On the Efficiency of Biased Random Walk with History on a Graph

Ryo Yoshitsugu, Hiroyuki Ohsaki
School of Engineering Kwansei Gakuin University
Email: {ryo-y,ohsaki}@kwansei.ac.jp

***Abstract*—Random walks are used for graph exploration in a wide range of engineering applications. In recent years, their mathematical properties have been actively studied. To optimize random walks, various mobility models have been developed, such as weighted random walk, non-backtracking random walk, self-avoiding random walk, and vicinity-avoiding random walk. Although some of these random walk models can be combined, it has not yet been adequately clarified how to combine those random walk models under different environments. The impact of memory management strategies of random walks with history on different types of graphs has also not been clarified. Therefore, this study focuses on (α, k) random walk, with the transition probability weight parameter α and the self-avoidance memory size parameter k , to clarify the relationship between two parameter settings and its efficiency of node search and graph exploration. Additionally, we investigate how the choice of memory management strategy (i.e., FIFO (First-In, First-Out) and LRU (Least Recently Used)) affects the efficiency of graph exploration. Simulation experiments are conducted to measure the node search time (average hitting time) and the graph exploration time (average cover time) when running different random walks with varying (α, k) parameter settings and memory management strategies on graphs generated with seven different network generation models.**

***Index Terms*—Random Walk, Node Search, Graph Exploration**

I. Introduction

Random walks on a graph are used in a wide range of engineering applications.In recent years, the mathematical aspects of random walks on graphs have been extensively studied.To improve the efficiency of random walks, multiple mobility models have been developed, including weighted random walk, non-backtracking random walk [1], self-avoiding random walk [2], and vicinity-avoiding random walk [3].

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 communication networks. While conventional deterministic algorithms may suffice for small-scale and static communication networks, the landscape changes in large-scale and dynamic networks. 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 rely solely on the local information available to an agent (i.e., a random walker), often represents the sole viable solution.

Random walks on a graph form the basis of numerous critical technologies aimed at enhancing the Quality of Service (QoS) and Quality of Experience (QoE) within large-scale, high-performance communication networks. 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.

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 [4]. In these contexts, they are employed for network sampling [5], node centrality assessment [6, 7], community detection [8, 9], node classification [10] and node embedding [11]. 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.

In this paper, we investigate the (α, k) random walk, which incorporates the transition probability weight parameter α and memory size parameter k.We aim to clarify the relationship between the settings of these two parameters and the efficiency of node search and graph exploration as well as the relationship between memory management strategies and the efficiency of random walks.

The (α, k) random walk is a discrete random walk on a graph conducted by a single agent.It is a generalized mobility model that encompasses weighted random walk and random walk with history. Weighted random walk is a type of mobility model in which transition probabilities are weighted, and random walk with history try to avoid re-visiting previously visited nodes using their memory. In a connected, unweighted, and undirected graph, denoted as $G = (V, E)$, the (α, k) random walk agent avoids nodes visited in the past k steps and determines its next movement based on the α-th power of the weights of the degrees of neighboring nodes.

In this paper, we aim to answer the following research questions.

- How can we combine various random walk movement algorithms on a graph?

- In particular, when combining weighted random walk with random walk that incorporate memory, to what extent does it improve the efficiency of node search and graph exploration through random walks?
- When the agent has limited memory space, how can the utilization of this memory space lead to an efficient random walk?

We conduct simulation experiments to measure the *average hitting time* and the *cover time* when running different random walks with varying (α, k) parameter settings and memory management strategies on graphs generated with seven different network generation models. Two essential characteristics of random walks on a graph are the hitting time and cover time. The hitting time represents the number of steps it takes for an agent to arrive at the target node from the starting node. The cover time is the number of steps it takes for the agent to visit all nodes at least once.

The main contributions of this work are summarized as follows.

- We devised an (α, k) random walk that combines weighted random walk and random walk with history.
- We quantitatively revealed the characteristics of (α, k) random walk, such as node search time and graph exploration time, under various conditions.
- We revealed that the optimal setting of the weight parameter α for the transition probabilities significantly depends on the size of the agent's memory space, particularly in graphs with scale-free properties.
- We proposed a memory management method using LRU and showed that it can improve the efficiency of random walk by approximately 6%.

The rest of this paper is organized as follows. In Section II, we explain random walks on graphs. Section III introduces the (α, k) random walk used in this paper and explains an example of its operation. Section IV investigates the relationship among the two-parameter settings, memory management strategies, and the efficiency of the (α, k) random walk through a large number of experiments. Finally, in Section V, we summarize this paper and discuss future challenges.

II. Random walk on a graph

A random walk on a graph is a probabilistic mobility in which, at each step, the agent randomly selects the next node to transit. A simple random walk on graphs is simple, making them local and memoryless; further, they require minimal processing capabilities at the nodes and rely only on local information. Owing to their minimal memory requirements, they are not significantly affected by changes in the environment, such as dynamic evolving graphs or edge disruptions in the graph. Consequently, they are particularly useful in scenarios in which the environment changes frequently, such as the Internet, peer-to-peer networks (P2P), and wireless ad-hoc networks. However, simple random walk is often local and may not fully utilize the global graph structure, and they have the drawback of not capturing the underlying physical model.

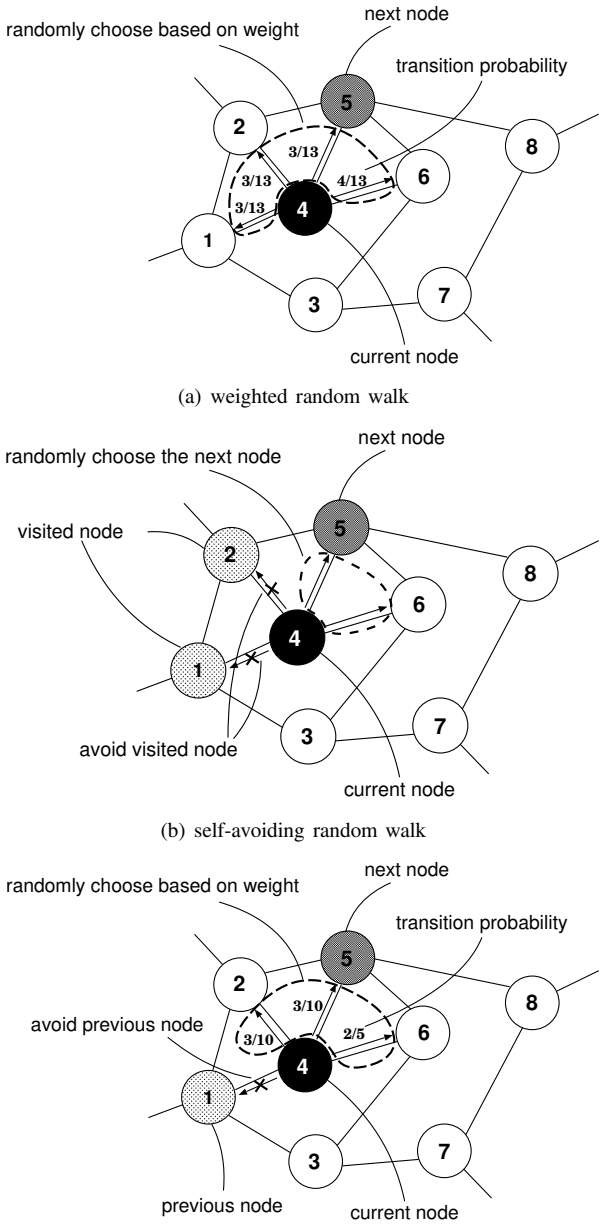


Fig. 1. Operation of random walks

In recent years, studies have been conducted to optimize random walks on graphs, leading to the development of multiple derivative mobility models such as weighted random walk, self-avoiding random walk, and vicinity-avoiding random walk. Weighted random walk introduces a mechanism that assigns weights to transition probabilities in simple random walks. Self-avoiding random walk has a mechanism for avoiding revisiting nodes previously visited in simple random walks as much as possible. Vicinity-avoiding random walk has a mechanism for avoiding visiting recently visited nodes and their adjacent nodes in simple random walks.

Fig. 1 illustrates examples of weighted random walks and

self-avoiding random walks. In weighted random walks, the weights of transition probabilities are based on the degrees of adjacent nodes. With smaller weights, the probability of transitioning increases, making it more likely to transition to nodes with lower degrees. Self-avoiding random walks prohibit revisiting previously visited nodes.

III. (α, k) RANDOM WALK

A. (α, k) *Random Walk*

This section provides an overview of the (α, k) random walk model under consideration. The (α, k) random walk is a discrete random walk on a graph conducted by a single agent; it serves as a generalized mobility model, encompassing weighted random walk and k-history random walk. Weighted random walks have a mechanism that assigns weights to transition probabilities, and k-history random walks record the nodes visited in the past k steps and prohibits revisiting those nodes. The (α, k) random walk model allows for the flexible adjustment of exploration strategies by controlling the parameters α and k. This model enables efficient exploration adapted to different graph characteristics by leveraging α for weighted transition probabilities and k for avoiding revisits based on the history of the past k steps. Consequently, it is expected to improve exploration efficiency in various topologies, such as large scale-free and random graphs.

In a connected, unweighted, undirected graph denoted as $G = (V, E)$, the (α, k) random walk agent avoids nodes visited in the past k steps and determines the next node to transition to based on the probability weighted by the α power of the degrees of adjacent nodes. The transition probability $P(u \to v)$ in the (α, k) random walk is defined by the following equation.

$$P(u \to v) = \begin{cases} \frac{1/\deg(v)^\alpha}{\sum_{w \in N(u) \setminus M} 1/\deg(w)^\alpha} & \text{if } v \in N(u) \text{ and } v \notin M, \\ 0 & \text{if } v \notin N(u) \text{ or } v \in M, \end{cases} \tag{1}$$

Here, the symbols are defined as follows.

- $P(u \to v)$: The transition probability from node u to node v.
- $\deg(v)$: The degree of node v (the number of neighboring nodes of v).
- α: The weighting parameter for the transition probability.
- $N(u)$: The set of neighboring nodes of node u.
- M: The set of nodes visited within the past k steps (the agent's memory).

For example, when $\alpha = 0$ and $k = 1$, the $(0, 1)$ random walk behaves like non-backtracking random walk where the agent moves to one of the neighboring nodes with equal probability while avoiding the node visited in the previous step. On the other hand, when $\alpha = 1$ and $k = 0$, the $(1, 0)$ random walk behaves like weighted random walks where the transition probability is determined based on the weights assigned to the degrees of the neighboring nodes. Furthermore, when $\alpha = 1$ and $k = 1$, the $(1, 1)$ random walk avoids the node visited in the previous step while transitioning to a neighboring node with a probability determined by the weights assigned to the degrees of the nodes.

Fig. 1 (c) illustrates the operation of $(1, 1)$ random walk when the agent is currently positioned at node 4. In this case, the set of neighboring nodes for node 4 is $N(4) = \{2, 5, 6, 1\}$. Since $k = 1$, the agent refers to its memory, which contains the node visited in the previous step, $M = \{1\}$ .Consequently, the actual transition candidates are determined as $N(4) \setminus M = \{2, 5, 6\}$, excluding node 1 from the possible transitions. Furthermore, when $\alpha = 1$, the transition probability $P(u \to v)$ to a candidate node is calculated using Eq. (1). This transition probability is calculated based on the degrees of the candidate nodes, with nodes of lower degrees being assigned higher probabilities.

The degrees of nodes 2, 5, and 6 are as follows.

$$\deg(2) = 4 \quad \deg(5) = 4 \quad \deg(6) = 3$$

Using these values, the transition probability to node 2 is calculated as follows.

$$\begin{aligned} P(4 \to 2) &= \frac{1/\deg(2)^\alpha}{\sum_{w \in N(4) \setminus M} 1/\deg(w)^\alpha} \\ &= \frac{1/4}{1/4 + 1/4 + 1/3} = \frac{3}{10} \end{aligned}$$

Similarly, the transition probabilities to nodes 5 and 6 are calculated as follows.

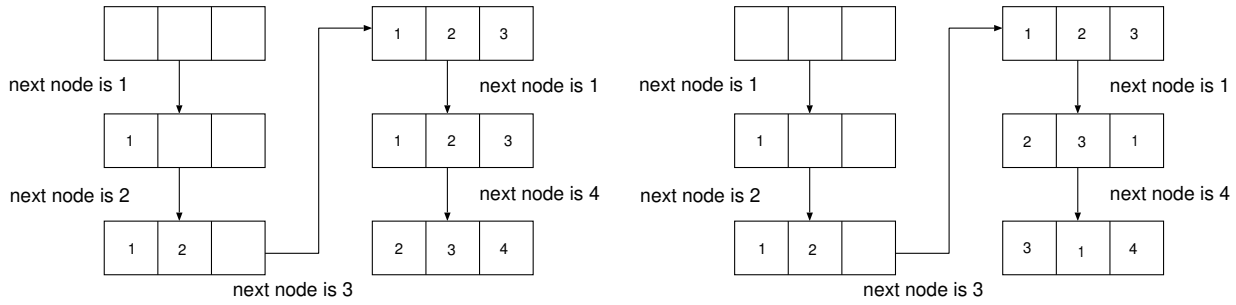
$$P(4 \to 5) = \frac{3}{10} \quad P(4 \to 6) = \frac{2}{5}$$

From this, it can be seen that node 6 is selected with the highest probability compared to the other nodes. On the other hand, nodes 2 and 5 are selected with the same probability. Thus, the (α, k) random walk achieves flexible graph exploration by combining degree-based weighting with memory-based self-avoidance.

B. *Memory storage management strategies*

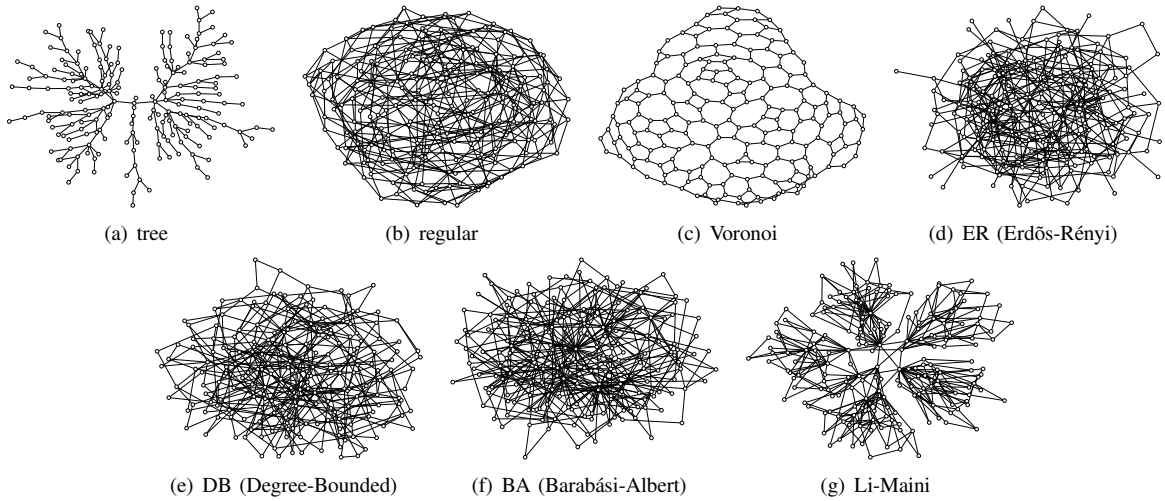
Memory storage is an area in the agent where visited node data can be stored. When the size of the memory storage is 1, the agent can store the ID of one visited node. Memory storage management strategies are potential factors that significantly influence the exploration behavior of agents in (α, k) random walks. How visit histories are managed is expected to affect not only the prevention of revisits but also the efficiency of reaching unexplored areas. However, the impact of memory storage management on exploration efficiency remains largely unexplored.Thus, investigating appropriate management strategies is an important task that may contribute to improving exploration efficiency. For instance, appropriately utilizing memory storage may reduce unnecessary revisits and allow agents to efficiently reach unvisited nodes. To evaluate these potential effects, this study compares different memory management strategies, such as FIFO and LRU, and examines their effectiveness.

This study focuses on two methods: FIFO and LRU. In the FIFO method, the node IDs stored in memory storage are



(a) FIFO (b) LIFO

Fig. 2. Operation of FIFO and LIFO



(a) tree (b) regular (c) Voronoi (d) ER (Erdõs-Rényi)

(e) DB (Degree-Bounded) (f) BA (Barabási-Albert) (g) Li-Maini

Fig. 3. Examples of 7 types of graphs (number of nodes: 200)

managed in the order they were added, with the oldest entries being removed first when memory is full. In contrast, the LRU method removes the node IDs that have not been accessed for the longest period, ensuring that the most recently accessed nodes remain in memory.

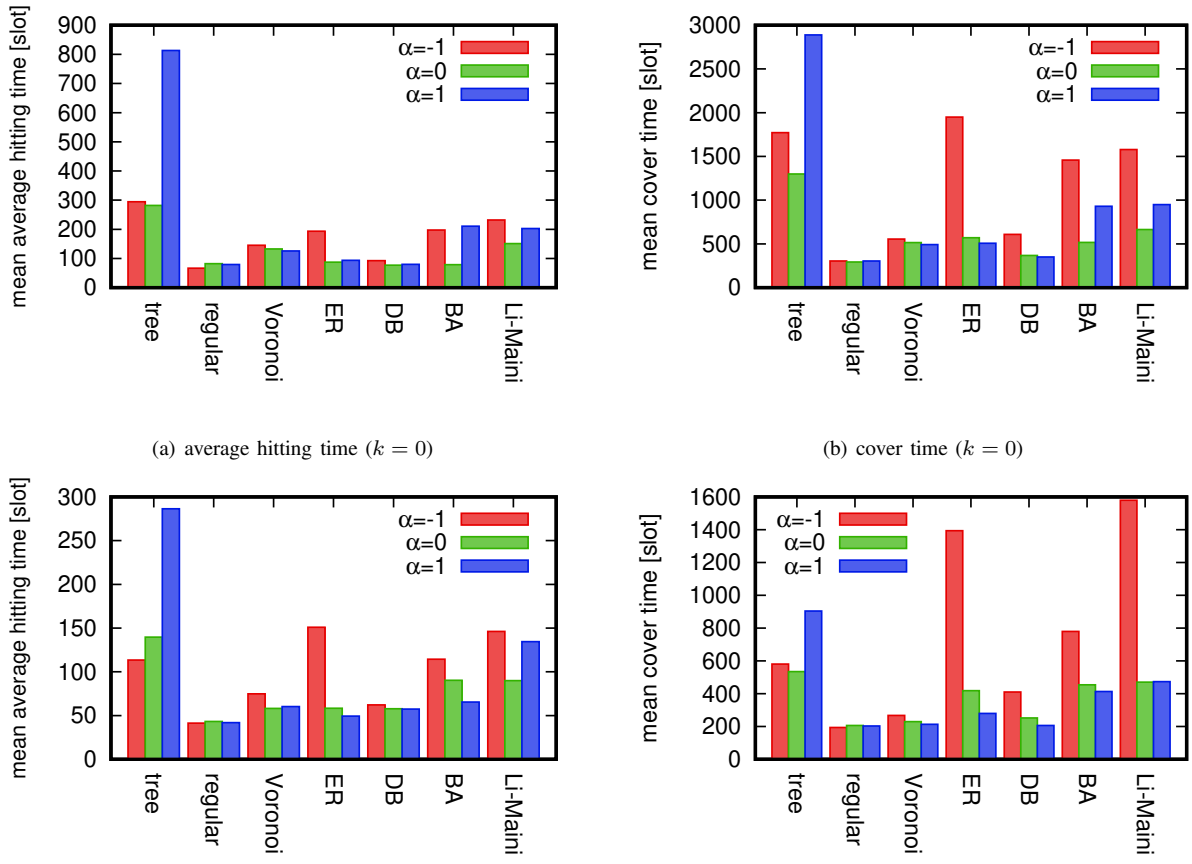
Fig. 2 shows an example of memory storage for a size of 3 by the FIFO and LRU methods. In the FIFO method, if the ID of the newly visited node is already present in memory (which happens when the agent has stored the IDs of all neighboring nodes of the current node), the ID stored the longest is removed, and the order of IDs is preserved. By contrast, the LRU method removes the ID of the node that has not been accessed for the longest time, ensuring that recently accessed nodes remain in memory.

IV. Experiments

A. Experimental design

To investigate the relationship between the weight parameter α and memory storage size k used for self-avoidance in (α, k) random walks and their impact on the efficiency of graph exploration, we analyze the characteristics (hitting time and cover time) of (α, k) random walks by varying the values of α and k in graphs generated by multiple network generation models. Furthermore, we focus on $(\alpha, 3)$ random walks when the memory storage size is set to 3. To understand the effect of two memory management methods, FIFO and LRU, on the graph exploration efficiency, we analyze the characteristics of $(\alpha, 3)$ random walks by changing the memory management method in graphs generated by multiple network generation models.

As network generation models, we used three simple graphs (tree, Voronoi, and random regular graph), two random graphs (ER (Erdõs-Rényi [12]) and DB (Degree-Bounded [13]), and two scale-free network generation models (BA (Barabási-Albert [14]) and Li-Maini [15]). Fig. 3 shows examples of the seven types of networks with 200 nodes. The tree graph has a structure where no cycles exist, and all nodes are connected in a linear or hierarchical manner, typically branching out from a single root. The Voronoi graph is a structure where the plane is divided into cells based on a set of specified points, with each cell forming the nearest region to its corresponding center point. A random regular graph is a graph in which all nodes have the same degree, with edges randomly distributed while maintaining a specific degree distribution. The ER (Erdõs-Rényi) model generates a structure where edges between nodes are randomly placed with a specified probability, producing a disordered and unpredictable graph. The DB (Degree-Bounded) model generates a random graph in



(a) average hitting time ($k = 0$)

(b) cover time ($k = 0$)

(c) average hitting time ($k = 1$)

(d) cover time ($k = 1$)

Fig. 4. Relationship between network topology and characteristics of (α, k) random walk (network size: 50)

which the degree of each node is constrained within a specified range. The BA (Barabási-Albert) model generates a scale-free network by preferentially attaching new nodes to existing high-degree nodes. The Li-Maini model evolves dynamically, where nodes form links across multiple communities, creating a network with community structures.

In the generated graphs, we measured the number of steps it took for an agent to arrive at a randomly selected destination node, excluding the starting node (i.e., hitting time). Similarly, we measured the number of steps it took for the agent to visit all nodes except the starting node (i.e., cover time). By conducting 100 trials of random walks starting from the same node in a graph, we calculated the average hitting time and cover time.

B. Performance evaluation of (α, k) random walk

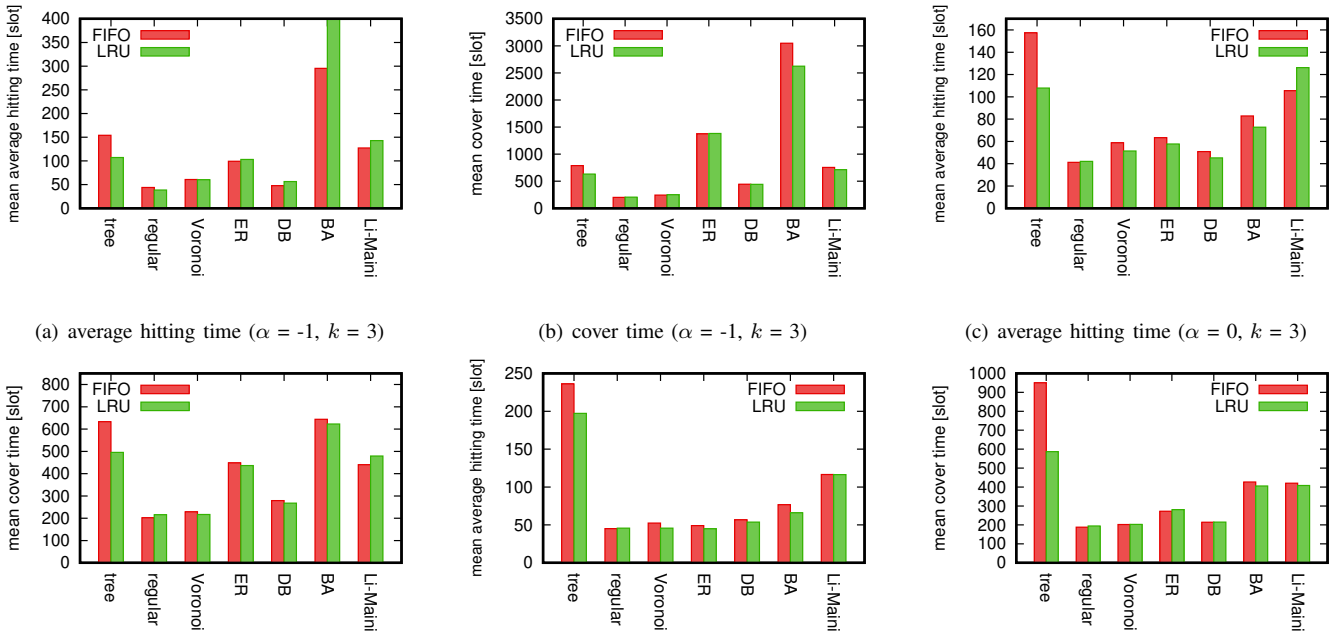
The hitting time and cover time of the (α, k) random walk when varying the transition probability weight parameter α are shown in Fig. 4. The hitting time and cover time of the (α, k) random walk when changing the memory storage management method are presented in Fig. 5.

These results indicate that the optimal transition probability weight parameter α, varies based on the size of the memory space the agent possesses when the graph topology exhibits scale-free properties. The interaction between α and k was found to have a significant impact on the agent's exploration strategy. Particularly in scale-free networks, where the degree distribution is highly skewed, an appropriate combination of α and k is suggested to enhance the efficiency of the agent's exploration.

Furthermore, managing the agent's limited memory space with LRU instead of FIFO was found to enhance the efficiency of random walks. This effect was particularly pronounced in the case of a graph topology resembling a tree. The LRU memory management strategy is considered particularly efficient because it prioritizes avoiding nodes that were recently visited, thereby facilitating transitions based on new information during exploration. This approach allows past visitation data to be quickly incorporated into new exploration strategies, enabling efficient graph traversal. This effect is especially pronounced in tree graph topologies.

V. Conclusion

We conducted random walks with different (α, k) parameter settings on graphs with diverse topological structures and measured the hitting time and cover time. We also measured



(a) average hitting time (α = -1, k = 3)

(b) cover time (α = -1, k = 3)

(c) average hitting time (α = 0, k = 3)

(d) cover time (α = 0, k = 3)

(e) average hitting time (α = 1, k = 3)

(f) cover time (α = 1, k = 3)

Fig. 5. Relationship between network topology and characteristics of (α, k) random walk with different memory storage management methods (network size: 50)

these times when running (α, k) random walks with different memory storage management methods (i.e., FIFO and LRU).

This study demonstrates that, to enhance random walk efficiency, it is not just important to optimize individual parameters, but also to understand and leverage the interactions between these parameters. By adopting this perspective, we lay the groundwork for creating universal exploration algorithms that can adapt to a wide range of network structures.

Future challenges include exploring appropriate (α, k) settings depending on the graph structure, investigating suitable memory storage management methods according to the graph structure, and proposing applications utilizing (α, k) random walks.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 24K02936.

References

[1] R. Fitzner and R. van der Hofstad, "Non-backtracking random walk," *Journal of Statistical Physics*, vol. 150, pp. 264–284, Dec. 2013.

[2] V. M. L. Millán, V. Cholvi, L. López, and A. F. Anta, "A model of self-avoiding random walks for searching complex networks," *Journal of Networks*, vol. 60, pp. 71–85, Sept. 2012.

[3] K. Kitaura, R. Matsuo, and H. Ohsaki, "Random walk on a graph with vicinity avoidance," in *Proceedings of the 36th IEEE International Conference on Information Networking (ICOIN 2022)*, pp. 232–237, Jan. 2022.

[4] J. Pan, F. Jiang, and J. Xu, "Influence maximization in social networks based on non-backtracking random walk," in *Proceedings of the 2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, pp. 260–267, June 2016.

[5] Y. Li, Z. Wu, S. Lin, H. Xie, M. Lv, Y. Xu, and J. C. Lui, "Walking with perception: Efficient random walk sampling via common neighbor awareness," in *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 962–973, Apr. 2019.

[6] M. E. Newman, "A measure of betweenness centrality based on random walks," *Journal of Social networks*, vol. 27, pp. 39–54, Sept. 2005.

[7] J. D. Noh and H. Rieger, "Random walks on complex networks," *Journal of Physical review letters*, vol. 92, p. 118701, Mar. 2004.

[8] M. Okuda, S. Satoh, Y. Sato, and Y. Kidawara, "Community detection using restrained random-walk similarity," *Journal of IEEE transactions on pattern analysis and machine intelligence*, vol. 43, pp. 89–103, July 2019.

[9] P. T. H. Duong *et al.*, "Detecting communities in large networks using the extended walktrap algorithm," in *Proceedings of the 2022 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 100–105, IEEE, Dec. 2022.

[10] S. Chanti, A. Angadi, S. Muppidi, and V. Rachapudi, "Influential node-based random walk graph embedding with graph neural networks in complex graphs," in *Proceedings of the 2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, pp. 27–35, IEEE, Oct. 2023.

[11] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 855–864, Association for Computing Machinery, Aug. 2016.

[12] P. Erdös and A. Rényi, "On random graphs I.," *Journal of Mathematicae*, vol. 6, pp. 290–297, Nov. 1959.

[13] K. Yamashita, R. Nakamura, and H. Ohsaki, "A study on robustness of complex networks against random node removals," in *Proceedings of the 42nd IEEE Signature Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2018)*, pp. 966–969, July 2018.

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

[15] C. Li and P. K. Maini, "An evolving network model with community structure," *Journal of Physics A: Mathematical and General*, vol. 38, pp. 9741–9749, Oct. 2005.