Received 5 September 2020; accepted 25 September 2020. Date of publication 8 October 2020; date of current version 3 November 2020.
*Digital Object Identifier 10.1109/OJCOMS.2020.3028871*

# Detouring Skip Graph: Efficient Routing via Detour Routes on Skip Graph Topology

TAKESHI KANEKO [1], RYOHEI BANNO [1,2] (Member, IEEE), KAZUYUKI SHUDO [1] (Member, IEEE), KOTA ABE [3,4] (Member, IEEE), AND YUUICHI TERANISHI [5,6] (Member, IEEE)

[1] Department of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo 152-8552, Japan
[2] Department of Information and Communications Engineering, Kogakuin University, Tokyo 192-0015, Japan
[3] Graduate School of Engineering, Osaka City University, Osaka 558-8585, Japan
[4] Social Innovation Unit, National Institute of Information and Communications Technology, Tokyo 184-8795, Japan
[5] Network System Research Institute, National Institute of Information and Communications Technology, Tokyo 184-8795, Japan
[6] Cybermedia Center, Osaka University, Osaka 558-8585, Japan

CORRESPONDING AUTHOR: T. KANEKO (e-mail: kaneko.t.ay@m.titech.ac.jp)

This work was supported in part by Japan Society for the Promotion of Science KAKENHI under Grant 19K20253, in part by New Energy and Industrial Technology Development Organization (NEDO), and in part by SECOM Science and Technology Foundation.

**ABSTRACT** Skip graph is a distributed data structure that provides a scalable structured overlay network by routing in logarithmic time for resource location and dynamic node addition/deletion. However, most of the routing paths are quite longer than the shortest paths because each node in the network knows only its neighbors, rather than the global topology. In general, long routing paths lead to the high latency and the low fault tolerance. Herein, we propose Detouring Skip Graph, which performs more efficient routing through the use of detour routes. It does not require construction of extra links or modification of its topology; thereby, it shortens the paths without additional costs while maintaining the advantages of Skip Graph. Our evaluation experiments show that the proposed method tends to shorten the paths considerably, and in particular, that the average path length is approximately 20%–30% shorter than that of Skip Graph.

**INDEX TERMS** Detour route, routing algorithm, skip Graph, structured overlay.

## I. INTRODUCTION

An OVERLAY network is an application-level logical network built on an existing network such as the Internet. Each node in overlay networks is organized in a decentralized manner and flexibly adapts to the dynamic underlying network [1]. Especially a structured overlay constructs an autonomous distributed network according to a specific data structure or some protocols; thereby providing reachability to target nodes, high scalability, high fault tolerance, and some useful functions. Owing to these properties, application to a variety of large-scale distributed systems has been proposed; e.g., distributed key/value stores [2], video streaming [3], and online games [4]. In recent years, application to the fields of IoT and Blockchain is also expected [5], [6].

Skip Graph [7] is a distributed data structure that provides a scalable structured overlay managing pairs of a key and a value. By hashing keys, it works as a distributed hash table (DHT) and archives good load balancing for data management. On the other hand, even if keys are not hashed, it constructs a balanced topology that performs routing in logarithmic time for resource location and dynamic node addition/deletion, unlike some typical DHTs [8], [9]. Moreover, Skip Graph without hashing supports range queries [10], [11] as a result of preserving the order of keys.

However, it is still a challenge for Skip Graph to ensure that each node takes full advantage of the existing links. The routing paths tend to be quite longer than the shortest paths because it knows only its neighbors, rather than the global topology. In general, an overlay network with long routing paths leads to the high latency and the low fault tolerance. In a homogeneous environment such as a local area network and a cloud network, a routing path length dominates the latency. In contrast, note that locality awareness [12], [13], [14], which considers the proximity of the underlying network, is also important in a non-homogeneous environment on the Internet. Regarding fault tolerance, it is effective to use the stabilization methods [15]. However, the probability of encountering a fault in a routing process increases as the path length increases.

Since most application mentioned above of overlay networks requires the high responsiveness and high reliability, shortening routing paths is a critical demand for overlays including Skip Graph. We propose an extension of Skip Graph which is called Detouring Skip Graph. It shortens the path lengths through a more efficient use of the existing links while maintaining the advantages of Skip Graph. Specifically, its routing algorithm is different from that of Skip Graph in two ways: each node 1) utilizes detour routes and 2) traverses adjacent nodes from its maximum level.

This article is an extended version of our previous work [16]. The main differences are comprehensive comparison experiments and mathematical analysis. This article provides comparisons in maximum path lengths and path length distribution, an additional comparison target `searchNLIOp`, and additional evaluation scenarios (in Section IV); and proves correctness and complexity of the proposed method (in Sections III-E and III-F). This article also provides all the pseudocodes including `searchDSGOp` (Algorithm 3). The rest of this article is organized as follows. Section II presents an overview of Skip Graph and the related work on shortening path lengths. Section III presents Detouring Skip Graph in detail. Section IV presents the evaluation experiments for the proposed method and the results. Finally, Section V presents the conclusion of this study.

## II. RELATED WORK

### A. SKIP GRAPH

Skip Graph is a distributed data structure designed based on Skip List [17], and each node belongs to multiple sorted doubly linked lists. Fig. 1 shows an example of a topology of Skip Graph. Each node has a key in a totally ordered set and a random string called membership vector (MV), which plays a key role in constructing the topology of Skip Graph. In the figure, the alphabet that are elements of MV is $\{0, 1\}$. In a linked list at level $l$, the leading $l$ digits of the MV of every node is the same as that of all others. Particularly at level 0, all nodes belong to one linked list. Therefore, each node belongs to $O(\log n)$ linked lists. Further, by using the same method as Skip List, Skip Graph achieves a routing path length of $O(\log n)$ for a query to search a key $k_{target}$.

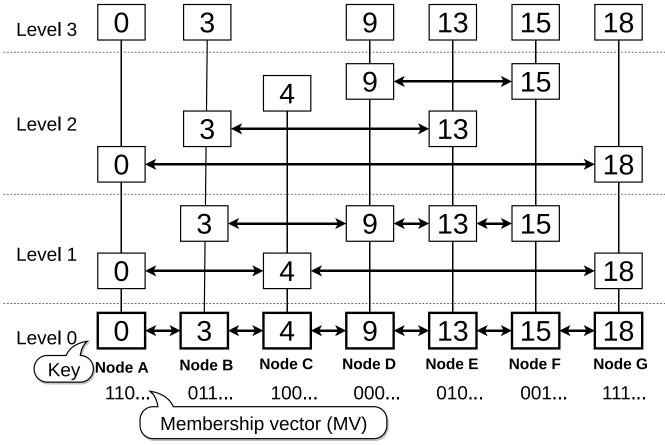


**FIGURE 1.** An example of Skip Graph.

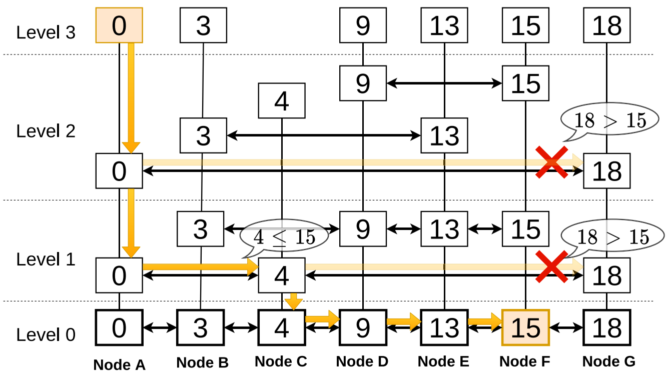


**FIGURE 2.** A `searchOp` routing to search a node whose key is 15 from a node *A*.

A detailed explanation of the routing algorithm is presented below, where it is assumed that the keys in the linked lists are sorted in ascending order from left to right.

Algorithm 4 is a pseudocode of routing algorithm for search queries of Skip Graph. Suppose a node $v_{current}$ in Skip Graph is receiving a query `searchOp` to search a node that has a specific key. The query has three information $(v_{start}, k_{target}, l_{prev})$: a start node $v_{start}$, a target key $k_{target}$, and the level $l_{prev}$ at which the previous node sends the query. If $v_{current}.key$, the key of $v_{current}$, is equal to $k_{target}$, $v_{current}$ sends a query `foundOp` to $v_{start}$ since it means that $v_{current}$ is the target node. If $v_{current}.key < k_{target}$, $v_{current}$ traverses the right adjacent nodes at the levels from $l_{prev}$ in descending order and sends a search query `searchOp` to the first adjacent node $v_{next}$ where $v_{next}.key \leq k_{target}$. If $v_{current}.key > k_{target}$, $v_{current}$ traverses the left adjacent nodes and sends a search query `searchOp` to the next node in a similar manner. If $v_{current}.key \neq k_{target}$ and $v_{current}$ cannot find such a next node, $v_{current}$ sends a query `notFoundOp` to $v_{start}$ since it means that there are no target nodes in the topology.

Fig. 2 shows a routing process when a query to search a node whose key is $k_{target} = 15$ is issued at a node $A$ on the condition that each node follows the above method in the topology shown in Fig. 1. Then, the key sequence of the nodes on the routing path is $(0, 4, 9, 13, 15)$, and the path length is 4. This routing specifies the target node is a node $F$.

### B. SHORTENING PATH LENGTHS

The path length is 4 in Fig. 2, however, there are shorter paths in reality. For instance, if the node $A$ chooses a node $G$ at level 2 instead of a node $C$ at level 1 as the next node, the key sequence of the nodes on the path would be (0, 18, 15), and the path length would be 2 (which is shorter than 4). Thus, the routing of Skip Graph is inefficient in that it cannot fully utilize the existing links. Therefore, various approaches have been proposed for shortening the path lengths.

In the above example, at the route from the node $A$ to the node $G$, the magnitude relationship between the key of the current node and the target key $k_{target}$ is reversed. Routes like this are hereinafter referred to as "detour routes." The proposed method (described later in detail) utilizes detour routes. It is similar to the method proposed by Higuchi *et al.* [18] in terms of use of detour routes. However, the subject of their method is not ordinary Skip Graph but Skip Graph whose topology is balanced by means of using linear hashing preserving the order.

There are already several methods to shorten path lengths of Skip Graph routing: e.g., methods that involve construction of extra links in the original Skip Graph and appropriate routing on the topology including the links [19], [20], and methods that reconstruct or refine the unbalanced topology resulting from randomly generated MVs into the ideal topology [21], [22], [23], [24]. However, the construction of extra links and the modification of the topology lead to an increase in necessary transfer messages and maintenance costs. Our proposed method has an advantage in that it shortens the path lengths without such additional costs.

## III. PROPOSED METHOD

The main idea of Detouring Skip Graph is the utilization of detour routes. Moreover, the idea can be combined with the technique of traversing from maximum levels. This section presents these two techniques and the details of Detouring Skip Graph whose routing algorithm combines them.

In the following, let $K$ be a set of keys. To simplify, we assume that $K$ is a subset of real numbers.[1] Thus, arithmetic operations and absolute values are well-defined on $K$. Moreover, we also assume that there is at most one node for each key. In practice, the uniqueness of keys is not always guaranteed because replication [26], [27] is often used to provide data availability and fault tolerance. A simple way to eliminate duplicate keys is to add some random bits on the right side of identical keys [28].

### A. UTILIZING DETOUR ROUTES

The routing algorithm of Detouring Skip Graph utilizes detour routes. The idea is based on the following argument.

Fig. 3 shows a part of the topology of Fig. 1. Let $v_{next}$ and $v_{lower}$ be the right neighbors of a node $v_{current}$ at level 1 and 0, respectively. Now, suppose $v_{current}$ is receiving a query to search a node whose key equals $k_{target} \in K$ where $9 \leq k_{target} < 18$. In a situation where $v_{current}$ follows the routing algorithm of Skip Graph, it selects $v_{lower}$ as the next node since $v_{next}.key > k_{target}$ and $v_{lower}.key \leq k_{target}$. If $k_{target} = 15$, the key sequence of the nodes on the path would be (4, 9, 13, 15), and the path length would be 3. However, in a situation where it selects $v_{next}$ as the next node, i.e., it uses the detour route, the key sequence would be (4, 18, 15), and the path length would be 2. The path length of the latter is shorter than that of the former. The effectiveness of such detour routes depends on the position of the target key; if $k_{target} = 13$, the path length of using the detour route would be longer than that of using the ordinary route.

1. As a mathematical fact, any countable totally ordered set can be order embedded into the set of rational numbers [25], which is a subset of real numbers.

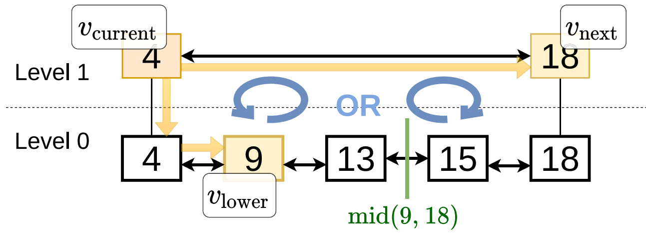


FIGURE 3. Should the node $v_{current}$ select the detour route?

As shown in Fig. 3, it can be determined from the center of the node sequence at the lower level (level 0 in Fig. 3) whether $v_{current}$ should select $v_{next}$ or $v_{lower}$ to shorten the path length. This section presents the routing algorithm that each node determines the next node based on a detour criterion at each level.

Algorithm 1 is a pseudocode of this algorithm. Herein, $v_{current}$ traverses the adjacent nodes in the same way as Skip Graph when a node $v_{current}$ receives a query `searchDROp`. However, if $v_{current}$ judges that using a detour route is better than not using it, $v_{current}$ selects the end node of the detour route as the next node. Function $closeToRight(k_{target}, k_{left}, k_{right})$ can be used to make this judgment. Let $I_{right}$ be $\{k \in K | k \geq k_{right}\}$. The function returns *true* if the signed distance from $k_{target}$ to $I_{right}$ is smaller than that from $mid(k_{left}, k_{right})$ to $I_{right}$, i.e., $k_{right} - k_{target} < k_{right} - mid(k_{left}, k_{right})$; otherwise, it returns *false*. Intuitively, it means that $k_{target}$ is closer to $k_{right}$ than $mid(k_{left}, k_{right})$. Further, $mid$ is a design parameter defined as a function before the construction of the topology to satisfy that the following (1) and (2) are approximately equal for the set $V$ of all participating nodes at any point in time and any $k_1, k_2 \in K$ $(k_1 \leq k_2)$;

$$\#\left\{k \in K \,\middle|\, k_1 \leq k \leq k_{mid} \wedge^{\exists} v \in V, v.key = k\right\} \quad (1)$$

$$\#\left\{k \in K \,\middle|\, k_{mid} \leq k \leq k_2 \wedge^{\exists} v \in V, v.key = k\right\} \quad (2)$$

where $k_{mid} = mid(k_1, k_2)$. Equation (1) means the number of nodes $v$ where $v.key \in [k_1, k_{mid}]$, and (2) means the number of nodes $v$ where $v.key \in [k_{mid}, k_2]$. Thus, $mid(k_1, k_2)$ implies the center between $k_1$ and $k_2$ for the key distribution of the

**Algorithm 1:** `searchDROp` in Node $v_{current}$

```
/* utilizing Detour Routes */
upon receiving ⟨searchDROp, v_start, k_target, l_prev⟩ then
    if v_current.key = k_target then
        send ⟨foundOp, v_current⟩ to v_start;
        return;
    else if v_current.key < k_target then
        for l_current ← l_prev downTo 0 do
            v_next ← v_current.neighbors[R][l_current];
            if v_next = ⊥ then
                continue;
            if v_next.key ≤ k_target then
                send ⟨searchDROp, v_start, k_target, l_current⟩ to v_next;
                return;
            else if l_current > 0 then
                v_lower ← v_current.neighbors[R][l_current − 1];
                if closeToRight(k_target, v_lower.key, v_next.key) then
                    send ⟨searchDROp, v_start, k_target, l_current⟩ to v_next;
                    return;
    else
        for l_current ← l_prev downTo 0 do
            v_next ← v_current.neighbors[L][l_current];
            if v_next = ⊥ then
                continue;
            if v_next.key ≥ k_target then
                send ⟨searchDROp, v_start, k_target, l_current⟩ to v_next;
                return;
            else if l_current > 0 then
                v_lower ← v_current.neighbors[L][l_current − 1];
                if ¬ closeToRight(k_target, v_next.key, v_lower.key) then
                    send ⟨searchDROp, v_start, k_target, l_current⟩ to v_next;
                    return;
    send ⟨notFoundOp, v_current⟩ to v_start;
function closeToRight(k_target, k_left, k_right)
    k_mid ← mid(k_left, k_right);
    return k_mid < k_target;
```

participating nodes. We give some examples of defining *mid*. Let $v.key$, the key of a node $v$, be regarded as a random variable.

If

$$K = \{0, 1, \ldots, n-1\} \text{ and } P\{v.key = k\} = \frac{1}{n}, \tag{3}$$

then

$$mid(k_1, k_2) := \frac{k_1 + k_2}{2}. \tag{4}$$

In Fig. 3 and for this *mid* definition, $mid(9, 18) = \frac{27}{2}$ holds, and thus $closeToRight(k_{target}, 9, 18) \iff \frac{27}{2} < k_{target}$. This means the target node is located to the right of the estimated center if $\frac{27}{2} < k_{target}$, and is located to the left if $k_{target} \leq \frac{27}{2}$.

Next, if

$$K = [\alpha, \beta] \text{ and } P\{v.key \leq k\} = \int_{\alpha}^{k} c\kappa^{\gamma}\, d\kappa \tag{5}$$

with constants $\alpha, \beta, \gamma, c \in \mathbb{R}$ where $\alpha < \beta, \gamma > 1$ and $\int_{\alpha}^{\beta} ck^{\gamma}\, dk = 1$; then

$$\begin{aligned} \mathbb{E}\left[\frac{(1)}{|V|}\right] &= P\{k_1 \leq v.key \leq k_{mid}\} \\ &= c\frac{k_{mid}^{\gamma+1} - k_1^{\gamma+1}}{\gamma + 1} \end{aligned} \tag{6}$$

and

$$\begin{aligned} \mathbb{E}\left[\frac{(2)}{|V|}\right] &= P\{k_{mid} \leq v.key \leq k_2\} \\ &= c\frac{k_2^{\gamma+1} - k_{mid}^{\gamma+1}}{\gamma + 1}. \end{aligned} \tag{7}$$

When (1) and (2) are approximately equal, i.e.,:

$$c\frac{k_{mid}^{\gamma+1} - k_1^{\gamma+1}}{\gamma + 1} = c\frac{k_2^{\gamma+1} - k_{mid}^{\gamma+1}}{\gamma + 1}, \tag{8}$$

we get

$$k_{mid} = \left(\frac{k_1^{\gamma+1} + k_2^{\gamma+1}}{2}\right)^{\frac{1}{\gamma+1}}. \tag{9}$$

Thus, in this case, we should define (9) as *mid*.

By defining *mid* in this way, $mid(v_{lower}.key, v_{next}.key)$ or $mid(v_{next}.key, v_{prev}.key)$ refers to a key estimation of the center of the lower-level node sequence. Thus, *closeToRight* plays the appropriate role of a detour judgment.

In practice, it is difficult to obtain the distribution of the keys beforehand. However, from the evaluation experiments presented in Section IV, we observed that it is effective in many cases for shortening path lengths by defining *mid* as:

$$mid(k_1, k_2) := \frac{k_1 + k_2}{2}. \tag{10}$$

Fig. 4 shows a routing process when a query to search a node whose key is $k_{target} = 15$ is issued at a node $A$ on the condition that each node follows Algorithm 1 in the topology shown in Fig. 1. The key sequence of the nodes on the path is (0, 18, 15), and the path length is 2, which is shorter than that of Fig. 2.

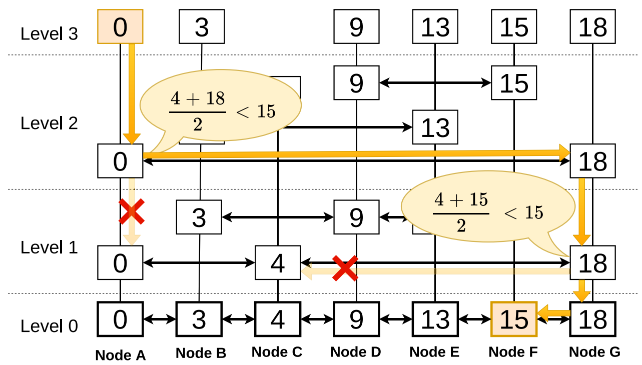


FIGURE 4. A `searchDROp` routing to search a node whose key is 15 from a node *A* where $mid(k_1,k_2) = \frac{k_1+k_2}{2}$.

### B. TRAVERSING FROM THE MAXIMUM LEVEL

In addition to utilizing detour routes, we can improve the routing of Skip Graph by making each node to traverse from the maximum level.

As discussed in Section II-A, in the routing of Skip Graph, a node $v_{current}$ traverses the adjacent nodes at the levels from the reception level $l_{prev}$ in descending order and determines the first adjacent node $v_{next}$ that satisfies the condition as the next node. The levels are monotonically decreasing for the entire routing. However, there are cases where an adjacent node at a level larger than $l_{prev}$ satisfies the condition. Moreover, the larger the level at which $v_{current}$ sends a query to the next node, the larger the difference in the keys between adjacent nodes. Therefore, the difference between the key of the next node and the target key $k_{target}$ when traversing from level $v_{current}.maxLevel$ is smaller than or equal to the difference when traversing from level $l_{prev}$, where $v.maxLevel$ is the maximum level of a node $v$. Thus, it is effective in shortening the path lengths that each node traverses from its maximum level.

Algorithm 2 is a pseudocode of this algorithm. When a node $v_{current}$ receives a query `searchMLOp`, $v_{current}$ traverses the adjacent nodes from $v_{current}.maxLevel$ and sends a query `searchMLOp` to the first node that satisfies the condition. This routing differs from that of Skip Graph only in the start level of traversing. Note that it is possible to use binary search for finding a next node instead of linear search, which is faster, but this code uses the latter for simplicity.

Fig. 5 shows a routing process when a query to search a node whose key is $k_{target} = 15$ is issued at a node $A$ on the condition that each node follows Algorithm 2 in the topology shown in Fig. 1. The key sequence of the nodes on the path is $(0, 4, 9, 15)$, and the path length is 3, which is shorter than that of Fig. 2. While `searchMLOp` uses a vertical link from each level toward the maximum level at each node, the routing algorithm `searchOp` of Skip Graph does not use it. However, the change does not affect the topology because vertical links are virtual links involving only the software process of the node. Thus, `searchMLOp` requires no modification of the topology.

**Algorithm 2:** `searchMLOp` in Node $v_{current}$

`/* traversing from Max Level */`
**upon receiving** $\langle$`searchMLOp`$, v_{start}, k_{target}\rangle$ **then**
  **if** $v_{current}.key = k_{target}$ **then**
    **send** $\langle$`foundOp`$, v_{current}\rangle$ to $v_{start}$;
    **return**;
  **else if** $v_{current}.key < k_{target}$ **then**
    **for** $l_{current} \leftarrow v_{current}.maxLevel$ **downTo** 0 **do**
      $v_{next} \leftarrow v_{current}.neighbors[R][l_{current}]$;
      **if** $v_{next} = \perp$ **then**
        **continue**;
      **if** $v_{next}.key \leq k_{target}$ **then**
        **send** $\langle$`searchMLOp`$, v_{start}, k_{target}\rangle$ to $v_{next}$;
        **return**;
  **else**
    **for** $l_{current} \leftarrow v_{current}.maxLevel$ **downTo** 0 **do**
      $v_{next} \leftarrow v_{current}.neighbors[L][l_{current}]$;
      **if** $v_{next} = \perp$ **then**
        **continue**;
      **if** $v_{next}.key \geq k_{target}$ **then**
        **send** $\langle$`searchMLOp`$, v_{start}, k_{target}\rangle$ to $v_{next}$;
        **return**;
  **send** $\langle$`notFoundOp`$, v_{current}\rangle$ to $v_{start}$

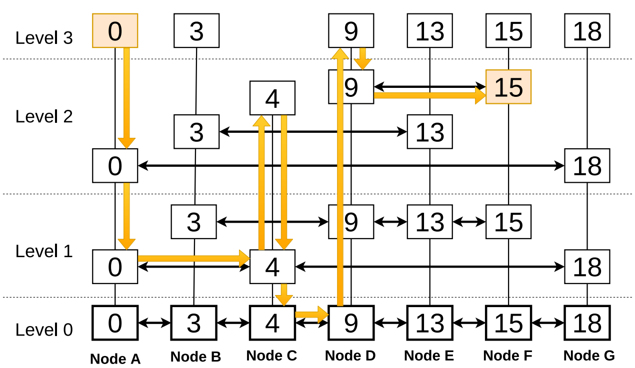


FIGURE 5. A `searchMLOp` routing to search a node whose key is 15 from a node *A*.

It should be noted that Algorithm 2 has the disadvantage of increasing the computation costs incurred between receiving a query and determining the next node, although it has the advantage of shortening the path lengths. Let $l_{MAX}$ be the maximum value of the maximum levels of all nodes, which is $O(\log n)$, and let $H$ be the path length, which is $O(\log n)$. Then, the sum of the time required for each node on a routing path to determine the next node until finishing a routing process, except for the communication time and the I/O processing time, is $O(l_{MAX} + H) = O(\log n)$ for Skip Graph. On the other hand, in the case of the following Algorithm 2, it is $O(H \cdot l_{MAX}) = O(\log^2 n)$. Especially when using binary search, it is $O(H \log l_{MAX}) = O(\log n \cdot \log(\log n))$. These are

**Algorithm 3:** `searchDSGOp` in node $v_{current}$

/* Detouring Skip Graph */

**upon receiving** $\langle$`searchDSGOp`$, v_{start}, k_{target}\rangle$ **then**
  **if** $v_{current}.key = k_{target}$ **then**
    **send** $\langle$`foundOp`$, v_{current}\rangle$ to $v_{start}$;
    **return**;
  **else if** $v_{current}.key < k_{target}$ **then**
    **for** $l_{current} \leftarrow v_{current}.maxLevel$ **downTo** $0$ **do**
      $v_{next} \leftarrow v_{current}.neighbors[R][l_{current}]$;
      **if** $v_{next} = \perp$ **then**
        **continue**;
      **if** $v_{next}.key \le k_{target}$ **then**
        **send** $\langle$`searchDSGOp`$, v_{start}, k_{target}\rangle$ to $v_{next}$;
        **return**;
      **else if** $l_{current} > 0$ **then**
        $v_{lower} \leftarrow v_{current}.neighbors[R][l_{current} - 1]$;
        **if** $closeToRight(k_{target}, v_{lower}.key, v_{next}.key)$ **then**
          **send** $\langle$`searchDSGOp`$, v_{start}, k_{target}\rangle$ to $v_{next}$;
          **return**;
  **else**
    **for** $l_{current} \leftarrow v_{current}.maxLevel$ **downTo** $0$ **do**
      $v_{next} \leftarrow v_{current}.neighbors[L][l_{current}]$;
      **if** $v_{next} = \perp$ **then**
        **continue**;
      **if** $v_{next}.key \ge k_{target}$ **then**
        **send** $\langle$`searchDSGOp`$, v_{start}, k_{target}\rangle$ to $v_{next}$;
        **return**;
      **else if** $l_{current} > 0$ **then**
        $v_{lower} \leftarrow v_{current}.neighbors[L][l_{current} - 1]$;
        **if** $\neg closeToRight(k_{target}, v_{next}.key, v_{lower}.key)$ **then**
          **send** $\langle$`searchDSGOp`$, v_{start}, k_{target}\rangle$ to $v_{next}$;
          **return**;
  **send** $\langle$`notFoundOp`$, v_{current}\rangle$ to $v_{start}$;

inferior to Skip Graph in terms of computational complexity. However, the time taken for a routing process is typically dominated by the communication time, hence it is more important to shorten path lengths in most cases.

### C. DETOURING SKIP GRAPH: COMBINING TWO IMPROVEMENTS

Because the two improved routing algorithms described above are independent changes from the routing algorithm of Skip Graph, an algorithm combining them can be defined naturally. We refer an extension of Skip Graph that performs such routing as Detouring Skip Graph.

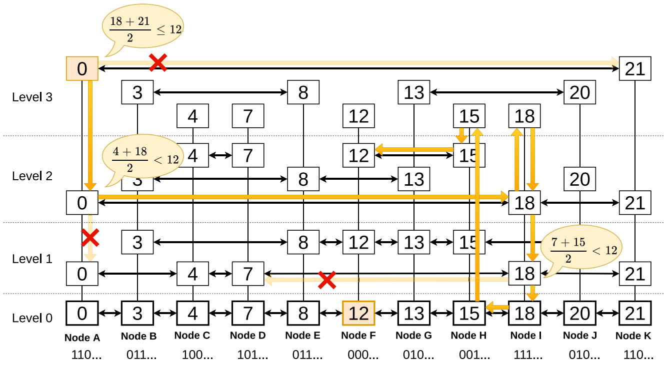


**FIGURE 6.** A `searchDSGOp` routing to search a node whose key is 12 from a node *A* where $mid(k_1, k_2) = \frac{k_1 + k_2}{2}$.

Algorithm 3 is a pseudocode of this routing algorithm. When node $v_{current}$ receives a query `searchDSGOp`, $v_{current}$ traverses the adjacent nodes from $v_{current}.maxLevel$ in the same way as described in Section III-B and sends a query `searchDSGOp` to the first node that satisfies the condition. In the process, `searchDSGOp` uses detour routes based on the detour judgment as described in Section III-A.

Fig. 6 shows a routing process when a query to search a node whose key is $k_{target} = 12$ is issued at a node $A$ on the condition that each node follows Algorithm 3 in the topology. The key sequence of the nodes on the path is $(0, 18, 15, 12)$, and the path length is 3.

From the foregoing, Detouring Skip Graph can bring about the shortening of the path lengths for search queries. Unlike the existing methods discussed in Section II-B, it does not require construction of extra links or modification of its topology. Therefore, there is no increase in message transfer and management costs, and it maintains the good properties of Skip Graph. Additionally, the extension is simple and can be easily applied to existing Skip Graph.

### D. REACHABILITY OF DETOURING SKIP GRAPH

Detouring Skip Graph has a property that the key sequence of the nodes on a routing path is not necessarily monotonic in order since the path is via detour routes, while Skip Graph does not have the property. You might consider that the routing process may lead to an infinite loop. However, the reachability is guaranteed, which is proved in this section.

Suppose a search query whose target key is $k$ is being issued. Then, we introduce the following notations:

- Let $(v_1, v_2, \ldots)$ be the node sequence on the routing path.
- $S_k := \{x \in K | x < k\}$.
- $T_k := \{x \in K | x > k\}$.
- Let $(v_{s_1}, v_{s_2}, \ldots)$ be the subsequence of $(v_i)_i$, whose elements are all $v_i$ satisfying $v_i.key \in S_k$.
- Let $(v_{t_1}, v_{t_2}, \ldots)$ be the subsequence of $(v_i)_i$, whose elements are all $v_i$ satisfying $v_i.key \in T_k$.

- Let binary relation $\prec_k, \succ_k$ on $S_k \cup \{k\} \times T_k \cup \{k\}$ be:

$$\prec_k := \left\{ \begin{matrix} (x, y) \in \\ S_k \cup \{k\} \times T_k \cup \{k\} \end{matrix} \middle| \mathit{closeToRight}(k, x, y) \right\} \tag{11}$$

$$\succ_k := \left\{ \begin{matrix} (x, y) \in \\ S_k \cup \{k\} \times T_k \cup \{k\} \end{matrix} \middle| \neg \mathit{closeToRight}(k, x, y) \right\}. \tag{12}$$

Intuitively, $x \prec_k y$ implies that $y$ is closer to $k$ than $x$, and $x \succ_k y$ implies that $x$ is closer to $k$ than $y$.

The query reaches the target node if and only if $(v_i)_i$ is a finite sequence. Thus, it is sufficient to show that $(v_i)_i$ is finite. If both $(v_{s_i}.key)_i$ and $(v_{t_i}.key)_i$ are strictly approaching $k$ (i.e., they are strictly increasing and strictly decreasing, respectively), $(v_i.key)_i$ converges to the key of the target node in finite steps and $(v_i)_i$ is a finite sequence.

Next, to show the strict monotonicity of each subsequence, the function *mid* used as the detour judgment must exhibit the following properties.

*Property 1:*

$$^\forall x, y \in K, x \leq y \Rightarrow x \leq \mathit{mid}(x, y) \leq y \tag{13}$$

*Property 2:* $^\forall x, x', y, y' \in K$,

$$\max(x, x') \leq y \Rightarrow \left[x \leq x' \Leftrightarrow \mathit{mid}(x, y) \leq \mathit{mid}(x', y)\right] \tag{14}$$

$$x \leq \min(y, y') \Rightarrow \left[y \leq y' \Leftrightarrow \mathit{mid}(x, y) \leq \mathit{mid}(x, y')\right] \tag{15}$$

These properties are satisfied whenever *mid* is defined as the median estimation based on any probability distribution, e.g., $\mathit{mid}(x, y) := \frac{x+y}{2}$. Then, the following lemmas hold.

*Lemma 1:* $^\forall x, x' \in S_k \cup \{k\}, ^\forall y, y' \in T_k \cup \{k\}$,

$$x \prec_k y \Rightarrow x' \leq x \Rightarrow x' \prec_k y \tag{16}$$

$$x \prec_k y \Rightarrow y' \leq y \Rightarrow x \prec_k y' \tag{17}$$

$$x \succ_k y \Rightarrow x \leq x' \Rightarrow x' \succ_k y \tag{18}$$

$$x \succ_k y \Rightarrow y \leq y' \Rightarrow x \succ_k y' \tag{19}$$

*Proof:* Suppose $x \prec_k y$ and $x' \leq x$. From $\max(x, x') \leq k \leq y$ and Property 2, we have $\mathit{mid}(x', y) \leq \mathit{mid}(x, y)$. Since $x \prec_k y$ means $\mathit{mid}(x, y) < k$, $\mathit{mid}(x', y) < k$ ($\Leftrightarrow x' \prec_k y$) holds. The other propositions can also be shown in the same way. ■

*Lemma 2:* $^\forall x_1, x_2 \in S_k \cup \{k\}, ^\forall y_1, y_2 \in T_k \cup \{k\}$,

$$\left[^\exists x \in S_k \cup \{k\} \text{ s.t. } (x \prec_k y_1 \wedge x \succ_k y_2)\right] \Rightarrow y_1 < y_2 \tag{20}$$

$$\left[^\exists y \in T_k \cup \{k\} \text{ s.t. } (x_1 \succ_k y \wedge x_2 \prec_k y)\right] \Rightarrow x_1 > x_2 \tag{21}$$

*Proof:* Suppose there exists $x \in S_k$ such that $x \prec_k y_1$ and $x \succ_k y_2$. If $y_1 \geq y_2$, then $x \succ_k y_1$ from $x \succ_k y_2$ and Lemma 1, however it contradicts $x \prec_k y_1$. Therefore, we have that $y_1 < y_2$. The latter proposition can also be established in the same way.

These lemmas derive the following theorem.

*Theorem 1:* $(v_{s_i}.key)_i$ and $(v_{t_i}.key)_i$ are strictly increasing and strictly decreasing, respectively.

*Proof:* It is sufficient for each step $i$ to show that:

$$\begin{cases} j > 1 \Rightarrow v_{s_j}.key > v_{s_{j-1}}.key & \left(\text{if } ^\exists j \text{ s.t. } s_j = i\right) \\ j > 1 \Rightarrow v_{t_j}.key < v_{t_{j-1}}.key & \left(\text{if } ^\exists j \text{ s.t. } t_j = i\right), \end{cases} \tag{22}$$

where step $i$ represents the process on the $i$-th node in the routing path. This can be shown using mathematical induction.

Suppose (22) holds at step $1, 2, \ldots, i$.

1) If $v_i.key = k$, then step $i + 1$ does not exist because the routing process is complete (and the current node sends a query `foundOp` to the start node).
2) If $v_i.key \in S_k$, then the four cases are considered: (i) $v_{i+1}.key \in S_k$, (ii) $v_{i+1}.key \in T_k$, (iii) $v_{i+1}.key = k$, and (iv) $v_{i+1}.key$ does not exist. In case (i) and (iii), (22) holds at step $i + 1$ because $v_i.key < v_{i+1}.key$ and $v_{i+1}.key \notin S_k \cup T_k$, respectively. In case (iv), step $i + 1$ does not exist because the routing process is complete (and the current node sends a query `notFoundOp` to the start node). In case (ii), because a detour route is used, exist $j$ and $x \in S_k \cup \{k\}$ such that $t_j = i + 1$, $v_i.key < x$, and $x \prec_k v_{t_j}.key$. If $j > 1$, then it implies that a detour route was used at step $t_{j-1}$ and that no detour route was used at step $t_{j-1} + 1, t_{j-1} + 2, \ldots, i - 1$ owing to the definition of subsequence $(v_{t_{j'}})_{j'}$. Thus, $v_{t_{j-1}+1}.key, v_{t_{j-1}+2}.key, \ldots, v_i.key \in S_k$, and there exists $y \in T_k \cup \{k\}$ such that $y < v_{t_{j-1}}.key$ and $v_{t_{j-1}+1}.key \succ_k y$. From Lemma 1, we have $v_{t_{j-1}+1}.key \succ_k v_{t_{j-1}}.key$. In addition, $v_{t_{j-1}+1}.key < v_{t_{j-1}+2}.key < \cdots < v_i.key$ holds by the induction hypothesis. From Lemma 1, we have $v_i.key \succ_k v_{t_{j-1}}.key$. Thus, $v_{t_j}.key < v_{t_{j-1}}.key$ holds because of Lemma 2, i.e., (22) holds at step $i + 1$.
3) If $v_i.key \in T_k$, then (22) holds at step $i + 1$, which can be shown in the same way as 2). ■

Therefore, the reachability for any search query is guaranteed, regardless of whether there is a node with a target key $k$ in the topology.

### E. CORRECTNESS OF DETOURING SKIP GRAPH

In this section, we prove the correctness of any search query of Detouring Skip Graph. Thus, the goal of this section is to ensure that a search operation returns a query `foundOp` if the target node $v$ exists in the topology and that it returns a query `notFoundOp` if the target node $v$ does not exist.

From Section III-D, the node sequence $(v_i)_i$ on a routing path is a finite sequence. Let the length of the sequence be $M$, i.e., $v_M$ denotes the last node. Moreover, Algorithm 3 indicates that $v_M$ sends `foundOp` if $v_M.key = k$ and that $v_M$ sends `notFoundOp` if $v_M.key \neq k$. Thus, the correctness is guaranteed if the following theorem holds.

*Theorem 2:*

$$v_M.key = k \Rightarrow ^\exists v \in V, v.key = k \tag{23}$$

$$v_M.key \neq k \Rightarrow {}^{\forall} v \in V, v.key \neq k \tag{24}$$

where $V$ denotes the participating nodes (when $v_M$ is in the routing process).

*Proof:* Equation (23) holds since $v_M \in V$. Suppose $v_M.key \neq k$. It means that $v_M$ cannot find a next node from the adjacent nodes of $v_M$.

1) If $v_M.key \in S_k$:
   a) If there exists the right adjacent node of $v_M$ at level 0, let $u$ be the node. Then, $u.key \in T_k$ since $v_M$ cannot find a next node. There is only one sorted linked list at level 0, and the set of all the nodes equals $V$. It means that there does not exist $v \in V$ such that $v_M.key < v.key < u.key$. Thus, ${}^{\forall}v \in V, v.key \in S_k \cup T_k$, thereby ${}^{\forall}v \in V, v.key \neq k$ holds.
   b) Otherwise, $v_M$ has no right adjacent node. It means that $v_M.key$ is the largest key in the topology. Thus, ${}^{\forall}v \in V, v.key \in S_k$, thereby ${}^{\forall}v \in V, v.key \neq k$ holds.
2) If $v_M.key \in T_k$, ${}^{\forall}v \in V, v.key \neq k$ holds, which can be shown in the same way as 1).

Therefore, (24) holds.

## F. COMPLEXITY OF DETOURING SKIP GRAPH

The search query in Skip Graph with $n$ nodes takes expected $O(\log n)$ messages [7]. In this section, we prove that the message complexity for a search query of Detouring Skip Graph is also expected $O(\log n)$.

We introduce the following notations for Detouring Skip Graph with alphabets $\Sigma$.

- Let $m(v) \in \Sigma^{\infty}$ be the membership vector of a node $v \in V$.
- Let $|w|$ be the length of a word $w$, with $|w| = \infty$ if $w \in \Sigma^{\infty}$.
- Write $w \uparrow i$ for the prefix of a word $w$ of length $i$.
- Write $w_1 \wedge w_2$ for the longest common prefix of words $w_1$ and $w_2$.

Suppose a search query whose target key is $k$ is being issued. Without loss of generality, we assume that $v_1.key \in S_k$. Then, the goal of this section is to ensure that the number $M$ of nodes in the routing path is expected $O(\log n)$ since the query takes $O(M)$ messages.

*Lemma 3:* ${}^{\forall}i, j,$

$$s_i < t_j \Rightarrow v_{s_i}.key \prec_k v_{t_j}.key \tag{25}$$
$$s_i > t_j \Rightarrow v_{s_i}.key \succ_k v_{t_j}.key \tag{26}$$

*Proof:* These propositions hold from Lemma 1 and Theorem 1. ■

*Lemma 4:* A sequence $(|m(v_1) \wedge m(v_i)|)_{i=1}^{M}$ is weakly decreasing.

*Proof:* Suppose, for proof by contradiction, that

$$\begin{aligned} &{}^{\exists}i \in \{1, \ldots, M-1\} \\ &\text{s.t. } |m(v_1) \wedge m(v_{i+1})| > |m(v_1) \wedge m(v_i)|. \end{aligned} \tag{27}$$

Let $l$ be $|m(v_1) \wedge m(v_i)|$ and let $j$ be the maximum number of $j' \in \{1, \ldots, i-1\}$ satisfying $\left|m(v_1) \wedge m(v_{j'})\right| \geq l+1$ (there exists such a $j$ since $|m(v_1) \wedge m(v_1)| = \infty \geq l+1$).

1) If $v_j.key \in S_k$, then from $\left|m(v_1) \wedge m(v_j)\right| \geq l+1$ and $\left|m(v_1) \wedge m(v_{j+1})\right| \leq l$, we have $l_1 \leq l$ where $l_1 = \left|m(v_j) \wedge m(v_{j+1})\right|$. It means that the node $v_{j+1}$ is the right adjacent node of $v_j$ at level $l_1 (\leq l)$. In addition, from $\left|m(v_1) \wedge m(v_j)\right| \geq l+1$ and $|m(v_1) \wedge m(v_{i+1})| \geq l+1$, we have $l_2 \geq l+1$ where $l_2 = \left|m(v_j) \wedge m(v_{i+1})\right|$. It means that there exists a right adjacent node $u$ of $v_j$ at level $l_2 (\geq l+1)$ such that $v_{j+1}.key < k < u.key \leq v_{i+1}.key$ (possibly $u = v_{i+1}$). From Lemma 3 and Lemma 1, $v_{j+1}.key \prec_k u.key$ holds. Let $u'$ be the right adjacent node of $v_j$ at level $l_1 + 1$, and $k < u'.key \leq u.key$. From Lemma 1, $v_{j+1}.key \prec_k u'.key$ holds. On the other hand, since $v_j$ selects $v_{j+1}$ as the next node, $v_{j+1}.key \succ_k u'.key$ holds. This contradiction proves that ${}^{\forall}i \in \{1, \ldots, M-1\}$, $|m(v_1) \wedge m(v_{i+1})| \leq |m(v_1) \wedge m(v_i)|$, i.e., a sequence $(|m(v_1) \wedge m(v_i)|)_{i=1}^{M}$ is weakly decreasing.
2) If $v_j.key \in T_k$, then we can prove it in the same way as 1). ■

*Lemma 5:* Let $(u_i)_i$ be the node sequence on the routing path for a `searchOp` query whose target key is $k$ issued at the node $v_1 (= v_{s_1})$. Then, the node sequence $(v_{s_i})_i$ is a subsequence of $(u_i)_i$.

*Proof:* $(v_{s_i}.key)_i$ is strictly increasing from Theorem 1, and $(u_i.key)_i$ is also strictly increasing. Thus, it is sufficient to show that $\{v_{s_i}\}_i \subset \{u_i\}_i$.

For each level $l$, let $W_{k,l}$ be:

$$W_{k,l} := \left\{ w \in V \,\middle|\, \begin{array}{l} |m(v_1) \wedge m(w)| = l \wedge \\ v_1.key < w.key \leq k \wedge \\ {}^{\forall}w' \in V, \\ \left[\left|m(v_1) \wedge m(w')\right| \geq l+1 \Rightarrow \right. \\ \quad \left. w'.key \leq k \Rightarrow w'.key < w.key\right] \end{array} \right\}. \tag{28}$$

From routing algorithm of `searchOp`, we have $\{u_i\}_i = (\bigsqcup_{l=0}^{\infty} W_{k,l}) \sqcup \{v_1\}$ where $\sqcup$ denotes disjoint union. It means that it is sufficient to show that: for each level $l$,

$$\left\{v_{s_i} \,\middle|\, \left|m(v_1) \wedge m\left(v_{s_i}\right)\right| = l\right\} \subset W_{k,l}. \tag{29}$$

Herein, fix any level $l \in \{0, 1, \ldots\}$. If there does not $i$ satisfying $\left|m(v_1) \wedge m(v_{s_i})\right| = l$, (29) holds. Suppose there exists such a $i$, and let $j$ be the minimum number of $i$ satisfying it. From Lemma 4 and Theorem 1, the following proposition holds:

$$(29) \Leftrightarrow \begin{array}{l} \neg {}^{\exists}w \in V \text{ s.t.} \\ |m(v_1) \wedge m(w)| \geq l+1 \wedge v_{s_j}.key \leq w.key \leq k. \end{array} \tag{30}$$

Suppose, for proof by contradiction, that

$${}^{\exists}w \in V \text{ s.t.} |m(v_1) \wedge m(w)| \geq l+1 \wedge v_{s_j}.key \leq w.key \leq k. \tag{31}$$

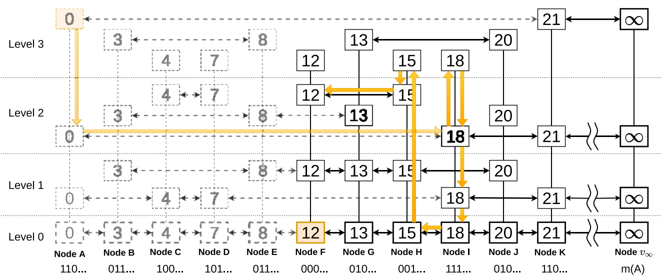


**FIGURE 7.** A topology $G'$ for the topology $G$, the start node $v_1 = A$, and the target key $k = 12$ of Fig. 6.

Let $\iota$ be the maximum number of $i$ satisfying $|m(v_1) \wedge m(v_i)| \geq l+1$, and let $l'$ be $|m(v_1) \wedge m(v_\iota)|$.

1) If $v_{s_j - 1} \in S_k$, then $\iota = s_j - 1 = s_{j-1}$, i.e., the node $v_\iota$ selects the right adjacent node $v_{s_j}$ at level $l$ as the next node. On the other hand, from the hypothesis (31), the node $v_\iota$ should select the right adjacent node ($\neq v_{s_j}$) at level $l'$ as the next node. This contradiction proves (29).
2) If $v_{s_j - 1} \in T_k$:
   a) If $\iota = s_j - 1$, then the node $v_\iota$ has the left adjacent node $v_{s_j}$ at level $l$ and the left adjacent node $w$ at level $l' (> l)$ where $v_{s_j}.key < w.key < k$. It contradicts a property of the topology of Skip Graph. Thus, (29) holds.
   b) If $\iota < s_j - 1$, the node $v_\iota$ selects the left adjacent node $v_{\iota+1}$ at level $l$ as the next node where $v_{\iota+1}.key \in T_k$. In addition, $v_\iota$ has the left adjacent node $w$ at level $l'$ where $v_{s_j}.key \leq w.key \leq k$. From Lemma 3, we have $v_{s_j}.key \succ_k v_{\iota+1}$. From Lemma 1, we have $w.key \succ_k v_{\iota+1}$. Thus, the node $v_\iota$ should select the left adjacent node $w$ at level $l'$ as the next node instead of $v_{\iota+1}$. This contradiction proves (29).

Therefore, $(v_{s_i})_i$ is a subsequence of $(u_i)_i$. ■

*Lemma 6:* The length of $(v_{s_i})_i$ is expected $O(\log n)$.

*Proof:* From Lemma 5, the lengths of $(v_{s_i})_i$ is expected $O(\log n)$ since the routing path length of `searchOp` is expected $O(\log n)$. ■

*Lemma 7:* Suppose there exists $t_1$, in other words, at most 1 detour route is used in the routing of `searchDSGOp`. Let $(u_i)_i$ be the node sequence on the routing path for a `searchOp` query whose target key is $k$ issued at the node $v_{t_1}$. Then, the node sequence $(v_{t_i})_i$ is a subsequence of $(u_i)_i$.

*Proof:* Consider a search query whose target key is $k$ is being issued at the node $v_{t_1}$. Then, we can prove it in the same way as Lemma 5. ■

We would like to show that the length of $(v_{t_i})_i$ is expected $O(\log n)$. However, we cannot prove it in the same way as Lemma 6 because a node $v_{t_1}$ is determined by the start node $v_1$ and the topology dependent on random numbers. Herein, let $G$ be the current topology and we define a topology $G'$ for $G$, $v_1$, and $k$ as follows:

- Let $K' := K \cup \{\infty\}$ where ${}^\forall x \in K, x < \infty$.
- Let $v'_\infty$ be a node with $v'_\infty.key = \infty$ and $m(v'_\infty) = m(v_1)$.
- Let $V' := \{v \in V | v.key \in T_k \cup \{k\}\} \cup \{v'_\infty\}$.
- Let $G'$ be the Skip Graph topology determined by $V'$.

For example, if $G$, $v_1$, and $k$ are the topology, the start node, and the target key in Fig. 6, respectively; the topology of Fig. 7 shows $G'$. Then, the following lemmas hold.

*Lemma 8:* Let $(u'_i)_i$ be the node sequence on the routing path for a `searchOp` query whose target key is $k$ issued at the node $v'_\infty$ in the topology $G'$. Then, the node sequence $(v_{t_i})_i$ is a subsequence of $(u'_i)_i$.

*Proof:* If there does not exist $t_1$, then $(v_{t_i})_i$ is a subsequence of $(u'_i)_i$ since $(v_{t_i})_i$ is empty.

Suppose there exists $t_1$. Let $(u_i)_i$ be the node sequence on the routing path for a `searchOp` query whose target key is $k$ issued at the node $v_{t_1}$ in the topology $G$. From Lemma 7, $(v_{t_i})_i$ is a subsequence of $(u_i)_i$. Thus, it is sufficient to show that $(u'_i)_i$ contains a node $v_{t_1}$, because it means that $(u_i)_i$ equals the contiguous subsequence of $(u'_i)_i$ where the first element is $v_{t_1}$ and the last element is the last node of $(u'_i)_i$.

Let $v'_{max}$ be the node with the maximum key in $\arg\max_{v' \in V' - \{v'_\infty\}}\{|m(v_1) \wedge m(v')|\}$ (in Fig. 7, $v'_{max}$ denotes a node $K$). From the definition, $v'_{max}$ equals a node $u'_2$.

1) If $|m(v_1) \wedge m(v_{t_1})| = |m(v_1) \wedge m(v'_{max})|$, then $v_{t_1}.key \leq v'_{max}.key$ holds from a property of the topology $G$. Thus from routing algorithm of `searchOp`, $(u'_i)_i$ contains $v_{t_1}$.
2) If $|m(v_1) \wedge m(v_{t_1})| < |m(v_1) \wedge m(v'_{max})|$, for each $v_i$ $(i = 1, \ldots, t_1 - 2)$, there does not exist a right adjacent node $w$ of $v_i$ where $w.key \in T_k \;\wedge\; |m(v_1) \wedge m(w)| > |m(v_1) \wedge m(v_{t_1})| \;\wedge\; v_{i+1}.key \succ_k w.key$. From Lemma 2, Lemma 3, and a property of the topology $G$, it means that there does not exist a node $w \in V$ where $|m(v_1) \wedge m(w)| > |m(v_1) \wedge m(v_{t_1})| \;\wedge\; k \leq w.key \leq v_{t_1}.key$. Thus from routing algorithm of `searchOp`, $(u'_i)_i$ contains $v_{t_1}$.

Therefore, $(v_{t_i})_i$ is a subsequence of $(u'_i)_i$. ■

*Lemma 9:* The length of $(v_{t_i})_i$ is expected $O(\log n)$.

*Proof:* From Lemma 8, it is sufficient to show that the length of $(u'_i)_i$ is expected $O(\log n)$.

Let $\{U'_w\}_{w \in \Sigma^*}$ be the family of doubly linked lists of the topology $G'$. Each $U'_w$ denotes the set of nodes $v \in V$ with $m(v) \uparrow |w| = w$. Then, the sequence $U'_{m(v_\infty)\uparrow 0}, U'_{m(v_\infty)\uparrow 1}, U'_{m(v_\infty)\uparrow 2}, \ldots$ is the skip list restriction of the node $v_\infty$. Hence, we can show the length of $(u'_i)_i$ is expected $O(\log n)$ in the same way as the proof for message complexity of `searchOp` given in [7]. ■

*Theorem 3:* The routing path length $M$ of `searchDSGOp` is expected $O(\log n)$.

*Proof:* From definitions of $(v_{s_i})_i$ and $(v_{t_i})_i$,

$$M \leq \left[\text{the length of } \left(v_{s_i}\right)_i\right] + \left[\text{the length of } \left(v_{t_i}\right)_i\right] + 1. \tag{32}$$

**TABLE 1.** Routing algorithms used as the evaluation subjects.

| | | |
|---|---|---|
| `searchOp` | : Skip Graph | (Sec. II-A) |
| `searchDROp` | : Utilizing detour routes | (Sec. III-A) |
| `searchMLOp` | : Traversing from max level | (Sec. III-B) |
| `searchDSGOp` | : Detouring Skip Graph | (Sec. III-C) |
| `searchNLIOp` | : Using node list information | (Sec. IV) |

From Lemma 6 and Lemma 9, the lengths of $(v_{s_i})_i$ and $(v_{t_i})_i$ are expected $O(\log n)$, respectively. Therefore, $M$ is also expected $O(\log n)$. ∎

## IV. EVALUATION

We evaluated the path lengths by conducting routing simulations and observed the effect of the proposed method. The network topologies for the experiments are constructed as Skip Graph topologies with the following key generation methods:

- Generated by uniform distribution.
- Generated by power-law distribution.
- Random English titles on Wikipedia.
- Hashed Random English titles on Wikipedia.

It is important to evaluate path lengths for various key distributions because the routing paths depend on not only the topology, but also the key distribution for participating nodes.

Moreover, the routing algorithms used as the evaluation subjects are `searchOp`, `searchDROp`, `searchMLOp`, `searchDSGOp`, and `searchNLIOp`. Table 1 is a correspondence table between the names of the algorithms and the section numbers with their descriptions.

Herein, we introduce a routing algorithm `searchNLIOp`, which is "ideal" for using detour routes. In Detouring Skip Graph, *mid* is set as a function before the construction of the topology to estimate the key of the center of the lower-level node sequence. However, each node can perfectly determine the use of detour routes without the estimation if the node sequence information is known. We define `searchNLIOp` as a routing algorithm performing such perfect detour judgment and traversing from maximum levels. Although this routing is not practical due to the additional large cost of obtaining the node sequence information, we use it as a comparison target in the experiments to show the limitations of the proposed method.

### A. GENERATED BY UNIFORM DISTRIBUTION

We used pseudorandom numbers to generate keys so that the keys of participating nodes follow uniform distribution $P\{v.key = k\} = \frac{1}{2^{30}}$ $(k \in \{0, 1, \dots, 2^{30} - 1\})$ where $v.key$ is the key of a node $v$ regarded as a random variable. Then, the center estimation *mid* for this distribution is

$$mid_{uniform}(k_1, k_2) := \frac{k_1 + k_2}{2}. \tag{33}$$

This evaluation gives the effectiveness of the proposed method on unbiased key distribution.

On the topology constructed based on the keys generated by the above method, every node issued 100 search queries whose target keys are the keys of randomly sampled nodes. We plotted the average and the maximum of all routing path lengths for these queries in Fig. 8 and Fig. 9, respectively. The horizontal axis represents the number of participating nodes in increments of 100, and the vertical axis represents the average and the maximum path length, respectively. Each line corresponds to each routing method, where `searchDSGOp(mid:uniform)` and `searchDROp(mid:uniform)` represent routing that involves the use of $mid_{uniform}$ as a center estimation for the keys. Further, every routing method is executed on the same topology for each number of nodes; and each topology is built by adding nodes to the existing topology, rather than rebuilt from scratch each time. These conditions are the same for the other experiments discussed in the subsequent sections.

Moreover, we plotted the path length distribution on the above experiments where the number of participating nodes is 10000 in Fig. 10. The horizontal axis represents path lengths, and the vertical axis represents the frequency of each path length.

As a result in Fig. 8, the average path lengths are shorter in the order of `searchNLIOp`, `searchDSGOp(mid:uniform)`, `searchDROp(mid:uniform)`, `searchMLOp`, and `searchOp`. Furthermore, `searchDSGOp(mid:uniform)` of Detouring Skip Graph shortens the average path lengths by about 30% compared to `searchOp` of Skip Graph, and `searchNLIOp` shortens them by about 32% compared to `searchOp`. It shows that Detouring Skip Graph takes advantage of detour routes to shorten the path lengths. Similar trends are shown in Fig. 9 and Fig. 10. The proposed method also provides more stable performance for any queries compared to Skip Graph since the standard deviations of the path lengths of `searchOp` and `searchDSGOp(mid:uniform)` are 4.59 and 2.78 in Fig. 10, respectively.

In addition, we conducted the same experiment on the condition that the target keys $k_{target} \in \{0, 1, \dots, 2^{30} - 1\}$ are generated by uniform distribution for comparison purposes. In this evaluation, since the maximum number of nodes is $10^4$, the probability that a target key is a non-existent key is $1 - \frac{10^4}{2^{30}} \simeq 1$. Thus, there does most likely not exist a target node for each target keys.

Fig. 11, Fig. 12, and Fig. 13 show the evaluation result. `searchDSGOp(mid:uniform)` of Detouring Skip Graph shortens the average path lengths by about 33% compared to `searchOp` of Skip Graph, and `searchNLIOp` shortens them by about 36% compared to `searchOp`. The result is similar to those for existent keys. We consider that it is because the distribution of target keys is the same in the two experiments.

### B. GENERATED BY POWER-LAW DISTRIBUTION

We converted pseudorandom numbers to generate keys so that the keys of participating nodes follow power-law distribution $P\{v.key \leq k\} = \int_0^k f(\kappa)d\kappa$ $(0 \leq k \leq 2^{30})$ where $v.key$ is the key of a node $v$ regarded as a random variable, $f$ denotes a probability density function

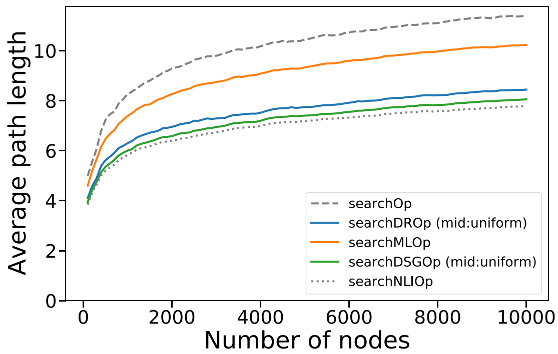


**FIGURE 8.** Average path lengths on a topology whose keys were generated by uniform distribution. The target keys are the keys of randomly sampled nodes.

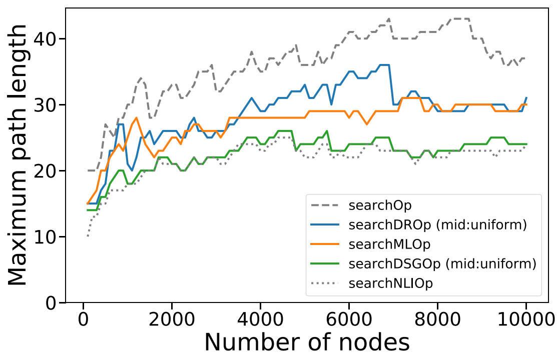


**FIGURE 9.** Maximum path lengths on a topology whose keys were generated by uniform distribution. The target keys are the keys of randomly sampled nodes.

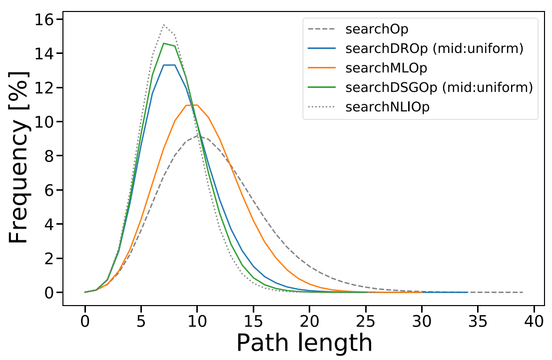


**FIGURE 10.** Path length distribution on a topology whose keys were generated by uniform distribution where $n = 10000$. The target keys are the keys of randomly sampled nodes.

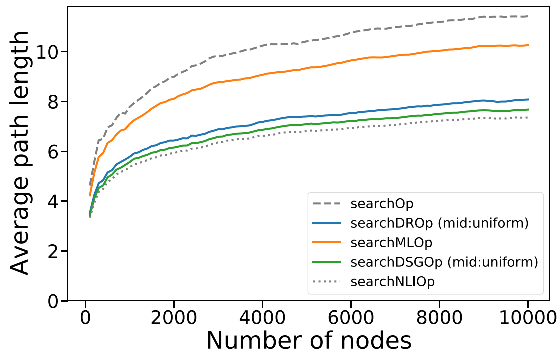


**FIGURE 11.** Average path lengths on a topology whose keys were generated by uniform distribution. The target keys are generated by uniform distribution.

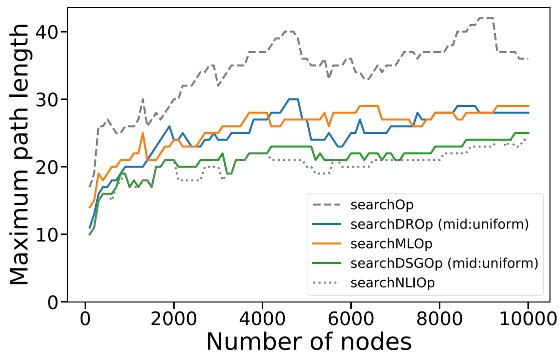


**FIGURE 12.** Maximum path lengths on a topology whose keys were generated by uniform distribution. The target keys are generated by uniform distribution.

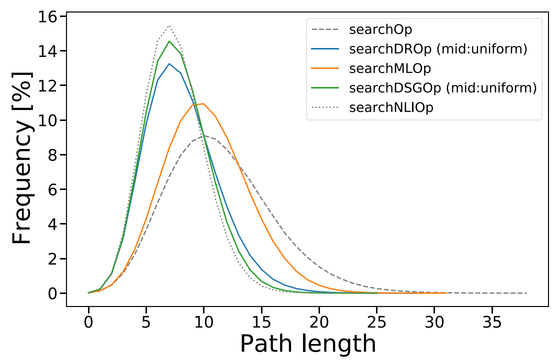


**FIGURE 13.** Path length distribution on a topology whose keys were generated by uniform distribution where $n = 10000$. The target keys are generated by uniform distribution.

$f(k) = ck^{10}$ $(0 \leq k \leq 2^{30})$, and $c$ denotes a constant that satisfies $\int_0^{2^{30}} f(k)dk = 1$. Then, from (9), the center estimation *mid* for this distribution is

$$mid_{power}(k_1, k_2) := \left(\frac{k_1^{10+1} + k_2^{10+1}}{2}\right)^{\frac{1}{10+1}}. \qquad (34)$$

The key distribution may be biased in the actual use of the proposed method without hashed keys. The purpose of using power-law distribution is to evaluate the effectiveness of the proposed method on simply biased key distribution.

On the topology constructed based on the keys generated by the above method, every node issued 100 search queries whose target keys are the keys of randomly sampled nodes. We plotted the average and the maximum of all routing path lengths for these queries in Fig. 14 and Fig. 15, respectively. `searchDSGOp(mid:power)` and `searchDROp(mid:power)` represent routing that involves the use of $mid_{power}$ as a center estimation of keys. Moreover, we plotted the path length distribution where the number of participating nodes is 10000 in Fig. 16.

As a result in Fig. 14, the average path lengths were almost the same in `searchDSGOp(mid:uniform)` and `searchDSGOp(mid:power)`, and we discovered that using $mid_{uniform}$ as a detour criterion is effective even if the key

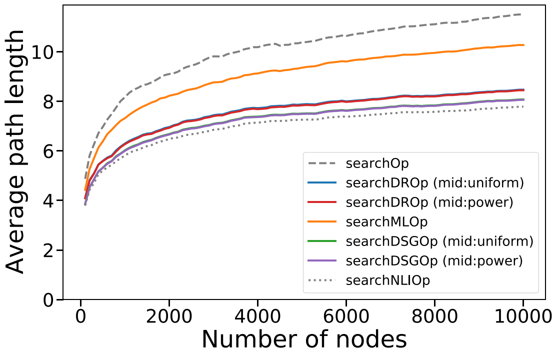


**FIGURE 14.** Average path lengths on a topology whose keys were generated by power-law distribution. The target keys are the keys of randomly sampled nodes.

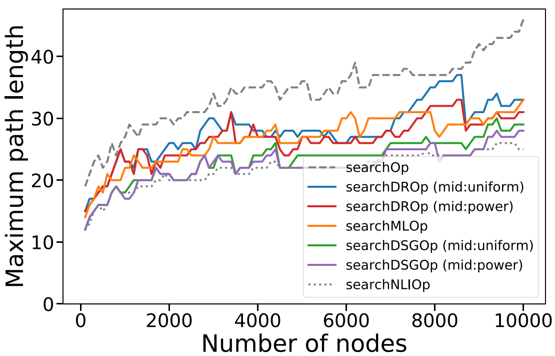


**FIGURE 15.** Maximum path lengths on a topology whose keys were generated by power-law distribution. The target keys are the keys of randomly sampled nodes.

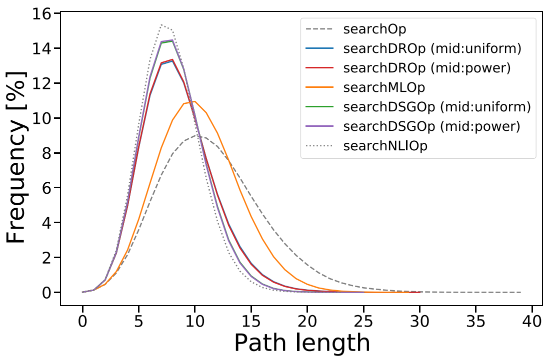


**FIGURE 16.** Path length distribution on a topology whose keys were generated by power-law distribution where $n = 10000$. The target keys are the keys of randomly sampled nodes.

**TABLE 2.** Average path lengths on a topology whose keys were generated by power-law distribution where $n = 100, 1000, 10000$. The target keys are generated by uniform distribution.

| | $n = 100$ | 1000 | 10000 |
|---|---|---|---|
| searchOp | 4.87 | 8.17 | 11.50 |
| searchDROp(mid : uniform) | 4.10 | 6.30 | 8.47 |
| searchDROp(mid : power) | 4.09 | 6.27 | 8.45 |
| searchMLOp | 4.42 | 7.32 | 10.27 |
| searchDSGOp(mid : uniform) | 3.86 | 6.02 | 8.08 |
| searchDSGOp(mid : power) | 3.85 | 6.00 | 8.06 |
| searchNLIOp | 3.79 | 5.82 | 7.79 |

distribution is biased. Table 2 lists the average path lengths where the number of nodes $n$ is 100, 1000, and 10000. The numerical values also indicate that the average path lengths of

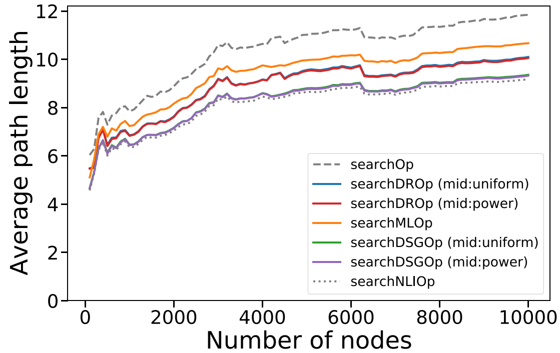


**FIGURE 17.** Average path lengths on a topology whose keys were generated by power-law distribution. The target keys are generated by uniform distribution.

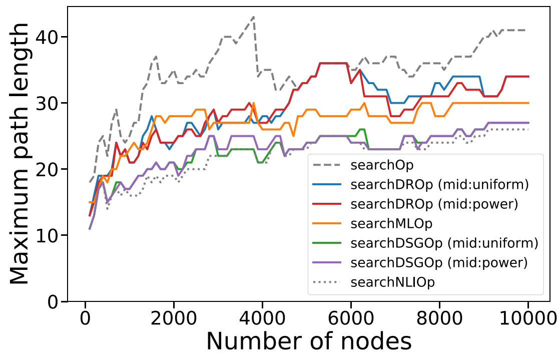


**FIGURE 18.** Maximum path lengths on a topology whose keys were generated by power-law distribution. The target keys are generated by uniform distribution.

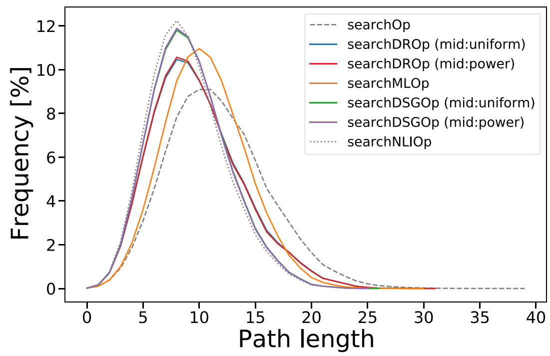


**FIGURE 19.** Path length distribution on a topology whose keys were generated by power-law distribution where $n = 10000$. The target keys are generated by uniform distribution.

the routing following searchDSGOp(mid:uniform) and searchDSGOp(mid:power) are almost the same. In both routing, the average path length of searchOp was shortened by about 30%. In addition, the result shows that Detouring Skip Graph takes advantage of detour routes to shorten the path lengths even in this experiments since searchNLIOp shortens the average path lengths by about 32%. Similar trends are shown in Fig. 15 and Fig. 16. The proposed method also provides more stable performance for any queries compared to Skip Graph since the standard deviations of the path lengths of searchOp and searchDSGOp(mid:uniform) are 4.54 and 2.76 in Fig. 16, respectively.

In addition, we conducted the same experiment on the condition that the target keys $k_{target} \in \{0, 1, \ldots, 2^{30} - 1\}$ are generated by uniform distribution for comparison purposes. In this evaluation, all of them are non-existent keys.

Fig. 17, Fig. 18, and Fig. 19 show the evaluation result. `searchDSGOp(mid:uniform)` and `searchDSGOp(mid:uniform)` of Detouring Skip Graph shortens the average path lengths by about 21% compared to `searchOp` of Skip Graph, and `searchNLIOp` shortens them by about 22% compared to `searchOp`. The shortening rate by the proposed method is smaller than that in the experiment with existent keys. We consider that it is because the target keys are biased relative to the key distribution of the nodes, thus a few detour routes were used.

### C. RANDOM ENGLISH TITLES ON WIKIPEDIA

We used 10000 random English titles obtained on Apr. 15, 2020 from an API[2] published by Wikipedia as keys. Specifically, we encoded each title as a character string in UTF-8, and we used the strings as $256(= 2^8)$-based integer keys. The purpose of using random titles is to evaluate the effectiveness of the proposed method in realistic situations.

On the topology constructed based on the keys obtained by the above method, every node issued 100 search queries whose target keys are the keys of randomly sampled nodes. We plotted the average and the maximum of all routing path lengths for these queries in Fig. 20 and Fig. 21, respectively. Moreover, we plotted the path length distribution where the number of participating nodes is 10000 in Fig. 22.

As a result, `searchDSGOp(mid:uniform)` of Detouring Skip Graph shortens the average path lengths by about 26% compared to `searchOp` of Skip Graph, and `searchNLIOp` shortens them by about 31% compared to `searchOp`. The effectiveness of `searchDSGOp(mid:uniform)` is smaller than that of `searchNLIOp` because the detour criterion $mid_{uniform}$ is not a center estimation for this key distribution. However, the result shows that the proposed method has a large effect on shortening the path lengths despite the mismatch detour criterion. Similar trends are shown in Fig. 21 and Fig. 22. The proposed method also provides more stable performance for any queries compared to Skip Graph since the standard deviations of the path lengths of `searchOp` and `searchDSGOp(mid:uniform)` are 4.62 and 3.08 in Fig. 22, respectively.

### D. HASHED RANDOM ENGLISH TITLES ON WIKIPEDIA

Since Skip Graph is can be used as a DHT by hashing keys for load balancing, it is important to evaluate the effectiveness of the proposed method in such cases. In reality, the hash value of some properties of each node, e.g., the hashed IP address. In this scenario, we used the hash values of the titles of IV.C as keys using SHA3-512 hash function.

This experimental settings the same as IV.C except that the keys is hashed. Fig. 23, Fig. 24, and Fig. 25 show the evaluation result.

2. https://www.mediawiki.org/wiki/API:Main_page (accessed Apr. 15, 2020).

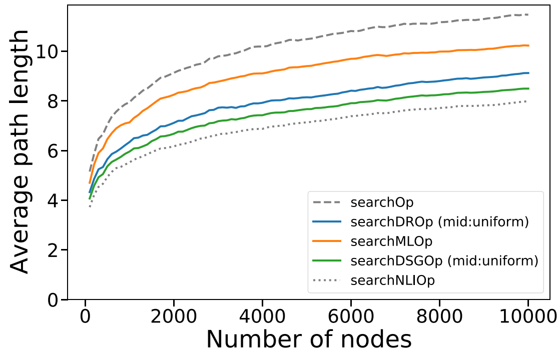


**FIGURE 20.** Average path lengths on a topology whose keys are random English titles on Wikipedia.

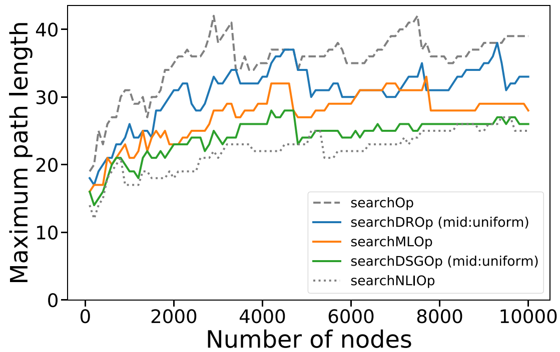


**FIGURE 21.** Maximum path lengths on a topology whose keys are random English titles on Wikipedia.

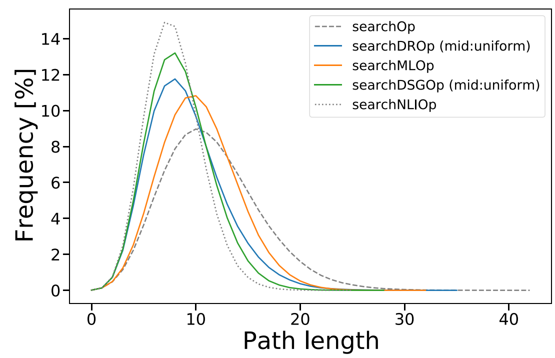


**FIGURE 22.** Path length distribution on a topology whose keys are random English titles on Wikipedia where $n = 10000$.

As a result, `searchDSGOp(mid:uniform)` of Detouring Skip Graph shortens the average path lengths by about 29% compared to `searchOp` of Skip Graph, and `searchNLIOp` shortens them by about 32% compared to `searchOp`. The proposed method also provides more stable performance for any queries compared to Skip Graph since the standard deviations of the path lengths of `searchOp` and `searchDSGOp(mid:uniform)` are 4.44 and 2.78 in Fig. 25, respectively.

Although the effectiveness is slightly less than that of Section IV-A, the result is similar. We consider that this is because the hash function brings the key distribution closer to uniform distribution and makes the detour criterion $mid_{uniform}$

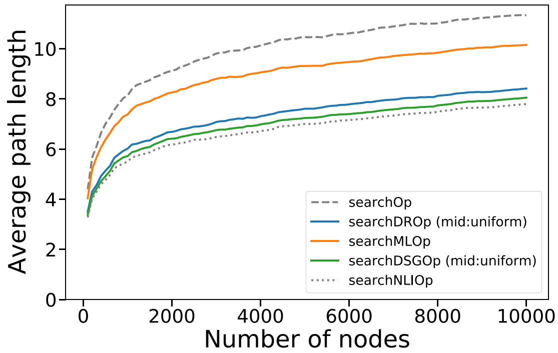


FIGURE 23. **Average path lengths on a topology whose keys are random English titles on Wikipedia.**

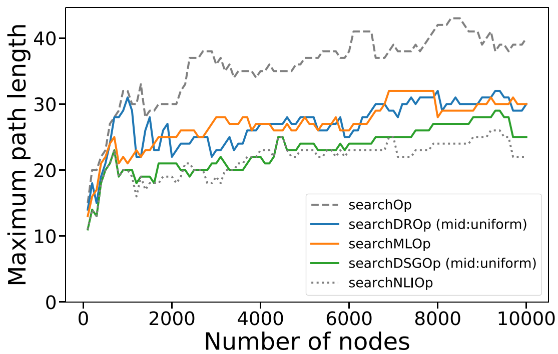


FIGURE 24. **Maximum path lengths on a topology whose keys are random Endlish titles on Wikipedia.**

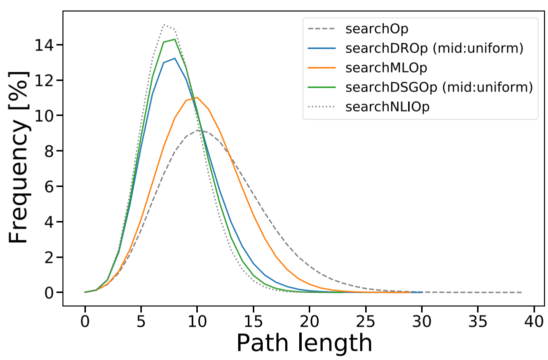


FIGURE 25. **Path length distribution on a topology whose keys are random English titles on Wikipedia where $n = 10000$.**

appropriate. The result confirmed a sufficient effect for this experimental scale although the bias of the distribution is a concern with a small number of nodes. In addition, an important finding is that the proposed method is more effective with hashing than without it.

## V. CONCLUSION

In this article, we proposed Detouring Skip Graph, which shortens the path lengths by using effectively the topology that Skip Graph constructs. It introduces two techniques in the routing algorithm of Skip Graph: each node 1) utilizes detour routes and 2) traverses the adjacent nodes from the maximum level. The simple extension enables to apply the proposed method to existing Skip Graph. In addition, we proved the reachability and correctness for any search query and that the routing path length is expected $O(\log n)$.

Detouring Skip Graph does not require construction of extra links and modification of its topology; thereby, it maintains the good properties of Skip Graph without additional costs. Through the evaluation experiments, we confirmed the large effect on shortening the path lengths, and especially the average path lengths were shortened by approximately 20%-30% in comparison with Skip Graph. Further, it was experimentally found that $mid_{uniform}$, the average of the keys belonging to two nodes, is effective as a detour criterion even for biased or realistic key distribution.

**Algorithm 4:** `searchOp` in node $v_{current}$

```
/* Skip Graph                                                  */
upon receiving ⟨searchOp, v_start, k_target, l_prev⟩ then
    if v_current.key = k_target then
        send ⟨foundOp, v_current⟩ to v_start;
        return;
    else if v_current.key < k_target then
        for l_current ← l_prev downTo 0 do
            v_next ← v_current.neighbors[R][l_current];
            if v_next = ⊥ then
                continue;
            if v_next.key ≤ k_target then
                send ⟨searchOp, v_start, k_target, l_current⟩ to
                  v_next;
                return;
    else
        for l_current ← l_prev downTo 0 do
            v_next ← v_current.neighbors[L][l_current];
            if v_next = ⊥ then
                continue;
            if v_next.key ≥ k_target then
                send ⟨searchOp, v_start, k_target, l_current⟩ to
                  v_next;
                return;
    send ⟨notFoundOp, v_current⟩ to v_start
```

## APPENDIX

In this Appendix, we give Algorithm 4, which is a pseudocode of routing algorithm for search queries of Skip Graph.

## REFERENCES

[1] A. Malatras, "State-of-the-art survey on P2P overlay networks in pervasive computing environments," *J. Netw. Comput. Appl.*, vol. 55, pp. 1–23, Sep. 2015.
[2] G. DeCandia *et al.*, "Dynamo: Amazon's highly available key-value store," in *Proc. 21st ACM SIGOPS Symp. Oper. Syst. Principles (SOSP)*, 2007, pp. 205–220.

[3] N. Ramzan, H. Park, and E. Izquierdo, "Video streaming over P2P networks: Challenges and opportunities," *Signal Process. Image Commun.*, vol. 27, no. 5, pp. 401–411, May 2012.

[4] A. Yahyavi and B. Kemme, "Peer-to-peer architectures for massively multiplayer online games: A survey," *ACM Comput. Surveys*, vol. 46, no. 1, pp. 1–51, Jul. 2013.

[5] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT," in *Proc. 2nd IEEE/ACM Int. Conf. Internet Things Design Implement. (IoTDI)*, 2017, pp. 173–178.

[6] Y. Hassanzadeh-Nazarabadi, A. Küpçü, and Ö. Özkasap, "LightChain: A DHT-based blockchain for resource constrained environments," Mar. 2019. [Online]. Available: arXiv:1904.00375.

[7] J. Aspnes and G. Shah, "Skip graphs," *ACM Trans. Algorithms*, vol. 3, no. 4, pp. 1–25, Nov. 2007.

[8] I. Stoica *et al.*, "CHORD: A scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[9] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Int. Conf. Distrib. Syst. Platforms Open Distrib. Process. (Middleware)*, Nov. 2001, pp. 329–350.

[10] A. González-Beltrán, P. Milligan, and P. Sage, "Range queries over skip tree graphs," *Comput. Commun.*, vol. 31, no. 2, pp. 358–374, Feb. 2008.

[11] R. Banno and K. Shudo, "An efficient routing method for range queries in skip graph," *IEICE Trans. Inf. Syst.*, vol. E103-D, no. 3, pp. 516–525, Mar. 2020.

[12] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun. (SIGCOMM)*, Aug. 2003, pp. 381–394.

[13] T. Miyao, H. Nagao, and K. Shudo, "A method for designing proximity-aware routing algorithms for structured overlays," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2013, pp. 508–514.

[14] T. Toda, Y. Tanigawa, and H. Tode, "Autonomous and distributed construction of locality aware skip graph," in *Proc. 14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2017, pp. 33–36.

[15] R. Jacob, A. Richa, C. Scheideler, S. Schmid, and H. Täubig, "SKIP$^+$: A self-stabilizing skip graph," *J. ACM*, vol. 61, no. 6, pp. 1–26, Dec. 2014.

[16] T. Kaneko, R. Banno, K. Shudo, Y. Aoki, K. Abe, and Y. Teranishi, "Detouring skip graph: A structured overlay utilizing detour routes," in *Proc. IEEE 17th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2020, pp. 1–7.

[17] W. Pugh, "Skip lists: A probabilistic alternative to balanced trees," *Commun. ACM*, vol. 33, no. 6, pp. 668–676, Jun. 1990.

[18] K. Higuchi, M. Yoshida, T. Tsuji, and N. Miyamoto, "Correctness of the routing algorithm for distributed key-value store based on order preserving linear hashing and skip graph," in *Proc. 18th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distrib. Comput. (SNPD)*, Jun. 2017, pp. 459–464.

[19] M. Naor and U. Wieder, "Know thy neighbor's neighbor: Better routing for skip-graphs and small worlds," in *Proc. Int. Workshop Peer-to-Peer Syst. (IPTPS)*, Feb. 2004, pp. 269–277.

[20] A. G. Beltran, P. Sage, and P. Milligan, "Skip tree graph: A distributed and balanced search tree for peer-to-peer networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2007, pp. 1881–1886.

[21] F. Makikawa, T. Tsuchiya, and T. Kikuno, "Balance and proximity-aware skip graph construction," in *Proc. 1st Int. Conf. Netw. Comput.*, Nov. 2010, pp. 268–271.

[22] T. Kawaguchi, R. Banno, M. Hojo, M. Ohnishi, and K. Shudo, "Self-refining skip graph: Skip graph approaching to an ideal topology," in *Proc. 14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2017, pp. 441–448.

[23] S. Huq and S. Ghosh, "Locally self-adjusting skip graphs," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 805–815.

[24] A. Goyal, S. Batra, N. Kumar, G. S. Aujla, and M. S. Obaidat, "Adaptive skip graph framework for peer-to-peer networks: Search time complexity analysis," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[25] P. Keef and D. Guichard. *Introduction to Higher Mathematics*. Accessed: Apr. 15, 2020. [Online]. Available: https://www.whitman.edu/mathematics/higher_math_online/

[26] K. Shudo, "Churn tolerance improvement techniques in an algorithm-neutral DHT," in *Proc. IFIP 3rd Int. Conf. Auton. Infrastruct. Manag. Security (AIMS)*, 2009, pp. 42–55.

[27] Y. Hassanzadeh-Nazarabadi, A. Küpçü, and Ö. Özkasap, "Decentralized and locality aware replication method for DHT-based P2P storage systems," *Future Gener. Comput. Syst.*, vol. 84, pp. 32–46, Jul. 2018.

[28] K. Abe and M. Yoshida, "Constructing distributed doubly linked lists without distributed locking," in *Proc. IEEE Int. Conf. Peer Peer Comput. (P2P)*, Sep. 2015, pp. 1–10.


**TAKESHI KANEKO** received the B.S. degree in information science from the Tokyo Institute of Technology, Tokyo, Japan, in 2019, where he is currently pursuing the master's degree. His research interests include distributed systems.

**RYOHEI BANNO** (Member, IEEE) received the Bachelor of Engineering degree and the Master of Information Science and Technology degree from Hokkaido University, Sapporo, Japan, in 2010 and 2012, respectively, and the Ph.D. degree in science from the Tokyo Institute of Technology, Tokyo, Japan, in 2018.

From 2012 to 2018, he was a Researcher with NTT Network Innovation Laboratories. From 2018 to 2020, he was a Researcher with the Tokyo Institute of Technology. Since 2020, he has been an Assistant Professor with Kogakuin University, Tokyo. His research interests include distributed systems and Internet of Things.

Dr. Banno's awards and honors include the Outstanding Paper Award from Information Processing Society of Japan in 2015, the Inoue Research Award for Young Scientist from Inoue Foundation for Science in 2020, and the Funai Research Award from Funai Foundation for Information Technology in 2020.

**KAZUYUKI SHUDO** (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer science from Waseda University in 1996, 1998, and 2001, respectively.

He worked as a Research Associate with Waseda University, from 1998 to 2001. He later served as a Research Scientist with the National Institute of Advanced Industrial Science and Technology. In 2006, he joined Utagoe, Inc., as a Director and a Chief Technology Officer. Since December 2008, he has been an Associate Professor with the Tokyo Institute of Technology. His research interests include distributed computing, programming language systems, and information security. He has received the Best Paper Award at SACSIS in 2006, the Information Processing Society Japan (IPSJ) Best Paper Award in 2006, the Super Creator certification by Japanese Ministry of Economy Trade and Industry and Information Technology Promotion Agency in 2007, the IPSJ Yamashita SIG Research Award in 2008, the Funai Prize for Science in 2010, the Young Scientists' Prize, the Commendation for Science and Technology by the Minister of Education, Culture, Sports, and Technology in 2012, and the IPSJ Nagao Special Researcher Award in 2013. He is a member of IEEE Computer Society, IEEE Communications Society, and ACM.

**KOTA ABE** (Member, IEEE) received the M.E. and Ph.D. degrees from Osaka University, Japan, in 1994 and 2000, respectively.

In 1994, he joined Nippon Telegraph and Telephone Corporation, Japan. In 1996, he started working as a Research Associate with the Media Center, Osaka City University, Japan, where he has been a Professor with the Graduate School of Engineering, since 2018. Also, since 2015, he has been a Cooperative Visiting Researcher with the National Institute of Information and Communications Technology. His research interests include distributed systems and system software. He received the Information Processing Society of Japan Best Paper Award in 2013.

**YUUICHI TERANISHI** (Member, IEEE) received the M.E. and Ph.D. degrees from Osaka University, Japan, in 1995 and 2004, respectively.

From 1995 to 2004, he was engaged Nippon Telegraph and Telephone Corporation. From 2005 to 2007, he was a Lecturer with the Cybermedia Center, Osaka University, where he was an Associate Professor with Graduate School of Information Science and Technology, from 2007 to 2011. Since August 2011, he has been a Research Manager with the National Institute of Information and Communications Technology and a Guest Associate Professor with the Cybermedia Center, Osaka University. His research interests include technologies for distributed network systems and applications. He received the IPSJ Best Paper Award in 2011.