J Inf Process Syst, Vol.16, No.1, pp.201~209, February 2020
https://doi.org/10.3745/JIPS.04.0159

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Combining Local and Global Features to Reduce 2-Hop Label Size of Directed Acyclic Graphs

Jinhyun Ahn[*] and Dong-Hyuk Im[**]

Abstract
The graph data structure is popular because it can intuitively represent real-world knowledge. Graph databases have attracted attention in academia and industry because they can be used to maintain graph data and allow users to mine knowledge. Mining reachability relationships between two nodes in a graph, termed reachability query processing, is an important functionality of graph databases. Online traversals, such as the breadth-first and depth-first search, are inefficient in processing reachability queries when dealing with large-scale graphs. Labeling schemes have been proposed to overcome these disadvantages. The state-of-the-art is the 2-hop labeling scheme: each node has *in* and *out* labels containing reachable node IDs as integers. Unfortunately, existing 2-hop labeling schemes generate huge 2-hop label sizes because they only consider local features, such as degrees. In this paper, we propose a more efficient 2-hop label size reduction approach. We consider the topological sort index, which is a global feature. A linear combination is suggested for utilizing both local and global features. We conduct experiments over real-world and synthetic directed acyclic graph datasets and show that the proposed approach generates smaller labels than existing approaches.

Keywords
Degree, Directed Acyclic Graphs, Topological Sort, 2-Hop Label

1. Introduction

A graph is a pair of a set of nodes and a set of edges connecting two nodes [1]. Several types of graph exist. We focus on the directed acyclic graph (DAG). A directed graph has a set of edges consisting of ordered pairs of two nodes. A DAG is a directed graph with no cycles. A cycle in a graph is a sequence of nodes starting and ending at the same node, connected via edges. Reachability over a graph and its corresponding DAG are equivalent because a graph can be transformed into a DAG. Cycles in a graph can be removed by condensing nodes in a connected component into an imaginary node [2].

Most real-world knowledge can be represented by graphs; examples are biological networks [3], software version management [4], social networks [5], data preprocessing in machine learning [6], and sensor networks [7,8]. In the biology domain, for example, a protein corresponds to a node whereas a signaling connection between two proteins corresponds to an edge. A graph query expresses a user's intention to locate specific knowledge in a graph. One of the fundamental graph queries is to check if

Manuscript received June 18, 2018; first revision September 6, 2019; accepted October 21, 2019.
Corresponding Author: Dong-Hyuk Im (dhim@hoseo.edu)
* Dept. of Management Information Systems, Jeju National University, Korea (jha@jejunu.ac.kr)
** Dept. of Computer and Information Engineering, Hoseo University, Korea (dhim@hoseo.edu)

there exists a graph path starting from a node and ending at another node [9]. Two nodes u and v in a graph are reachable if there exists a sequence of adjacent edges that starts at u and ends at v. In the biology graph example, the reachability corresponds to the existence of signaling paths between proteins.

Labeling schemes, such as prime number labeling scheme, tree cover approach, and 2-hop labeling scheme [9], have been proposed to determine the reachability in a DAG. We focus on the state-of-the art approach, i.e., 2-hop labeling schemes that assign two kinds of labels $L_{out}(v)$ and $L_{in}(v)$ to each node [10]. The formal definition is given in Section 3. The 2-hop label size is calculated via summation over node IDs in $L_{out}(v)$ and $L_{in}(v)$, where node IDs are assumed to be integers for simplicity.

There exist two lines of research to reduce 2-hop label size. In the first case, the cardinality of $L_{out}(v)$ and $L_{in}(v)$ is reduced while minimizing the loss of reachability query processing performance [2,11]. That is, several reachable nodes are selected to be included in $L_{out}(v)$ and $L_{in}(v)$ while maintaining reachability query processing performance. $L_{out}(v)$ and $L_{in}(v)$ only maintain at most k node IDs, where k is a positive integer. In the second case, each $L_{out}(v)$ and $L_{in}(v)$ is set to have node IDs as small as possible [12,13]. These approaches have strategies to assign IDs to each node. The idea is motivated by the fact that arbitrary IDs can be assigned to each node if no two nodes have the same id [2]. The difference between the first and the second research directions is as follows. The first one tries to maintain small cardinality of $L_{out}(v)$ and $L_{in}(v)$, while the second one focuses on how small integer numbers are included in $L_{out}(v)$ and $L_{in}(v)$. The second line of approaches expects that smaller integer numbers with $L_{out}(v)$ and $L_{in}(v)$ enable obtaining smaller 2-hop label sizes. Since previous works are still inefficient in reducing 2-hop label size, in this paper we propose an improved way to assign node IDs that consider both local and global features of nodes.

The paper is organized as follows. Section 2 describes related works, the 2-hop labeling scheme is explained in Section 3, the proposed approach is demonstrated in Section 4, experimental results are discussed in Section 5, and the paper is concluded in Section 6.

2. Related Works

The simplest ways of checking reachability over a given DAG are online graph traversals such as depth-first search (DFS) and bread-first search (BFS). For a given pair of nodes, we can check if there exists a path using DFS and BFS. Computing transitive closures (TCs) for every node allows us to answer a reachability query in $O(1)$ time [14]. However, computing TCs is expensive and the index could be huge for even small, dense graphs. Various approaches have been made to address the tradeoffs between the index size and the speed of query processing.

To avoid such expensive index construction, labeling schemes have been proposed. Prime number labeling schemes [15,16] assign each node v the product of all its direct parents' labels and the self-label (a unique prime number). Reachability queries can be answered via divisibility checking. However, prime numbers grow quickly. The tree cover approach labels each node with the compressed TC of the subtree rooted at the node [1]. GRAIL (GRaph Indexing based on Pre- and Postorder numbering) is an interval label scheme that proposes randomized multiple interval labeling [17]. FERRARI utilizes a mixture of exact and approximate reachability intervals [18].

Recently, a state-of-the-art variation of 2-hop labeling was proposed in [2], which reported outstanding performance compared to existing approaches. A permutation of nodes is first generated randomly.

MinHash is adapted to construct 2-hop labels; at most k reachable nodes are maintained in the 2-hop labels. For pairs that cannot be determined by k reachable nodes only, an online search is performed. Some heuristics are also presented to minimize cases that require exhaustive online DFS searches.

3. Preliminaries

Typical 2-hop labeling schemes of a DAG assign each node v two labels, $L_{out}(v)$ and $L_{in}(v)$ [6]. $L_{out}(v)$ is a set of nodes that v can reach and $L_{in}(v)$ is a set of nodes reachable at v. A permutation-based 2-hop labeling scheme was proposed in [2]. In the scheme, $L_{out}(v)$ and $L_{in}(v)$ retain at most k reachable nodes. For reachability that cannot be answered by the labels, online graph traversal is used instead. We give a formal definition of [2] in Definition 1.

Definition 1. (2-Hop Labeling) For a given DAG $G(V,E)$ where V is a set of nodes and E is a set of edges, we consider $u \Rightarrow v$ if the node v is reachable from the node u, and $u \not\Rightarrow v$ otherwise. A permutation mapping of V can be represented as $\sigma: V \rightarrow V$. For a DAG G and a positive integer k, we build a permutation mapping σ_i and create a 2-hop label $I^k_{\sigma_i}: V \rightarrow H$, where $H := (L_{out}(v), L_{in}(v))$ such that

$$L_{out}(v) \coloneqq min_k\{\sigma(u) | v \Rightarrow u\} \tag{1-1}$$

$$L_{in}(v) \coloneqq min_k\{\sigma(u) | u \Rightarrow v\} \tag{1-2}$$

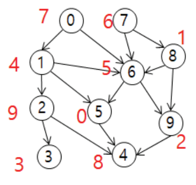
Here, $min_k\{A\}$ is defined as $|min_k\{A\}| \leq k$ that satisfies $A_i < A_j$ if $i < j$.

The label size of the 2-hop labeling scheme is defined in Definition 2.

Definition 2. (2-Hop Label Size) Assume we have a DAG G, a positive integer k, and a permutation mapping σ_i. The 2-hop label size of G with respect to k and σ_i is the summation of node IDs in $I^k_{\sigma_i}(v)$.

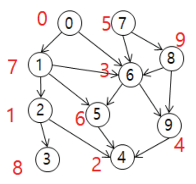
$$\sum_{v \in V} \left(\sum_{w \in L_{out}(v)} w + \sum_{q \in L_{in}(v)} q \right) \tag{2}$$

According to Definition 2, it is easy to see that the 2-hop label size depends on the choice of permutation mapping. Fig. 1 shows an example DAG that results in different 2-hop label sizes according to the choice of permutation mapping.



v	$L_{out}(v)$	$L_{in}(v)$
0	0,2,3	7
1	0,2,3	4,7
2	3,8,9	4,7,9
3	3	3,4,7
4	8	0,1,2
5	0,8	0,1,4
6	0,2,5	1,4,5
7	0,1,2	6
8	0,1,2	1,6
9	2,8	1,2,4

$size\left(I^3_{\psi_1}(V)\right) = 162$



v	$L_{out}(v)$	$L_{in}(v)$
0	0,1,2	0
1	1,2,3	0,7
2	1,2,8	0,1,7
3	8	0,1,7
4	2	0,1,2
5	2,6	0,3,5
6	2,3,4	0,3,5
7	2,3,4	5
8	2,3,4	5,9
9	2,4	0,3,4

$size\left(I^3_{\psi_2}(V)\right) = 139$

Fig. 1. Two different ways of assigning IDs to each node that result in different 2-hop label sizes. The red number beside the circles represents permutated IDs.

In Fig. 1, the permutation mapping on the right-hand side is better for the example graph in terms of reducing the 2-hop label size. In this example, for simplicity, we assume that k is 3, which means that each label contains at most three elements. As can be seen, there are smaller IDs in the right-hand graph. For example, consider the *in* label of node 2: the left-hand graph has 4,7,9 while the right-hand one has 0,1,7. Likewise, we have 7 for the *in* label of node 0 in the left-hand graph while we have 0 in the right-hand graph. The remainder of this paper is devoted to devising permutation mappings to reduce the 2-hop label size.

4. Combining Local and Global Features

This section explains our proposed approach to reduce the 2-hop label size. The 2-hop label size varies according to the choice of permutation mapping $\sigma(v)$. We can consider permutation mapping as the sorting of nodes in V. In other words, it is important to order nodes in such a way that each label contains the smallest ID possible. However, it is not easy to appropriately order nodes because the label of a node consists of *in* and *out* labels. These two labels contain node IDs in opposite directions. Both the *in* and *out* labels can only be reduced by considering the descendants and ancestors of the node.

The study in [13] tries to reduce the label size by the degree of each node, as defined in Definition 3.

Definition 3. (Labeling by Degree) For a given DAG $G(V, E)$, we create a permutation mapping $\sigma(v) = i$, where v is the vth element in V^d such that V^d is obtained by sorting V by decreasing degrees of v.

"Degree" here means the summation of incoming and outgoing edges. The rationale behind the degree approach is that the degree can be used to guess the total number of descendants and ancestors of a node. Descendant and ancestor numbers are related to the *out* and *in* labels, respectively.

According to Definition 3, a smaller number is assigned to $\sigma(v)$ if v has a bigger degree. A variation is possible, that is, a smaller number is assigned to $\sigma(v)$ if v has a smaller degree. However, the latter case does not make sense in general. We call the degree of each node the local feature because it does not account for the total number of descendants and ancestors in a given DAG. Rather, the approach only considers the number of neighbors of a node. The advantage of the approach is that it is easy to obtain the degree of each node. The drawback is that the degrees cannot capture the overall graph structure. The number of edges does not reflect the number of reachable nodes.

To address the disadvantage of the degree approach, the research in [12] proposed the idea of considering a global feature that utilizes the topological sort index, which is defined in Definition 4.

Definition 4. (Labeling by Topological Sort Index) For a given DAG $G(V, E)$, V^{ts} is an ordered set of sorted V obtained from the visiting order of the topological sort. We create a permutation mapping $\sigma(v) = i$, where v is the ith element in V^{ts}.

A topological sort is an ordering of nodes according to the visiting sequence from nodes with no incoming edges. We call the topological approach a global feature because the topological index of a node reflects the visiting history of a node with no incoming edges. The topological index represents the number of connected nodes from a node with no incoming edges. A higher topological index of a node means more nodes are reachable to the node. Unlike the degree approach, the topological sort approach can successfully capture reachability relationships.

A variation of the topological sort approach is possible. We could reverse V^{ts} to obtain V^{tsr} and create a permutation mapping $\sigma(v) = i$, where v is the ith element in V^{tsr}. As 2-hop label consists of *in* and *out* labels, the topological sort approach works better for one of them. Which one is better depends on the structure of the target DAG. This observation leads us to combine various features to create a permutation mapping. We combine the local and global features. We sort V by decreasing order of the value of a linear combination of degree and topological sort index. The formal definition is in Definition 5.

Definition 5. (Labeling by Degree and Topological Sort Index)

$$\alpha \times deg(v) + \beta \times top(v)$$

where $deg(v)$ is the degree of v, that is, the number of edges connected to v ; $top(v)$ is assigned the order index visited by a topological sort; α and β are the weights of the degree and topological sort indexes, respectively.

It should be noted that $top(v)$ ranges from 1 to $|v|$. In real-world cases, $deg(v)$ is much smaller than $|v|$. There are few cases where a node v is connected to every other node, which yields $deg(v) = |v| - 1$. Normally, a node is connected to several other nodes. To adjust this skew, α needs to be much bigger than β. We will examine the effect of β in experiments.

5. Experiments

Two previous approaches exist such which assume that top stands for topological sort (global feature) and deg stands for degree (local feature). The approach proposed here is called deg+top as a combination of degree and topological sort. We performed extensive experiments to check if deg+top can successfully reduce the 2-hop label size (according to Eq.(2)), compared to previous approaches. Experiments were carried out on a machine with a 3.4 GHz CPU and 64 GB RAM.

Two kinds of DAG datasets—synthetic and real-world—are listed in Table 1. Synthetic datasets are random DAGs. The number of nodes of synthetic DAG datasets range from 1,000 to 10,000. For each number of nodes, we generated 20 random DAGs, varying the number of edges. We had a total of 60 synthetic DAGs. The real-world datasets were obtained from [18].

Table 1. Synthetic and real-world DAG datasets

Name	Nodes	Edges
Random DAG	1,000	4,360–10,900
	5,000	21,800–54,500
	10,000	43,600–109,000
arxiv	6,000	66,707
go	6,793	13,361
PubMed	9,000	40,028

Experimental results against 60 synthetic datasets are depicted in Figs. 2–4. For deg+top, we only report the best weights for α and β. The effects of the weights will be discussed only for real-world datasets. For DAGs with a small number of edges, top was the best for random DAGs with 5,000 and 10,000 nodes. The effect of combining global and local features is not significant when the number of edges is small. When the number of edges becomes large, deg+top shows the best performance. The detailed analysis will be discussed against real-world datasets as follows.

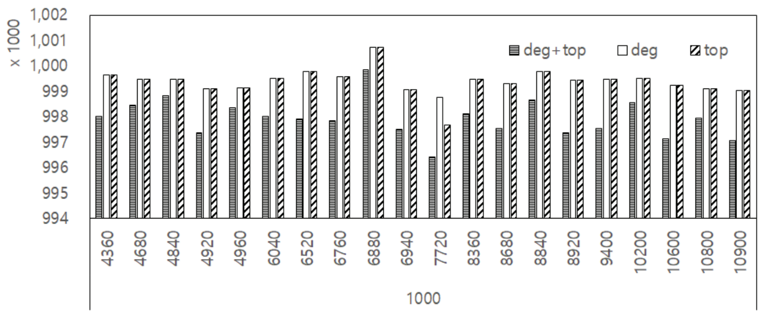


Fig. 2. Two-hop label sizes of random DAGs with 1,000 nodes. The X-axis represents the number of edges. For deg+top, only the best case of $\boldsymbol{\alpha}$ is depicted here.

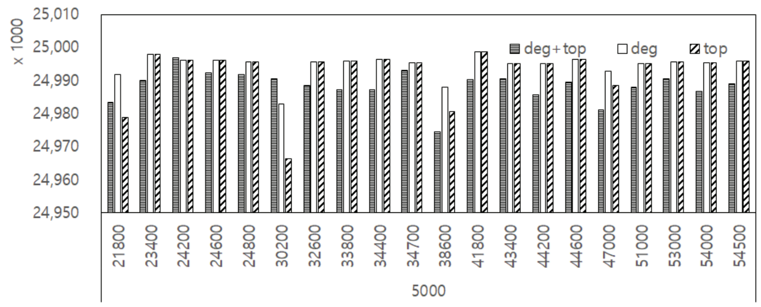


Fig. 3. Two-hop label sizes of random DAGs with 5,000 nodes. The X-axis represents the number of edges. For deg+top, only the best case of $\boldsymbol{\alpha}$ is depicted here.

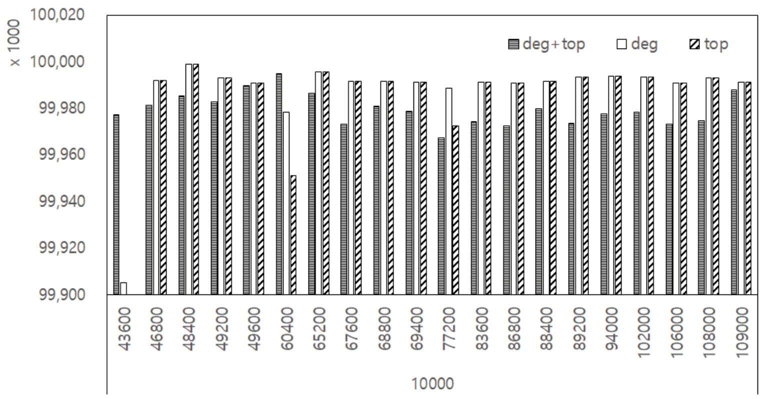


Fig. 4. Two-hop label sizes of random DAGs with 10,000 nodes. The X-axis represents the number of edges. For deg+top, only the best case of $\boldsymbol{\alpha}$ is depicted here.

Experimental results against real-world datasets are depicted in Fig. 5. The effects of the weights α and β will be discussed in Fig 6. For the arxiv dataset, let u be the last node visited by the topological sort. $top(u)$ would then become 6,000, which is much bigger than $deg(u)$ in that the average degree is 22.2 and the maximum is 700. To equally exploit both the degree and topological sort index, we would like to assign more weight to β. For the arxiv and pubmed datasets, deg+top generates smaller labels than top and deg. top generates the largest labels. The reason top is the worst is that the 2-hop label scheme is designed to construct both $L_{out}(v)$ and $L_{in}(v)$. Topological sort traverses a one-way direction. The size of either $L_{out}(v)$ or $L_{in}(v)$ would be reduced by the topological sort approach. However, it would not work successfully for reducing both $L_{out}(v)$ and $L_{in}(v)$. deg+top is worse than deg in the case of the go dataset. The reason is that go has smaller edges than the other datasets. The proposed approach would not work well for simple graphs as shown for the case of synthetic datasets above.

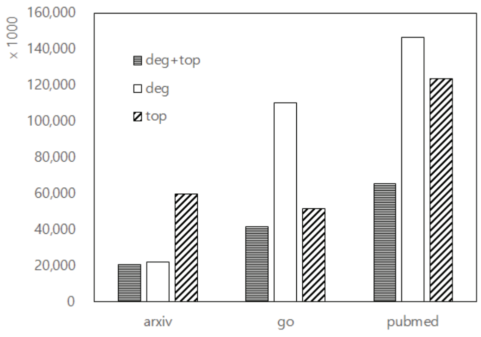


Fig. 5. 2-hop label sizes of real-world DAGs. For deg+top, only the best case of $\boldsymbol{\alpha}$ is depicted here.

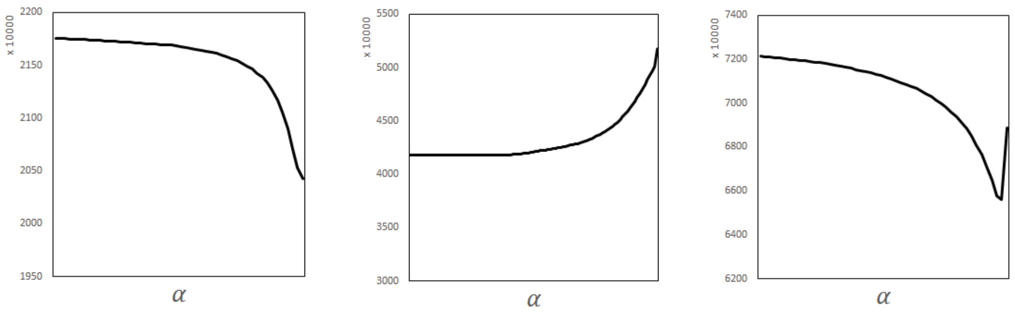


Fig. 6. Two-hop label sizes of real-world DAGs with deg+top while varying $\boldsymbol{\alpha}$. The values on the X-axis are represented in order from big to small; that is, the rightmost value is closest to zero. (a) arxiv, (b) go, and (c) PubMed.

Fig. 6 shows the effects of the parameters used in deg+top. Generally, β must be bigger than α to reduce the 2-hop label size because the degree of most nodes is smaller than the total number of nodes. The X-axis represents α, and the value in the rightmost graph is close to zero. Significantly different results were obtained with arxiv and go. For arxiv, the 2-hop label size becomes smaller when α becomes smaller and, at the same time, β becomes larger due to the constraint $\alpha + \beta = 1$. For go, the 2-hop label size becomes larger when α becomes smaller. The reason is the same as that mentioned above; that is, deg+top would not show better performance for simple graphs. For pubmed, the 2-hop label size decreases as α becomes smaller. However, no reduction is observed when α is too small. In other words, the 2-hop label size increases when α is close to zero. The reason is that some nodes in pubmed have many degrees (the highest degree is 55,758), which means that it is not a good idea not to consider the local feature (degree) for the reduction of the 2-hop label size when some nodes have large degrees.

6. Conclusion

In this paper, we propose a node id permutation approach to reduce the 2-hop label size. The characteristics of a node in a graph data structure was leveraged; local (degree) and global (topological sort index) features of each node were combined to determine a permutation mapping. Experiments with synthetic and real-world DAG datasets confirmed that the proposed approach produces smaller labels than previous approaches. The effects were significant when the number of edges were large. In future works, we plan to devise an approach that updates only a portion of an existing 2-hop label. To achieve the required efficiency, relabeling should be avoided when the target graph changes. To deal with massive graphs that cannot be loaded into a single machine, we will make use of distributed computing frameworks such as MapReduce and Spark.

Acknowledgement

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant No. NRF-2017R1C1B1003600; in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program Supervised by the Institute for Information & Communications Technology Promotion (IITP), under Grant No. IITP-2019-2018-0-01417; in part by the IITP Grant funded by the Korean Government (MSIP) under Grant No. R0113-15-0005; in part by the Development of a Unified Data Engineering Technology for Large-scale Transaction Processing and Real-Time Complex Analytics; in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant No. NRF-2018R1D1A1B07048380; and in part by the 2019 scientific promotion program funded by Jeju National University.

References

[1] R. Agrawal, A. Borgida, and H. V. Jagadish, "Efficient management of transitive relationships in large data and knowledge bases," *ACM SIGMOD Record*, vol. 18, no. 2, pp. 253-262, 1989.

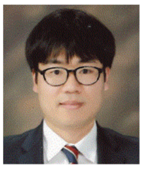
[2] H. Wei, J. X. Yu, C. Lu, and R. Jin, "Reachability querying: an independent permutation labeling approach," *Proceedings of the VLDB Endowment*, vol. 7, no. 12, pp. 1191-1202, 2014.

[3] H. Gabr and T. Kahveci, "Signal reachability facilitates characterization of probabilistic signaling networks," *BMC Bioinformatics*, vol. 16, article no. S6, 2015.

[4] D. H. Im, S. W. Lee, and H. J. Kim, "A version management framework for RDF triple stores," *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 1, pp. 85-106, 2012.

[5] S. M. Albladi and G. R. Weir, "User characteristics that influence judgment of social engineering attacks in social networks," *Human-centric Computing and Information Sciences*, vol. 8, article no. 5, 2018.

[6] K. Sriwanna, T. Boongoen, and N. Iam-On, “Graph clustering-based discretization of splitting and merging methods (GraphS and GraphM),” *Human-centric Computing and Information Sciences*, vol. 7), article no. 21, 2017.
[7] Y. Derdour, B. Kechar, and M. Faycal-Khelfi, “Using mobile data collectors to enhance energy efficiency and reliability in delay tolerant wireless sensor networks,” *Journal of Information Processing Systems*, vol. 12, no. 2, pp. 275-294, 2016.
[8] K. E. Moustafa and H. Hafid, “Self-Identification of Boundary's nodes in wireless sensor networks,” *Journal of Information Processing Systems*, vol. 13, no. 1, pp. 128-140, 2017.
[9] J. X. Yu and J. Cheng, “Graph reachability queries: a survey,” in *Managing and Mining Graph Data*. Boston, MA: Springer, 2010, pp. 181-215.
[10] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, “Reachability and distance queries via 2-hop labels,” *SIAM Journal on Computing*, vol. 32, no. 5, pp. 1338-1355. 2003.
[11] J. Su, Q. Zhu, H. Wei, and J. X. Yu, “Reachability querying: Can it be even faster?” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 3, pp. 683-697, 2016.
[12] J. Ahn, “Optimization techniques for 2-hop labeling of dynamic directed acyclic graphs,” in *Proceedings of the Doctoral Consortium at the 15th International Semantic Web Conference (ISWC 2016)*, Kobe, Japan, 2016, pp. 1-8.
[13] A. D. Zhu, W. Lin, S. Wang, and X. Xiao, “Reachability queries on large dynamic graphs: a total order approach,” in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Snowbird, UT, 2014, pp. 1323-1334.
[14] K. Simon, “An improved algorithm for transitive closure on acyclic digraphs,” *Theoretical Computer Science*, vol. 58, no. 1-3, pp. 325-346, 1988.
[15] G. Wu, K. Zhang, C. Liu, and J. Li, “Adapting prime number labeling scheme for directed acyclic graphs,” in *Database Systems for Advanced Applications*. Heidelberg: Springer, 2016, pp. 787-796.
[16] J. Ahn, T. Lee, and D. H. Im, “Efficiently answering reachability queries for tree-structured data in repetitive prime number labeling schemes,” *Applied Sciences*, vol. 8, article no. 785, 2018.
[17] H. Yildirim, V. Chaoji, and M. J. Zaki, “GRAIL: a scalable index for reachability queries in very large graphs,” *The VLDB Journal*, vol. 21, no. 4, pp. 509-534, 2012.
[18] S. Seufert, A. Anand, S. Bedathur, and G. Weikum, “Ferrari: flexible and efficient reachability range assignment for graph indexing,” in *Proceedings of 2013 IEEE 29th International Conference on Data Engineering (ICDE)*, Brisbane, Australia, 2013, pp. 1009-1020.



Jinhyun Ahn **https://orcid.org/0000-0002-2331-004X**

He is currently an assistant professor of the department of management information systems at Jeju National University. His research interests include distributed/parallel algorithms, graph processing, knowledge engineering, and data privacy. He received his B.S. and M.S. degrees in computer science education from Korea University, in 2005 and 2007, and Ph.D. degree in computer science and engineering from Seoul National University, in 2017.

Dong-Hyuk Im **https://orcid.org/0000-0002-0290-755X**

He is currently an associate professor of the department of computer and information engineering at Hoseo University. His research interests include database system, semantic web technology and big data processing. He received his B.S. degree in computer science education from Korea University, in 2003 and his M.S. and Ph.D. degrees in school of computer science and engineering from Seoul National University, in 2005 and 2011, respectively.