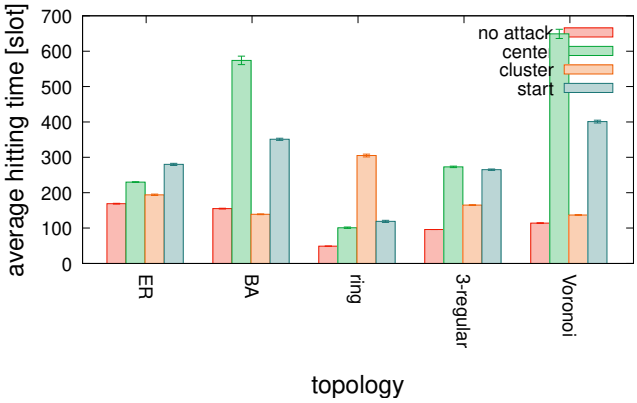


(a) attack interval of $A = 50$

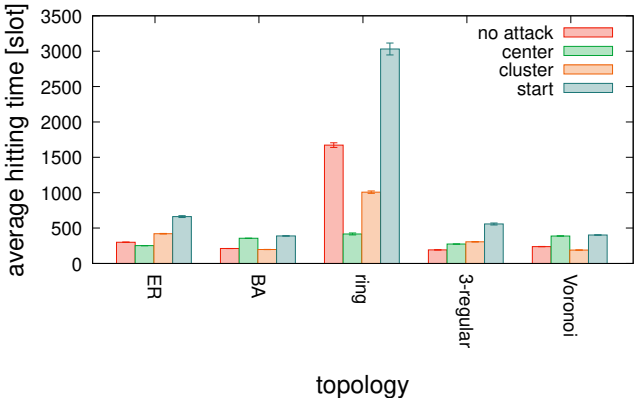


(b) attack interval $A = 100$

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

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.

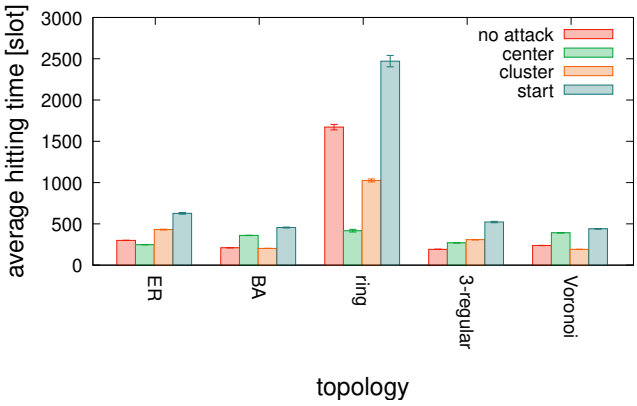
[20] C. Avin, M. Koucký, and Z. Lotker, "How to Explore a Fast-Changing World," in *ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming - Volume Part I*, pp. 121–132, July 2008.

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

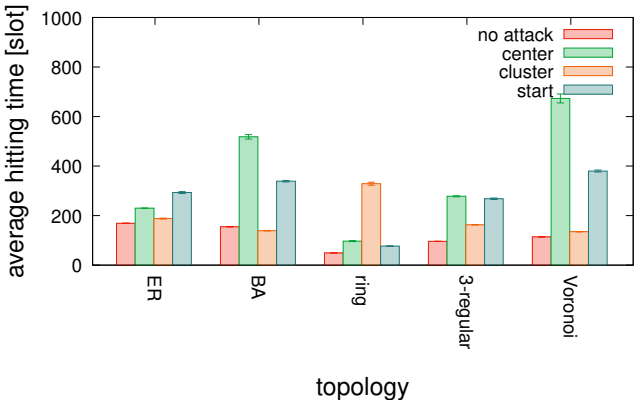
[22] A. L. Barabasi and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, pp. 509–512, Oct. 1999.

[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.

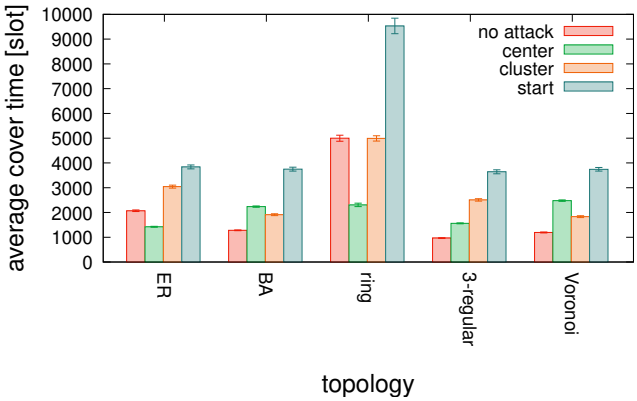


(a) attack interval of $A = 50$

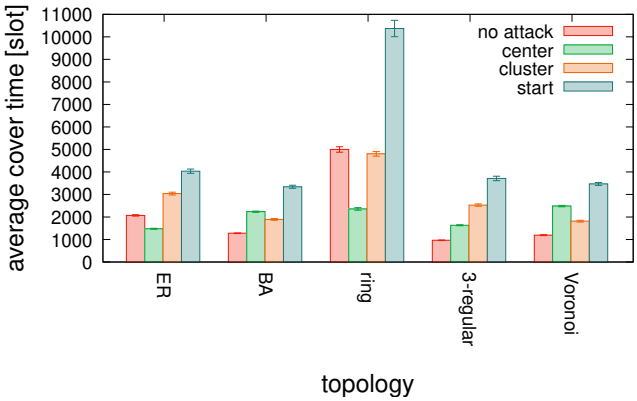


(b) attack interval $A = 100$

Fig. 8. Average hitting time with VARW

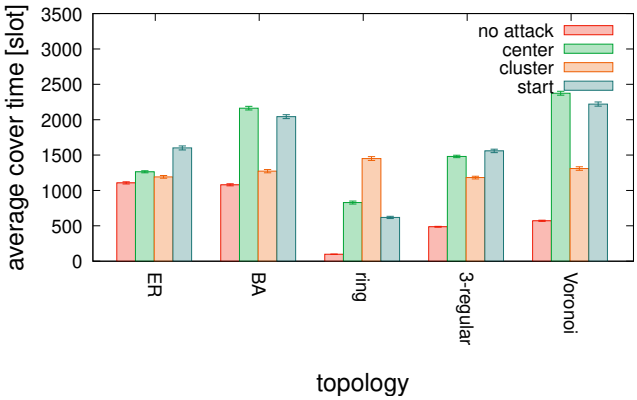


(a) attack interval of $A = 50$

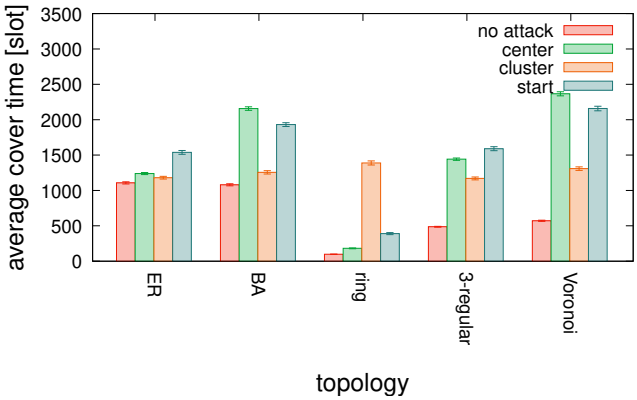


(b) attack interval $A = 100$

Fig. 9. Average cover time with SRW

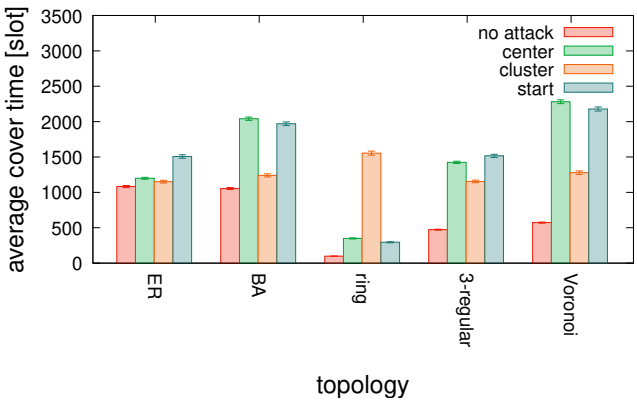


(a) attack interval of $A = 50$

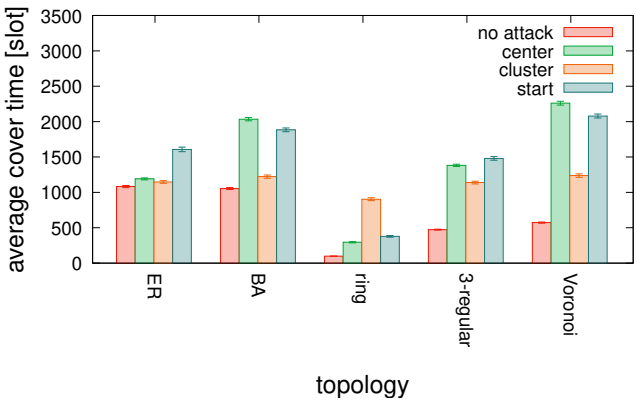


(b) attack interval $A = 100$

Fig. 10. Average cover time with NBRW

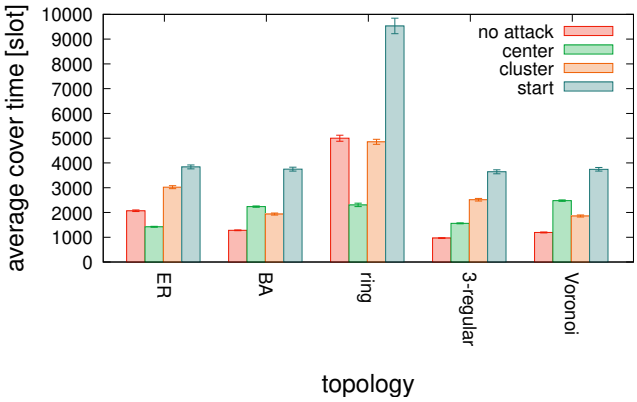


(a) attack interval of $A = 50$

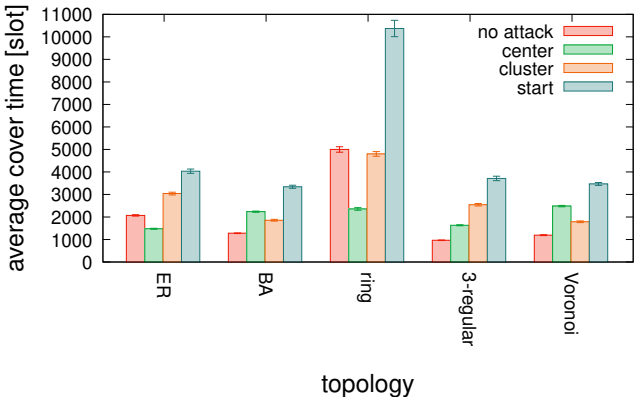


(b) attack interval $A = 100$

Fig. 11. Average cover time with kHistory

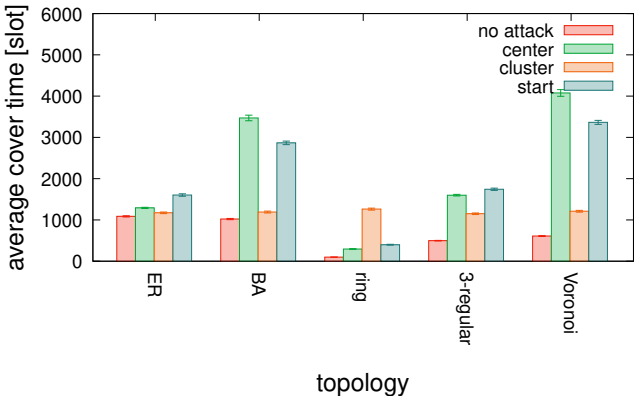


(a) attack interval of $A = 50$

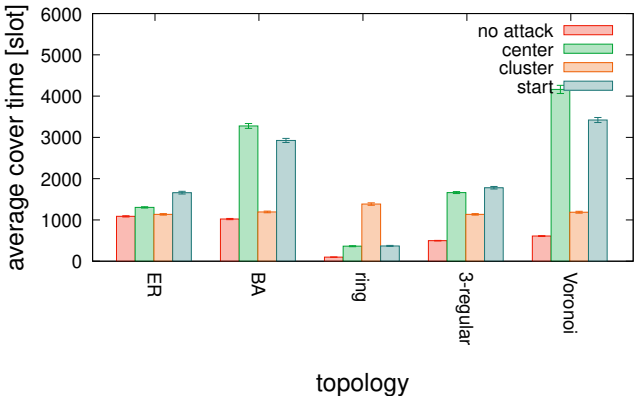


(b) attack interval $A = 100$

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