NOLTA, IEICE

Paper

Evolutionary core-periphery structure and its application to network function virtualization

Mari Otokura [1 a)], *Kenji Leibnitz* [2,1 b)], *Tetsuya Shimokawa* [2,3 c)], *and Masayuki Murata* [1 d)]

[1] *Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan*

[2] *Center for Information and Neural Networks (CiNet), NICT and Osaka University, 1-4 Yamadaoka, Suita, Osaka 565-0871, Japan*

[3] *Graduate School of Frontier Biosciences, Osaka University, 1-3 Yamadaoka, Suita, Osaka 565-0871, Japan*

[a)] *m-otokura@ist.osaka-u.ac.jp*
[b)] *leibnitz@nict.go.jp*
[c)] *shimokawa@nict.go.jp*
[d)] *murata@ist.osaka-u.ac.jp*

Received August 20, 2015; Revised November 15, 2015; Published April 1, 2016

Abstract: Biological networks, such as that of the human brain, show a remarkable ability to adapt to changing environments by long-term evolution and short-term adaptations. This is facilitated by the core-periphery structure of the network, where a core of densely connected network nodes provides long-term functionality and robustness, while the periphery is responsible for adaptation on short time scales to changing conditions in the environment. In this paper, we discuss the characteristics of the core-periphery network structure and its relation with adaptability and evolvability, which we also illustrate with own experimental data from the brain's functional network. Based on this concept, we propose a mechanism for the adaptive placement of network functions to virtual servers in network function virtualization (NFV) under time-varying user requests. Our simulation results show that the number of server manipulations (migrations, merges, and replications of virtual machines) necessary to accommodate changes in network function requests can be greatly reduced compared to the conventional placement method.

Key Words: evolvability, core-periphery structure, modularity, network function virtualization (NFV), software defined networking (SDN)

Nonlinear Theory and Its Applications, IEICE, vol. 7, no. 2, pp. 202–216 DOI: 10.1587/nolta.7.202

1. Introduction

Biological systems are well known for their high adaptability and intrinsic robustness [1, 2]. Genetic evolution drives the process of adapting an organism to newly arising environmental conditions to ensure the survival of the species while at the same time keeping the changes within a limited energy budget. This results in biological networks having a high degree of *modularity* [3], i.e., the network can be easily decomposed into "building blocks" of smaller network modules that can be reassembled to perform high-level complex functions. For example, modularity in the human brain network is considered as a driver for the interplay between segregation of the specialized brain regions and their integration [4] facilitating cognitive behavior.

Module decomposition is mostly regarded in complex network studies for stationary networks when the connections between nodes do not change over time. However, in the recent past also an increased interest in temporal networks [5] has arisen, where the connectivity between network elements may change over time. Particularly from evolutionary viewpoint, it is appealing to investigate how nodes may join or leave modules at different time. One approach for evaluating the temporal dynamics of the community structure in graphs is by using a multi-slice modularity technique [6], where each slice consists of the same network at a certain time or time interval, and where each node is interconnected to itself in successive slices. Instead of finding the optimal modularity for each time instance (slice) independently, multi-slice modularity optimizes the community structure simultaneously across all time instances, leading to a more consistent view of the dynamic communities. This method has been applied to investigating how learning a motor skill is represented in the functional connectivity structure of the human brain [7].

For investigating how modularity spontaneously arises in a network, Kashtan et al. [8, 9] provide a model of an evolving biological system with *Modularly Varying Goals* (MVG) and show that changes in the environment from alternating evolutionary goals lead to a modular topology. Once the system has converged, switching from one goal to another by using different fitness functions does not change the major modules, but only the interconnections between them. The result is that the part of the network that has common functionality for both goals remains stable, while the part that deals with the individual functionality required for the current goal is activated only when needed. Such a behavior has been observed as *core-periphery structure* of a network [10].

The fast adaptation scheme toward time-varying specific goals with a low number of manipulations to the network configuration is also appealing for engineered systems, such as information networks. Recently, the Internet has shown a growing trend toward *Software Defined Networking* (SDN) and *Network Function Virtualization* (NFV), where network components, e.g., routers, and functions, e.g., deep packet inspection or firewalls, are implemented in software instead of using dedicated hardware. While SDN is based on separating the control plane, i.e., the control mechanisms of how data is transmitted, from data plane, i.e., the actual transmission mechanism, NFV facilitates running network functions, which were previously only available on dedicated hardware, as software instances in *virtual machines* (VM) on commodity servers. While the reconfiguration of conventional networks has been traditionally difficult, NFV enables an adaptive control of the network's resources in a fast and dynamic way.

One common application example of SDN/NFV is *service chaining*. It provides each user to individually customize a chain of services composed of multiple *Virtualized Network Functions* (VNFs) realized by NFV and enables dynamic provisioning of services according to the requests of users as well as auto-scaling, i.e., increasing or decreasing the scale of network functions according to their current level of demand. In order to control service chaining, it is necessary to provide services rapidly in response to the requests from users [11]. Furthermore, considering the current development of *Internet of Things* (IoT) technology, the number of users/devices that will be requesting service chains is expected to become very large in the future. Therefore, deciding the placement of resources and traffic routes by periodical re-optimization will be infeasible due to the long calculation time.

In this paper, we follow an interdisciplinary approach by applying an evolutionary core-periphery model from biology to information networking. Our contribution in this paper is to interpret the

MVG model as a temporal core-periphery model based on which we derive an efficient method that can be applied to the problem of allocating VNFs to virtual servers in the physical infrastructure of an information network with time-varying demand. We compare our proposed method by simulation with a conventional method using optimization [12, 33]. Simulation results show that the number of manipulations (migrations, merges, and replications of VMs) to accommodate for changes in traffic demand are in our proposed method about 33% lower than with the optimization.

The remainder of this paper is organized as follows. First we discuss in Section 2 general properties of modularity and core-periphery, as well as describe how we can characterize MVG as a temporal core-periphery mechanism. In order to show a real-world example, we identify core and periphery in the brain functional network extracted from *functional magnetic resonance imaging* (fMRI) data where the subjects alternate between two tasks. In Section 3, we then propose an evolutionary mechanism to efficiently control the server allocation of network functions in an NFV scenario under varying traffic conditions. We compare our proposed mechanism with conventional optimization by simulations in Section 4 and show numerical evidence that our proposal reduces the number of required adaptations necessary to accommodate changes in traffic. Finally, the paper is concluded in Section 5 including an outlook on future work.

2. Core-periphery structure and modularity of networks

Studying the topological structure of networks is necessary for understanding their internal organization, particularly if they are continuously evolving and changing over time. This is not only important for biological networks, e.g., the human brain, but also for most technical systems, e.g., the Internet. Methods from complex network theory can be used for evaluating the network topology based on a large variety of topological measures, e.g., clustering coefficient or characteristic path lengths. Some of these measures, like betweenness or modularity, are of particular importance for understanding the global network structure since they reveal important information about the role and topological importance of each individual node. While most complex network studies only look at a steady-state from geometrical viewpoint, it is challenging to also include temporal aspects. One way is to characterize the temporal network dynamics by observing changes in modularity.

2.1 Topological properties of network graphs

Estrada [13] provides a general classification of networks into 4 classes of topologies based on the spectral properties characterized by the eigenvalues of the adjacency matrix. Beside homogeneous networks that are free from any bottlenecks, the other classes are modular networks, core-periphery networks, and networks having features of both (quasicliques and quasibipartites). This classification is performed by comparing how the elements of the eigenvector of the adjacency matrix deviate from a straight line which corresponds to a homogeneous network with perfect spectral scaling. In this paper, we decouple the strictly topological distinction from [13] between core-periphery and modular, but rather observe the time-dependent behavior of core-periphery structures. This also permits combining both notions by having a network with modular topology, but where some modules frequently change over time, while others remain stable.

2.1.1 Modularity structure

Biological networks have in many cases modular structures such as protein-protein interaction networks, gene expression control networks, and brain functional networks [14]. Modularity can be informally characterized as the formation of groups of densely interconnected nodes that have sparse connections among groups. These groups are referred to as *modules*. The connection density is evaluated through comparison with randomly connected networks serving as null models.

Modularity is formally defined in [15] as the partition of network nodes that maximizes the modularity quality function Q defined as

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(s_i, s_j) \tag{1}$$

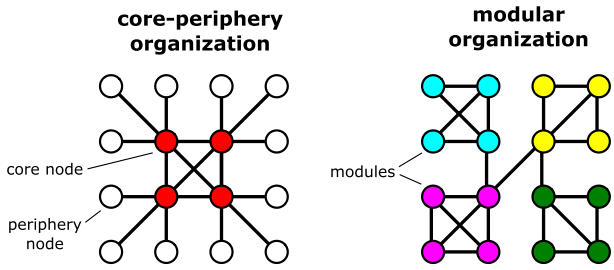


Fig. 1. Core-periphery versus modularity structure.

where $2m = \sum_{ij} A_{ij}$ is the number of all links, i.e., the sum of all entries A_{ij} in the adjacency matrix, k_i is the degree of node i, i.e., the number of connected neighbors of node i, and $\delta(s_i, s_j)$ is an indicator function that is 1 if nodes i and j are in the same module and 0 otherwise. The partition that maximizes Q in Eq. (1) is also called the *community structure* of the network.

2.1.2 Core-periphery organization

Core-periphery organization can often be found in real-world networks [10]. Although there appears to be no fully consistent formal definition of a core-periphery network, the general consensus is that the core forms a densely interconnected group of nodes with rather low adaptability located in the "center" of the network, while the periphery is formed by nodes with sparser connections, usually connecting only to the core and having a high adaptability. This is in contrast to modularity which has a more decentralized type of organization as shown in Fig. 1.

Various theoretical models for core-periphery networks and real-world networks have been proposed, often originating from studies of *elites* in social networks [16, 17]. Especially in the field of information networking there seem to be abundant examples of core-periphery, such as the Jellyfish model for the network of autonomous systems (AS) in the Internet [18], the bowtie structure of WWW links [19], or the evolution of network protocols that leads to the characteristic hourglass shape of the TCP/IP protocol stack [20]. *Bowties* and *rich clubs* are regarded as special cases of core-periphery [10], where a bowtie distinguishes the periphery into directed links leading to the core and links leading from the core, while a rich club has a clique of high degree nodes as core. Both, bowties and rich clubs, have been also studied in the context of the organization of functional [21] and anatomical brain networks [22].

2.2 Modularly varying goals vs. Temporal core-periphery

One of the factors which generates modularity in biological networks is evolution under varying environments as shown in [8]. When organisms evolve towards varying goals, they genetically change to adapt to the new environments and the underlying biological networks also need to change to fulfill the new goal. Regarding the trajectory in the system's state space, evolution toward a goal is the process of moving the system's state closer to the location that represents maximum fitness. When the goal changes, it means that the maximum fitness state is located at a new position toward which evolution will occur, starting from the state that was optimized for the previous goal. If the goals cyclically change, the ideal state will converge somewhere in the middle between both goals making both easy to reach. Kashtan et al. [8] refer to this concept as *Modularly Varying Goals* (MVG).

2.2.1 Modularly Varying Goals (MVG)

Genetic algorithms (GAs) are computational heuristics which imitate biological evolution with respect to an objective. In GAs, genomes are evolved toward a goal by using genetic operations such as selection, mutation, and crossover. Reference [8] proposes MVG based on GA to introduce the concept of biological evolutions in varying environments. Unlike conventional GAs which only optimize toward a fixed and single goal, MVG requires 2 or more goals to represent the varying environments. Since switching between the goal does not involve re-initialization, this method differs from simply performing sequential GAs with fixed goals.

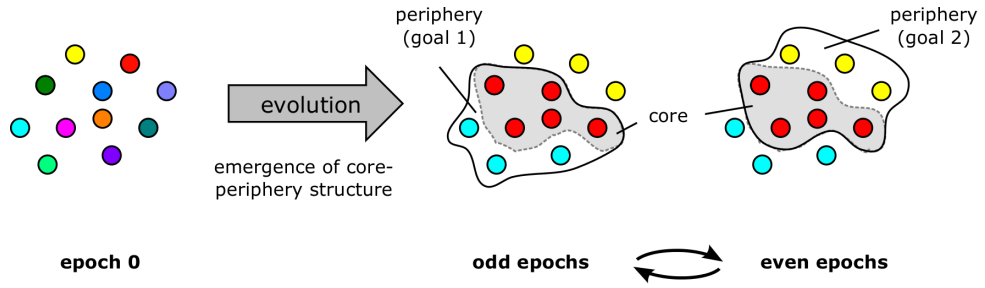


Fig. 2. Basic concept of evolutionary core-periphery structure in MVG.

In the examples presented in [8], the goals consist of combinations of sub-goals, i.e., boolean functions consisting of nested expressions in parentheses of the goals $G_1 = (X \oplus Y) \vee (W \oplus Z)$ and $G_2 = (X \oplus Y) \wedge (W \oplus Z)$, where "$\oplus$", "$\vee$", and "$\wedge$" are the logical XOR, OR, and AND operators, respectively. When evolving a network with varying goals, MVG obtains a structure, where modules correspond to sub-goals $(X \oplus Y)$ and $(W \oplus Z)$ and only the AND operator needs to be replaced by an OR, or vice versa. Since goals are combinations of sub-goals, a genome can adapt by only changing that part of the genome corresponding to links between the modules by few mutations without changing the modules themselves. Therefore, we can obtain solutions from genomes by adapting to a new goal by only few operations.

2.2.2 Evolutionary Varying Goals (EVG)

Our proposed method in this paper is based on MVG, but unlike the graph-based problem definition in [8], our formulation does not break down into individual modules, but only into two groups: core and periphery (Fig. 2). Similar to MVG, we assume that goals remain fixed during one *epoch*, comprising several generations, but then change slightly at the begin of the next epoch.

For each change in goal, the core corresponds to the common part that doesn't change regardless of the goal, whereas the periphery reflects the different part that is required when the new goal is active. We will refer to the MVG-based method as *Evolutionary Varying Goals* (EVG) and its temporal core-periphery structure can be regarded as a generalization of MVG.

2.3 Temporal core-periphery in brain functional networks

In order to illustrate the existence of dynamic core-periphery structures in brain functional networks, we evaluated data that was obtained from fMRI measurements where each subject alternates between 2 tasks at an auditory cue signal: (i) COUNT in which the subject silently counts down from 100, and (ii) NAM in which the subject recalls a negative autobiographical experience from memory, each performed 3 times for epochs of 32 s at a temporal resolution (TR) of 2 s. The data set consists of functional magnetic resonance imaging (fMRI) measurements from 11 subjects (5 males, 21–37 years) and was made available by kind permission of the authors of [25], where further details on the acquisition of the data can be found.

Each COUNT and NAM epoch is separated by a short REST phase of 16 s to avoid overlaps of the hemodynamic responses from task changes. The time series consists of 3 repetitions of COUNT-REST-NAM-REST epochs, resulting in a total time series length of 144 scans. The time series data from all voxels was reduced to 264 regions of interest (ROIs) based on the definition of putative functional areas by Power et al. [26] by averaging over the BOLD signals of all voxels' time series located within spheres centered at the ROI locations with a 5 mm radius.

The general process of constructing a network graph from fMRI time series is often described in the literature [27, 28] and will be only roughly sketched here. Each ROI acts as node in the functional network graph and two nodes are connected by a link if their time series are strongly correlated. Since the duration of each task-specific epoch is very short with only 16 scans, the correlation matrices becomes very noisy if we attempt to determine an individual network for each epoch. We, therefore, reordered the time samples so that all COUNT epochs are concatenated in one data set and all NAM epochs are concatenated in another (Fig. 3). These time series are sufficient in length to

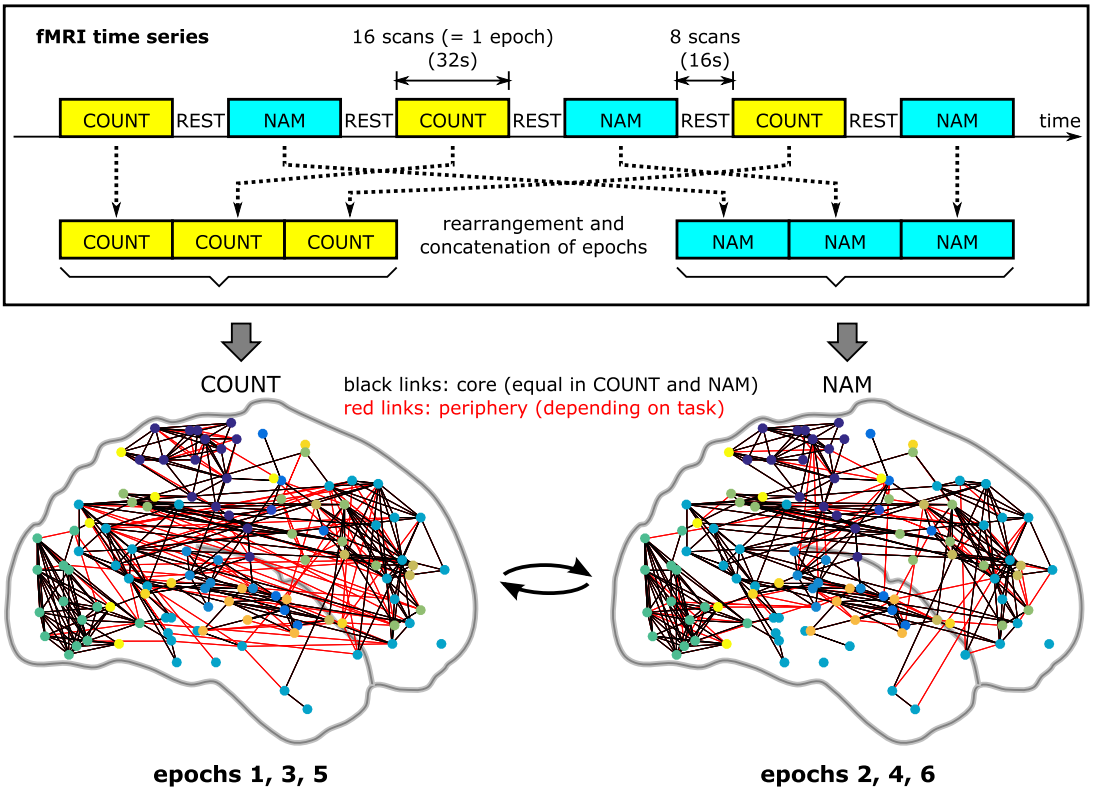


Fig. 3. Reordering of fMRI epochs and extraction of functional brain network graphs for both tasks reveals temporal core-periphery structure. Black links are common in both networks (core) and red links vary depending on task (periphery). Colors of the nodes are chosen based on their anatomical regions.

compute a correlation matrix for each task and we calculated the average correlation matrix from all 11 subjects and applied a proportional threshold of $\theta = 0.05$ to pick only the 5% strongest values as links. Figure 3 shows the resulting networks for the COUNT and NAM tasks. For sake of clarity, we only depict nodes located on the left hemisphere. Links shown in black are common in both networks (core) and red links vary depending on the task (periphery). Colors of the nodes are chosen based on their functional areas according [29]: motor and somatosensory network (SMN), cingulo-opercular network (CON), auditory, default mode network (DMN), visual, fronto-parietal network (FPN), salience network (SAN), subcortical, ventral and dorsal attention networks (VAN and DAN).

3. Application to traffic-dependent VNF placement

In this section, we present the application of EVG as a dynamic core-periphery mechanism to the problem of optimizing the VNF placement to servers in a physical network with respect to the changes in demand for the network functions.

3.1 Modeling the VNF placement problem

Let us consider an SDN/NFV scenario, consisting of a physical network connecting routers and servers as shown in Fig. 4. We model the service chain of each VNF as a sequence of 3 function components FC_c with $c = 1, 2, 3$, where similar to [30] the first component handles receiving of packets, the second deals with finding flow states from a flow table, which contains information on the current flow state and the corresponding processes, and the last component is for processing packets and outputting them. Each network function needs to be processed by the components FC_c in their exact order $c = 1, 2, 3$, while the components run on VMs located on any server in the network, not necessary on the same server.

VNFs for service chaining are placed on servers in carrier clouds [31]. In the future, carrier clouds will be composed of multiple data centers and their interconnections. In such case, it is important

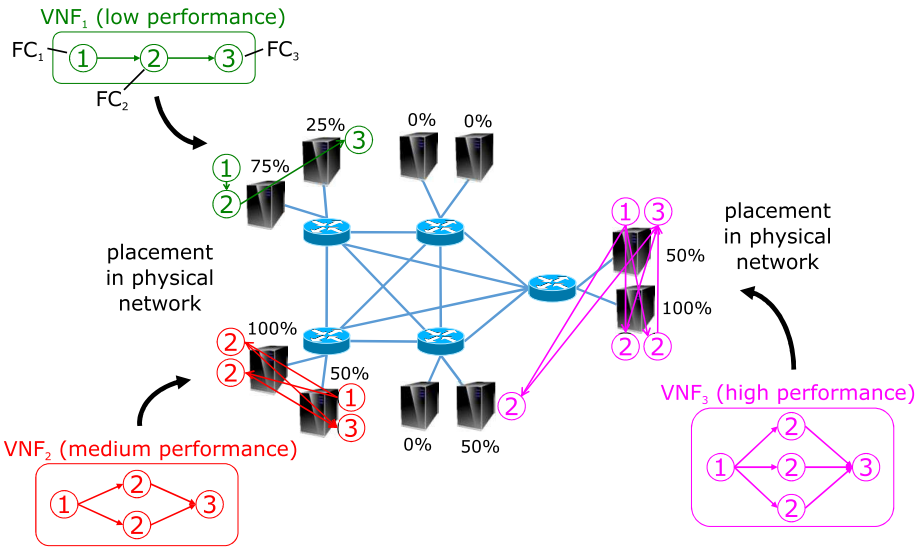


Fig. 4. Conceptional figure of our considered SDN/NFV model. An example placement is shown where each server has 16 cores and the number of necessary cores for components FC_c is 4, 8, 4 for $c = 1, 2, 3$, respectively. The percentages shown beside each server are the utilization of their CPU cores based on the current placement.

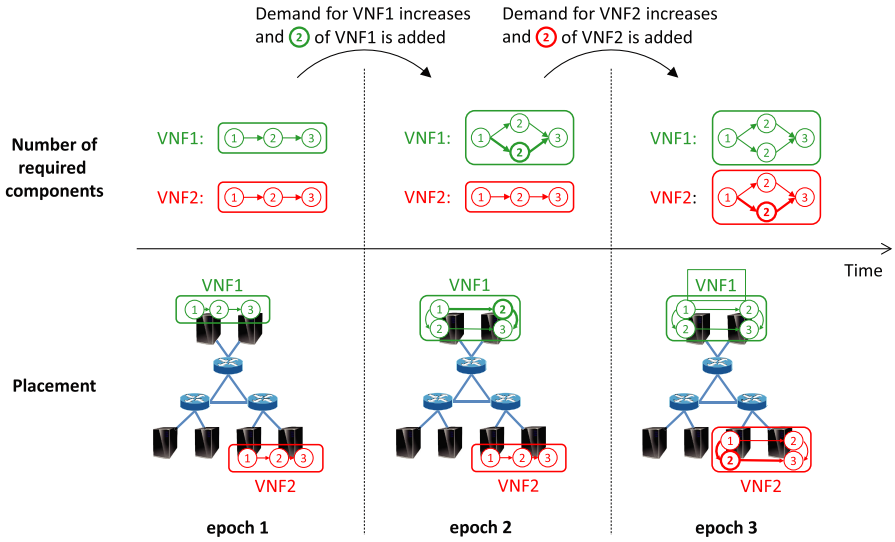


Fig. 5. An example of time-dependent placement due to changes in demand over 3 epochs is shown where each server has 16 cores and the number of necessary cores for components FC_c is 4, 8, 4 for $c = 1, 2, 3$, respectively.

to place the components of the same VNF on the same data center to avoid unnecessary delays in the communication between data centers. In this paper we assume that the network is composed of routers that each connect to multiple servers, so our scenario is equivalent to a router being a data center and the network connecting routers as the network between data centers.

When demand for a certain network function VNF v increases, the second component FC_2 often becomes the bottleneck of VNF v [32], while we assume that FC_1 and FC_3 remain nearly unaffected by the number of traffic flows. Therefore, when the demand increases, we need to increase the performance of FC_2, which is achieved by scaling up the number of VMs running the software of FC_2 (Fig. 5). We will therefore focus in our notation only on FC_2 when we consider the number of required components by representing its demand only.

However, the number of components that can be placed on each server is finite. In [12], a limit of the number of VNFs that can be placed on each server is set by introducing the concept of the

number of cores[1] to their model. Given the number of necessary cores for running a VNF on a VM, and the number of cores for each server, the VNFs are placed in the network in such a way that no server uses more cores than it actually has. Since our smallest unit size is one VNF component, we limit the number of components by providing the number of necessary cores for running a certain component and the total number of available cores for each server. Then, our proposed method EVG places the components on the servers without exceeding their number of cores.

3.2 Application of EVG to VNF placement problem

As discussed in earlier sections, EVG is based on MVG and has a core-periphery structure that we use for adapting to time-varying goals in the form of requests for VNFs. In order to solve the VNF placement problem, we now propose the EVG method to make use of the core which remains same when goals are changed and only need to accommodate for the changes in the periphery. In order to define the EVG model, we now need to elaborate on its formulation as a genetic algorithm.

An instantaneous configuration in our model reflects the current assignment of component VMs to the servers in the network, which is encoded as a genome. Therefore, the population of genomes represents the candidates of possible assignment configurations of VNF placement to the physical infrastructure. Switching between goals in EVG is accomplished by comparing the two traffic requests for required VNF components at generation t and $t+1$. Since this transition from t to $t+1$ is very likely to have overlapping requirements, this common part forms the core of the core-periphery structure discussed in Section 2. The quality of the output from the GA is evaluated by a time-varying fitness function, which distinguishes it from most conventional GA implementations. The algorithm of EVG that we use in the following is outlined in Algorithm 1. We refer to all steps within the loop (steps 3–9) as one generation. Note that, in Algorithm 1, no re-initialization of the population occurs when the goal changes. This property of EVG lets the genomes in the population obtain the core-periphery structure which can easily adapt to varying goals.

Algorithm 1 Genetic algorithm for solving VNF placement problem

1: randomly generate population of N genomes
2: **while** maximum number of generations is not reached **do**
3: **if** required components changed from previous generation **then**
4: change goal
5: **end if**
6: evaluate the fitness of each genome and rank genomes by their fitness
7: save η genomes from the top of the ranking ("elites") and remove from population
8: mutate each genome in the population with probability p_m and crossover with probability p_c
9: generate the new population by combining η and the remaining population
10: **end while**

Fitness F evaluates the quality of the current configuration of VNF placement and is determined by Eq. (2).

$$F = \kappa - \frac{P_{util} + P_{core} + P_{delay}}{\kappa} \tag{2}$$

Here, κ is the coincidence of the number of components in the current placement to the required number of components, P_{util} is the penalty imposed according to the utilization of the resources of the physical network, and P_{core} is the penalty imposed when more cores are placed on servers than they have. The coincidence κ is calculated as follows.

$$\kappa = 1 - \frac{\sum_{c=1}^{3}\sum_{v=1}^{V} D_{cv}}{3V} \qquad \text{with} \qquad D_{cv} = \begin{cases} 0 & \text{if} \quad R_{cv} = \sum_{s=1}^{S} x_{scv} \\ 1 & \text{otherwise} \end{cases}$$

[1]The term "cores" refers in this section to CPU cores in a server and should not be confused with the cores in the core-periphery structure discussed in Section 2.

Here, V is the number of VNFs that need to be placed and S is the number of servers in the system. The number of placed components is represented by variable x_{scv}, where $s \in \{1, 2, \ldots, S\}$ is the server index, $c \in \{1, 2, 3\}$ is the component index, and $v \in \{1, 2, \ldots, V\}$ is the index of the corresponding VNF. The variable R_{cv} is the number of required components c of VNF v, and D_{cv} indicates if the number of required components matches with the number of placed components ($D_{cv} = 1$) or not ($D_{cv} = 0$). D_{cv} can be determined from x_{scv} and x_{scv} can be decided from the genome.

The utilization penalty on the fitness P_{util} is calculated as follows:

$$P_{util} = \begin{cases} U - U_{th} & \text{if} \quad U \geq U_{th} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Here, U is the utilization of the resources in the physical network, which we will refer to in the following as "resource utilization" and U_{th} is its threshold. U is defined as

$$U = \frac{1}{2}\left(\frac{\hat{C}}{C} + \frac{\hat{L}}{L}\right)$$

where

$$\hat{C} = \sum_{s=1}^{S}\sum_{c=1}^{3}\sum_{v=1}^{V} C_c \times x_{scv} \qquad C = \sum_{s=1}^{S} S_s \qquad \hat{L} = \sum_{k=1}^{L} y_k.$$

C is the sum of the number of cores of all servers in the system and L is the number of all links between routers in the physical network. C_c is the number of required cores for running component c, S_s is the number of cores of server s, and y_k is a variable which indicates with 1 if link k is used, otherwise it is 0. $\hat{L}$ is the sum of the used links between routers and $\hat{C}$ is the sum of the number of cores which the placed components need. U, $\hat{C}$, $\hat{L}$, and y_k can be determined from x_{sjv}.

The penalty P_{core} of assigning too many cores is calculated as follows:

$$P_{core} = P_{ex}\sum_{s=1}^{S} E_s \qquad \text{with} \qquad E_s = \begin{cases} 1 & \text{if} \quad \sum_{c=1}^{3}\sum_{v=1}^{V} x_{scv}C_c > S_s \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

Here, E_s indicates with 1 if the required number of components exceeds the capacity of server s (0 if it doesn't exceed) and the sum over all E_s is the number of servers which require more cores for running the components than they actually have. In the following, we refer to such servers as "exceeded servers". P_{ex} is the penalty value which is imposed on the fitness for each exceeded server and is given as parameter.

The penalty P_{delay} of exceeding a delay threshold is calculated as follows:

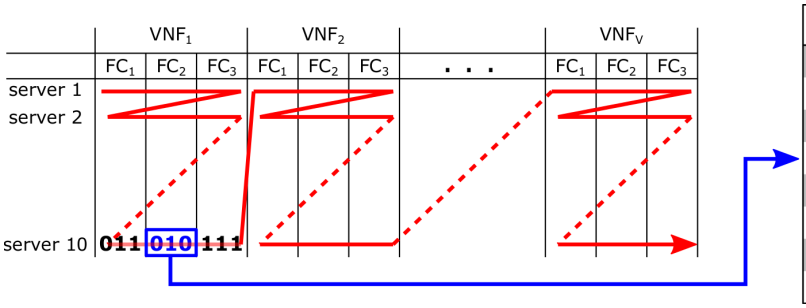
$$P_{delay} = \begin{cases} D - D_{th} & \text{if} \quad D > D_{th} \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

Here, D is the sum of delays of each VNF and D_{th} is its threshold. D is defined as

$$D = \sum_{l=1}^{L}\sum_{v=1}^{V}(w_l^{v\,(1,2)} + w_l^{v\,(2,3)})\psi_l$$

where ψ_l is the propagation delay of the physical link l, $w_l^{v\,(1,2)}$ is a variable which represents the number of virtual links of VNF v from component 1 to component 2 which are allocated to physical link l, and $w_l^{v\,(2,3)}$ is a variable which represents the number of virtual links of VNF v from component 2 to component 3 which are allocated to physical link l, where $l \in \{1, 2, \ldots, L\}$. Both variables of the number of links can be determined from x_{scv}.

By using Eq. (2) as the fitness function, we can achieve a fitness of 1 for specific genomes. These genomes represent placement configurations which have the following 3 properties:



bit string	number of FC_1, FC_3	number of FC_2
000	0	0
001	0	0
011	0	1
010	0	2
110	0	3
111	1	0
101	0	0
100	0	0

Fig. 6. The structure of a genome and table for converting 3-bit strings to number of components. In this example, the part of the genome "011010111" for VNF 1 and server 10 is translated to 0 FC_1, 2 FC_2, and 1 FC_3 number of components.

- the placements have the appropriate number of components
- resource utilization of the placement is below a specified threshold, and
- there is no exceeded server among them.

4. Simulation evaluation of VNF placement mechanism

We now evaluate the performance of EVG as VNF placement mechanism. Our objective is to reduce the number of configuration changes while also achieving a good quality of placement in terms of utilization.

4.1 Evaluation environment

In this subsection we explain the settings we used in the simulation regarding the structure of the network and how we model traffic changes. We used for our evaluations the same physical network as previously shown in Fig. 4, which is composed of 5 routers connected in a full mesh and where each router has 2 servers connected to it. Each server and router has an id number, ranging from 1 to 10 for servers and from 1 to 5 for routers. Furthermore, we assume that each server is equipped with 16 cores in total. Each server must use a certain number of cores for running a VNF component. We assume this number to be 4, 8, and 4 cores for components FC_i, $i = 1, 2, 3$, respectively.

In this paper, we consider the problem of placing $V = 5$ equal VNFs on the physical infrastructure. As mentioned in Section 3.2, our focus is on the time-varying demand for components of VNFs, particularly for FC_2. Denoting r_v as the number of required FC_2 of VNF v, the number of required components at generation t is represented as a V-dimensional vector $R(t) = [r_1, \ldots, r_V]$. This vector $R(t)$ randomly changes after every T generations (1 epoch), such that $|R(t+1) - R(t)| = 1$ if generation t and $t+1$ belong to different epochs, and $R(t+1) = R(t)$ if they belong to the same epoch. In other words, $R(t)$ performs a discrete random walk in $\mathbb{N}_+^V$ with step size 1 on the time scale of epochs.

4.2 Proposed genetic algorithm

As explained in Section 3.2 our proposal is based on the evolutionary method of MVG. In the following of this section, we will explain the design and settings of the GA simulation.

The structure of a genome is illustrated on the left in Fig. 6. The number of components for FC_i, $i \in \{1, 2, 3\}$ for each VNF $v \in \{1, ..., V\}$ and server $s \in \{1, ..., S\}$, is a binary string of 3 bits and the entire genome is comprised of such concatenations in the order shown by the red arrow in Fig. 6. In order to convert from a 3 bit binary sequence to the number of components, the table on the right of Fig. 6 is used.

In the conversion of FC_1 or FC_3, only the bit string of "111" leads to 1, while all others are converted to 0 (middle column of table). This is because the number of FC_1 and FC_3 components is likely to become greater than 1 due to the genome structure shown in the left side of Fig. 6, although one component of each of FC_1 and FC_3 is needed. On the other hand, in the conversion of FC_2, the difference between two values which are converted from two neighboring bit strings in the Gray code

Table I. Parameter settings of proposed VNF placement method.

number of genomes in a population (N)	1000
size of elites (η)	100
crossover probability (p_c)	0.9
mutation probability (p_m)	0.9
number of generations (G_{max})	20000
interval between goal changes (T)	20
threshold of resource utilization (U_{th})	0.5
threshold of delay (D_{th})	120
penalty imposed per exceeded server (P_{ex})	0.1

is 1, except for "110" and "111" so that the number of FC_2 doesn't change drastically when a bit is changed by a mutation.

The parameters in the evaluation of our proposal are summarized in Table I. In our simulations we use a population size of $N = 1000$ genomes. The elite consists of $\eta = 100$ genomes with highest fitness values. Crossovers and mutations are performed with probabilities p_c and p_m, respectively. Every $T = 20$ generations the traffic requirement vector $R(t)$ is changed. The threshold U_{th} in Eq. (3) is set to 0.5, the penalty per exceeded server P_{ex} in Eq. (4) is set to 0.1, and the threshold of delay $D_{th} = 120$ in Eq. (5). The maximum number of generations is $G_{max} = 20000$.

The solution for each goal is the genome which has maximum fitness among all other genomes obtained in the generation prior to the goal change. That is, we obtain a solution for the current goal and change the goal every $T = 20$ generations, which we refer to as an epoch. Therefore, we obtain a time series of 1000 ($= G_{max}/T$) epochs from a single simulation run. Furthermore, we also store the time series of R_{cv} corresponding to the traffic requirements as input to the simulation.

4.3 Reference placement method by ILP

Beside our proposed method, we also consider the placement results obtained by solving the problem formulated by an *Integer Linear Problem* (ILP) which was used to obtain optimized VNF placements in previous research [12, 33]. In the following, we refer to this method as "ILP method". We use the same input R_{cv} from the simulation of EVG method for which we also determine the placement of components by the ILP method. We obtain the placements which have minimal resource utilization and satisfy 2 constraints:

- the placements have the appropriate number of components, and
- there is no exceeded server.

In the following, we explain the formulation of the problem by ILP. We use the same variables and constants as in Section 3.2. The decision variable is x_{scv}, the objective function is $U + D$, and the constraints are shown below:

$$
\begin{aligned}
& \kappa = 1 \\
\forall s \quad & \sum_{c}^{3} \sum_{v}^{V} (x_{scv} \times C_c) \leq S_s \\
\forall s, c, v \quad & x_{scv} \text{ is an integer value and } 0 \leq x_{scv} \leq 4 \\
\forall l \quad & y_l \text{ is a binary value} \\
\forall l \quad & (\exists v \quad (\tilde{x}_{r1v} = 1 \wedge \tilde{x}_{r'2v} \geq 1) \rightarrow y_l = 1) \\
\forall l \quad & (\exists v \quad (\tilde{x}_{r3v} = 1 \wedge \tilde{x}_{r'2v} \geq 1) \rightarrow y_l = 1) \\
\forall l, v \quad & w_l^{v\ (1,2)} \geq (\tilde{x}_{r2v} - \tilde{x}_{r'1v}) + (\tilde{x}_{r'2v} - \tilde{x}_{r1v}) \\
\forall l, v \quad & w_l^{v\ (2,3)} \geq (\tilde{x}_{r2v} - \tilde{x}_{r'3v}) + (\tilde{x}_{r'2v} - \tilde{x}_{r3v})
\end{aligned}
$$

Table II. Mean and standard deviation of the number of each manipulation.

	EVG		ILP		FG	
	mean	SD	mean	SD	mean	SD
replications	0.51	0.45	1.61	0.67	25.91	2.13
merges	0.54	0.49	1.60	0.67	12.10	3.02
migrations	0.50	0.29	13.63	2.19	6.14	0.85

Here, $\tilde{x}_{rcv}$ is the variable which represents the number of component c of VNF v on the servers connected to router r with $\tilde{x}_{rcv} = x_{(2r)cv} + x_{(2r+1)cv}$. The terms r and r' are the router ids which are connected to link l.

4.4 Reference placement method by FG

In order to show the difference between EVG and ordinary GA, we also consider the placement results obtained by *Fixed-Goal* genetic algorithm (FG) in our comparison. FG is the repetition of the ordinary GA whose maximum number of generations is T. That is, the only difference we have in Algorithm 1 is that we change "change goal" in line 1 of to "regenerate population and change goal." All parameters of FG are the same as in EVG (Table I).

4.5 Evaluation metrics

We use the number of VM manipulations that are required to change the VNF placement from the old configuration to the new one as our evaluation metric. In the following, we evaluate it as the number of manipulations and resource utilization obtained from simulation runs.

We assume three types of VM manipulations: migrations, replications, and merges for changing from one VNF configuration to another. *Migration* means moving a VM from an origin server to another server, *replication* means copying an existing VM and additionally launching it on the same or a different server, and *merge* means combining multiple VMs into a single VM. In this paper, we use the number of migrations, replications, and merges needed for each change of the placements as metrics for comparing our proposal with ILP and FG. We only count the number of manipulations that traverse links between routers because manipulations that take place between servers connected to the same router do not consume significant transmission time for transferring disk and memory images [34].

4.6 Numerical results

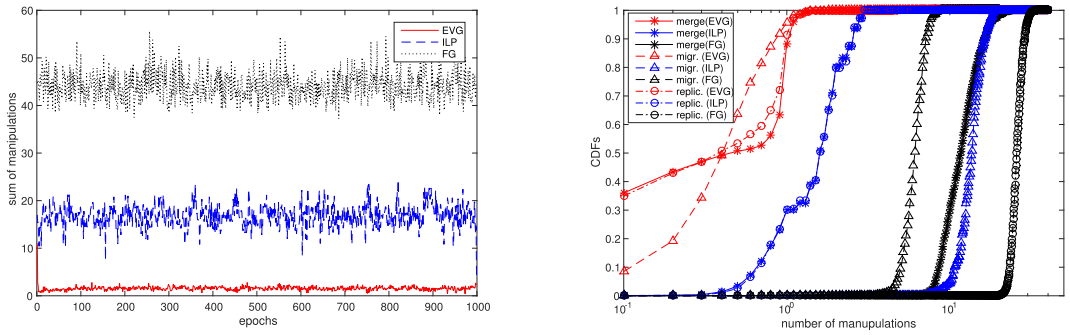
To perform a comparison under the exactly same conditions, we first generated the input R_{cv} which we then used for 10 repetitions of the proposed EVG simulation, ILP with random seeds, and FG simulation.

4.6.1 Number of manipulations for placements

Figure 7(a) shows the sum of the 3 kinds of manipulations to match the requirement R_{cv} over the whole time course averaged across all simulation runs. It can be clearly seen that both the mean and variance are significantly smaller for EVG than for ILP and FG (Table II). If we look at the cumulative distribution functions (CDFs) for each manipulation separately in Fig. 7(b), we can see that each manipulation is on average smaller for the EVG method than ILP and FG. This is due to the core-periphery type of configuration which requires only few manipulations to switch the periphery while the core remains unchanged. In contrast, FG is much worse than the other two methods. This is because FG is often unable to generate even a valid placement in only $T = 20$ generations as we explain further in Section 4.6.2.

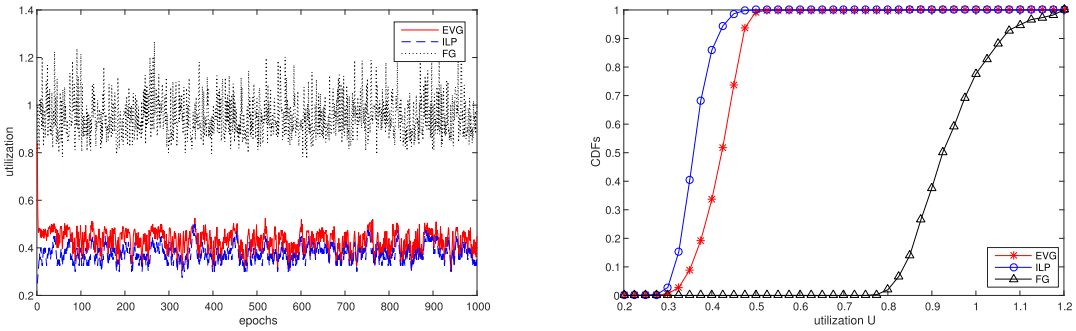
4.6.2 Comparison of resource utilization

While the proposed MVG-based method shows a significant decrease in the number of manipulations compared to ILP, it is expected that the quality of the solutions achieved by the proposal is inferior since ILP is able to find the best placement. However, Fig. 8(a) shows that the utilization of EVG



(a) Time series of sum of manipulations (b) CDFs of the number of each manipulation

Fig. 7. Evaluation of number of manipulations for VNF placement for EVG, ILP, and FG.



(a) Time series of average utilization (b) CDFs of the utilization

Fig. 8. Evaluation of VNF placement quality by utilization for EVG, ILP, and FG.

is not significantly worse than ILP when we look at its time series. The solution by ILP has a mean utilization of 0.37 (SD: 0.037), while EVG has a mean of 0.43 (SD: 0.046). Moreover, we can also recognize that both time series are positively correlated ($\rho = 0.745$), which means that EVG has a tendency to come relatively close to the optimal solution from ILP. In contrast, the quality of the solutions achieved by FG is much worse than EVG and ILP. The solutions often have utilizations of more than 1.0 because evolution of $T = 20$ generations is insufficient to get even a valid result in those cases. Figure 8(b) shows the plots of the CDFs of all distributions and although the variances of EVG and ILP appear similar in the slope of the CDF, EVG has a larger median by about 0.06, which indicates that the placement is not as efficient as ILP, but it is much more efficient than FG.

5. Conclusion

In this paper, we provided an interdisciplinary view of the core-periphery structure and its relationship with evolution. While core-periphery structure is a phenomenon that is often observed in static networks, it can also be observed in the evolutionary adaptation of a system to varying goals. This concept was introduced as MVG that showed how a modular structure could arise, but we generalized this notion to EVG, where an underlying network does not need to exist for showing modularity, but we can characterize a core, which remains stable, and a periphery which flexibly adapts due to the change of goals. We showed by a simple evaluation of fMRI data that the partitioning into core and periphery can also be observed in the human brain when the subject's tasks are switched.

We proposed the application of EVG to the problem of VNF placement on a physical network with varying traffic requirements. Our proposed method evolves the genomes which represent the placements of VNF components towards structures which are strong against goal changes. This is mostly the effect of alternating between the goals every time the number of required components changes. It was confirmed that the proposed method can significantly reduce the number of manipulations needed to change the placement when adapting to new component requirements. Thus, the proposed

method can control the network adaptively to the varying traffic and we consider the results in this paper as a proof of concept.

We intend to pursue extensions of the work described here in various directions. From the application viewpoint, we want to enhance EVG-based VNF placement to further improve its performance. In order to do this, we will perform studies in more realistic scenarios, i.e., taking more complex physical network topologies into account. We further intend to place service chains composed of multiple VNFs as future work. Furthermore, for a better comparison with task switching in the human brain, we are planning on acquiring new data from subjects which have longer task epochs to study the dynamics of temporal core-periphery in brain functional networks.

Acknowledgments

The authors would like to thank Norberto Eiji Nawa for providing the fMRI data set, as well as Yuki Koizumi and Daichi Kominami for their valuable comments during the course of this research. This research has been supported by the Humanware Innovation Program of Osaka University.

References

[1] H. Kitano, "Biological robustness," *Nat. Rev. Genet.*, vol. 5, pp. 826–837, November 2004.

[2] H. Kitano and K. Oda, "Self-extending symbiosis: A Mechanism for increasing robustness through evolution," *Biol. Theory*, vol. 1, no. 1, pp. 61–66, 2006.

[3] G.P. Wagner, M. Pavlicev, and J.M. Cheverud, "The road to modularity," *Nat. Rev. Genet.*, vol. 8, no. 12, pp. 921–931, 2007.

[4] G. Tononi, O. Sporns, and G.M. Edelman, "A measure for brain complexity: relating functional segregation and integration in the nervous system," *Proceedings of the National Academy of Sciences*, vol. 91, no. 11, pp. 5033–5037, 1994.

[5] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, Temporal Networks, vol. 519, no. 3, pp. 97–125, 2012.

[6] P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, and J.P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *Science*, vol. 328, no. 5980, pp. 876–878, 2010.

[7] D.S. Bassett, N.F. Wymbs, M.P. Rombach, M.A. Porter, P.J. Mucha, and S.T. Grafton, "Task-based core-periphery organization of human brain dynamics," *PLoS Comput. Biol.*, vol. 9, no. 9, pp. 1–16, 2013.

[8] N. Kashtan and U. Alon, "Spontaneous evolution of modularity and network motifs," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 102, no. 39, pp. 13773–13778, 2005.

[9] N. Kashtan, E. Noor, and U. Alon, "Varying environments can speed up evolution," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 104, no. 34, pp. 13711–13716, 2007.

[10] P. Csermely, A. London, L.Y. Wu, and B. Uzzi, "Structure and dynamics of core-periphery networks," *J. Complex Networks*, vol. 1, no. 2, pp. 93–123, 2013.

[11] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in network service chaining," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, pp. 1–7, November 2013.

[12] M.F. Bari, S.R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions in NFV," *eprint arXiv:1503.06377*, March 2015.

[13] E. Estrada, "Topological structural classes of complex networks," *Phys. Rev. E*, vol. 75, no. 1, pp. 1–12, 2007.

[14] G.P. Wagner, M. Pavlicev, and J.M. Cheverud, "The road to modularity," *Nature Reviews Genetics*, vol. 8, pp. 921–931, December 2007.

[15] M.E.J. Newman, "Modularity and community structure in networks," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 103, no. 23, pp. 8577–82, 2006.

[16] S.P. Borgatti and M.G. Everett, "Models of core/periphery structures," *Soc. Networks*, pp. 375–395, 2000.

[17] C. Avin, Z. Lotker, D. Peleg, Y.A. Pignolet, and I. Turkel, "Core-periphery in networks: An axiomatic approach," *arXiv preprints*, no. arXiv:1411.2242, 2014.

[18] G. Siganos, S.L. Tauro, and M. Faloutsos, "Jellyfish: A conceptual model for the AS internet topology," *Commun. Networks*, vol. 8, no. 3, pp. 339–350, 2006.

[19] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the Web," *Comput. Networks*, vol. 33, pp. 309–320, June 2000.

[20] S. Akhshabi and C. Dovrolis, "The evolution of layered protocol stacks leads to an hourglass-shaped architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, p. 206, 2011.

[21] M.P. van den Heuvel and O. Sporns, "Rich-club organization of the human connectome," *J. Neurosci.*, vol. 31, no. 44, pp. 15775–15786, 2011.

[22] N.T. Markov, M. Ercsey-Ravasz, D.C. Van Essen, K. Knoblauch, Z. Toroczkai, and H. Kennedy, "Cortical high-density counterstream architectures," *Science*, vol. 342, no. 6158, pp. 1238406–1238406, 2013.

[23] M.P. Rombach, M.A. Porter, J.H. Fowler, and P.J. Mucha, "Core-periphery structure in networks," *SIAM J. Appl.*, pp. 1–27, 2014.

[24] X. Zhang, T. Martin, and M.E.J. Newman, "Identification of core-periphery structure in networks," *Phsysical Rev. E*, vol. 91, no. 3, pp. 1–10, 2015.

[25] N.E. Nawa and H. Ando, "Classification of self-driven mental tasks from whole-brain activity patterns," *PLoS One*, vol. 9, no. 5, 2014.

[26] J.D. Power, A.L. Cohen, S.M. Nelson, G.S. Wig, K.A. Barnes, J.A. Church, A.C. Vogel, T.O. Laumann, F.M. Miezin, B.L. Schlaggar, and S.E. Petersen, "Functional network organization of the human brain," *Neuron*, vol. 72, no. 4, pp. 665–678, 2011.

[27] O. Sporns, *Networks of the Brain*. MIT Press, 2010.

[28] M. Rubinov and O. Sporns, "Complex network measures of brain connectivity: Uses and interpretations," *Neuroimage*, vol. 52, no. 3, pp. 1059–1069, 2010.

[29] M.W. Cole, J.R. Reynolds, J.D. Power, G. Repovs, A. Anticevic, and T.S. Braver, "Multi-task connectivity reveals flexible hubs for adaptive task control," *Nat. Neurosci.*, vol. 16, no. 9, pp. 1348–1355, 2013.

[30] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, pp. 227–240, April 2013.

[31] J. Soares, C. Goncalves, B. Parreira, P. Tavares, J. Carapinha, J. Barraca, R. Aguiar, and S. Sargento, "Toward a telco cloud environment for service functions," *Communications Magazine, IEEE*, vol. 53, pp. 98–106, February 2015.

[32] Y.K. Chang, C.I. Lee, and C.C. Su, "Multi-field range encoding for packet classification in tcam," in *Proceedings of the 30th IEEE international conference on Computer Communications*, pp. 196–200, April 2011.

[33] D. Bhamare, R. Jain, M. Samaka, G. Vaszkun, and A. Erbad, "Multi-cloud distribution of virtual functions and dynamic service deployment: Open ADN perspective," in *2015 IEEE International Conference on Cloud Engineering, IC2E 2015, Tempe, AZ, USA, March 9–13, 2015*, pp. 299–304, 2015.

[34] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proceedings of the 31st International Conference on Distributed Computing Systems*, pp. 559–570, June 2011.