Received 30 January 2023, accepted 11 February 2023, date of publication 14 February 2023, date of current version 17 February 2023.
Digital Object Identifier 10.1109/ACCESS.2023.3244890

RESEARCH ARTICLE

Generative Service Provisioning for IoT Devices Using Line Graph Structure

JOOHYUN KIM[1] AND JAE-HOON KIM[1,2], (Member, IEEE)
[1]Department of Industrial Engineering, Ajou University, Suwon 16499, South Korea
[2]Department of AI Convergence Network, Ajou University, Suwon 16499, South Korea
Corresponding author: Jae-Hoon Kim (jayhoon@ajou.ac.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion (IITP) funded by the Korean Government (Ministry of Science and Information Technology) (Manufacturing Software (S/W) Platform Based on Digital Twin and Robotic Process Automation) under Grant 2021000292, and in part by the National Research Foundation of Korea (NRF) from the Korean Government (Ministry of Science and Information Technology) under Grant 2020R1F1A1049553.

ABSTRACT A service subgraph helps Internet-of-Things devices access resources in a dynamic Internet-of-Things device network. We propose a service subgraph generation method for Internet-of-Things device networks. Service subgraph generation aims to find more capable neighboring Internet-of-Things devices for service provisioning. We apply a line graph structure for an adequate representation of device resources. The line graph structure effectively represents the resources in the generated service subgraph. A general node classification problem constituting the generated service subgraph identifies the appropriate resource binding for service provisioning. A node in the service subgraph corresponds to a unique relationship between devices. Service provisioning is guaranteed by reinforcement learning based on the resource binding identified by node classification. The proposed line graph structure and resource binding significantly enhance the traditional intelligent resource allocation method. In addition, the proposed scheme can effectively attain service subgraphs with very low computational complexity. The proposed generative service provisioning generally has a significantly lower occupation probability than the swarm intelligence-based algorithm. The average value of the occupation probability is 0.49 with the proposed method. It is 0.12 lower than that of swarm intelligence-based algorithm.

INDEX TERMS Internet of Things, line graph, reinforcement learning, service provisioning, subgraph.

I. INTRODUCTION

In the Internet of Things (IoT) environment, a large pool of devices cooperates. Sensors and actuators in an IoT environment activate several IoT-based services. For example, a smart building collects data from thousands of sensors, and actuators, such as air conditioners, respond accordingly to sustain an appropriate environment. Another example of an IoT-based service is self-driving cars, which use various sensor data sources to make driving decisions. Conventional IoT services must cooperate with other services. A resource can be forwarded to the neighboring devices to satisfy the service requirements of a service requester. Resources from multiple devices can be combined in a small area. The devices in the IoT network can function as service requesters and resource providers simultaneously.

The network typically consists of many requesters and a relatively small number of providers. Service provisioning binds various primitive resources to meet the requirements of requesters [1], [2]. Service provisioning enhances resource utilization in various fields. In industry, predictions on services can be made to reserve resources [3]. The network slicing becomes gains efficiency by mapping resource status [4]. The quality of services (QoS) provided by primitive resources must be well coordinated in a single service provisioning to satisfy the service requests. A service requester typically has a list of desired services with different requirements. For each requester, service provisioning presents a pool of resource providers that can offer sufficient resources.

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

Requesters prefer providers that offer higher QoS. A graph is an appropriate representation of the provider pool [5]. Considering that all devices are connected in an IoT environment, the service-provisioning problem is equivalent to finding the best subgraph from an overall device network. A subgraph is a subset of the nodes and links selected from the original graph. Graph-based learning techniques are advantageous in complex applications. Recently, graph-based learning has focused on relational features between nodes or between links [5], [6], [7], [8]. Compared with the feature aggregation or concatenation of traditional graph-based learning, relational features increase data informativeness. Graph-based techniques have been applied to subgraph detection or subgraph generation.

In prior studies, the objective of subgraph detection or generation was to find a subgraph that could delegate the original graph. A frequently detected subgraph exhibits recurring or common features in the original graph. For example, a common connection between molecules can represent a specific chemical characteristic in the molecular structure. In IoT environments, subgraph detection is used to detect malware or botnets in an IoT network [8].

A provider pool for the service requester can be represented as a subgraph of the IoT network. A well-constructed service subgraph includes the most capable candidate providers for the requesters. Moreover, the subgraph enhances its usability by including the next candidate provider in the subgraph. The information required to construct a subgraph is the provider's resource status. A line-graph structure is useful for constructing a subgraph. A line graph converts node features into link features or vice versa. The link feature of the original graph contains the relational information (i.e., connectivity) of the two nodes. By applying a line graph, the node features in the line graph correspond to the relational information in the original graph. The line graph has the advantage of representing both the device's resources (i.e., CPU, storage, or sensors) and the relational information (i.e., connectivity between two devices). In [9], the link prediction problem was changed into a classification task by applying a line graph. The line graph generally has a smaller number of training parameters, which increases the learning efficiency and convergence speed of the problem.

Deep reinforcement learning (RL) implements a practical RL scheme for various problems. The cost-minimization problem with multistep decision-making is a typical deep RL problem. A recent study [10] used deep RL for network planning. Deep RL effectively prunes the search space to increase solution exploitation speed. A well-designed state representation is a critical factor for guaranteeing the performance of deep RL. Many previous studies have adopted a new state-representation framework [11], [12], [13]. In the literature, exploitation of the Markov decision process in graph representation outperformed matrix representation.

In this study, we proposed a generative service provisioning method, which uses deep RL and a line graph structure to address service provisioning in IoT networks. An IoT network comprises devices with different resources. These devices can be service requesters, resource providers, or both. Deep RL determines a specific subgraph from the original graph. We compared the proposed generative service provisioning with ant colony optimization (ACO), a swarm intelligence-based algorithm [14]. The ACO algorithm can identify a service subgraph. However, the proposed scheme exhibits faster convergence.

The remainder of this paper is organized as follows. The system model and problem formulation are presented in Sections II and III, respectively. Then, the proposed scheme is described in Section IV. In Section V, the results of the simulation and a discussion of the results are presented. Finally, the conclusions of this study are presented in Section VI.

II. SYSTEM MODEL

Consider an IoT network with N devices, M services, and Z resource types: the total set of devices $D_{total} = \{d_r\} \cup D$ where d_r denotes a service requester and D is a set of $N-1$ devices. A service subgraph is the feature graph of a requester. When we designate d_r as d_1 and let $D = \{d_2, d_3, \cdots, d_N\}$, the proposed model aims to find a service subgraph of a requester device d_1. The total set of services $S_{total} = \{s_0\} \cup S$ where $S = \{s_1, s_2, \cdots, s_M\}$ and s_0 defines "no service," which requires no resources. Each device can function as a requester, a provider, or both. We assume that Z fixed resource types exist in each IoT environment. A resource can be either a value-described resource (e.g., CPU, storage, RAM) or an existence-described resource (e.g., sensors and actuators).

A requester (d_r) requests service j with probability p_{rj}. We assume that all the services in S_{total} have an equal probability of occurrence. A requester has an individual list of preferred services (i.e., services can be classified as "preferred" or "not-preferred" by the requesters). Let a_{rj} be 1 if the service j is the preferred service of the requester (d_r), and 0 otherwise. The probability α_r, where the requester (d_r) requests any service, is as follows:

$$\alpha_r = \frac{\sum_j a_{rj}}{M+1} \tag{1}$$

where $r = 1, 2, \cdots, N$ and $j = 0, 1, \cdots, M$. We divide by $M+1$, which denotes the total number of M services and a "no-service."

At each time step t (i.e., iteration), the requester attempts to request a service. The system model assumes that a service forms a one-to-one or one-to-many relationship between a requesting device and a providing device. Similar to service preference in requesters, providers can "offer" or "refuse" services. The term "offer" refers to services that require fewer resources than the resources possessed by the devices.

Let $\boldsymbol{b_{pj}}$ be 1 if service $\boldsymbol{j}$ is offered by the provider ($\boldsymbol{d_p}$), and 0 if it is refused. Under this assumption, let $\boldsymbol{\beta_{pj}}$ be the probability of service $\boldsymbol{j}$ being provided by the provider ($\boldsymbol{d_p}$)

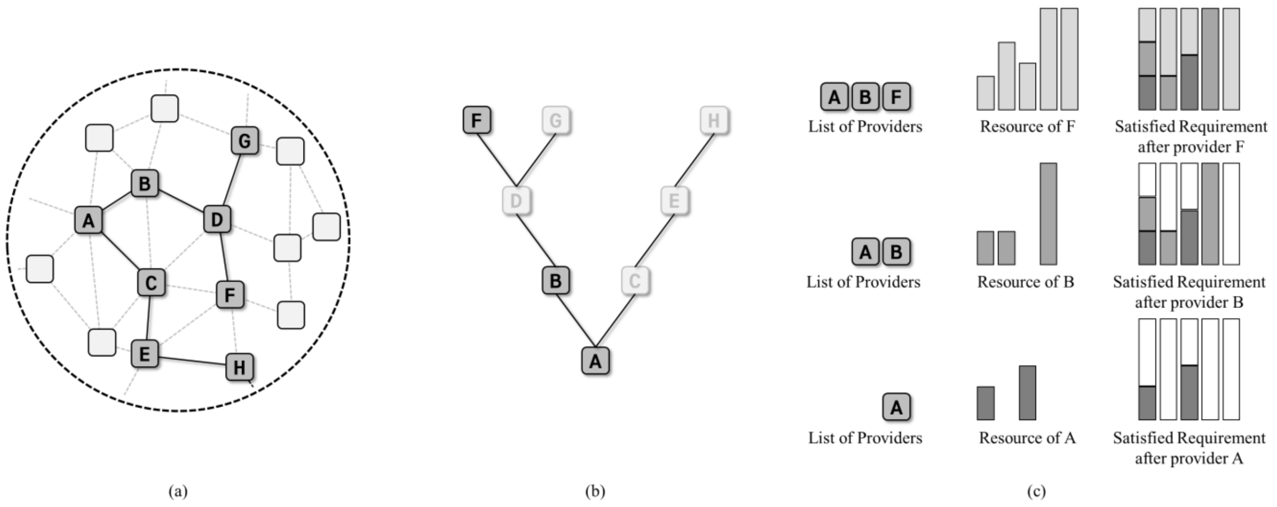
A
B
C
D
E
F
G
H
List of Providers
Resource of F
Satisfied Requirement after provider F
Resource of B
Satisfied Requirement after provider B
Resource of A
Satisfied Requirement after provider A
(a)
(b)
(c)

FIGURE 1. (a) A generated service subgraph in the Original Network, (b) A service subgraph of device A, (c) Process of resource-binding in the service subgraph using devices A, B, and F.

in the IoT environment. The probability $\boldsymbol{\beta_{pj}}$ is as follows:

$$\beta_{pj} = \frac{1}{\sum_j b_{pj}} \frac{\sum_p a_{pj}}{M+1} \tag{2}$$

where $p = 1, 2, \cdots, N$ and $j = 0, 1, \cdots, M$.

To model device occupation in the IoT environment, we adopted a probability distribution to predict the "unoccupied" devices. The term "unoccupied" refers to a device that does not operate any service and can provide future requested service. Let a random variable X_{pj} denote the service j offered by the provider (d_p). At each iteration, X_{pj} follows a binomial distribution with probability β_{pj}. As $n \rightarrow \infty$, the binomial distribution approximates a Poisson distribution. Let the random variable Y_p denote the occupation of the provider (d_p). Then, the random variable $Y_p = \sum_j X_{pj}$ follows a Poisson distribution.

Note that a device node acts as either a requester or provider. Whenever a service request is generated in the network, we identify the service requester (d_r) and search for resource providers (d_p). The device node resets whenever a service request is generated. A resource provider for a previous service request can be a service requester and vice versa.

III. PROBLEM FORMULATION

All devices have different available resources, and services have different service requirements. This heterogeneity complicates the service subgraph generation. A service subgraph is a tree of devices rooted in a requester. To find the service subgraph (SG$_{rj}$) of a requester (d_r) for service j, we make the following assumptions on the service subgraph:

- The requester itself is the first device that can provide service to the requester.
- A device can offer resources only once. The resources deplete after being offered. Hence, reexploring the device is meaningless.

A well-designed service subgraph allows a service requester to be assigned resource providers who offer desired services. The quality of service (QoS) for service j provided to a requester (d_r) is defined as $Q_{SG_{rj}}$. This value is determined by the product of the unoccupied probabilities of all resource providers in the service subgraph SG$_{rj}$ for service j. The service subgraph SG$_{rj}$ refers to the set of resource providers that are used to provide service j to requester d_r. The occupation of each provider (d_p) is represented by Y_p, and the unoccupied probability of each provider is represented by $(1 - Y_p)$. The product of the unoccupied probabilities of all providers in SG$_{rj}$, $\prod_{p \in SG_{rj}} (1 - Y_p)$, represents the overall unoccupied probability of the service subgraph for service j. This unoccupied probability is directly related to the resource availability in the service subgraph and is used as a measure of the QoS for service j provided to requester d_r.

$$Q_{SG_{rj}} = \prod_{p \in SG_{rj}} (1 - Y_p) \tag{3}$$

Note that, the average QoS achieved by the service subgraph for all services of requester (d_r) is calculated by equation (4).

$$Q_{SG_r} = \frac{1}{|S_r|} \sum_{j \in S_r} Q_{SG_{rj}} \tag{4}$$

where S_r is the set of services preferred by the requester (d_r). $|S_r|$ means the number of services in the set S_r.

We define all devices before the provider (d_p) as the predecessors of d_p. Other devices located after d_p are successors. A single provider can provide a service. However, we can assume a case in which no device can provide service alone. The service may require extremely high computational power or various types of sensor data. Service provisioning states that multiple devices cooperate in providing the requested service. The idea of service provisioning can help requesters provide services faster while satisfying QoS. With good service provision, a combination of low-performance devices can replace a single high-performance device. High-performance devices are more likely to be occupied by other service requesters. The resource-binding scheme incorporates its predecessors' available resources and is useful in determining available resource providers. Resource-binding scheme emulates the transpiration process in plants. Plants transport water from the roots to the leaves, which is required for photosynthesis. The leaf and root are the requester and potential provider, respectively. Similar to transpiration, the available resources are combined from the requester (root) to the providers (leaves) in the service subgraph to offer the service (transpiration).

Fig. 1 shows the service subgraph of a requester from the original graph and how to bind resources through the service subgraph. The resource provided to the device r for service j is the sum of the resources from unoccupied predecessors.

$$A_{SG_{rj}} = \sum\nolimits_{p\in SG_{rj}} (1 - Y_p)R_p \tag{5}$$

$A_{SG_{rj}}$ represents the combined resource of all devices in the service subgraph SG_{rj}. R_p represents the resources owned by each provider (d_p) in SG_{rj}, and $(1 - Y_p)$ represents the resource availability of provider (d_p) (remember that Y_p is the occupation probability of provider (d_p), and $(1 - Y_p)$ is the unoccupied probability). The product of R_p and $(1 - Y_p)$ represents the current resource status of provider (d_p). The sum of $(1 - Y_p)R_p$ for all providers in SG_{rj} represents the combined resource of all devices in SG_{rj}. In other words, it is the total resource available for use in SG_{rj}. The service subgraph quality can be obtained by computing (3), where the providers of each accepted service j are the devices that maximize $Q_{SG_{rj}}$ under a resource-binding scheme.

ACO-based algorithm shows effectiveness in resource allocation and task offloading problems. The basis for the ACO-based algorithm is that ants travel through different routes to accomplish certain objectives. The modification can be made by giving different constraints on traveling or pheromone update [15], [16].

The remaining problem is to determine a good service subgraph. The service subgraph connects the devices in an IoT environment. The heterogeneity of resources and devices increases the complexity of connections between devices. In the following section, we present the solution to finding a service subgraph with the idea of a line graph and deep reinforcement learning.

IV. GENERATIVE SERVICE PROVISIONING METHOD

The proposed generative service provisioning method focuses on generating a service subgraph. The service subgraph helps the requester find a set of adequate providers for all services. Service provisioning finds a group of devices

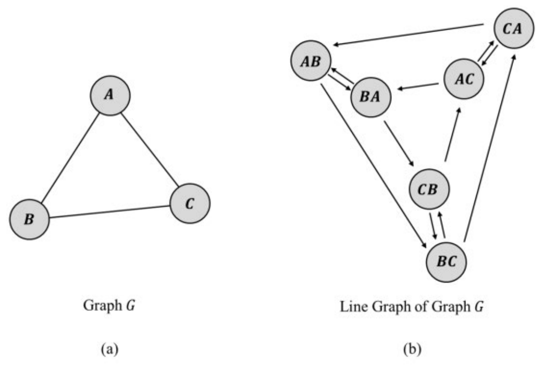

FIGURE 2. Transformation of original graph into a line graph form.

that offers a service in a graphical the form of a service subgraph. Another essential part of the generative service provisioning is resource-binding, which allows the device to gather available resources in the service subgraph. The proposed scheme is an iterative process consisting of two steps: the first step involves generating a service subgraph, and the second step involves evaluating the generated service subgraph. Equation (4) represents an evaluation metric for the generated service subgraph. Table 1 is the pseudo code of the proposed method.

In the proposed service provisioning, pairing devices should inform the resource difference. We used a line graph of the original graph for service subgraph generation. Every pair of devices in the original graph creates two nodes, indicating resource differences. For example, device pairs A and B in Fig. 2(a) are converted to nodes AB and BA, respectively, in Fig. 2(b). The line graph converts the link selection problem in the original graph G to a node classification problem in the line graph $L(G)$. If a node in the line graph is classified as "active," then the corresponding two devices and the directed link between the devices are selected for the service subgraph.

The link prediction problem requires combining the features of individual nodes before generating a subgraph. A common approach for this on the original graph is to use message passing, which combines the features of neighboring nodes to create an edge feature. However, the proposed line graph structure simplifies this process by replacing message passing with a simple subtraction operation. This results in a significant reduction in computation time. In the original graph, updating node features through message passing requires $O(n)$ computation for a single node (where n is the number of nodes in the network), and $O(kn)$ computation for k iterative updates. When applied to all nodes in the original network, the computational complexity becomes $O(kn^2)$. In contrast, the subtraction in the line graph only requires $O(1)$ computation for a single update, and $O(k)$ computation for k iterative updates. The number of subtractions needed for all nodes in the line graph (or edges in the original graph) is $O(km)$, where m is the number of nodes in the line graph. The conversion process from an original graph to a line graph may require additional computation, but the most advanced conversion methods have a computational

TABLE 1. Service subgraph generation pseudo code.

```
Algorithm: Service Subgraph Generation
Initialize
    $D_{total}$ : Device set
    $S_{total}$ : Service Set
    $a_{rj}$ : request( $r$ )'s preferred service( $j$ )index probability
    $b_{pj}$ : provider($p$)'s offered service($j$) index
    $Y_p$: occupation of provider ($p$)
Set
    $\alpha_r = \frac{\sum_j a_{rj}}{M+1}, r = 1,2,\cdots,N, j = 0,1,\cdots,M$
    $\beta_{pj} = \frac{1}{\sum_j b_{pj}} \frac{\sum_p a_{pj}}{M+1}, p = 1,2,\cdots,N, j = 0,1,\cdots,M.$
While
    Get line graph $L(G)$ of $D_{total}$ with the resource difference between devices
    Update node features of $L(G)$ using binary representation
    Use Deep RL to get subset of $L(G)$ as service subgraph
    While
        Evaluate service subgraph
    End while
    Update Deep RL and $D_{total}$
End while
END
```

complexity of $O(n)$. The overall feature update complexity for the line graph is $O(n) + O(km)$, which can be approximated as $O(km)$ when m is larger than n. In the most complex cases, m is equal to $n(n-1)$, but in typical communication networks, m is significantly less than $n(n-1)$. Therefore, the line graph structure can greatly reduce the amount of computation required.

Service subgraph generation is a node-classification problem for identifying active nodes in $L(G)$. The node features in $L(G)$ are the inputs for node classification and represent the resource difference between the devices. Feature subtraction is a simple but effective way to show the difference. However, simple subtraction still requires serialization in the standard form. The range and unit of each resource type are different (i.e., value-described resource or existence-based resource). The following insights are advantageous to operate as a practical input in a neural network for node classification:

- The providers in the service subgraph should have at least one better providable resource than the requester (i.e., higher-performance CPU or more storage).
- The diversity of resources must be more recommendable than the sufficiency of the resource.

Given these insights, we propose a representation method applied to generate the line graph $L(G)$. The representation of device features in $L(G)$ is a binary vector of size $(1 \times Z)$. The component of the binary vector is 1 if the node has a better providable resources and 0 otherwise. The device with a zero vector can be neglected because it has no available resources. The proposed representation method reduces the number of devices in $L(G)$. Additionally, the binary vector representation eliminates the need to standardize or form resources in advance of learning. The binary vector representation reduces the complexity of the problem.

TABLE 2. Deep RL model.

Agent	Node
State	Resources (for Provider Node)
Action	Activation/Deactivation (for Connections between Devices)
Reward	QoS Value

One of the candidate methods for service subgraph generation is the naïve service subgraph generation. The method has a severe problem caused by the mutually connected devices. The mutually connected devices generate a cyclic service subgraph between them. Effective service provisioning is complex owing to the presence of cycles. Hence, we designed a modified service subgraph generation method that includes resource binding. Modified service subgraph generation was performed using deep RL. The neural network used to classify the active nodes in the line graph should be updated to generate an effective service subgraph. Deep RL is a technique that updates neural networks using a reward function. We used the average quality value of the generated service subgraphs defined in (5), as the reward. On average, deep RL updates the neural network to be effective for all service requesters. The service subgraph enables all service requesters to have a similar QoS.

Deep RL uses different service requests from all requesters to develop a single neural network. The rewards may use average quality value or utility-based rewards such as the providers' occupation. The occupations differ in each service subgraph. Some requesters may have a service subgraph with a high occupation but a low occupation for other requesters. The average QoS of all generated service subgraphs determines the overall performance and develops the neural network with general knowledge.

The deep RL model uses the resources of each device as the input feature which is the state in deep RL. The action of the deep RL agents is to activate or deactivate the node, which is the connection between the devices. Then, the deep RL model evaluates the overall learning through the average QoS of the generated service subgraph (see the equation (3) to calculate QoS). The Deep RL descriptions are represented in Table 2.

The resources of these devices are not constant and can vary over time. In order to take this fluctuation into account, we apply the time variation to R_p and Y_p in equation (5). Specifically, we define $R_p(t)$ as the resource of the provider (d_p) at iteration t, where $R_p(t)$ represents the original resource owned by d_p in the IoT system model. The formulation for $R_p(t)$ can be represented as follows:

$$R_p(t) = (1 - Y_p(t))R_p(0) \qquad (6)$$

The product of the unoccupied probability, represented as $(1 - Y_p(t))$, and the original owned resource $(R_p(0))$ represents the current resource status at iteration t. The first step in this iteration involves classifying the nodes of the line graph generated from resource $R_p(t)$. In the second step, the resource is updated to $R_p(t+1)$. At each iteration t, each device generates a service subgraph. It is worth noting that the service subgraph, SG_{rj}, can be redefined as $SG_{rj}(t)$ to account for the iteration index t.

The neural network for each iteration classifies the active nodes of the service subgraphs presented in a line graph form. The reward is the average quality of all service subgraphs. The neural network is updated in response to the reward. Reinforcement learning using line-graph-based node classification generates an actual service subgraph from the original graph.

The objective of the proposed deep RL model is to use fewer parameters while maintaining a good performance. It is also important to find resource-binding to provide services in a distributed manner. We compared the proposed method with an ACO-based swarm intelligence-based algorithm in the same environment model. A swarm intelligence-based algorithm develops the collective behavior of life groups. Swarm intelligence has been applied to many domains and routing problems [14], [17], [18]. ACO (Ant Colony Optimization) is an effective Swarm Intelligence-based algorithm that uses the concept of ants using pheromones to guide the other ants. The ACO-based algorithm was used to address routing problems in wireless sensor networks [14]. In the literature, the trailing ant can obtain promising perspectives from the leading ant using pheromones. The pheromones indicate route quality. If trials with different levels of pheromones are left to determine routes to the prey, the pheromone-based solution addresses the presence of desired resources in the service subgraph. We modify the method in the literature [14] for comparison. In the proposed system model, the ant travels through the device until sufficient resources are accumulated. The pheromone level is the probability of selecting a device. We define the pheromone level ρ_u of a link in u-th travel as follows (see Li et al. [18] equation (9)~(10)):

$$\rho_{u+1} = \gamma \rho_u + (1-\gamma)x \tag{7}$$

where γ is the pheromone fading rate, and $\gamma = 0.9$. The variable x is 1 if the device is chosen for travel u. For each travel u, the unoccupied devices are initialized following the probability defined in Section II. The ant selects an untraveled device with respect to ε-greedy algorithm. If the selected device is unoccupied, the ant accumulates resources from it. The ant repeatedly selects the next device until it accumulates enough resources to provide all the requested services. The pheromone level is updated according to the travel route of the ant. Note that in u-th travel, the ant uses the most recently updated pheromone map. The pheromone map is a service subgraph generated by the ACO algorithm. With sufficient travel, the tree of links with high pheromone levels can be regarded as a service subgraph. We assume that the service subgraph consists of links with an above-average pheromone level.

TABLE 3. Resource types in the designed environment.

Types	Resource	Range
Value-described	CPU	1.6 GHz ~ 2.4 GHz (Quantified to 1.6, 1.8, 2.0, 2.2, 2.4 GHz)
	Storage	256 GB ~ 1024 GB (Quantified to 256, 512, 1024 GB)
	RAM	4 GB ~ 32 GB (Quantified to 4, 8, 16, 32 GB)
Existence-described	Temperature sensor	0 (not equipped), 1 (equipped)
Types	Resource	Range
	Humidity sensor	0 (not equipped), 1 (equipped)
	Location sensor	0 (not equipped), 1 (equipped)

Equation (3) denotes overall quality metric of generated subgraph. The overall quality metric consists of quality achievement of individual services (equation (4)). The constraint of service provisioning is the resource provided by devices. Equation (5) denotes the combined resource of devices in the generated subgraph. To calculate the resource provided in the subgraph, we must monitor the current status of available resource of devices. Equation (6) denotes the resource availability according to time flow. Additionally, equation (7) indicates the pheromone level in each iteration for the ACO-based service provisioning.

V. EXPERIMENTAL RESULTS

The system model depicts an IoT environment consisting of N devices, M services, and Z resources. Each service's parameters, N, M, and Z, for each service influence the model complexity. We present the simulation results to demonstrate the performance excellence of the proposed methods. We constructed an IoT environment using devices and services. The constructed IoT environment was randomly generated from pre-defined perspectives. We limited resource variations by assuming the possible range of each resource. Table 3 presents the resource types used for the proposed model and ACO-based model. We present six resource types in the designed IoT environment (individual resources belong to one resource type). The resources of each device were initialized based on the resource types. Then, the actual resource is selected with a pre-defined probability (for example, a 1.6 GHz CPU with 0.5, a 2.0 GHz CPU with 0.3, and a 2.4 GHz CPU with 0.2).

The resource difference between service providers was used as the input of the neural network. Quantification to form a binary representation improves the convergence speed. The value-described resource was quantified in several steps.

TABLE 4. Service types in the designed environment.

Name of Types	Specifications					
	Value-described			Existence-described		
	CPU	Storage	RAM	Temperature	Humidity	Location
GPS	2.4	1024	32	0	0	1
Environment Monitoring	2.4	256	8	1	1	1
Video Streaming	2.2	512	8	0	0	0
Smart Air Conditioning	2.0	512	4	1	0	0
Data Computation	2.0	256	32	0	0	0
Humidity Monitoring	1.8	1024	4	0	1	0
Temperature Forecasting	1.8	512	8	1	0	0
Humidity Forecasting	1.8	256	16	0	1	1
Smart Farm Monitoring	1.6	512	4	1	1	0
Database	1.6	256	4	0	0	0

It is practical to describe the actual resource of IoT devices. Existence-described resources are naturally represented in binary form.

The simulation testbed comprised 100 IoT devices. Each testbed constituted randomly generated resources as defined in Table 3. The service types are presented in Table 4. We generated 100 service requests using randomly selected service requesters. The learning iterations were limited to 200 iterations. At each iteration, the reinforcement learning uses 10 random samples for the generated service requests. Note that each device is initialized following the resource types in Table 3.

Fig. 3 demonstrates the average size of the service subgraphs generated by the two methods. The size of the service subgraph generated by the ACO-based model fluctuated over a relatively long time (see Fig.3 (b)). The ant has less knowledge until it converges on the pheromone map. The ACO-based model is a random-walk model [19]. The proposed generative service provisioning succeeds in determining the required resources for every graph search. Fig. 3 (a) illustrates rapid convergence to the ideal size of the service subgraph. The service subgraph determination accelerates as the learning progresses. The learning speed is critical for achieving higher RL performance. The learning speed can be regulated to balance the subgraph finding time and size of the subgraph.

The service subgraph generation latency is the time spent to make the service subgraph. Shorter latency refers to the faster service provisioning. Fig. 4 depicts the variance of latency for each method. The average latency of the proposed method is 1.70 ms, while the ACO-based service provisioning requires 3.54ms to make the service subgraph. The latency pattern demonstrates stepwise increment. The proposed method needs about 2ms when the subgraph size

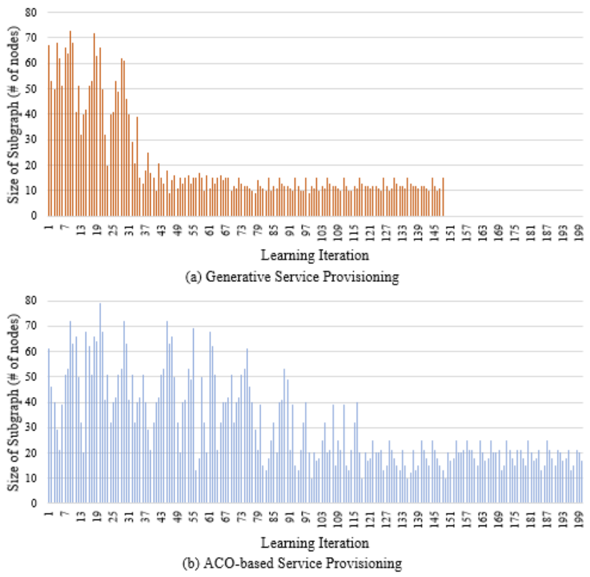

FIGURE 3. Average depth of the service subgraph.

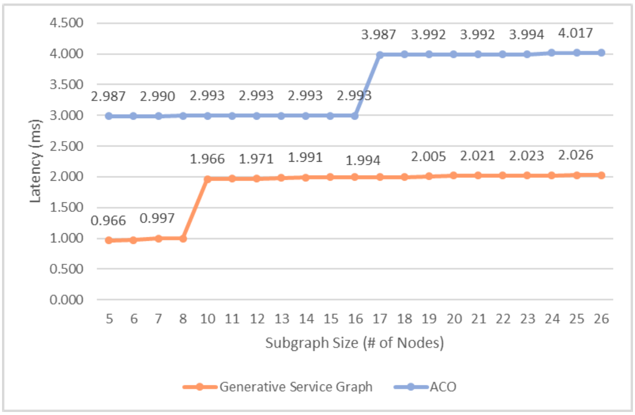

FIGURE 4. Latency of the service subgraph.

reaches more than 10 nodes while the ACO needs 4ms from 16 nodes.

Fig. 5 shows the number of assigned services per resource provider in the service subgraph. The proposed generative service provisioning illustrates the focused device participation in service provisioning. Approximately 80% of resource providers support less than five service requests at the same time. In ACO-based service provisioning, 60% of service providers support 6∼10 service requests simultaneously. Generative service provisioning has an advantage in managing relatively simplified resource-binding. The resource-binding from the small number of service providers induces a small number of service support of resource providers, which leads to simple resource finding for future service requests.

During the initial stages of learning, it is inevitable that the unoptimized search and activation of connections, performed by the deep RL algorithm, may result in longer paths to reach the appropriate resource providers. The QoS, which is the product of the unoccupied probabilities of all providers in the service subgraph, is maximized when the path to

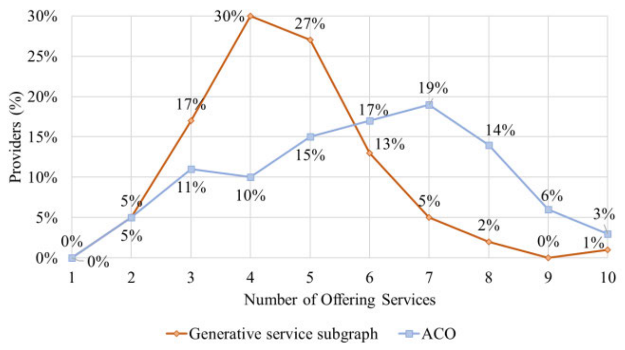

FIGURE 5. Ratio of service providers in service subgraphs generated by the generative service provisioning (red) and the ACO-based provisioning (blue).

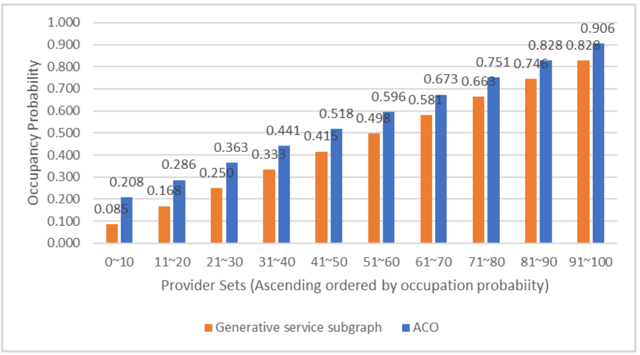

FIGURE 6. Distribution of occupation probability for providers in service subgraphs generated by the generative service provisioning (red) and the ACO-based provisioning (blue).

reach the providers is shorter. Even the addition of a single node to the path can cause a decrement in the QoS value. As the unoccupied probability of a node is less than one, the multiplication value of unoccupied probabilities for QoS calculation (i.e., $\prod_{p \in SG_{rj}} (1 - Y_p)$ in equation (3)) decreases. However, with proper training, the deep RL algorithm ensures the maximization of QoS values for each requester. Once the deep RL training reaches a converged stage, search paths are minimized then every service subgraph achieves its minimum size. The providers in the generated service subgraph are classified to occupied providers. The optimized service subgraphs for service requesters in the network ensures the smallest number of device participation to support service requests.

The availability of a service subgraph is related to the occupation level of the resource providers. If the occupation probability Y_p is high for the providers in the service subgraph, the service subgraph is likely to fail to satisfy a service request.

Fig. 6 shows the distribution of occupations (i.e., Y_p). Compared to ACO-based service provisioning, the proposed generative service provisioning has a lower load for each service provider. The average value of the occupation probability (Y_p) is 0.49 with the proposed method. It is 0.12 lower than that of ACO-based provisioning.

The proposed generative service provisioning method generally has a lower occupation probability compared to the ACO-based provisioning method. We have divided the resource providers into ten sets based on their ascending order of occupation probability, as shown in Figure 6. The lowest set, consisting of resource providers with the bottom 10% of occupation probability, has a 0.2 difference in occupation probability between the two service provisioning methods. Although the differences become smaller as we move up the provider sets, the differences in occupation probability are still observed for all provider sets. The provider sets with the top 10% of occupation probability show a difference of 0.078.

The proposed generative service provisioning uses line graph structure to ensure the low complexity for problem resolution. The proposed method effectively makes tree of service providers to inform the requester in finding appropriate provider for the requested services. The generated service subgraph consists of providers that are less likely to be occupied. The low occupation probability indicates the chance of having sufficient resource for the service.

VI. CONCLUDING REMARKS

We introduced a service subgraph generation process using a line-graph structure and deep RL. A service subgraph is a tree of devices for better service offerings. Line graphs reduce the complexity of the subgraph generation problem. The search space to determine the available providers decreases, and the feature representation becomes easier to learn. The proficiency of the service subgraph increases with deep RL. RL learns and changes the service subgraph to create additional unoccupied devices. A service subgraph is a group of devices organized in a tree-like form for service provisioning. A well-constructed service subgraph is effective in environments with low-performance devices. We apply resource binding to the generated service subgraph to combine the resources of the low-performance devices. The proposed service subgraph generation method was compared with the ACO-based method. The proposed method outperformed the ACO-based method in terms of the convergence speed and availability of the service subgraph. An important characteristic of an IoT environment is its scalability. The proposed generative service provisioning should be effective in environments with many devices. An additional characteristic to consider is heterogeneity. The term ‘‘heterogeneity’’ can be derived from many aspects. Diversity of services can result in heterogeneity. The RL design implies the qualification of the best service subgraph in a heterogeneous IoT network environment.

Additional work is suggested. First, we used a probabilistic model to design an IoT environment in the proposed experiment. However, real-world applications have anomalies, noise, and randomness. Using a more realistic model to capture the behaviors of real devices would make the proposed model more applicable. Second, the proposed model can be improved by implementing multi-agent RL. The proposed model predicts the service provision of other devices with probability. By implementing multi-agent RL, the model can learn the dynamics of co-existing devices in the environment, which can prevent encounters with the same provider.

REFERENCES

[1] N. Kashyap, A. C. Kumari, and R. Chhikara, "Service composition in IoT—A review," in *Intelligent Data Communication Technologies and Internet of Things*, vol. 38. Cham, Switzerland: Springer, 2019.

[2] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Service composition approaches in IoT: A systematic review," *J. Netw. Comput. Appl.*, vol. 120, pp. 61–77, Oct. 2018.

[3] H. Yang, J. Yuan, C. Li, G. Zhao, Z. Sun, Q. Yao, B. Bao, A. V. Vasilakos, and J. Zhang, "BrainIoT: Brain-like productive services provisioning with federated learning in industrial IoT," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 2014–2024, Feb. 2022.

[4] H. Yang, A. Yu, J. Zhang, J. Nan, B. Bao, Q. Yao, and M. Cheriet, "Data-driven network slicing from core to RAN for 5G broadcasting services," *IEEE Trans. Broadcast.*, vol. 67, no. 1, pp. 23–32, Mar. 2021.

[5] M. Gorawski and K. Grochla, "Graph representation of linear infrastructure in smart city IoT systems," in *Proc. 11th Int. Congr. Ultra Modern Telecommun. Control Syst. Workshops (ICUMT)*, Oct. 2019, pp. 1–4.

[6] A. Khanfor, A. Nammouchi, H. Ghazzai, Y. Yang, M. R. Haider, and Y. Massoud, "Graph neural networks-based clustering for social Internet of Things," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 1056–1059.

[7] H. Vulchi, R. P. Leal, and A. G. Armada, "Graph based interference analysis and resource allocation in mmWave IoT networks," in *Proc. IEEE-APS Topical Conf. Antennas Propag. Wireless Commun. (APWC)*, Sep. 2019, pp. 269–271.

[8] A. A. Abusnaina, H. Alasmary, M. Abuhamad, S. Salem, D. Nyang, and A. Mohaisen, "Subgraph-based adversarial examples against graph-based IoT malware detection systems," in *Computational Data and Social Networks (CSoNet)*, vol. 11917. Cham, Switzerland: Springer, 2019.

[9] L. Cai, J. Li, J. Wang, and S. Ji, "Line graph neural networks for link prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5103–5113, Sep. 2022.

[10] O. Rivlin, T. Hazan, and E. Karpas, "Generalized planning with deep reinforcement learning," 2020, *arXiv:2005.02305*.

[11] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *Comput. Commun.*, vol. 196, pp. 184–194, Dec. 2022.

[12] W. Jiang, "Graph-based deep learning for communication networks: A survey," 2021, *arXiv:2106.02533*.

[13] V. Waradpande, D. Kudenko, and M. Khosla, "Deep reinforcement learning with graph-based state representations," 2020, *arXiv:2004.13965*.

[14] A. Sharmin, F. Anwar, and S. M. Motakabber, "Efficient and scalable ant colony optimization based WSN routing protocol for IoT," *Adv. Sci., Technol. Eng. Syst. J.*, vol. 5, no. 6, pp. 1710–1719, 2020.

[15] X. Liu and G. Zhang, "Joint optimization offloading and resource allocation in vehicular edge cloud computing networks with delay constraints," in *Proc. IEEE Int. Conf. Prog. Informat. Comput. (PIC)*, Dec. 2020, pp. 363–368.

[16] Z. Shafahi and A. Yari, "An efficient task scheduling in cloud computing based on ACO algorithm," in *Proc. 12th Int. Conf. Inf. Knowl. Technol. (IKT)*, Dec. 2021, pp. 72–77.

[17] C. Shin and M. Lee, "Swarm-intelligence-centric routing algorithm for wireless sensor networks," *Sensors*, vol. 20, no. 18, p. 5164, Sep. 2020.

[18] X. Li, B. Keegan, and F. Mtenzi, "Energy efficient hybrid routing protocol based on the artificial fish swarm algorithm and ant colony optimisation for WSNs," *Sensors*, vol. 18, no. 10, p. 3351, Oct. 2018.

[19] F. Xia, J. Liu, H. Nie, Y. Fu, L. Wan, and X. Kong, "Random walks: A review of algorithms and applications," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 2, pp. 95–107, Apr. 2020.

JOOHYUN KIM received the B.S. degree in industrial engineering from Ajou University. His research interests include the IoT networking, intelligent network operation, and blockchain networks. He has developed reinforcement learning methodology and IoT network simulator.

JAE-HOON KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in management science from the Korea Advanced Institute of Science and Technology (KAIST), in 1996, 1998, and 2003, respectively. He has developed various network operation frameworks and simulation test-beds. Also, he has experience on the mobile service design and strategy. He worked as a System Architect in wireless systems with Samsung Electronics and a System Engineer with SK Telecom, South Korea. Currently, he is a Faculty Member of Industrial Engineering with Ajou University, South Korea. His research interests include the IoT networks, blockchain platforms, and ubiquitous networks. He is a member of the IEEE Communications Society and the IEEE Vehicular Technology Society.