# Network Partitioning Problem for Effective Management of Multi-domain SDN Networks

Hidenobu Aoki, Norihiko Shinomiya
Graduate School of Engineering
Soka University
Tokyo, Japan
Emails: aoki39h@gmail.com, shinomi@soka.ac.jp

***Abstract*—In Software-Defined Networking, a network with distributed controllers can be partitioned into sub-networks as controller's administrative domains. Network partitioning could affect various aspects of network performances, such as controller load and reliability of controller's domains because network resources and topology are logically divided into sub-networks. By focusing on network partitioning, this paper handles the issue as a mathematical problem based on graph clustering and analyzes effective network partitioning methods with different indicators related Software-Defined Networking. The simulation results indicate the effectiveness of our clustering method in terms of the load balance of controllers and the reliability of controller domains.**

***Keywords*—*Software-Defined Networking; distributed controllers; network partitioning; graph clustering.***

## I. Introduction

This paper extends the conference paper presented in SOFT-NETWORKING 2015 [1], which evaluates clustering algorithms from diversified standpoints of the network partitioning in Software-Defined Networking (SDN).

SDN has been emerging as a new networking paradigm. The fundamental concept of SDN is to achieve programmable networking by separating the control and the data planes in an individual network device, such as a switch and router [2]. In an SDN network, a controller is in charge of generating data forwarding rules. In contrast, network devices in the data plane are responsible for forwarding data according to the rules. This centralized architecture where a controller manages network devices enables network operators to dynamically configure network devices and to flexibly manage their networks. Currently, its applications have been extended to various types of networks, such as campus, datacenter, and carrier networks [3].

However, it has been discussed that a single controller has raised scalability and reliability issues. In large-scale networks, the load could converge on a single controller even though its processing capacity could be limited [4]. Moreover, if a failure occurs on a single controller, an entire network managed by the controller could take a risk of breakdown [5]. Furthermore, in wide-area networks, it could cause communication latency because some switches may be located far away from the controller. As a result, it might not be feasible to process events requiring real-time operations, such as failure recovery [6]. To handle those issues on a single controller, it has been studied to deploy multiple controllers over a network as one of main SDN-related research topics [7][8].

In such an SDN network with distributed controllers, there are mainly two types of control: the hierarchical and flat controls [9]. The hierarchical control is to organize controller's functions vertically. For example, a function dealing with flow setup messages or maintaining the states of network devices on data plane while another exchanges network information with other controllers to keep a global view of a network.

In contrary, the flat control is to partition a network into sub-networks, and controllers are assigned to one of them as their administrative domains. Because of the network partitioning, each sub-network can be managed independently by a controller. Hence, it could reduce the overall complexity of the whole network management and the computational load of each controller as well as handling flow setup requests faster and more efficiently. Moreover, it would be preferable to limit the scope of network operations for the deployment of new technology or infrastructure, such as adding or relocating network devices. Furthermore, when a controller failure occurs, it could alleviate to spread its negative effects to the rest of the network [10].

On partitioning a network, various aspects of the network performance could be affected since network resources and topology are logically divided into sub-networks. For instance, the load of controllers would be regarded as one of the major issues. Generally, the controller load is originated from network provisioning and status collection overhead within a domain and collaboration load among controllers, and these overheads could depend on how network resources are distributed in sub-networks [11]. Additionally, it would be necessary to consider the reliability of sub-networks as well as the controller load. In reality, controllers will be located in the same position as switches, and the control and data messages are flowed through the same communication links in the in-band control model. As a result, it would be desirable to partition a network so that switches in each sub-network have multiple paths reaching to controllers and other switches in case of link failures.

Therefore, this paper focuses on the network partitioning which is related to the flat control. In order to analyze its effective way, we provide four clustering algorithms and evaluate them based on the different indicators associated with

SDN networks.

The organization of this paper is as follows: Section II introduces the related work of the hierarchical control of distributed SDN controllers and network partitioning. Section III explains the layered architecture of control plane and addresses the issues of network partitioning. Section IV provides the definitions of the graphs and formulates Network Partitioning Problem (NPP). Section V details clustering algorithms as solutions of network partitioning. Section VI describes the simulation results and discussions, and Section VII concludes this paper.

## II. Related Work

In this section, the related work of the hierarchical control of distributed SDN controllers and network partitioning are presented.

### A. Hierarchical Control Plane

Onix [12] describes network topology as a graph. Each partitioned network is contracted to a logical node and used as a unit to share network information among controllers. This enables a controller to communicate with other controllers without knowing specific network states and topology of other partitioned networks. In this way, the reduction of the amount of network information possessed by a controller can be achieved.

Kandoo [13] provides a hierarchical control method consisting of the root controller and some local controllers. The root controller manages all local controllers and is responsible for the events which requires information over the whole network. On the other hand, local controllers deal with the local events, such as flow setups and network statistics collections of a local network. This layered control defines the scope of operations to process different requests efficiently, which could offload the burden of the root controller.

Although the ideas of hierarchical control plane have been proposed in those researches, they do not discuss how to partition a network to decide controller's domains.

### B. Network Partitioning

On partitioning a network, the major issue to consider would be the controller load. The controller load is generally thought to be the overhead to configure and mange network devices within domains. Simply, the more switches a controller needs to configure, the heavier load it is imposed on the controller. Thus, it has been studied to partition a network aiming at balancing the controller load by equalizing the number of switches in domains [14][15].

Contrary, Yao et al. in [16] argues that the importance of switches should be considered for load balance of controllers because it should differ from each other depending on networks. In the research, weight of the switch is given as the degree of nodes representing the number of flow setup requests required by the switch. Although the research considers the new metric for load balance of controllers, the focus of the research is where to place controllers to satisfy the metric and does not discuss how to partition a network in detail.

On the other hand, connectivity of a network is considered as one of the objectives for the controller placement problem [17][18]. Those researches aim to maximize the number of disjoint paths between a controller and switches or between switches so as to ensure the connection between them in case of link failure. Nevertheless, their approaches do not focus on network partitioning methods but controller locations to enhance reliability of domains.

Our previous work [1] proposes a network partitioning method whose objective is to minimize the number of inter-domain links in terms of the reduction of the controller load in sharing topology information. However, the work does not evaluate the load balance of controllers and reliability of controller domains. Thus, as the extension of our previous work, this paper provides clustering algorithms and evaluates the load balance of controllers, the reliability of each domain as well as inter-control load.

## III. Layered control plane

This section discusses the architecture of the layered control plane and addresses the issues on the network partitioning.

### A. Definitions of Two-tier Control Plane

In an SDN network with multiple controllers, the network can be logically partitioned into sub-networks as controller domains. In each domain, a controller is mainly in charge of two roles: (1) control of switches in own domain and (2) federation of a whole network by communicating with other controllers. As a result, control plane can be layered in two tiers: the local and the federation tiers are responsible for (1) and (2), respectively.

### B. Network Topology in Local and Federation Tiers

In the local tier, the local control function abstracts and possesses the network topology of each administrative domain as a local graph. Moreover, the local graph is contracted to a single node, which is used as a unit of communication with other controllers. In the federation tier, the global control function gathers the contracted nodes from all controllers and aggregates them to form a federation graph, which describes global network topology. Note that edges in a federation graph correspond to the edges between local graphs, which means inter-domain links are recognized as global topology information. Due to this topology contraction, it can be expected to reduce the amount of global network information shared among controllers.

Figure 1 illustrates an example of the layered control plane. In Figure 1, there are two domains described as local graphs 1 and 2. On the other hand, in the federation tier, the federation graph has two contracted nodes, and three edges correspond to the edges between the local graphs.

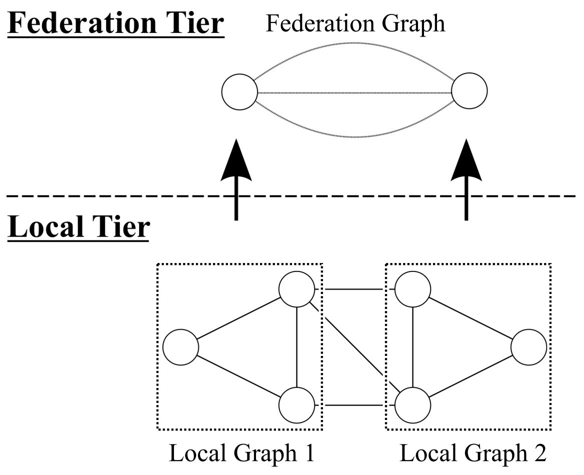


Fig. 1. Topology contraction in layered control plane.

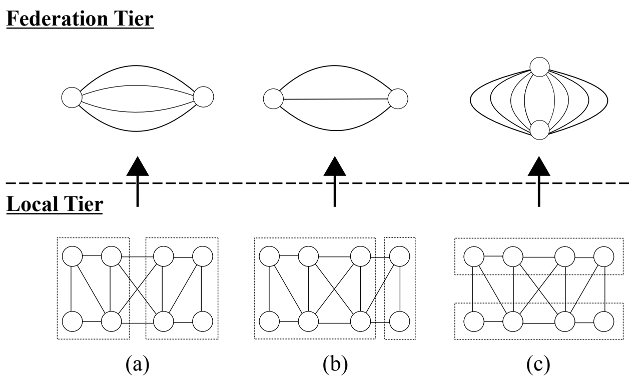


Fig. 2. Examples of network partitioning.

### C. Issues on Network Partitioning

Because a network partitioning decides the distribution of network resources and topology into sub-networks, it would have an influence on the aspects of network operations and performances in SDN [14].

Figure 2 illustrates examples of the network partitioning in different ways. In Figure 2, comparing (a) and (b), there are four nodes in each domain in (a) although the domains in (b) contain different number of nodes: 6 and 2 nodes, respectively. This implies that the balance of the controller load to manage switches is determined by network partitioning.

Moreover, there are four edges in the federation graph (a) while the federation graph (c) has eight edges even though the domains in both (a) and (b) contain the same number of nodes. This might indicate the increase of the amount of shared information and collaboration overhead among controllers.

Additionally, although there are multiple paths between any pairs of nodes in domains of (a), there is only a single path connecting any pairs of nodes within domains of (c). As a result, when a link failure occurs on a link in the domains (c), even the communication to a switch in the same domain has to be via a path crossing another domain. This requires the extra inter-controller communication which leads to additional load on controllers.

As those examples imply, network partitioning would be an important issue to address for SDN networks. Therefore, this paper defines the problem to determine controller domains as Network Partitioning Problem (NPP) and analyzes effective way of the network partitioning.

## IV. Problem Formulation

This section describes the graph definitions related to the layered control plane and formulates NPP.

### A. Definitions

For a network graph $G = (V, E)$, a set of vertices $V$ denotes network devices, such as routers and switches, and a set of edges $E$ represents links between those devices. Considering the network partitioning, sub-networks called local graph are denoted as

$$G_1^l = (V_1^l, E_1^l), G_2^l = (V_2^l, E_2^l), \ldots, G_k^l = (V_k^l, E_k^l). \qquad (1)$$

Note that we assume that a node can exclusively belong to a local graph. In the federation tier, on the other hand, a federation graph is defined as

$$G^f = (V^f, E^f), \qquad (2)$$

where $V^f$ indicates a set of the contracted nodes of the local graphs, and $E^f$ represents that of the edges between the local graphs.

### B. Clustering

In graph theory, clustering is a fundamental problem in mathematics and the applied science, which classifies the data into groups or categories. Graph clustering is defined as a task of grouping nodes in a graph into subsets called clusters [19]. Based on graph clustering, suppose that a local graph $G_i^l$ in (1) is a cluster, and a set of local graphs $\mathbf{G^l}$ is denoted as a clustering:

$$\mathbf{G^l} = \{G_1^l, G_2^l, ..., G_k^l\}. \qquad (3)$$

Generally, the desirable clustering is defined that there are many edges within each cluster called intra-cluster edges and relatively few edges between clusters referred to inter-cluster edges. Considering the network topology treated in the layered control plane, intra-cluster edges correspond to the edges in $E_i^l$, and inter-cluster edges are equivalent to the edges in $E^f$. In addition, a clustering having the less number of inter-cluster edges and the more number of intra-cluster edges is regarded as a preferable one [20].

### C. Problem Formulation

In this paper, the primal objective of the network partitioning is to balance the load of controllers. The load of controllers would be composed of several factors, such as the management of local domains and the communication with other controllers for a global control. Among them, to handle flow setup requests might be one of the major loads of controllers because the potential bottlenecks of a network will be bandwidth, memory, and processor of controllers when they receive many flow setup requests at a time [21]. In general, the amount of flow setup requests to a controller could increase depending on the number of switches it has to manage. However, since the importance of switches in a domain would be distinguished from each other, not only the number of switches in a domain but also the importance of the switches may also need to be considered as stated in [16]. Hence, as a measure of the importance of switches, this paper takes account of the switch weight. In our model, it is abstracted as the degree of nodes because the number of flow setup requests required by a switch could be related to its connection with other devices.

Here, for a graph $G$, let the weight of node $u$ be given as its degree, $deg(u)$. Then, the total node weights in a domain is denoted as

$$D(G_i^l) = \sum_{u \in G_i^l} deg(u). \tag{4}$$

In order to evaluate the balance of intra-control load among controllers, the standard deviation of node weights in domains is calculated as follows:

$$\mathbf{G}_\sigma^\mathbf{l} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(D(G_i^l) - \overline{D(\mathbf{G}^\mathbf{l})})^2}. \tag{5}$$

Note that $\overline{D(\mathbf{G}^\mathbf{l})}$ stands for the mean value of the sum of node weights in a cluster, which is calculated as $\overline{D(\mathbf{G}^\mathbf{l})} = \frac{\sum_{u \in V} deg(u)}{k}$. Therefore, the objective function of NPP is to find a clustering $\mathbf{G}^\mathbf{l}$ such that $\mathbf{G}_\sigma^\mathbf{l}$ is minimized.

## V. Clustering Algorithms

In this section, four clustering algorithms are provided as solutions of NPP.

### A. Conductance Clustering

As one of clustering indices, conductance has been defined, which compares the sum of the number of inter-cluster edges and that of all edges yielded by a clustering [22]. By denoting a set of all edges that have their origin in $G_i^l$ and their destination in $G_j^l$ as $E(G_i^l, G_j^l)$, conductance of $G_i^l$ is denoted as

$$\Phi(G_i^l) = \frac{|E(G_i^l, \mathbf{G}^\mathbf{l} \setminus G_i^l)|}{\min(D(G_i^l), (D(\mathbf{G}^l \setminus G_i^l))}, \tag{6}$$

where $D(G_i^l)$ is the sum of node degree in $G_i^l$ as defined in (4), and $D(G_i^l, \mathbf{G}^\mathbf{l} \setminus G_i^l)$ is that of other clusters. Note that a cluster with smaller conductance represents a better cluster.

In general, finding a clustering with minimum conductance is known as NP-hard [20]. Hence, as proposed in [1], we construct Conductance clustering that chooses nodes one by one based on the conductance value shown in Algorithm 1. The algorithm begins with a random node. Then, one of neighbor nodes of the node, which the cluster obtains the smallest conductance value, is chosen. As this process, it expands the cluster by recursively choosing a neighbor node of the nodes in the cluster. If the number of nodes in the cluster reaches to an upper bound of the number of nodes in a cluster $\frac{|V|}{k}$, it starts again to create a new cluster with a random node, which has not belonged to any clusters.

**Algorithm 1** Conductance Clustering.

```
Input: a graph $G$=$(V, E)$, the number of node limitation $\frac{|V|}{k}$
 1: Clustering $\mathbf{G^1} \leftarrow \phi$
 2: $V' \leftarrow$ a list of nodes in a graph $G$
 3: while $V' \neq \phi$ do
 4:     $G_i^l \leftarrow \phi$
 5:     Choose a node $v_r$ from $V'$ at random
 6:     Add $v_r$ to $G_i^l$ and remove $v_r$ from $V'$
 7:     while the number of nodes in $G_i^l < \frac{|V|}{k}$ do
 8:         Find a neighbor node $v_n$ of nodes in $G_i^l$
            which minimizes $\Phi(G_i^l)$
 9:         Add $v_n$ to $G_i^l$ and remove $v_n$ from $V'$
10:     end while
11:     Add $G_i^l$ to $\mathbf{G^1}$
12: end while
Output: $\mathbf{G^1}$
```

### B. Spectral Clustering

Spectral clustering has been applied for various fields of data analysis [23][24]. It is a method to cluster data by using eigenvectors of the Laplacian matrix of a graph. The Laplacian matrix describes the structure of a graph where diagonal components represent the node degree, and others describe the adjacency relationship between corresponding nodes.

Here, suppose that the adjacency matrix of a graph $G$ be defined as $A_G$ with its elements determined by

$$A_G(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

On the other hand, the degree matrix of $G$ is described as $D_G$ which is a diagonal matrix composed of

$$D_G(u, u) = deg(u). \tag{8}$$

By using $A_G$ and $D_G$, the Laplacian matrix of $G$ is denoted as

$$L_G = D_G - A_G. \tag{9}$$

As described in Algorithm 2, Spectral clustering computes $L_G$ of $G$ and its eigenvectors. Then, based on the eigenvectors, it obtains a clustering by $k$-means clustering algorithm.

*1) k-means Clustering:* In Step 5 of Algorithm 2, $k$-means clustering is used, which is one of the most commonly used clustering algorithms [25]. For the $k$ initial cluster centers obtained in Step 4 of Algorithm 2, each node of a graph is assigned to one of the closest cluster centers so as to satisfy the following function:

$$\text{Minimize} \sum_{i=1}^{k} \sum_{u \in G_i^l} |u - c_i|^2. \tag{10}$$

**Algorithm 2** Spectral Clustering.

```
Input: a graph $G$=($V, E$), the number of domains to create $k$.
1: Compute the Laplacian matrix $L_G$
2: Obtain the first $k$ eigenvectors $u1, ..., u_k$ of $L_G$
3: Let $U \in \mathbf{R}^{n \times k}$ be the matrix containing the vectors $u1, ..., u_k$ as columns.
4: For $i = 1, ..., n$
   let $y_i \in R^k$ be the vector corresponding to the $i$-th row of $U$.
5: Cluster the points $y_i \in \mathbf{R}^k$
   with $k$-means clustering algorithm into cluster, $G^l_1, ..., G^l_k$.
Output: $\mathbf{G^l}$
```

*C. Betweenness Centrality Clustering*

As one of metrics for graph analysis, the betweenness centrality has been defined [26]. It implies how an edge is associated with the shortest paths between every pair of nodes in a graph. Thus, edges with high betweenness centrality are regarded to be important in the graph.

Here, the edge betweenness centrality is given by

$$C_B(u, v) = \sum_{s \neq u \neq v \neq t} \frac{\sigma_{st}(u, v)}{\sigma_{st}}, \tag{11}$$

where $\sigma_{st}$ is the total number of shortest paths from node $s$ to node $t$, and $\sigma_{st}(u, v)$ is the number of those paths that pass through edge $(u, v)$.

In general, the edges connecting different groups tend to have high edge betweenness centrality. Thus, by removing these edges, the underlying group structure of a graph would be revealed [27]. Based on the idea, Girvan and Newman have developed a clustering algorithm shown in Algorithm 3. The algorithm begins with an initial cluster $C_i$ containing all nodes in $G$. Then, it computes $C_B(u, v)$ for all edges in $C_i$ and removes the edge with the highest $C_B(u, v)$ recursively until $C_i$ is disconnected. For the larger one of two yielded clusters, start the edge removal process again, and this process is repeated until the number of clusters reaches to $k$.

*D. Repeated Bisection Clustering*

As an application method of $k$-means clustering, Repeated bisection clustering has been studied. Generally, it is known as a faster and more accurate method than $k$-means clustering [28]. As stated in Algorithm 4, it recursively separates a graph into two clusters by assigning each node to a closer cluster. Note that the Step 2 in Algorithm 4, we select the nodes with minimum and maximum ID in $C_i$ as two random nodes. Additionally, the distance of two nodes is defined as the shortest path length between the nodes.

## VI. Simulations and Results

In this section, the simulation settings and results are presented. We have developed a simulator in Python and NetworkX to conduct our simulations and evaluate the clustering algorithms presented in Section V.

**Algorithm 3** Betweenness Centrality Clustering.

```
Input: a graph $G$=($V, E$), the number of domains to create $k$.
1: Start with an initial cluster $C_i$ containing all nodes in $G$
2: while True do
3:   while $C_i$ is connected do
4:     Compute $C_B(u, v)$ for all edges in $C_i$
5:     Remove the edge with the highest $C_B(u, v)$
6:   end while
7:   if $|\mathbf{G^l}| - 2 = k$ then
8:     Add two yielded clusters to $\mathbf{G^l}$
9:     break
10:  else
11:    For two yielded clusters, add the smaller cluster to $\mathbf{G^l}$ and let the larger cluster be $C_i$.
12:  end if
13: end while
Output: $\mathbf{G^l}$
```

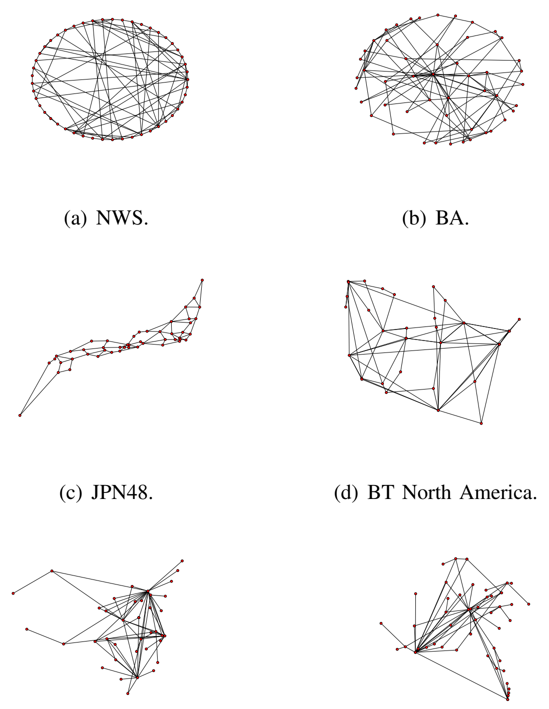
**Algorithm 4** Repeated Bisection Clustering.

```
Input: a graph $G$=($V, E$), the number of domains to create $k$.
1: Start with $C_i$ containing all nodes in $G$.
2: while True do
3:   Select two random nodes $v_a$ and $v_b$ from $C_i$ and create clusters $C_a$ and $C_b$ containing $v_a$ and $v_b$, respectively.
4:   for each node in $C_i$ do
5:     Calculate the distance with $v_a$ and $v_b$ and assign the node to the closer cluster.
6:   end for
7:   if $|\mathbf{G^l}| - 2 = k$ then
8:     Add $C_a$ and $C_b$ to $\mathbf{G^l}$
9:     break
10:  else
11:    For $C_a$ and $C_b$, add the smaller cluster to $\mathbf{G^l}$, and let the larger cluster be $C_i$
12:  end if
13: end while
Output: $\mathbf{G^l}$
```

*A. Network Models*

Our simulation makes use of two types of random graphs and four kinds of real network models. As random graphs, Newman Watts Strogatz (NWS) and Barabasi Albert (BA) are used because they are flexible to adjust their sparseness and denseness for theoretical analyses. A NWS graph is formed by connecting random pairs of nodes with a certain probability after creating a ring over $n$ nodes, which tends to be sparse [29]. In contrast, a BA graph is generated where nodes with higher degree have higher probability to be connected to a node during its generation process [30].

In addition to NWS ans BA graphs, the clustering algorithms are executed on the real network models: JPN48, BT North America, China Telecom, and Bell South provided in [31][32]. Figures 3(a) to 3(f) depict the network models used in our simulations, and Table I describes the parameters of the network models, such as the number of nodes, edges, and average degree.



(a) NWS. (b) BA. (c) JPN48. (d) BT North America. (e) China Telecom. (f) Bell South.

Fig. 3. Network models for simulation.

TABLE I
SIZE OF NETWORK MODELS.

| | NWS | BA | JPN48 |
|---|---|---|---|
| # of nodes | 50 | 50 | 48 |
| # of edges | 62 | 100 | 82 |
| Average degree | 2.48 | 3.88 | 3.41 |

| | BT North America | China Telecom | Bell South |
|---|---|---|---|
| # of nodes | 36 | 42 | 51 |
| # of edges | 76 | 66 | 66 |
| Average degree | 4.22 | 3.14 | 2.58 |

## B. Evaluation Indicators

In our simulations, three different indicators are evaluated while the number of controller domains is varied.

*1)* **Load balance of controllers:** As mentioned in Section IV-C, the standard deviation of the switch weight in domains $\mathbf{G}^1_\sigma$ is evaluated to examine the load balance of controllers.

*2)* **Inter-control load:** The amount of topology information shared among controllers could be related to Inter-controller load since they need to synchronize network information they possess to keep a global view. As stated in Section III-C, the amount of shared topology information corresponds to the size of federation graph $|G^f|$ in the layered control plane. However, since the number of nodes in a federation graph $|V^f|$ is equivalent to the number of controller domains, $|G^f|$ varies depending on the number of the edges in a federation graph $|E^f|$, which is the number of inter-domain links.

*3)* **Reliability of controller domains:** It might be desirable for any switches to have multiple paths reaching to other switches within the same domain in case of failures; otherwise, the extra inter-controller communication might be required as addressed in Section III-C. Thus, as a indicator to evaluate the reliability of controller domains, the average number of edge disjoint paths for all pairs of nodes in a domain is compared.

Assuming that the number of edge disjoint paths between nodes $u$ and $v$ is given as $\psi_{uv}$, the average number of edge disjoint paths in a domain is denoted as

$$\Psi(G_i^l) = \frac{\sum_{u,v\in E_i^l}\psi_{uv}}{{}_{|V_i^l|}C_2}. \qquad (12)$$

Then, the average number of edge disjoint paths of a clustering as a whole is calculated as

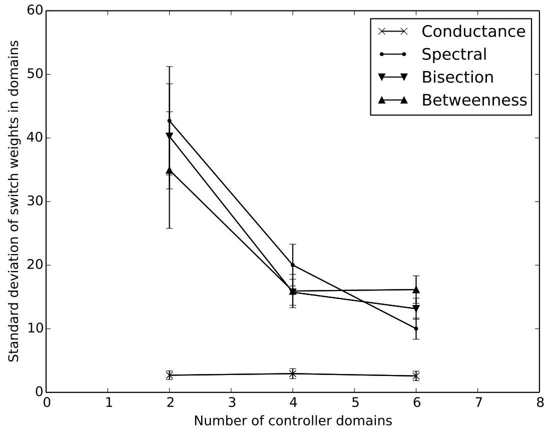
$$\overline{\Psi(\mathbf{G^1})} = \frac{\sum_{G_i^l\in\mathbf{G^1}}\Psi(G_i^l)}{k}. \qquad (13)$$

## C. Results and Discussions

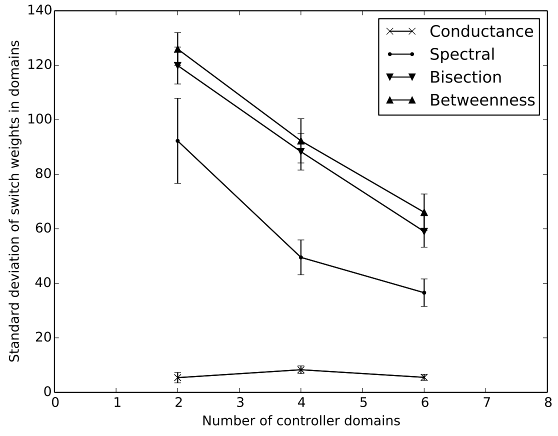
This section presents the simulation results and discussions. In our simulations, the number of controller domains is varied from 2 to 6 in all simulations, and each simulation is executed 20 times.

*1)* **Load balance of controllers:** Figures 4(a) to 4(f) illustrate the standard deviation of the number of the switch weight in domains representing the load balance of controllers. We can see from Figures 4(a) to 4(f) that Conductance clustering demonstrates better than any other algorithms for balancing the controller load. This would be because Conductance clustering generates a cluster until the number of nodes in the cluster reaches to the constraint of the number of nodes in a cluster, which is set as the number of nodes in an original graph divided by the number of controller domains. As a consequence, most clusters include almost the same number of nodes as the constraint even though last-produced cluster may contain less number of nodes compared with others. This depends on whether the number of nodes in an original graph can be divisible by the number of controller domains.

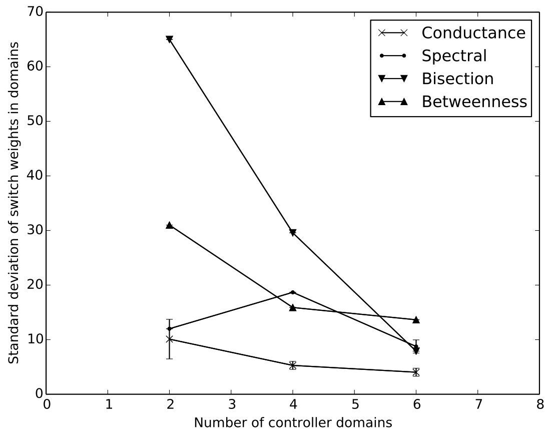
*2)* **Inter-control load:** Figures 5(a) to 5(f) indicate the number of inter-domain links implying the inter-control load. Those results show that Spectral clustering could generate clusters with less number of inter-domain links on NWS, JPN48, BT North America, and Bell South. On the other hand, Betweenness centrality clustering performs better on BA and China Telecom. As Spectral clustering separates a graph based on graph connectivity, it generally produces dense clusters and sparse relations among them, which yields less number of inter-domain links. However, the reason why Betweenness centrality clustering works better on BA and China Telecom would be because of their structural feature. Both models include several hub nodes, and the edges connecting such nodes



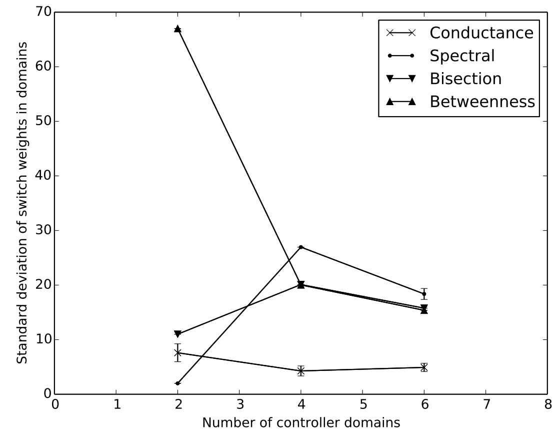
(a) NWS.



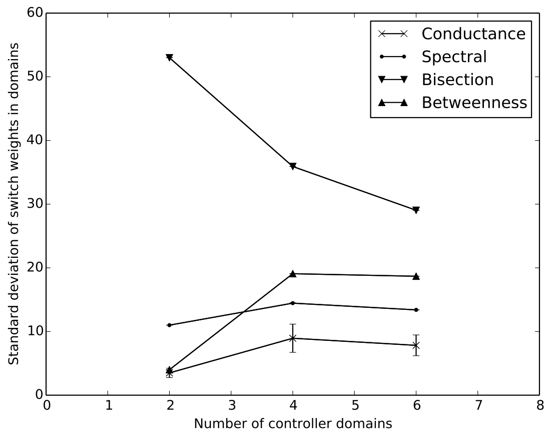
(b) BA.



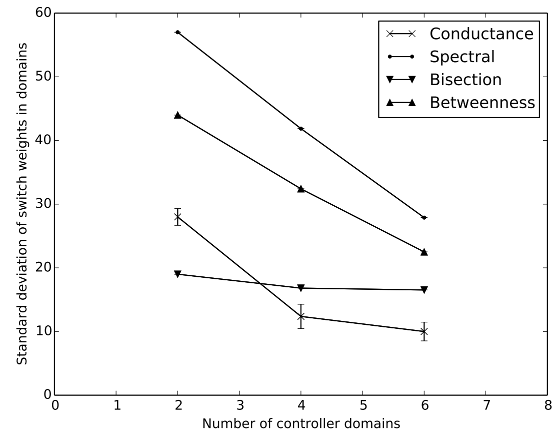
(c) JPN48.



(d) BT North America.

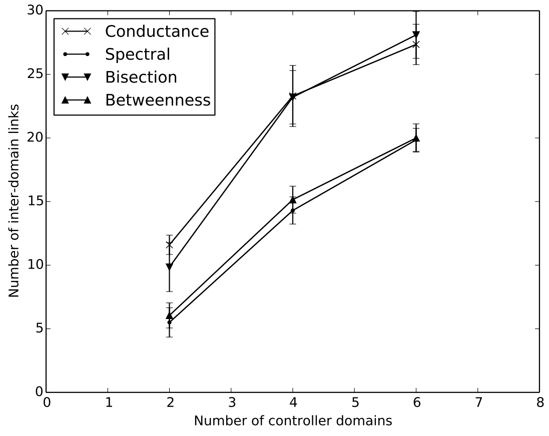


(e) China Telecom.

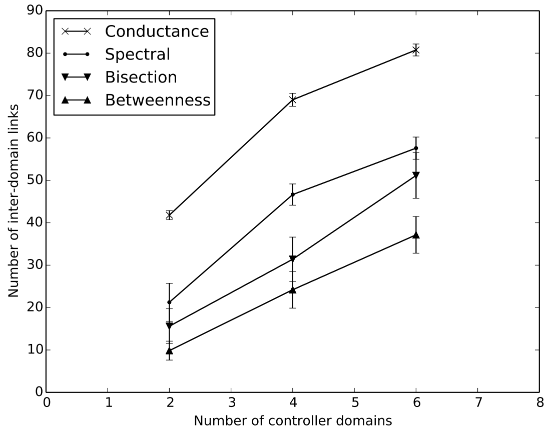


(f) Bell South.

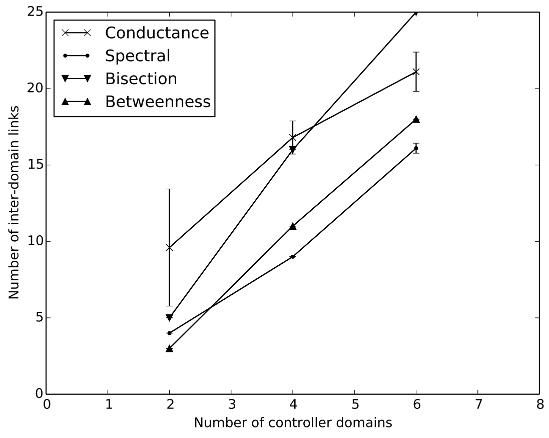
Fig. 4. Load balance of controllers.



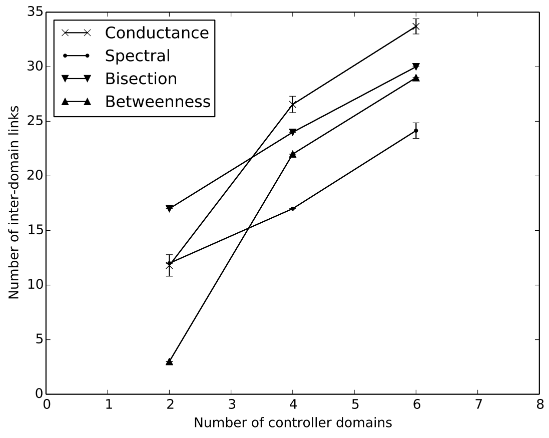
(a) NWS.



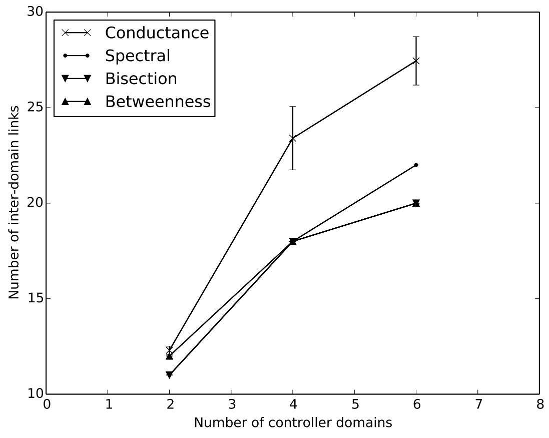
(b) BA.



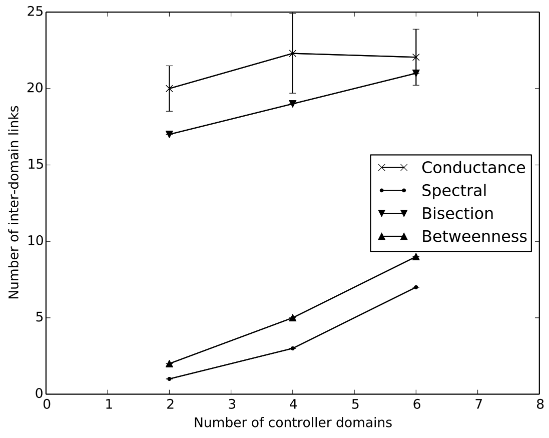
(c) JPN48.



(d) BT North America.

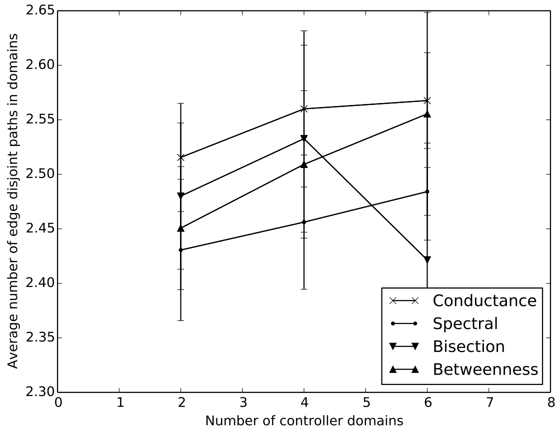


(e) China Telecom.

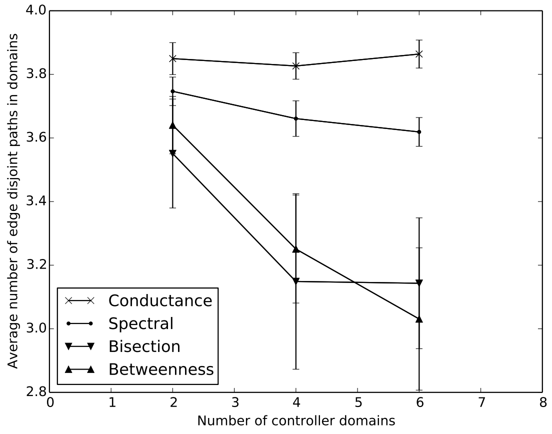


(f) Bell South.

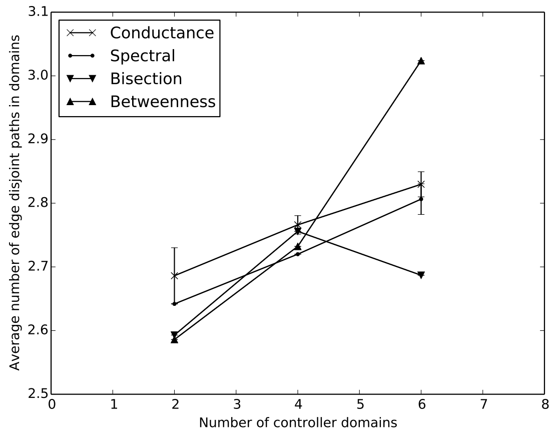
Fig. 5. Inter-control load.



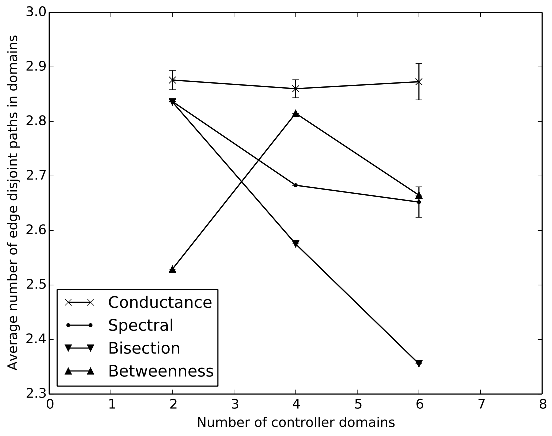
(a) NWS.



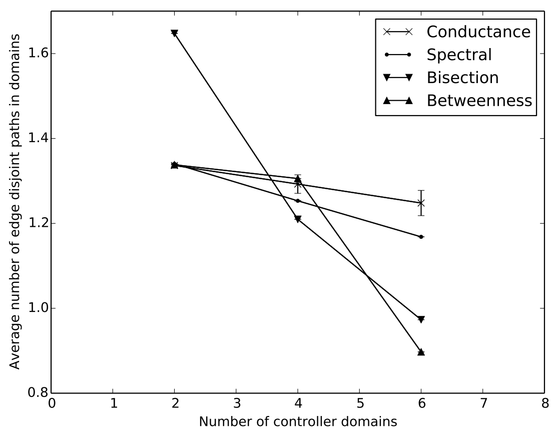
(b) BA.



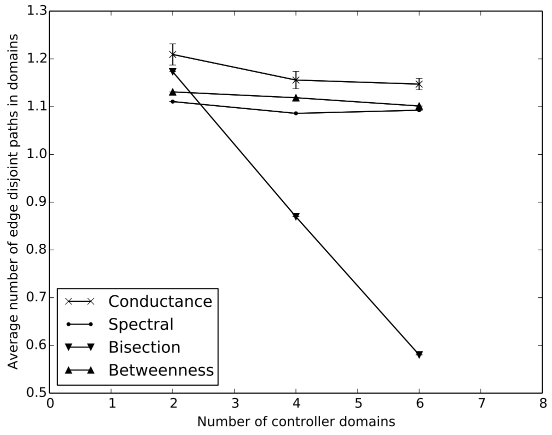
(c) JPN48.



(d) BT North America.



(e) China Telecom.



(f) Bell South.

Fig. 6. Reliability of domains.

would have high betweenness centrality. Thus, Betweenness centrality clustering can reveal the sub-structure of the graph by relatively a few edge removals. As a result, it may end up to produce the less number of inter-domain links.

*3)* **Reliability of domains:** Figures 6(a) to 6(f) depict the average number of edge disjoint paths in domains. As we can see from those results, Conductance clustering yields clusters with higher average number of edge disjoint paths on most of network models. Conductance clustering produces a cluster by choosing a node one by one so that the density of the cluster becomes high. As a result, it may increase the possibility to have many edge disjoint paths between nodes since the algorithm tries to contain many edges within each cluster.

In Figure 6(d), Betweenness centrality clustering results in the non-monotonic behavior when the number of controller domains is varied from 2 to 4. As mentioned in Section V-C, Betweenness centrality clustering partitions a graph by removing edges with high betweenness centrality until the graph is disconnected. Thus, it can be assumed that BT North America model might contain some dense substructures, and those might be coincidentally revealed by the edge removal process when 4 domains are created.

Moreover, in Figure 6(e), Repeated bisection clustering demonstrates the best result when two domains are created. However, this would be because it just separates a graph into two parts: extreme large cluster and small one. In fact, unbalanced clusters are created by the clustering method as seen in Figure 4(e). Hence, it could be said that the result is just originated from a single large cluster.

## VII. Conclusion and Future Work

This paper has focused on the network partitioning for multi-domain SDN networks. By abstracting an SDN network as a graph, this paper approached to the issue based on graph clustering and formulated Network Partitioning Problem (NPP). Then, four clustering algorithms have been provided and evaluated with different SDN-related indicators, such as the balance load of controllers, inter-control load, and reliability of domains. The simulation results show that Conductance clustering could contribute to better-balanced load of controllers and higher reliability of controller domains while Spectral clustering demonstrates less inter-control load. Our future work will include a comparison of the clustering methods in terms of the controller placement. Furthermore, because the current simulations are only theoretical approach, an examination of different partitioning and its effects on network performances under realistic SDN scenarios are also left as our future work.

## Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 26330120.7.

## References

[1] A. Hidenobu, N. Junichi, and S. Norihiko, "Network partitioning problem to reduce shared information in openflow networks with multiple controllers," ICN 2015, 2015, pp. 250–255.

[2] S. Sezer et al., "Are we ready for SDN? implementation challenges for software-defined networks," Communications Magazine, IEEE, vol. 51, no. 7, July 2013, pp. 36–43.

[3] S. Kuklinski and P. Chemouil, "Network management challenges in software-defined networks," IEICE Transactions on Communications, vol. 97, no. 1, 2014, pp. 2–9.

[4] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," Communications Magazine, IEEE, vol. 51, no. 2, February 2013, pp. 136–141.

[5] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. USENIX Association, 2012, pp. 10–10.

[6] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 7–12.

[7] D. Kreutz et al., "Software-defined networking: A comprehensive survey," proceedings of the IEEE, vol. 103, no. 1, 2015, pp. 14–76.

[8] B. Pankaj et al., "Onos: towards an open, distributed sdn os," in Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014, pp. 1–6.

[9] S. Schmid and J. Suomela, "Exploiting locality in distributed sdn control," in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013, pp. 121–126.

[10] H. Xie, T. Tsou, D. Lopez, H. Yin, and V. Gurbani, "Use cases for alto with software defined networks," Working Draft, IETF Secretariat, Internet-Draft draft-xie-alto-sdn-extension-use-cases-01. txt, 2012.

[11] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in sdn-based core networks," in Teletraffic Congress (ITC), 2013 25th International. IEEE, 2013, pp. 1–9.

[12] T. Koponen et al., "Onix: A distributed control platform for large-scale production networks." in OSDI, vol. 10, 2010, pp. 1–6.

[13] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 19–24.

[14] P. Xiao, W. Qu, H. Qi, Z. Li, and Y. Xu, "The sdn controller placement problem for wan," in Communications in China (ICCC), 2014 IEEE/CIC International Conference on. IEEE, 2014, pp. 220–224.

[15] E. Borcoci, R. Badea, S. G. Obreja, and M. Vochin, "On multi-controller placement optimization in software defined networking-based wans," ICN 2015, 2015, p. 273.

[16] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, "Controller placement and flow based dynamic management problem towards sdn," in Communications (ICC), 2015 IEEE International Conference on. IEEE, 2015, pp. 369–374.

[17] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," Communications, China, vol. 11, no. 2, 2014, pp. 38–54.

[18] L. F. Muller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: an enhanced controller placement strategy for improving sdn survivability," in Global Communications Conference (GLOBECOM), 2014 IEEE. IEEE, 2014, pp. 1909–1915.

[19] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," Journal of the ACM (JACM), vol. 51, no. 3, 2004, pp. 497–515.

[20] S. E. Schaeffer, "Graph clustering," Computer Science Review, vol. 1, no. 1, 2007, pp. 27–64.

[21] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," 2014.

[22] U. Brandes and T. Erlebach, "Network analysis." Springer Berlin Heidelberg, 2005.

[23] F. Jordan and F. Bach, "Learning spectral clustering," Adv. Neural Inf. Process. Syst, vol. 16, 2004, pp. 305–312.

[24] U. Von Luxburg, "A tutorial on spectral clustering," Statistics and computing, vol. 17, no. 4, 2007, pp. 395–416.

[25] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” Applied statistics, 1979, pp. 100–108.
[26] U. Brandes, “A faster algorithm for betweenness centrality*,” Journal of Mathematical Sociology, vol. 25, no. 2, 2001, pp. 163–177.
[27] M. Girvan and M. E. Newman, “Community structure in social and biological networks,” Proceedings of the national academy of sciences, vol. 99, no. 12, 2002, pp. 7821–7826.
[28] S. M. Savaresi and D. L. Boley, “On the performance of bisecting k-means and pddp.” in SDM. SIAM, 2001, pp. 1–14.
[29] M. E. Newman and D. J. Watts, “Renormalization group analysis of the small-world network model,” Physics Letters A, vol. 263, no. 4, 1999, pp. 341–346.
[30] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” science, vol. 286, no. 5439, 1999, pp. 509–512.
[31] T. Sakano et al., “A study on a photonic network model based on the regional characteristics of japan (in japanese,” 2013.
[32] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The internet topology zoo,” Selected Areas in Communications, IEEE Journal on, vol. 29, no. 9, 2011, pp. 1765–1775.