Electronic Preprint for Journal of Information Processing Vol.24 No.6

Regular Paper

Adaptive Function to Ensure Robustness of MCMC-based Autonomous Decentralized Control Mechanism against Changing Environment

Yusuke Sakumoto[1,a)] Masaki Aida[1,b)] Hideyuki Shimonishi[2,c)]

Received: January 18, 2016, Accepted: July 5, 2016

Abstract: In our previous work, we proposed an autonomous decentralized control mechanism (ADCM) to be able to control the probability distribution of a system performance variable on the Markov chain Monte Carlo method. As interesting features of our ADCM, we proved the probability distribution controlled by our ADCM is described by a few macro parameters, and discovered a law between the macro parameters and the external environment of the system in our ADCM. In this paper, on the basis of the law, we design an autonomous decentralized adaptive function to adapt to change in the external environment. This function ensures the robustness of our ADCM against changing environment. We apply our ADCM with the proposed adaptive function to a virtual machine placement problem in data center networks (DCNs). Simulation experiments confirm that the proposed adaptive function effectively deals with several DCN scenarios with changing environment.

Keywords: large-scale and wide-area system, autonomous decentralized control, data center network, virtual machine placement problem, adaptive control

1. Introduction

Information technology plays important role in people's life. Several systems supporting information technology are constructed on worldwide networks, and its system scale is rapidly increasing to improve or maintain service quality required by people. Examples of such a large-scale and wide-area system are Microsoft and Google's data centers [1], [2]. Currently, these data centers are composed of approximately one million computers distributed all over the world. In the future, the number of computers in these systems will continue to increase at a rapid rate. Hence, such large-scale and wide-area networks need scalable control mechanisms to accommodate the sheer number of computers anticipated.

Centralized control is one commonly-used approach to system management. In a centralized control mechanism, a supervising node gathers and processes complete state information from the entire system, and then controls the other nodes. The supervising node knows system behavior from the gathered state information, and would be able to adjust the states of the other nodes appropriately. Thanks to the supervising node, centralized control mechanisms are simple, and offer good usability. However, it is impossible to apply centralized controls to large-scale and wide-area systems because of information gathering limitation, and control frequency requirement. In a wide-area system, the gathering of state information from all nodes will take too long, and so the supervising node cannot control the other nodes frequently. However, the appropriate states of the nodes in a large-scale system are always changing due to unpredictable reasons (e.g., device failure and sudden increase in service demand), and so the supervising node should control them frequently. Therefore, centralized controls are not scalable, and an alternative to centralized control mechanism is needed for large-scale and wide-area systems.

For controlling large-scale and wide-area systems, several autonomous decentralized control mechanisms (ADCMs) have been proposed in Refs. [3], [4], [5], [6], [7]. They do not depend on supervising nodes. In an ADCM, each node gathers information from its local area and takes action to control its state. ADCMs would be scalable, but the problem of not depending on supervising nodes is that it is difficult to control the entire behavior of a large-scale and wide-area system in a desirable direction. Fortunately, the ADCMs [3], [6], [7] solved this problem with the inspiration from the physical universe.

The physical universe contains many examples of large-scale and wide-area systems. For example, water in a beaker is composed of innumerable molecules, and its overall extent is many orders of magnitude larger than that of its constituent molecules. Each molecule behaves autonomously, but it is easy to know and control its overall behavior with a macroscopic operation, despite numerous degrees of freedom of molecules. We, for instance, can measure and control the water's temperature with a heater. This property is understood with the hierarchical structure (**Fig. 1**) to be able to simplify numerous degrees of freedom in the physical system. Thanks to this hierarchical structure, the system can be described by a few degrees of freedom at the macroscopic scale.

[1] Tokyo Metropolitan University, Hino, Tokyo 191–0065, Japan
[2] NEC Corporation, Kawasaki, Kanagawa 211–0011, Japan
[a)] sakumoto@tmu.ac.jp
[b)] aida@tmu.ac.jp
[c)] h-shimonishi@cd.jp.nec.com

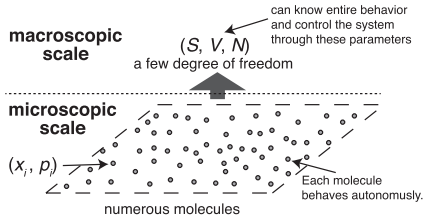

Fig. 1 The hierarchical structure in physical systems.

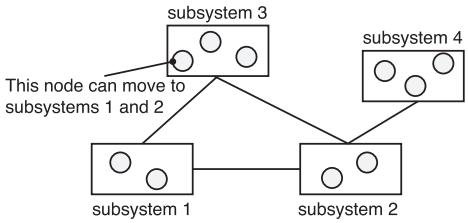

Fig. 2 An example of the system model for $N_S = 4$ and $N = 10$.

In Ref. [7], inspired by the hierarchical structure in the physical system, we have proposed an ADCM that offers good usability and quick response in large-scale and wide-area systems.

In Ref. [7], we designed an ADCM based on Markov chain Monte Carlo (MCMC) [8] for large-scale and wide-area systems. MCMC is used to design a state transition probability for controlling the probability distribution of a state quantity (e.g., energy) in models (e.g., Ising spin model) of statistical mechanics. In our ADCM, each node autonomously changes its state using the MCMC-based state transition probability that is able to indirectly control probability distribution of a system performance variable in a desirable direction. Moreover, we proved that the controlled probability distribution is described by just a few macro parameters, and discovered the law between these macro parameters and the external environment of the system. This law would have application potentiality to design macroscopic operations like physical systems. However, we have not discussed the application potentiality.

In this paper, to realize macroscopic operations like physical systems, we design an autonomous decentralized adaptive function based on the law of our ADCM. This function ensures the robustness of our ADCM. From the law, we first derive a condition to adapt to change in the external environment for ensuring the robustness. Then, we design an adaptive function with autonomous decentralized manner to satisfy the derived condition against changing environment. Moreover, we apply the proposed adaptive function to a virtual machine (VM) placement problem in a data center network (DCN) as in Ref. [7]. To investigate the performance of the proposed adaptive function, we perform simulation experiment with changing traffic rates among VMs. Simulation experiments confirm that the proposed adaptive function can deal effectively with several scenarios with changing environment. In the conference paper [9], we have reported simulation results when homogeneously changing traffic rates among VMs. In this paper, we also show the effectiveness of the proposed adaptive function against changing traffic rates heterogeneously. Moreover, we investigate the effect of the graph diameter on the effectiveness of the proposed adaptive function. This investigation would help in estimating performance of the proposed adaptive function in large-scale systems.

This paper is organized as follows. Section 2 explains the system model used in this paper. Section 3 introduces our ADCM proposed in Ref. [7]. Section 4 proposes the autonomous decentralized adaptive function that ensures control robustness against changing environment. Section 5 details the experiments conducted to investigate the effectiveness of the proposed adaptive function. Section 6 describes the related work for the MCMC-based ADCM and the proposed adaptive function. Finally, in Section 7, we conclude this paper and discuss future work.

2. System Model

We assume a system with N_S subsystems and N nodes. Each node is assigned to a subsystem, and collaborating with some other nodes by using the subsystem's resource. Node state x_i of node i is the identification number of its assigning subsystem ($x_i \in \{1, \ldots, N_S\}$). Let ϕ_k be the set of nodes assigned to subsystem k. Each subsystem has adjacency relationship with other subsystems. Let a_k be the set of subsystems that have adjacency relationship with subsystem k. A node $i \in \phi_k$ is permitted to move to one of the other subsystems, $l \in a_k$. **Figure 2** shows an example of the system model for $N_S = 4$ and $N = 10$.

System state $\boldsymbol{X}$ is given by all node states $(x_1, x_2, \ldots, x_N)$. In Ref. [7], we define system performance variable $M(\boldsymbol{X})$ for node collaboration as

$$M(\boldsymbol{X}) = \sum_{i=1}^{N} \sum_{j \in \chi_i} m_{ij}(x_i, x_j), \tag{1}$$

where χ_i is the set of nodes collaborating with node i. $m_{ij}(x_i, x_j)$ represents the weakness of the collaboration between nodes i and j. System performance variable $M(\boldsymbol{X})$ represents system-level weakness of the collaboration among nodes. Small $m_{ij}(x_i, x_j)$ means that nodes i and j collaborate strongly. If nodes i and j are assigned to one subsystem (i.e., $x_i = x_j$), they are most strongly collaborating. When all nodes are assigned to one subsystem, $M(\boldsymbol{X})$ is minimized. In this assignment, all nodes can most strongly collaborate, but load of the assigning subsystem is terrible. Hence, $M(\boldsymbol{X})$ should be controlled by considering the strengths of node collaboration and node distribution.

We assume $m_{ii}(x_i, x_i) = 0$ and $m_{ij}(x_i, x_j) = m_{ji}(x_j, x_i)$. Other values of $m_{ij}(x_i, x_j)$ are determined by information of the external environment. Hence, external environment affects $m_{ij}(x_i, x_j)$ and $M(\boldsymbol{X})$.

3. MCMC-based ADCM

3.1 Autonomous Node Action

In Ref. [7], on the basis of MCMC [8], [10], we designed an autonomous node action to indirectly control the probability distribution of system performance variable M. We showed the designed node action can adjust the strengths of node collaboration and node distribution, simultaneously.

In the designed node action, node i changes its state x_i to $x_i' \in a_{x_i}$ according to the following probability

$$T_i(x_i \to x_i') = \begin{cases} \dfrac{1}{|a_{x_i}|} e^{-\alpha\,\lambda\,\Delta m_i(x_i \to x_i')} \\ \qquad \text{if } \Delta m_i(x_i \to x_i') < 0 \\ \dfrac{1}{|a_{x_i}|} e^{-(1-\alpha)\lambda\,\Delta m_i(x_i \to x_i')} \\ \qquad \text{otherwise} \end{cases}, \tag{2}$$

where λ is the control parameter used in common by nodes, and α is a positive parameter ($0 < \alpha < 0.5$). In Eq. (2), $\Delta m_i(x_i \to x_i')$ is the difference in system performance variable M for node states x_i and x_i'; it is calculated by

$$\Delta m_i(x_i \to x_i') = \sum_{j \in \chi_i} \Big[m_{i\,j}(x_i', x_j) - m_{i\,j}(x_i, x_j) \Big]. \tag{3}$$

If $\lambda = 0$, $T_i(x_i \to x_i')$'s for any x_i' are the same value, and so x_i behaves as a uniformly distributed random variable. As λ increases, each node tends to select a node state to realize smaller $M(\boldsymbol{X})$, and so the collaboration among nodes becomes strong.

3.2 Global Property

In Ref. [7], we derived Eq. (2) so as to have stationary distribution $p(\boldsymbol{X}) = A\,e^{-\lambda M(\boldsymbol{X})}$ where A is a normalizing constant of $p(\boldsymbol{X})$. According to MCMC [10], system state $\boldsymbol{X}$ follows a stationary distribution $p(\boldsymbol{X}) = A\,e^{-\lambda M(\boldsymbol{X})}$ if state transition probability $p(\boldsymbol{X'}|\boldsymbol{X})$ satisfies the condition

$$p(\boldsymbol{X'}|\boldsymbol{X})\,p(\boldsymbol{X}) = p(\boldsymbol{X}|\boldsymbol{X'})\,p(\boldsymbol{X'}), \tag{4}$$

and the ergodic condition, that is, the probabilities of Markov chains of X with arbitrary length more than a certain value moving from one system state to any of the other system states is larger than 0.

$T_i(x_i \to x_i')$ given by Eq. (2) satisfies the condition Eq. (4). Consider $\boldsymbol{X'} = (x_1', \ldots, x_N')$ where $x_i \neq x_i'$ and $x_j = x_j'$ to focus on the autonomous node action of node i. If $p(\boldsymbol{X}) = Ae^{-\lambda M(\boldsymbol{X})}$, Eq. (4) is rewritten by

$$\begin{aligned} \frac{p(\boldsymbol{X'}|\boldsymbol{X})}{p(\boldsymbol{X}|\boldsymbol{X'})} &= \frac{p(\boldsymbol{X'})}{p(\boldsymbol{X})} = e^{-\lambda\,(M(\boldsymbol{X'}) - M(\boldsymbol{X}))} \\ &= e^{-\lambda \sum_{j \in \chi_i} (m_{i\,j}(x_i', x_j) - m_{i\,j}(x_i, x_j))} \\ &= e^{-\lambda\,\Delta m_i(x_i \to x_i')} \\ &= \frac{e^{-\alpha\,\lambda\,\Delta m_i(x_i \to x_i')}}{e^{(1-\alpha)\,\lambda\,\Delta m_i(x_i \to x_i')}} \\ &= \frac{\frac{1}{|a_{x_i}|} e^{-\alpha\,\lambda\,\Delta m_i(x_i \to x_i')}}{\frac{1}{|a_{x_i}|} e^{-(1-\alpha)\,\lambda\,\Delta m_i(x_i' \to x_i)}} \\ &= \frac{T_i(x_i \to x_i')}{T_i(x_i' \to x_i)}. \end{aligned} \tag{5}$$

In the process of deriving Eq. (5), we conventionally replace 2λ with λ. Hence, changing node state x_i using Eq. (2), probability distribution $p(\boldsymbol{X})$ is controlled to $A\,e^{-\lambda M(\boldsymbol{X})}$ according to the above discussion.

Following $p(\boldsymbol{X})$ by $A\,e^{-\lambda M(\boldsymbol{X})}$, system states $\boldsymbol{X}$'s with the same value of system performance variable $M(\boldsymbol{X})$ occur with the same probability, and cannot be distinguished. Hence, by summing probabilities $p(\boldsymbol{X})$ with the same $M(\boldsymbol{X})$, probability distribution $p(M)$ of system performance variable M is given by

$$p(M) = \frac{G(M)e^{-\lambda M}}{\sum_{Y \in \Omega_M} G(Y)e^{-\lambda Y}} = \frac{e^{-\lambda F(M)}}{\sum_{Y \in \Omega_M} e^{-\lambda F(Y)}}, \tag{6}$$

where Ω_M is the set of all possible M, and $F(M) := M - 1/\lambda \log G(M)$. $G(M)$ is the system state distribution, which is the number of system states $\boldsymbol{X}$ with the same value of system performance variable M.

Equation (6) shows that the autonomous node action can indirectly control the distribution of system performance variable M in the direction desired. If $\lambda = 0$, $p(M)$ is simply proportional to $G(M)$, and all possible system states are realized with equal probability. Hence, our ADCM with $\lambda = 0$ corresponds to a mechanism that randomly selects system state $\boldsymbol{X}$ from system state space Ω. As λ increases, the probability distribution is shifted according to $e^{-\lambda M}$, and system states with small $M(\boldsymbol{X})$ become realized with higher probability.

In statistical mechanics, the probability distribution given by Eq. (6) is called the *Boltzmann distribution*, which is well understood. Some variables in our ADCM are related to variables in statistical mechanics; λ, M, $F(M)$, and $\log G(M)$ correspond to the inverse of temperature, the energy, the Helmholtz free energy, and the entropy in statistical mechanics. In Ref. [7], we clarified the global property of our ADCM on the basis of statistical mechanics. According to the clarified global property, we found that our ADCM yields a hierarchical structure that has node-level and system-level layers. At the system-level layer, there is the law between statistics (i.e., average and standard deviation) of M and statistics depending on the external environment of the system.

We explain the law on the system-level layer found in Ref. [7]. According to the assumption of Ref. [7], we first assume that system performance variable M can be modeled as a continuous quantity. This assumption is valid for large-scale systems. In Ref. [7], we discussed the global property around M^* because it is dominant in large-scale systems. $F(M)$ in Eq. (6) determines the maximum point, M^*, of Boltzmann distribution. Namely, M^* can be derived by solving $dF(M)/dM = 0$. To discuss the global property around M^*, we convert $F(M)$ to a Taylor series at M^*. The Taylor series of $F(M)$ at M^* is given by

$$F(M) = F(M^*) + \frac{1}{\lambda} \sum_{k=2}^{\infty} \frac{C_k}{k!} (M - M^*)^k, \tag{7}$$

where

$$C_k := -\frac{d^k}{dM^k} \log G(M) \Big|_{M=M^*}. \tag{8}$$

Because M is generally a monotonically increasing function of N, the derivatives of $\log G(M)$ for $k \geq 3$ decrease more quickly than that for $k = 2$ as N increases. Hence, in large-scale systems, $F(M)$ can be approximated by

$$F(M) \simeq F(M^*) + \frac{C_2}{2\lambda} (M - M^*)^2. \tag{9}$$

By substituting Eq. (9) to Eq. (6) and normalizing the substituted equation with $\sum_{M \in \Omega_M} p(M) = 1$, $p(M)$ is given by the following normal distribution

$$p(M) \simeq \frac{1}{\sqrt{2\pi}\sigma_\lambda} \exp\left[-\frac{(M - \mu_\lambda)^2}{2\sigma_\lambda^2} \right], \tag{10}$$

where $\mu_\lambda = M^*$ and $\sigma_\lambda = 1/\sqrt{C_2}$. μ_λ and σ_λ are the average and standard variance of M when our ADCM uses control parameter

λ, respectively. By substituting Eq. (9) to $G(M) = e^{\lambda(M-F(M))}$ and normalizing the substituted equation with $\sum_{M\in\Omega_M} G(M) = |\Omega|$, $G(M)$ is approximated by

$$G(M) \simeq \frac{|\Omega|}{\sqrt{2\pi}\sigma_G}\exp\left[-\frac{(M-\mu_G)^2}{2\sigma_G^2}\right], \tag{11}$$

where $\mu_G = M^* + \lambda/C_2$ and $\sigma_G = 1/\sqrt{C_2}$. μ_G and σ_G are the average and standard variance of M, respectively, for all system states in Ω. They are also given by

$$\mu_G = \frac{1}{|\Omega|}\sum_{X\in\Omega} M(X), \tag{12}$$

$$\sigma_G^2 = \frac{1}{|\Omega|}\sum_{X\in\Omega} (M(X)-\mu_G)^2 . \tag{13}$$

When repeating the random selection of system state $\boldsymbol{X}$ from Ω, the average and standard variance of occurred M approach μ_G and σ_G, respectively. Hence, μ_G and σ_G mean the average and standard variance of M for such a random selection. Since $m_{ij}(x_i, x_j)$ in $M(\boldsymbol{X})$ is given by information of the external environment, μ_G and σ_G depend on the external environment. Hence, changes in the environment alter μ_G and σ_G. According to Eqs. (10) and (11), $\mu_\lambda = M^*$, $\mu_G = M^* + \lambda/C_2$, and $\sigma_\lambda = \sigma_G = 1/\sqrt{C_2}$. Hence, on the system-level layer, μ_λ and σ_λ are given by using μ_G and σ_G as follows

$$\mu_\lambda = \mu_G - \lambda\sigma_G^2, \tag{14}$$

$$\sigma_\lambda = \sigma_G. \tag{15}$$

According to the above equations, we find $\mu_0 = \mu_G$ and $\sigma_0 = \sigma_G$. From the law given by Eqs. (14) and (15), we can understand the relation between the external environment and our ADCM.

4. Autonomous Decentralized Adaptive Function to Ensure Control Robustness

4.1 Condition to Ensure Control Robustness against Changes in the Environment

To ensure the robustness, we should retain the control strength (i.e., strength of collaboration among nodes) of our ADCM against changing environment. Control strength is measured by the difference between states with randomly selected states ($\lambda = 0$) and states controlled by our ADCM. Since randomly selected states depend on the environment, changing environment fluctuates the average of system performance variable M for randomly selected states (i.e., μ_G). There are several definitions of control strength. In this paper, we define the control strength by the ratio of μ_λ divided by μ_G. In this definition, when the ratio of μ_λ is small, the control of our ADCM is strong. Hence, the ratio of μ_λ means the weakness of control by our ADCM. This definition is reasonable because it allows us to design an adaptive function with the autonomous decentralized manner.

Following the definition of the above-mentioned control strength, we introduce variable R by

$$R = \frac{M}{\mu_G}. \tag{16}$$

R represents the instantaneous control weakness of our ADCM, and its average is invariant against changes in the environment,

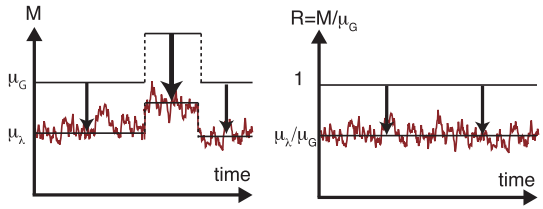

Fig. 3 System performance variable M vs. weakness index R; the average of R, μ_λ/μ_G, is invariant against changing environment if the target control strength of our ADCM is unchanged.

see **Fig. 3**. In what follows, we call R *weakness index* since the average of R represents the weakness of our ADCM.

In what follows, we derive a condition to retain the probability distribution of R against changes in the environment. With the change of variables from M to R for probability distribution $p(M)$, probability distribution $p(R)$ is given by

$$p(R) \simeq \frac{1}{\sqrt{2\pi}\sigma_R}\exp\left[-\frac{(R-\mu_R)^2}{2\sigma_R^2}\right], \tag{17}$$

where $\mu_R = \mu_\lambda/\mu_G$ and $\sigma_R = \sigma_\lambda/\mu_G$.

To retain control strength, μ_R and σ_R in $p(R)$ should remain constant against changes in μ_G and σ_G, that is

$$\mu_R = \frac{\mu_\lambda}{\mu_G} = 1 - \lambda\frac{\sigma_\lambda^2}{\mu_G} = K_\mu, \tag{18}$$

$$\sigma_R = \frac{\sigma_\lambda}{\mu_G} = K_\sigma, \tag{19}$$

where K_μ and K_σ are variables depending on the environment, but should be constant against change in μ_G and σ_G. The conditions Eqs. (18) and (19) are reduced to single condition

$$\lambda\mu_G = K, \tag{20}$$

where $K = (1-K_\mu)/K_\sigma^2$. K is set to a value considering target control strength of our ADCM by a system manager.

4.2 Design

According to condition Eq. (20), to ensure the robustness, our ADCM should reconfigure control parameter $\lambda(t)$ at time t by

$$\lambda(t) = \frac{\tilde{\mu}_G}{\mu_G}\,\lambda(t-\Delta T_\lambda), \tag{21}$$

where $\tilde{\mu}_G$ is the average used in the last reconfiguration, and ΔT_λ is time interval of the reconfiguration of λ. The value of ΔT_λ depends on the time needed to obtaining μ_G from the system. Hence, the rapid obtaining of μ_G leads to fast response to an environment fluctuation. To reduce the setting value of ΔT_λ, we design a method for obtaining μ_G in an autonomous decentralized manner.

For easy understanding, in what follows, we use $m_{ij}(x_i, x_j)$ defined as

$$m_{ij}(x_i, x_j) = f_{ij}\, d_{x_i x_j}, \tag{22}$$

where f_{ij} is a coefficient for the collaboration between nodes i and j, and d_{kl} is a cost to communicating between subsystems k and l. We assume that f_{ij} and d_{kl} are independent of each other. Smaller $d_{x_i x_j}$ means stronger collaboration between nodes i and j.

According to Eq. (12), μ_G with Eq. (22) is given by

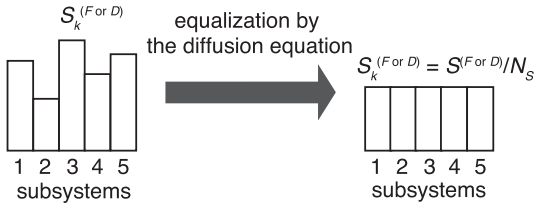

Fig. 4 Equalizing the parts of $S^{(X)}$, $S_k^{(X)}$ for all subsystems in the adaptive function.

$$\mu_G = \frac{1}{N_S(N_S-1)}\left(\sum_{i=1}^{N}\sum_{j\in\chi_i} f_{ij}\right)\left(\sum_{k=1}^{N_S}\sum_{l=1}^{N_S} d_{kl}\right) = \frac{1}{N_S(N_S-1)} S^{(F)} S^{(D)}, \tag{23}$$

where $S^{(F)}$ is the sum of f_{ij} for all pairs of nodes, and $S^{(D)}$ is the sum of d_{kl} for all pairs of subsystems. To calculate μ_G and reconfigure λ, we need to obtain $S^{(F)}$ and $S^{(D)}$.

We explain here that each subsystem can obtain $S^{(F)}$ and $S^{(D)}$ by using the autonomous decentralized method based on the diffusion equation. A subsystem cannot directly calculate $S^{(F)}$ and $S^{(D)}$ from own information, but it can calculate a part of $S^{(F)}$ and $S^{(D)}$. For example, subsystem k can calculate $\left(\sum_{i\in\phi_k}\sum_{i\in\chi_i} f_{ij}\right)$, which is a part of $S^{(F)}$. Each subsystem has a different part of $S^{(F)}$, but each subsystem obtains $S^{(F)}/N_S$ by equalizing their parts of $S^{(F)}$ while conserving the sum of parts of $S^{(F)}$ for all subsystems, see **Fig. 4**. This equalization of $S^{(F)}$ can be performed by using the autonomous decentralized method based on the diffusion equation. The same thing is possible for $S^{(D)}$. By using $S^{(F)}/N_S$ and $S^{(D)}/N_S$ obtained in the above manner, each subsystem can reconfigure control parameter λ by Eq. (21).

The autonomous decentralized method based on the diffusion equation is summarized as follows. Let $S_k^{(F)}(t)$ and $S_k^{(D)}(t)$ be parts of $S^{(F)}$ and $S^{(D)}$ at time t for subsystem k, respectively.

1) Every ΔT_λ, subsystem k updates $S_k^{(F)}(t)$ and $S_k^{(D)}(t)$ by using the following equations

$$S_k^{(F)}(t) \leftarrow S_k^{(F)}(t-\Delta T_\lambda) + \frac{1}{2}\sum_{i\in\phi_k}\sum_{j\in\chi_i}\left(f_{ij} - \tilde{f}_{ij}\right), \tag{24}$$

$$S_k^{(D)}(t) \leftarrow S_k^{(D)}(t-\Delta T_\lambda) + \frac{1}{2}\sum_{l=1}^{N_S}\left(d_{kl} - \tilde{d}_{kl}\right), \tag{25}$$

where f_{ij} and d_{kl} are the values at a referenced time. $\tilde{f}_{ij}$ and $\tilde{d}_{kl}$ are the values of f_{ij} and d_{kl} used in the last update by Eqs. (24) and (25). By using the previous information of $S_k^{(F)}(t-\Delta T_\lambda)$ and $S_k^{(D)}(t-\Delta T_\lambda)$), $S_k^{(F)}(t)$ and $S_k^{(D)}(t)$ can be equalized rapidly.

2) Every ΔT_D, subsystem k updates $S_k^{(X)}(t)$ $(X \in F, D)$ by using the following discrete diffusion equation

$$S_k^{(X)}(t) \leftarrow S_k^{(X)}(t-\Delta T_D) + \kappa_D \sum_{l\in a_k}\left(S_l^{(X)}(t-\Delta T_D) - S_k^{(X)}(t-\Delta T_D)\right), \tag{26}$$

where κ_D is the diffusion coefficient $(0 < \kappa_D < 1/\Delta T_D)$. Equation (26) can be calculated using only $S_l^{(X)}(t-\Delta T_D)$ of adjacency subsystems $l \in a_k$. ΔT_D can be determined from the maximum communication delay of adjacency subsystems. Subsystem k repeats the update by Eq. (26) N_D times. Thus, $\Delta T_\lambda > N_D \Delta T_D$.

3) Subsystem k reconfigures own control parameter $\lambda_k(t)$ at time t according to Eq. (21). Specifically, $\lambda_k(t)$ is updated by

$$\lambda_k(t) = \frac{S_k^{(F)}(t-\Delta T_\lambda) S_k^{(D)}(t-\Delta T_\lambda)}{S_k^{(F)}(t) S_k^{(D)}(t)} \lambda_k(t-\Delta T_\lambda). \tag{27}$$

Note that we assume that N_S is invariant during $[t-\Delta T_\lambda, t]$.

4) Node i in subsystem k uses control parameter λ_k to calculate state transition probability T_i.

The proposed adaptive function works in the MCMC-based ADCM. Hence, its restrictions to apply to solve a given problem are followed by those of the MCMC-based ADCM. Due to paper limits, we omit the explanation of the restrictions from this paper. Please see Section 2 of Ref. [7].

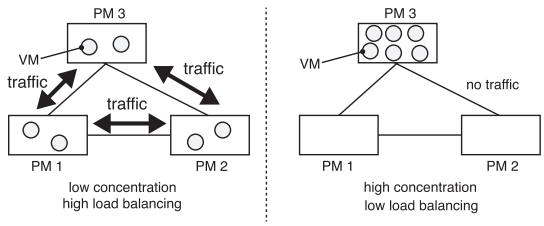

Fig. 5 Examples of VM placement in a DCN.

4.3 Application to Virtual Machine Placement

In Ref. [7], we applied our ADCM to a VM placement problem in DCNs. In a DCN, the distribution of traffic rates between VMs is uneven [11], and it is difficult to alter the topology to suit the traffic rates since they change frequently. Hence, for better network performance, a DCN controller should realize traffic concentration in the given topology by placing VMs that are handling high traffic rates in the neighborhood of physical machines (PMs). Traffic concentration may lead to concentrating VM loads on a few PMs, and thus degrade their computing performance (**Fig. 5**). Hence, load balancing between PMs should be performed with traffic concentration, simultaneously. In Ref. [7], we proposed an ADCM that performs traffic-aware VM placement with load balancing based on MCMC, and confirmed its effectiveness.

In Ref. [7], we formulated the VM placement problem on the basis of the system model explained in Section 2. VM and PM correspond to node and subsystem in our system model, respectively. As $m_{ij}(x_i, x_j)$, we use the traffic-cost product, which is a variable for traffic concentration also used in Ref. [11]. f_{ij} and d_{kl} in Eq. (22) correspond to the traffic rate between VMs i and j and the communication cost between PMs k and l, respectively. A smaller traffic-cost product $m_{ij}(x_i, x_j)$ is better for traffic concentration because a small $d_{x_i x_j}$ indicates that VMs i and j are in the neighborhood of PMs. By controlling the probability distribution of traffic-cost product sum M, our ADCM can adjust both the strengths of traffic concentration and load balancing in DCNs.

5. Experiment

Through simulation experiments, we confirm that the proposed adaptive function improves the robustness against changing envi-

ronment. The simulation examines the placement of VMs by our ADCM.

5.1 Experiment Model

Due to space limitation in this paper, we describe only the fat-tree topology, one of the network topologies often found in data center studies (e.g., Ref. [11]). PMs are placed in layer 0 of the network topology, and separated into N_G groups. The communication cost of a pair of PMs is given by the sum of link costs on the shortest path. PM k is assigned to group $\lfloor k/N_G + 1 \rfloor$. PMs in group i have an adjacency relationship (edge) with PMs in groups i and $i \pm 1$. As the exception, PMs in groups 1 and N_G have mutual adjacency relationship. VM i is permitted to migrate to a neighbor PM of PM x_i. Let diameter of the adjacency relationship graph of PMs be the maximum number of relationships between any PM pair. In the setting of this paper, the diameter is given by $N_G - 1$. In general, equalization speed of the diffusion equation in a graph depends on its diameter. Hence, the diameter of the adjacency relationship graph may affect the effectiveness of the proposed adaptive function.

Figure 6 shows an example of the network topology with 16 PMs and 4 groups. In the example, the diameter is 3.

For investigating the effectiveness of the proposed adaptive function, we introduce a traffic model that generates changing environment. The traffic model is based on measured results [12], [13]. In Ref. [12], the authors investigated the total traffic amount in a DCN, and showed that the total traffic amount increases and decreases rapidly. In Ref. [13], the authors investigated the link bandwidth utilization of a DCN router, and reported gradual changes (increase and decrease) in the utilization rate. According to the findings [12], [13], the proposed adaptive function should be effective regardless of the rate of traffic change. On a day timescale, the time evolution of traffic reported in Ref. [13] is divided into increasing periods and decreasing periods. Hence, we assume the overall period to consist of increasing and decreasing periods, and confirm the effectiveness of the proposed adaptive function in each period. According to the above discussion, the traffic model should be able to (a) adjust the average rate of traffic, and (b) switch the behavior of changing traffic to increase or decrease. We design such a traffic model that satisfies the above requirements with the fewest parameters to reduce the complexity involved in investigation.

We illustrate the basic ideas that underlie the traffic model in **Fig. 7**. In the traffic model, we divide the overall traffic behavior into increasing periods and decreasing periods, and approximate the continuous time evolution as a series of discrete time events. There are unpredictable factors that are likely to change the traffic in a real system, and so the simulation randomly generates each traffic change event. The time interval between each pair of consecutive traffic change events follows an exponential distribution with average time ΔT_f. When a traffic change event for traffic rate f_{ij} occurs, traffic rate f_{ij} increases by amount Δf_{ij} with probability of p_+, or otherwise decreases by amount Δf_{ij}. In the traffic model, the average change rate for f_{ij}, $\overline{f_{ij}}$, is given by

$$\overline{f_{ij}} = \frac{\Delta f_{ij}\,(2\,p_+ - 1)}{\Delta T_f}. \tag{28}$$

According to Eq. (28), the traffic model can adjust the average change rate $\overline{f_{ij}}$ by changing Δf_{ij} or ΔT_f, and switch increase period or decrease period by changing p_+.

At the start of each simulation run, we place N VMs in a randomly chosen PM, and set control parameter of PM k, λ_k, to λ_{INIT}. At each simulation time unit, a VM uses the local action to determine to which PM it should migrate. If the proposed adaptive function is activated, PM k adjusts its control parameter λ_k every ΔT_λ by using the procedures explained in Section 4.

Initially, each VM communicates with randomly chosen N_H (average) VMs with high traffic rate T_H, and other VMs with low traffic rate T_L. Let $f_{ij}^{(\mathrm{INIT})}$ be the initial value of traffic rate f_{ij} between nodes i and j. During a simulation, we change (a) all traffic rates or (b) only high rates (i.e., f_{ij} if $f_{ij}^{(\mathrm{INIT})} = T_H$), according to the above traffic model. It is unnatural to assume that the high rates change with the same rule as the low rates. Hence, we should consider not only homogeneous traffic changes like setting (a) but also heterogeneous traffic changes like setting (b). Since only some traffic rates are changed in traffic setting (b), the variance of traffic rates in setting (b) is larger than in setting (a). Unless explicitly stated, we use setting (a). As the configuration of initial traffic rates (T_H, T_L) in settings (a) and (b), we use values shown in **Table 1**.

The other parameters are set to the values shown in **Table 2**. In particular, we set Δf_{ij} to cover the large range that includes the change rates shown in Refs. [12], [13]. We examine a small-scale system with $N = 160$ due to our computation limits. However, thanks to the statistical effect, our ADCM works more effectively as N increases [7]. In principle, the results obtained in this simulation are also valid for large-scale networks.

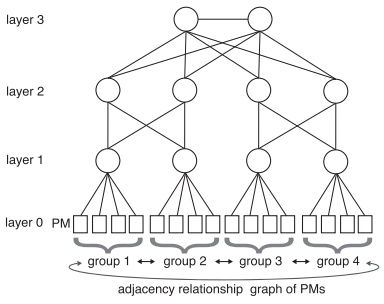

Fig. 6 An example of the network topology for $N_S = 16$ and $N_G = 4$.

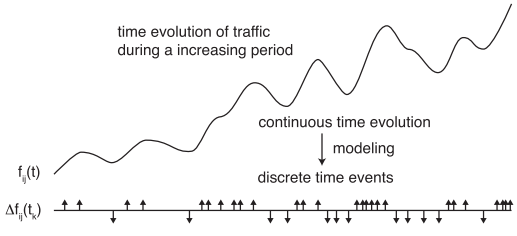

Fig. 7 Traffic model to generate changing environment.

Table 1 Configuration of initial traffic rates (T_H, T_L).

	$p_+ = 0.7$ $\kappa = 0.001, 0.1$	$p_+ = 0.3$ $\kappa = 0.001$	 $\kappa = 0.1$
setting (a)	(10, 0.1)	(810, 8.1)	(80,010, 800.1)
setting (b)	(10, 0.1)	(810, 0.1)	(80,010, 0.1)

Table 2 Parameter configuration.

number of PMs, N_S	16
number of VMs, N	160
number of groups, N_G	4
average number of high traffic VMs, N_H	10
internal communication cost of a PM	0.0001
link cost for the network topology	0.1
interval of traffic changes, ΔT_f	1
changing amount Δf_{ij} for $f_{ij}^{(\mathrm{INIT})} = T_H$	10κ
changing amount Δf_{ij} for $f_{ij}^{(\mathrm{INIT})} = T_L$	0.1κ
coefficient in Δf_{ij}, κ	0.1 or 0.001
VM load $\rho^{(VM)}$	1
parameter α	0.1
simulation time	150,000
diffusion coefficient κ_D	0.1
number of updates, N_D	5
update interval of λ, ΔT_λ	1

The VM placement with our ADCM yields load balancing of PMs. As the metric for load balancing in DCNs, we use PM load variance Var[$\boldsymbol{\rho}$] ($\boldsymbol{\rho} = (\rho_1, \ldots, \rho_{N_S})$) of PM loads

$$\mathrm{Var}[\boldsymbol{\rho}] = \frac{1}{N_S}\sum_{l=1}(\rho_l - \mathrm{E}[\boldsymbol{\rho}])^2, \tag{29}$$

where $\rho_l = \sum_{i\in\phi_l}\rho_i^{(\mathrm{VM})}$ and $\rho_i^{(\mathrm{VM})}$ is VM i's load. ρ_T is the average of PM loads, which is given by

$$\mathrm{E}[\boldsymbol{\rho}] = \frac{1}{N_S}\sum_{l=1}^{N_S}\rho_l. \tag{30}$$

There is a trade-off between weakness index R and PM load variance Var[$\boldsymbol{\rho}$]. The VM placement problem in DCNs is formulated by multi-objective functions R and Var[$\boldsymbol{\rho}$]. Following a given control strength, our ADCM tries to adjust the balance of R and Var[$\boldsymbol{\rho}$], but the attempt would fail against changing traffic. We expect that our ADCM with the proposed adaptive function retains R and Var[$\boldsymbol{\rho}$] against traffic changes, solving the problem of our ADCM. In what follows, through simulation experiments, we will confirm that the expectation is true whether

5.2 Simulation Results

The figures of this section illustrate the result of using or not using the proposed adaptive function with the legend *w adapt* or *w/o adapt*, and result of changing or not changing traffic rate with the legend *fluc.* or *no fluc.*, respectively.

We first visually confirm that the problem of our ADCM for changing environment occurs, and the adaptive function can solve the problem of our ADCM. **Figures 8** and **9** show the time evolution of weakness index R and PM load variance Var[$\boldsymbol{\rho}$] when the traffic rate gradually increases (i.e., $p_+ = 0.7$ and $\kappa = 0.001$). According to these figures, when the proposed adaptive function is disabled, weakness index R decreases drastically, and PM load variance Var[$\boldsymbol{\rho}$] increases dramatically. In this simulation, our ADCM places all VMs into a PM. This phenomenon means that our ADCM allows high traffic concentrations without the proposed adaptive function due to the increasing traffic. However, when the proposed adaptive function is enabled, weakness index R and PM load variance Var[$\boldsymbol{\rho}$] remain roughly constant in the face of the traffic changes. Hence, the proposed adaptive function can effectively deal with traffic fluctuation.

We next confirm the effectiveness of the proposed adaptive

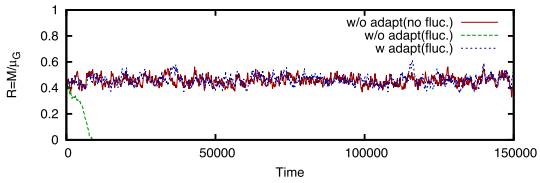

Fig. 8 Time evolution of weakness index R when traffic rate gradually increases (i.e., $\lambda_{\mathrm{INIT}} = 0.05$, $p_+ = 0.7$ and $\kappa = 0.001$); the red and blue lines are almost the same time evolution of R.

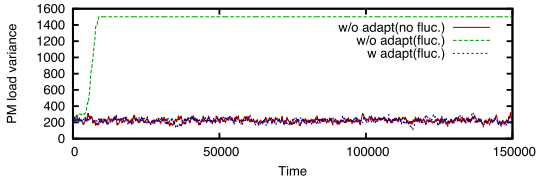

Fig. 9 Time evolution of PM load variance Var[$\boldsymbol{\rho}$] when traffic rate gradually increases (i.e., $\lambda_{\mathrm{INIT}} = 0.05$, $p_+ = 0.7$ and $\kappa = 0.001$); the red and blue lines plot almost the same time evolution of Var[$\boldsymbol{\rho}$].

function against traffic changes. **Figures 10** and **11** show the probability distribution of weakness index R and PM load variance Var[$\boldsymbol{\rho}$] when traffic rate gradually or rapidly increases (i.e., $p_+ = 0.7$ and $\kappa = 0.001$ or 0.1), and gradually or rapidly decreases (i.e., $p_+ = 0.3$ and $\kappa = 0.001$ or 0.1). These results confirm that the proposed adaptive function ensures control robustness against changing traffic. The proposed adaptive function can almost retain the probability distribution of weakness index R and PM load variance Var[$\boldsymbol{\rho}$] against changing traffic regardless of initial control parameter λ_{INIT}, so we can confirm its effectiveness. While there is a little gap in the probability distribution for our ADCM with the proposed adaptive function when traffic rates rapidly increase, we believe that this will not be a problem in practical use. The small gap is explained by the equalization delay of the diffusion equation in the proposed adaptive function. Such delay allows each node to use control parameter λ_k that does not really suit the actual environment (i.e., μ_G). **Figure 12** shows time evolution of the variance coefficient of subsystem k's control parameters λ_k when traffic rate increases. According to this result, the variance coefficient for $\kappa = 0.1$ is larger than that for $\kappa = 0.001$. This result implies that the equalization achieved by the diffusion equation is slightly slow if the traffic increase is rapid.

Figure 13 shows the average of weakness index R, μ_R, for different diameters of the adjacency relationship graph of PMs when traffic rate increases (i.e., $p_+ = 0.7$). According to this result, regardless the diameter, the proposed adaptive function retains μ_R against changing traffic, so the diameter hardly alters the effectiveness of the proposed adaptive function. This result implies that the proposed adaptive function would keep its effectiveness in large-scale DCNs that consist of many PMs, having a large diameter.

Finally, we confirm the effectiveness of the proposed adaptive function when using traffic setting (b). This traffic setting has a larger variance in traffic rates compared with traffic setting (a). **Figures 14** and **15** show the probability distribution of weakness index R and PM load variance Var[$\boldsymbol{\rho}$] when traffic rate gradually or rapidly increases (i.e., $p_+ = 0.7$ and $\kappa = 0.001$ or 0.1), and gradually or rapidly decreases (i.e., $p_+ = 0.3$ and $\kappa = 0.001$ or

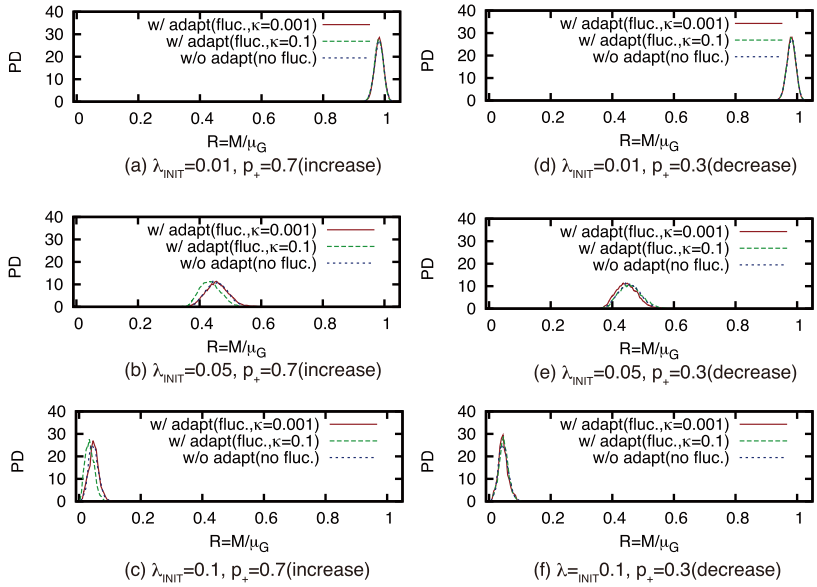

Fig. 10 Probability distribution (PD) of weakness index R when traffic rate increases ($p_+ = 0.7$) or decreases ($p_+ = 0.3$).

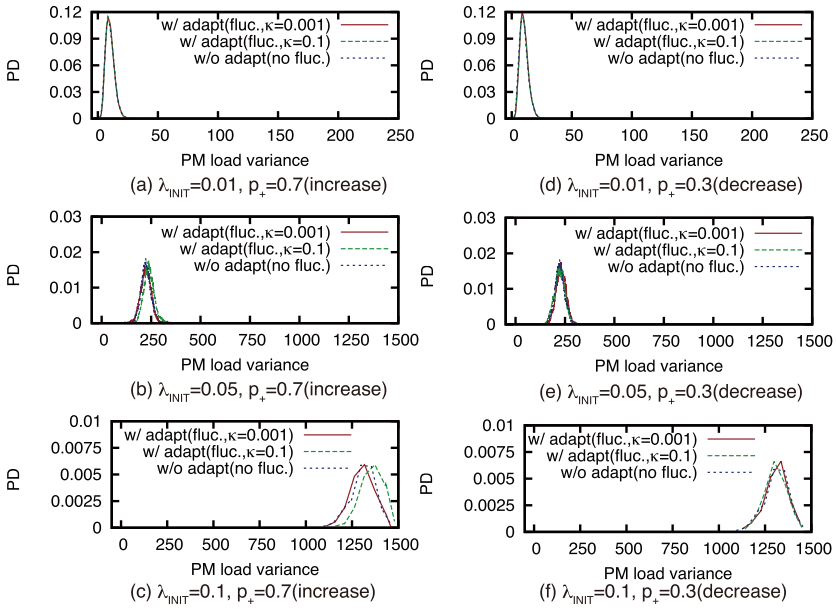

Fig. 11 Probability distribution (PD) of PM load variance Var[ρ] when traffic rate increases ($p_+ = 0.7$) or decreases (i.e., $p_+ = 0.3$).

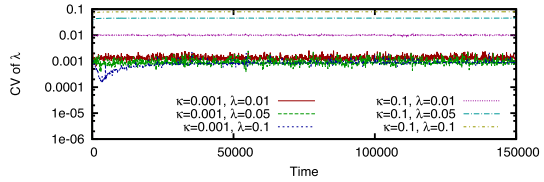

Fig. 12 Time evolution of variance coefficient of subsystem k's control parameters λ_k when traffic rate increases (i.e., $p_+ = 0.7$).

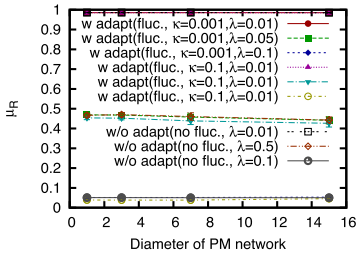

Fig. 13 Average of weakness index R for different diameters in the neighbor relationship graph when traffic rate increases (i.e., $p_+ = 0.7$).

0.1). These results also confirm the effectiveness of the proposed adaptive function similar to Figs. 14 and 15. Namely, the proposed adaptive function can almost retain the probability distribution of weakness index R and PM load variance Var[ρ] against changing traffic for $\lambda_{\mathrm{INIT}} = 0.01$ and 0.05, so we can confirm its effectiveness to retain weak and medium control strengths of the MCMC-based ADCM. For $\lambda_{\mathrm{INIT}} = 0.1$ and $p_+ = 0.7$, the proposed adaptive function needs more than $N_D = 10$ to keep the probability distribution unchanged. Hence, we should set N_D to a sufficiently large value if traffic rate variance is large.

6. Related Work

We first describe the difference between the MCMC-based

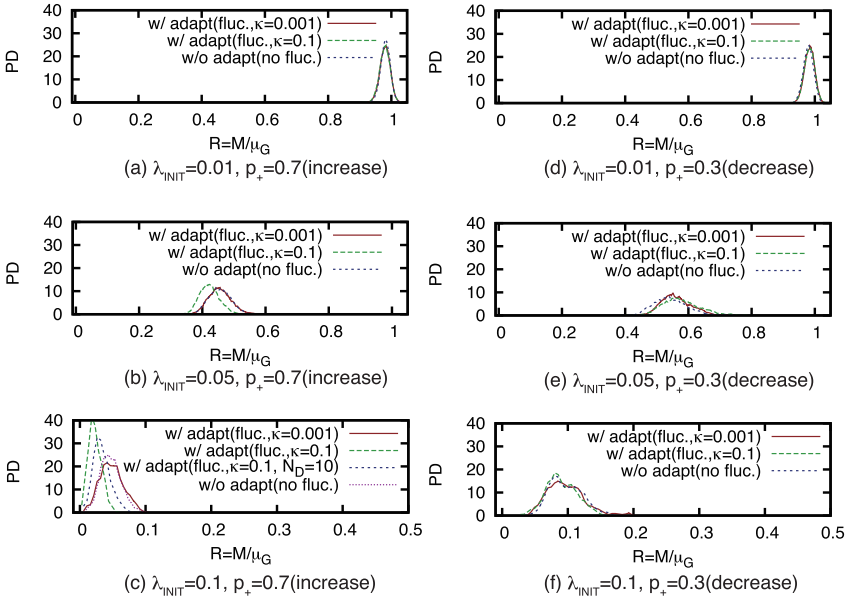

Fig. 14 Probability distribution (PD) of weakness index R when traffic rate increases ($p_+ = 0.7$) or decreases ($p_+ = 0.3$) under traffic setting (b).

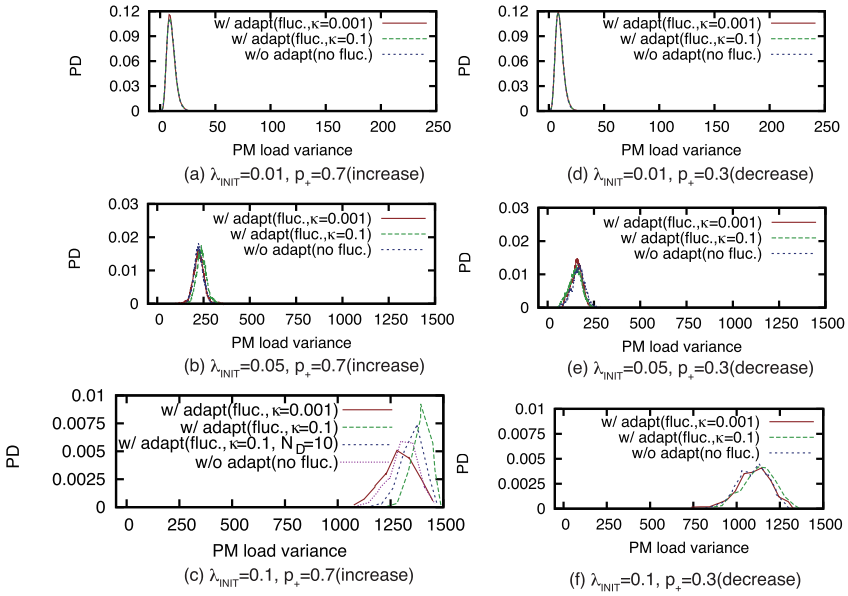

Fig. 15 Probability distribution (PD) of PM load variance Var[ρ] when traffic rate increases ($p_+ = 0.7$) or decreases (i.e., $p_+ = 0.3$) under traffic setting (b).

ADCM [7] and ADCMs [3], [6] inspired by the physical universe. ADCMs [3], [6] can make a spatially structure used for network clustering and layering. However, they do not address other engineering problems (e.g., resource allocation) formulated by a system performance variable. The MCMC-based ADCM remedies that omission. Since the proposed adaptive function works in the MCMC-based ADCM, it contributes to the solutions of such important engineering problems.

Many centralized control mechanism were proposed in Refs. [11], [14], [15], [16]. However, these mechanisms must gather information from the whole system, so their control times increase as the size of the system increases. On the contrary, the MCMC-based ADCM and the proposed adaptive function only use local information around each node. Hence, the MCMC-based ADCM with the proposed adaptive function offers quick response also in large-scale and wide-area systems.

7. Conclusion and Future Work

In this paper, we designed an autonomous decentralized adaptive function to ensure the robustness of our ADCM [7] against changing environment. We first derived the condition to adapt to change in the external environment on the basis of the global property of our ADCM [7]. Following the condition, we designed an autonomous decentralized adaptive function that ensures the robustness of our ADCM, and applied it to a VM placement problem in a DCN. Simulation experiments confirmed that the adaptive function effectively deals with several scenarios with changing environment. For the scenarios, we generated several patterns of traffic changes. We showed that the adaptive function can ensure the robustness regardless of traffic changes.

As future work, we are planning to clarify the relation between the frequency of environment changes and the optimum setting of

the control parameters (i.e., k_D and N_D) of the proposed adaptive function, and create a policy for setting control parameter K considering several situations. We intend to implement our ADCM with the proposed adaptive function in an actual environment, and investigate its effectiveness in realistic tests. Specifically, we will demonstrate the effectiveness of the proposed adaptive function in an actual system with many VMs and PMs.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number 15K15985.

References

[1] Ballmer, S.: Worldwide Partner Conference 2013 Keynote, available from ⟨http://www.microsoft.com/en-us/news/speeches/2013/07-08wpcballmer.aspx⟩ (accessed 2016-01-15).
[2] Google: Data Centers, available from ⟨http://www.google.com/about/datacenters/inside/index.html⟩ (accessed 2016-01-15).
[3] Takano, C., Masaki, A., Murata, M. and Imase, M.: Proposal for Autonomous Decentralized Structure Formation Based on Local Interaction and Back-Diffusion Potential, *IEICE Trans. Communications* (*Special Section on Frontiers of Information Network Science*), Vol.E95-B, No.5, pp.1529–1538 (2012).
[4] Jiang, J.W., Lan, T., Ha, S., Chen, M. and Chiang, M.: Joint VM Placement and Routing for Data Center Traffic Engineering, *Proc. IEEE INFOCOM 2012*, pp.2876–2880 (2012).
[5] Chen, M., Liew, S.C., Shao, Z. and Kai, C.: Markov Approximation for Combinatorial Network Optimization, *Proc. IEEE INFOCOM 2010*, pp.1–9, IEEE (2010).
[6] Masaki, A.: Using a Renormalization Group to Create Ideal Hierarchical Network Architecture with Time Scale Dependency, *IEICE Trans. Communications* (*Special Section on Frontiers of Information Network Science*), Vol.E95-B, No.5, pp.1488–1500 (2012).
[7] Sakumoto, Y., Aida, M. and Shimonishi, H.: Autonomous Decentralized Control for Indirectly Controlling System Performance Variable of Large-Scale and Wide-Area Networks, *IEICE Trans. Communications*, Vol.E98-B, No.11, pp.2248–2258 (2015).
[8] Hastings, W.: Monte Carlo Sampling Methods Using Markov Chains and Their Applications, *Biometrika*, Vol.57, No.1, pp.97–109 (1970).
[9] Sakumoto, Y., Aida, M. and Shimonishi, H.: An Autonomous Decentralized Adaptive Function for Retaining Control Strength in Large-Scale and Wide-Area System, *Proc. IEEE GLOBECOM 2014*, pp.1958–1964 (2014).
[10] Gilks, W.R., Richardson, S. and Spiegelhalter, D.J.: *Markov Chain Monte Carlo in Practice*, CRC press (1996).
[11] Meng, X., Pappas, V. and Zhang, L.: Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement, *Proc. IEEE INFOCOM 2010*, pp.1154–1162 (2010).
[12] Kandula, S., Sengupta, S., Greenberg, A., Patel, P. and Chaiken, R.: The Nature of Data Center Traffic: Measurements & Analysis, *Proc. ACM IMC 2009*, pp.202–208 (2009).
[13] Benson, T., Akella, A. and Maltz, A.D.: Network Traffic Characteristics of Data Centers in the Wild, *Proc. ACM IMC 2010*, pp.267–280 (2010).
[14] Hyser, C., Mckee, B., Gardner, R. and Watson, B.J.: Autonomic Virtual Machine Placement in the Data Center, *Technical Report of Hewlett Packard Laboratories* (*HPL-2007-189*) (2007).
[15] Tarighi, M., Motamedi, S.A. and Sharifian, S.: A New Model for Virtual Machine Migration in Virtualized Cluster Server Based on Fuzzy Decision Making, *Journal of Telecommunications*, Vol.1, No.1, pp.901–907 (2010).
[16] Xu, H. and Li, B.: Anchor: A Versatile and Efficient Framework for Resource Management in the Cloud, *IEEE Trans. Parallel and Distributed Systems*, Vol.24, No.6, pp.1066–1076 (2013).

Yusuke Sakumoto received M.E. and Ph.D. degrees in the Information and Computer Sciences from Osaka University in 2008 and 2010, respectively. He is currently an assistant professor at Graduate School of System Design, Tokyo Metropolitan University, Japan. His research work is in the area of autonomous decentralized control for large-scale and complex systems. He is a member of IEEE, IPSJ and IEICE.

Masaki Aida received B.S. degree in Physics and M.S. degree in Atomic Physics from St. Paul's University, Tokyo, Japan, in 1987 and 1989, respectively, and received Ph.D. in Telecommunications Engineering from the University of Tokyo, Japan, in 1999. After joining NTT Laboratories in April 1989, he has been engaged in research on traffic issues in computer communication networks. From April 2005 to March 2007, he was an Associate Professor at the Faculty of System Design, Tokyo Metropolitan University. He has been a Professor of the Graduate School of System Design, Tokyo Metropolitan University since April 2007. Prof. Aida is a member of IEEE, IEICE, and the Operations Research Society of Japan.

Hideyuki Shimonishi received M.E. and Ph.D. degrees from the Graduate School of Engineering Science, Osaka University, Osaka, Japan, in 1996 and 2002. He joined NEC Corporation in 1996 and has been engaged in research on traffic management in high-speed networks, switch and router architectures, and traffic control protocols. As a visiting scholar in the Computer Science Department at the University of California at Los Angeles, he studied next-generation transport protocols. He now works in Knowledge Discovery Research Laboratories at NEC Corp., engaged in researches on networking technologies including SDN, OpenFlow and NFV for carrier, data center and enterprise networks. He is a member of IEICE.