Designing Adaptive and Evolvable Software-defined Infrastructure Inspired by Biological Behaviors

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2019

Koki INOUE

List of publication

Journal papers

1. Koki Inoue, Shin'ichi Arakawa and Masayuki Murata, "A biological approach to physical topology design for plasticity in optical networks," *Optical Switching and Networking*, vol. 25, pp. 124–132, July 2017.

2. Koki Inoue, Shin'ichi Arakawa, Satoshi Imai, Toru Katagiri and Masayuki Murata, "Noise-induced VNE method for software-defined infrastructure with uncertain delay behaviors," *Computer Networks*, vol. 145, pp. 118–127, November 2018.

Refereed Conference Papers

1. Koki Inoue, Shin'ichi Arakawa and Masayuki Murata, "Achieving plasticity in WDM networks: Application of biological evolutionary model to network design," in *Proceedings of IEEE GLOBECOM 2015*, December 2015.

2. Koki Inoue, Shin'ichi Arakawa, Satoshi Imai, Toru Katagiri and Masayuki Murata, "Adaptive VNE method based on Yuragi principle for software defined infrastructure," in *Proceedings of IEEE HPSR 2016*, pp. 188–193, June 2016.

Non-Refereed Technical Papers

1. Koki Inoue, Shin'ichi Arakawa and Masayuki Murata, "A design method of WDM networks based on biological evolution model," *Technical Report of IEICE (PN2014-8)*, vol. 114, no. 109, pp. 41–46, June 2014.

2. Koki Inoue, Shin'ichi Arakawa, Satoshi Imai, Toru Katagiri, Motoyoshi Sekiya and Masayuki Murata, "Yuragi-based approach with delay profile for virtual network embedding in software defined infrastructure," *Technical Report of IEICE (IN2015-148)*, vol. 115, no. 484, pp. 235–240, March 2016.

3. Koki Inoue, Shin'ichi Arakawa and Masayuki Murata, "An evolvable network resource planning for adaptive virtual network control in software defined infrastructure," *Technical Report of IEICE (NS2017-160)*, vol. 117, no. 385, pp. 93–98, January 2018.

Preface

Software-defined infrastructure (SDI) is a promising framework to enable flexible and rapid deployment of new services on information networks by providing virtualized infrastructure to customers by slicing computing resources and network resources. That contributes shortening time-to-market of customers' services. Also, for network service providers, it is expected that there will be a merit of both capital and operating expense (CAPEX and OPEX) reduction by deploying a SDI framework.

However, several problems remain in virtual network control and physical network design towards enjoying the SDI framework. First, a resource control which can immediately response to demand fluctuations is required. The softwarized user interface in SDI enables responding to resource demands from various customers in short-term. Second, related to the above problem, a resource controller is required to work without a full knowledge of the whole network situation. Conventional approaches intend to solve a certain optimization problem, where a centralized network controller needs to collect precise information before calculating an optimized solution. However, this process will be difficult for a larger number of multiplexed virtual networks, and that disables the on-demand network operation. Third, drastic and unexpected demand fluctuations should be considered. With short-term and customized requests, demand fluctuations become difficult to predict. Then adaptation by a softwarized VNE control becomes more important in SDI, but an improper physical resource arrangement may course degradation of the performance of VNE controls.

The target of this thesis is to construct a virtual network embedding (VNE) control method for solving the first and the second problems, and physical resource planning for the third problem. In

this thesis, we present an adaptive VNE method that works with only a little information for large, complicated, and uncertain SDI frameworks. The method is based on a biological attractor selection model, which our research group has been adopting for virtual network topology control in optical networks. As a preliminary study for physical resource design, we examine a biological approach to physical topology design for plasticity in optical networks. The design considers plasticity just as better link utilization rate obtained through an evolutionary process. However, it does not consider a diverse set of potential virtual network topologies, so evolvability will not be obtained. Thus for the SDI framework, we consider physical resource design to increase a diversity of solutions reached by a VNE control. It is expected that providing candidates of more various VNE solutions will enhance adaptability of VNE control against various future environmental fluctuations.

We first propose a physical resource design method for optical networks, e.g., wavelength division multiplexing (WDM) networks, as a prior inspection for physical resource design method in SDI frameworks. We propose a design method for adding transceivers to IP routers in IP-over-WDM networks. The method defines correspondence between an evolution model and a WDM network, and simulates a process of biological evolution (i.e., mutation and selection of gene regulatory networks through generations) where transceivers arrangement is reflected by modifying the gene regulatory network. Then it measures performance of the VNT control method (i.e., average link utilization rate). Evaluation results show that our method accommodates more patterns of traffic fluctuation with lower link utilization than ad-hoc design methods do. Thus we confirm our approach is promising for physical resource design.

Second, we present a VNE method that works with limited information for large, complicated, and uncertain SDI frameworks. To achieve this, the proposed method applies the biological "Yuragi" principle. Therefore, we develop a Yuragi-based VNE method that deals with node attributes, has the generality to set a performance objective, and runs in multi-slice environments. We examine a complicated model of end-to-end delay and show that the proposed method can sustain its adaptability under various types of delay conditions. Simulation results show that the Yuragi-based method can decrease VN migrations by about 29% relative to a heuristic method to adapt to fluctuations in resource requirements.

Finally, we propose an SDI resource design strategy that increases diversity of VNE solutions,

which is derived by a variation of regulatory matrices under demand fluctuations. Our design strategy for the SDI system aims to achieve adaptability in the face of unpredictable environmental changes by increasing the diversity of considered VNE states. As a successful biological model, we consider the evolution of populations of organisms to better fit changing environments. The strategy imitates the evolvability of biological populations, adopting an evolutionary model that treats each VNE solution characterized as a biological phenotype. We use the proposed strategy to construct a method for reinforcing the computational capacity of physical nodes, and conduct experiments by computer simulation. Our results show that the probability of convergence with VNE control is improved by using the proposed physical-resource reinforcement, achieving a gain of up to 19% relative to a basis reinforcement method which optimizes an expected performance under predicted demand fluctuations.

Acknowledgments

I would like to express my sincere appreciation to everyone who supported me in various ways throughout my Ph.D. This thesis could not have been accomplished without their assistance.

First of all, I express my grate gratitude to my supervisor, Professor Masayuki Murata of Graduate School of Information Science and Technology, Osaka University, for his insightful suggestions and valuable discussions. He brought me to an attractive research field, and made my research life fruitful.

I am heartily grateful to the members of my thesis committee, Professor Takashi Watanabe, Professor Toru Hasegawa, and Professor Teruo Higashino of Graduate School of Information Science and Technology, Osaka University, and Professor Morito Matsuoka of Cyber Media Center, Osaka University, for their multilateral reviews and perceptive comments.

Furthermore, I would like to owe my special thanks to Associate Professor Shin'ichi Arakawa of Graduate School of Information Science and Technology, Osaka University, for his continuous support and encouragement. He taught me the fun of thinking about and solving problems.

Also, I would like to express my sincere appreciation to Dr. Motoyoshi Sekiya, Dr. Toru Katagiri and Dr. Satoshi Imai of Fujitsu Laboratories Ltd. for their helpful comments and fruitful discussions.

Moreover, I am deeply grateful to Assistant Professor Yuichi Ohsita, Specially Appointed Assistant Professor Tatsuya Otoshi of Graduate School of Information Science and Technology, Osaka University, Assistant Professor Daichi Kominami of Graduate School of Economics, Osaka University, and Assistant Professor Naomi Kuze of Graduate School of Engineering Science, Osaka university, for their valuable comments on my study.

I am really thankful to all of past and present colleagues, friends, and secretaries of the Advanced Network Architecture Research Laboratory, Graduate School of Information Science and Technology, Osaka University.

I express my special appreciation to Yoshimura Foundation for offering me a scholarship support.

Finally, I express my thanks to my family for invaluable supports throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Information networks are faced with new emerging services, such as mobile services, cloud computing services, and social services. Such services are now part of the social infrastructures and indispensable in people's lives. In the coming future, it is also anticipated that wider variety of services and applications utilizing network infrastructures are produced. For example, many conceptions are under development for various services such as smart grid [1], vehicular communication systems [2], Ultra-high-definition video delivery [3], telemedicine and health-care [4,5], augmented reality (AR) applications with edge computing [6], etc. Requirements and priorities for a network are different among the services, e.g., some services seek low latency while others requires a large capacity of data transportation, or many number of connections must be accommodated even with low processing power. It is also required within the rapidly changing society and economy that a service should be started with minimum implementation, scalable deployment should be accepted and the time-to-market should be shorten. Therefore, it is required that the network infrastructure be provided flexibly and quickly. Software-defined infrastructure (SDI) [7, 8] is a promising framework to enable flexible and rapid deployment of new services on information networks. An SDI framework, which is realized by orchestrating software-defined computing, software-defined storage, and software-defined network, provides virtualized infrastructure to customers with any

required capacities by slicing computing resources, storage and network resources.

A key to leveraging an SDI framework is network virtualization technologies and orchestration of them. Network virtualization technologies are in the research and development phase. In recent years, software-defined networking (SDN) and network-function virtualization (NFV) technologies have been expected to replace the conventional network management systems, and standardization of SDN/NFV technologies is being promoted. SDN/NFV technologies enable programmable and automated network control, while conventional systems require the network operator to configure various kinds of network devices [9–15]. The customers can order virtualized computational resources and network resources to their network service providers by making customized requests via a certain softwarized API, e.g., a graphical user interface (GUI) application on the Web. Then, a sliced virtual network is immediately assigned to the requesting customer by automated resource control. That contributes shortening time-to-market of customers' services. Also, for network service providers, it is expected that there will be a merit of both capital and operating expense (CAPEX and OPEX) reduction by deploying a SDI framework. CAPEX will be reduced by sharing infrastructure among different customers and services, and by flexibly scaling the amount of resources provided in response to demand fluctuation. OPEX will be reduced by replacing manual operations with automated operations, that also leads cutting of human error.

However, several problems remain in virtual network control and physical network design towards enjoying the SDI framework. First, a resource control which can immediately response to demand fluctuations is required. The softwarized user interface in SDI enables responding to resource demands from various customers in short-term. Second, and this is related to the above problem, a resource controller is required to work without a full knowledge of the whole network situation. Although virtual network embedding (VNE) problem has been addressed to obtain a proper assignment of resources satisfying demands [16–22], most of existing methods dissatisfy those requirements. Their approaches intend to solve a certain optimization problem, where a centralized network controller needs to collect precise information before calculating an optimized solution. However, this process will be difficult for a larger number of multiplexed virtual networks, and that disables the on-demand network operation. Third, drastic and unexpected demand fluctuations should be considered. With short-term and customized requests, demand fluctuations

become difficult to predict. Then adaptation by a softwarized VNE control becomes more important in SDI, but an improper physical resource arrangement may course degradation of the performance of VNE controls.

The target of this thesis is to construct a VNE control method for solving the first and the second problems, and physical resource planning for the third problem.

1.1.1 Virtual network control

The VNE problem is a placement problem in which virtual resources are to be allocated to the physical network with optimization of some performance objectives. In the VNE problem, service demands from customers are translated to virtual network requests. A virtual network consists of virtual nodes and virtual links. Each of the virtual nodes is hosted on a physical node as a form of virtual machine. Then, the virtual nodes are connected through a path of physical nodes, forming virtual links. The VNE problem is divided into two sub-problems: virtual node mapping and virtual link mapping. Virtual node mapping decides the location of the physical node for each virtual node. Note that each virtual node must be allocated to a physical node supporting its "node attribute." The node attribute allows classification of nodes in ways defined by the supported operating system (OS), storage type, or node use (e.g., computing, storage, or packet switching). Virtual link mapping decides the path on the physical network for virtual links between virtual nodes.

In [19–22], a centralized calculation was assumed to solve virtual node mapping and virtual link mapping. That is, a centralized component gathers traffic information and resource utilization for each virtual network and identifies the current situation (i.e., the current traffic demand and/or the current service demand) of the networks. Then, the component solves the optimization problem that optimizes some metric, such as maximizing revenue or minimizing resource utilization. However, when the network size gets larger and the number of multiplexed virtual networks increases, the identification of the current situation becomes complicated by the enormous amount of network information. As the network operators want to know the current situation more accurately and precisely, more information is necessary to collect. This will lead to increased used of link bandwidth, increased delay, and a bottleneck on network scalability [15]. Note that the calculation

time to obtain a solution of the optimization problem also gets larger. However, the calculation time is not crucial because it may be relaxed by some heuristic algorithms with some sacrifice of the quality of the solution. Our concern in adopting the centralized approach is the overhead of collecting information, and this overhead gets larger as the size of the infrastructure and number of virtual network requests increase. Moreover, the environments surrounding the Internet today are continuously changing, thus, adaptive control of VNE is required to handle uncertain changes in the environments. Although precise modeling of the end-to-end delay in SDI environment is difficult, it would be required to suppress the maximum delay in order to guarantee a specific quality of experience (QoE) for applications on virtual networks. There are several models of network delay proposed, which are constructed generally and disregard the data contexts of packets [23]. However, the processing delay on servers depends on multiple factors, including server specification; CPU and memory utilization (on virtual machines); and details of processing, which depend on the context of the data.

In this thesis, we present an adaptive VNE method that works with only a little information for large, complicated, and uncertain SDI frameworks. A process of the VNE method is executed for each virtual network request. Different from optimizing problems and related heuristics, our VNE method can avoid the necessity of collecting detailed information about the entire network. The process for a virtual network request needs only enough information for performance objective and does not need any information related to other virtual network requests.

1.1.2 Physical resource planning

Virtual network embedding (VNE) control is expected to allow properly configuring virtual resource allocation in response to environmental fluctuations, such as changes in virtual resource demands, but a VNE control may not result in good virtual network performance. Such failure is caused mainly by two factors. The first factor is the VNE algorithm itself, and many VNE algorithms have been studied with the aim of achieving better allocation of virtual resources [18–20, 22, 24–27]. The second factor is related to the physical resource design. When resource utilization levels become high, processing delays and data transfer delays will increase, resulting in worse performance of

services running on a virtual network. Despite the extensive research on VNE algorithms, the design of physical infrastructure for SDN/NFV applications has been scarcely considered to date.

Although physical network resource designs have been considered in traditional communication systems, such systems aim to have adequate capacity for future states as predicted from long-term traffic observation. Indeed, physical network designs have been studied to optimize performance on the basis of current demand or a predicted future demand. For example, in IP-optical networks, ref. [28] describes the design of an optical-cross-connect topology in which the number of distinct wavelengths is minimized by knowledge of the optical path demands. Reference [29] describes a design for an optical layer network with the capacity to accommodate a predefined IP-layer topology, decided on the basis of predicted future traffic and possible failure scenarios. However, such conventional methods of designing for capacity are unsuited to SDI frameworks. A fundamental difference between capacity planning in conventional frameworks and in the SDI framework is the time granularity of changes in demand. That is, with SDI, the resource demands from various users may change over short periods. This is inherent to SDI frameworks, where virtual network configurations are executed by a softwarized control instead of by conventional manual operation. Because of this, adaptation by a softwarized VNE control becomes more important in SDI frameworks for achieving rapid provisioning of resources to meet fluctuating demands, and physical resource design is an important factor in the adaptability of VNE control. As mentioned above, algorithms for finding better VNE solutions under given resource constraints have been considered, but a strategy for choosing a physical network design that promotes VNE adaptability has not been discussed.

In contrast with virtual resource allocation, which is nearly instant, installing physical resources in an SDI framework takes considerable time and manual work. It is thus not practical to adjust the physical resources in response to every demand fluctuation. Physical resource planning requires that short-term fluctuations be managed by a dynamic VNE control. Note that drastic fluctuations should be expected to occur in the future for SDI because user requests frequently arrive through user-friendly interfaces (e.g., GUI) and applications are customized to be suitable for their intended purpose. A promising way to enhance the ability of the VNE control to adapt to unexpected fluctuations is to reinforce the physical resources so that the VNE control can draw on this more robust infrastructure, which makes a higher number of VNE solutions feasible.

Therefore, we consider which physical resource designs will increase the diversity of feasible solutions considered by a VNE control. It is expected that providing more varied candidates for VNE solutions will enhance the robustness of VNE control against environmental fluctuations (i.e., will enhance its adaptability). Even in situations where it is difficult to predict demand changes, our design strategy aims to deploy physical resources such that the VNE control can accommodate various fluctuations in future demand.

1.2 Outline of thesis

A Biological Approach to Physical Topology Design for Plasticity in Optical Networks [30–32]

We first propose a physical resource design method for optical networks, e.g., wavelength division multiplexing (WDM) networks, as a prior inspection for physical resource design method in SDI frameworks.

We have an idea of applying biological evolution to physical resource design in SDI frameworks for adaptability against drastic demand fluctuations. The design approach intend to simulate a biological characteristic that biological evolution allows species to survive unexpected environmental changes with adaptively expressing phenotypes suitable for each novel environment. While adaptive virtual resource control is required to enjoy such the physical resource design, there is no specific adaptive VNE methods in SDI frameworks. As for in WDM networks, our research group has previously developed a virtual network topology (VNT) control method that works with only a little information [33]. The VNT control method has been experimented through actual implementation, and adaptability to traffic changes is demonstrated. We construct a physical resource design based on a biological evolutionary model with presupposing the adaptive VNT control, and confirm whether our approach can be promising for physical resource design.

One important characteristic of biological evolution is *plasticity*, which describes the changeability against environmental changes [34]. In Ref. [34], the authors develop a gene expression

dynamics model to explain how organisms can obtain both short-term (on the order of hours) robustness and long-term (on the order of days to years) plasticity. In an optical network capable of plasticity, it is expected that adaptive VNT control can enjoy plasticity of the physical infrastructure, and so network performance can avoid being degraded under various patterns of future traffic fluctuation, even unknown patterns. We propose a design method for adding transceivers to IP routers in IP-over-WDM networks. The method defines correspondence between an evolution model and a WDM network, and simulates a process of biological evolution (i.e., mutation and selection of gene regulatory networks through generations) where transceivers arrangement is reflected by modifying the gene regulatory network. Then it measures performance of the VNT control method (i.e., average link utilization rate). Evaluation results show that our method accommodates more patterns of traffic fluctuation with lower link utilization than ad-hoc design methods do.

Noise-induced VNE Method for Software-defined Infrastructure with Uncertain Delay Behaviors [27, 35, 36]

Second, we present a VNE method that works with limited information for large, complicated, and uncertain SDI frameworks. To achieve this, the proposed method applies the biological "Yuragi" principle. Yuragi is a Japanese word whose English translation is a small perturbation, both externally and internally generated, to the system. Yuragi is a mechanism that provides adaptability to organisms and is often expressed as an attractor selection model. Our research group has developed a virtual network control based on attractor selection for optical networks. Our results showed that our control mechanism has high adaptability to environmental fluctuations with restricted information. Unlike a virtual network on an optical network, a virtual network on an SDI framework has to consider various matters such as node attribute, computational performance of servers, and VN multiplexing. Therefore, we develop a Yuragi-based VNE method that deals with node attributes, has the generality to set a performance objective, and runs in multi-slice environments. One process of the method is executed for each VN slice, and each process needs information about only its own VN requests. Each of the processes behaves so as to improve its own performance function,

considering other VNs as a part of an external perturbation (i.e., Yuragi). We examine a complicated model of end-to-end delay and show that the proposed method can sustain its adaptability under various types of delay conditions. Simulation results show that the Yuragi-based method can decrease VN migrations by about 29% relative to a heuristic method to adapt to fluctuations in resource requirements.

A network resource planning for evolvability in software-defined infrastructure [37]

Finally, we propose an SDI resource design strategy that increases diversity of VNE solutions, which is derived by a variation of regulatory matrices under demand fluctuations. Our design strategy for the SDI system aims to achieve adaptability in the face of unpredictable environmental changes by increasing the diversity of considered VNE states. As a successful biological model, we consider the evolution of populations of organisms to better fit changing environments. One key to obtaining evolutionary adaptability is to increase genotypic evolvability (i.e., the phenotypic diversity that can arise from a genetic distribution) [34, 38]. Even when the environment drastically changes, genotypic evolvability lets the system produce phenotypes that are much different from the previously dominant phenotype and ultimately settle on a phenotype that is suitable for the changed environment. In this thesis, we propose an SDI resource design strategy that increases VNE solution diversity, which originates from control system variation under demand fluctuations. The strategy imitates the evolvability of biological populations, adopting an evolutionary model that treats each VNE solution characterized a biological phenotype.

As a preliminary work, we considered a method based on biological evolution that can increase the number of transceivers of IP routers in a WDM network. However, the method given in our earlier study, which incorporates the state of resource reinforcement into the gene regulatory network, is specific to the combination of a virtual network control method and increases in the number of IP transceivers in a WDM network. In addition, that method does not consider a diverse set of potential virtual networks, so evolvability is not obtained. In the SDI framework, demands of virtual resources become more complicated including node computing servers and network bandwidths.

Against fluctuations of such the demands, adaptation for a variety of situations by a virtual network control is prior to obtaining better performances for several estimated situations. We instead focus on improving evolvability (in the form of phenotype diversity caused by genetic mutation) and thereby contributing to improvement of environmental adaptability, analogous with biological evolution. For this, we develop an evolvability index to characterize the diversity of a VNE solution set in an SDI framework. This index is independent of the type of resource to be reinforced (e.g., node resources and link resources are treated the same), thus constructing a more general method of resource design.

We use the proposed strategy to construct a method for reinforcing the computational capacity of physical nodes, and conduct experiments by computer simulation. Our results show that the probability of convergence with VNE control is improved by using the proposed physical-resource reinforcement, achieving a gain of up to 19% relative to a reinforcement method which optimizes an expected performance under predicted demand fluctuations.

Chapter 2

A Biological Approach to Physical Topology Design for Plasticity in Optical Networks

2.1 Introduction

In wavelength division multiplexing (WDM) networks, optical cross connects (OXCs) switch optical signals without optical-electrical-optical (OEO) conversion by using wavelength routing. A wavelength channel, called a lightpath, is established between nodes. Since the upper-layer's traffic, such as IP traffic, can change its nature, much research has examined the construction of a virtual network topology (VNT) on top of a WDM network [39, 40]. A VNT is a logical network composed of lightpaths, and the connectivity among routers can be easily reconfigured by establishing or tearing down lightpaths. When the traffic demand changes and certain performance metrics degrade to the point where they are no longer acceptable, the VNT is changed to a new VNT that exhibits optimal or near-optimal performance under the network environment as it exists at that time.

The environment of the Internet is rapidly changing. With the appearance of new web services such as video streaming and cloud computing, traffic volumes have increased rapidly and fluctuate

drastically. Some VNT control methods have been studied for countering traffic fluctuations, showing good performance on metrics such as keeping link utilization lower by adaptively reconfiguring the VNT in accordance with traffic changes [33, 41]. However, when the traffic volume increases, VNT control methods may fail to find a suitable VNT. That is, there may be no solution that can provide good performance because of a lack of network resources or because of other problems. In such situations, network operators must reinforce the physical network resources. Much consideration has gone into physical network design [28, 42–44]. In Ref. [42], the authors consider designing a physical topology in which logical rings can be established for survivability while minimizing the number of physical links. In Ref. [28], the authors address both physical and logical topology design, and formulate the problem as an integer linear programming problem of minimizing the number of wavelengths used. In Ref. [43], the authors consider a routing and wavelength assignment problem in optical networks with the aim of minimizing the cost over the long term under a restricted budget. In Ref. [44], the authors consider designing a mixed-line-rates network with minimum cost. Most of these works solve optimization problems against predicted traffic demand. However, when the environment changes drastically, it is natural that future traffic demand cannot be estimated accurately. Even if we are able to 'specify' future traffic demand by incorporating environmental uncertainty and use it in the design method, the designed network is specialized to the pre-specified situation, which may lose adaptability against unexpected traffic changes. Therefore, a new design approach that can accommodate various patterns of future traffic in conjunction with the VNT control method is needed.

In order to develop a new design approach, we consider biological evolution, which allows species to survive environmental changes over the long term. One important characteristic of biological evolution is *plasticity*, which describes the changeability against environmental changes [34]. In Ref. [34], the authors develop a gene expression dynamics model to explain how organisms can obtain both short-term (on the order of hours) robustness and long-term (on the order of days to years) plasticity. Following the gene expression dynamics model, we propose a method for designing physical networks and develop a design method for adding transceivers to IP routers in optical networks, e.g., IP-over-WDM networks. The number of transceivers is equal to the degree of virtual links, i.e., lightpaths, connected to the node. Our method determines a set of nodes to

which transceivers should be added in order to give plasticity to the optical network. In an optical network capable of plasticity, it is expected that adaptive VNT control can enjoy plasticity of the physical infrastructure, and so network performance can avoid being degraded under various patterns of future traffic fluctuation, even unknown patterns. Through computational simulation, we confirm that our design method offers plasticity.

A preliminary version of this work has been presented in [31]. In our previous chapter, we have introduced a concept of plasticity in designing optical networks and have compared with a heuristic method on the European optical network (EON) [45]. In the current chapter, we introduce a mixed integer linear programming (MILP) solution for comparison and show the effects of the plasticity on the EON, the US nationwide network (USNET) and the Japan backbone network (JBN).

The rest of this chapter is organized as follows. In Sec. 2.2, we describe the purpose of our research. We then propose a method of optical network design capable of plasticity in Sec. 2.3 and show evaluation results in Sec. 2.4. We finally conclude this chapter and mention future work in Sec. 2.5.

2.2 Adaptive VNT control and physical network design method

When traffic changes drastically, a dynamic VNT control method that can adapt to various changes in traffic is needed. We previously proposed a VNT control method based on attractor selection that exhibits high adaptability to unexpected changes in traffic demand [33]. In this VNT control method, lightpath reconfiguration is driven by the following expression:

$$\frac{dx_i}{dt} = \alpha \cdot f(\mathbf{x}) + \eta, \tag{2.1}$$

where x_i is a variable indicating that a lightpath between the node-pair i is configured when it exceeds a certain threshold. The function $f(\mathbf{x})$ represents deterministic behavior that causes the VNT to converge to one of the equilibrium point, that is, to an attractor. The activity α represents feedback of the network condition. When α is high, the system stays at an attractor that offers good conditions. When the network condition worsens due to traffic fluctuations, α decreases towards

zero until stochastic behavior dominates the system. That is, lightpaths are reconfigured at random in the search for another attractor. After a while, the VNT again converges on a new attractor thereby adapting to the traffic fluctuation.

Although our VNT control method is more successful in terms of obtaining robustness against traffic changes than other existing methods are, it fails to obtain a good VNT when the network resources are insufficient for the increased traffic. This is a fundamental limit that also applies to other methods. The aim of this chapter is therefore to consider a physical network design method for accommodating future unknown traffic demand as much as possible while keeping the adaptability of the attractor-based VNT control method. Figure 2.1 illustrates the relation between VNT reconfiguration and our physical network design method. VNT control reconfigures the VNT over the physical network and adapts to traffic fluctuations. When traffic volume increases, we might not able to find a good VNT because of a shortage of network resources. We then need to add network resources such as physical links, IP routers, optical switches, and transceivers. Since the adaptability of VNT control depends on the underlying physical network, an improperly designed physical network may reduce the ability of VNT control to adapt to traffic fluctuations. We also note that our proposed design method is easily extended to incorporate other network resources. This proposal is applicable to not only our VNT control methods but also other existing dynamic VNT control methods.

2.3 Method for designing optical networks to have plasticity

In this chapter, we apply a biological evolution model that mimics the robustness and plasticity of biological systems. This is introduced in the next subsection. Note that while we understand it is a rather lengthy explanation, it is necessary for readers to understand how biological plasticity can be applied to our case.

2.3.1 Biological model

Organisms adapt to the environment through the evolution of a genetic network. Robustness and plasticity are thought to be basic characteristics in evolutionary biology. Robustness is the capacity

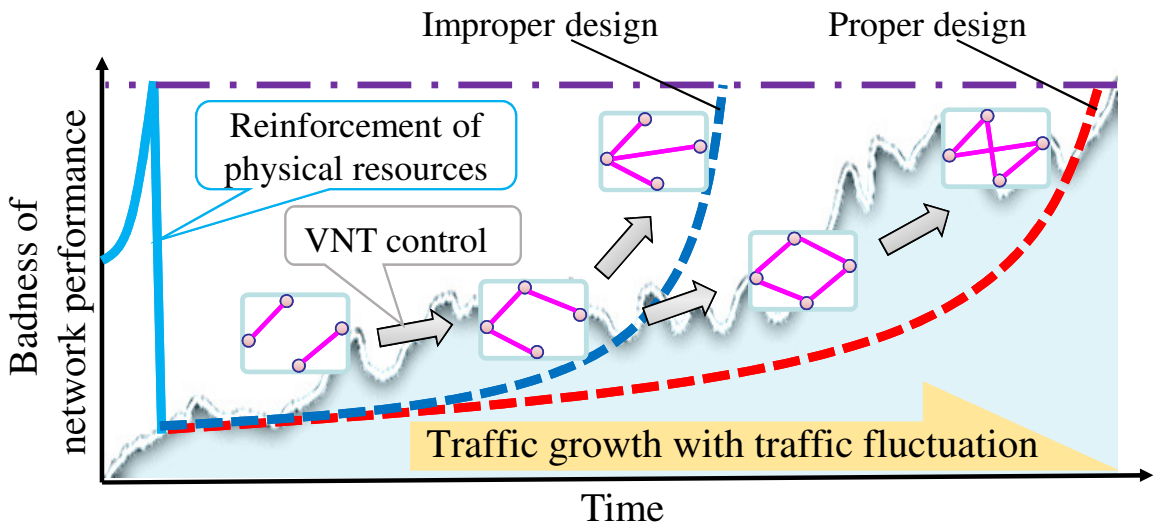


Figure 2.1: VNT control and network design to prepare for traffic growth

of an organism to maintain its own state and function against disturbances. In contrast, plasticity is changeability or flexibility in response to environmental fluctuations [34]. Organisms are able to adapt to new and/or unexperienced environments by greatly changing state as the external environment changes. Plasticity expresses sensitivity to external perturbations, and is an important characteristic for adaptive evolution.

In Ref. [34], the author formulates a model of the evolution process by taking account both biological robustness and plasticity. In the model, an organism optimizes the value of fitness against various kinds of environmental changes by changing gene expression (phenotype), in which the dynamics are governed by activation/inhibition between genes (genotype). The model consists of several elements (Fig. 2.2), each of which is explained below.

gene: There are M genes. Each gene i has its own expression level $x_i(-1 \leq x_i \leq 1)$. When x_i exceeds some threshold θ_i, gene i is expressed. Otherwise, gene i is not expressed.

input gene: k_{inp} genes among the M genes are input genes, and their gene expression levels are given initially and do not change regardless of the gene expression dynamics. Without loss of generality, we regard genes $x_i(1 \leq i \leq k_{inp})$ as the input genes. Changes in the expression levels of these input genes represents a change in the environment.

phenotype: As a result of the gene expression dynamics, the gene expression levels $x_i(k_{inp} < i \leq M)$ converge to some set of values. Note that the input gene expression levels are independent of the gene expression dynamics. Some genes are expressed and others are not expressed, thus forming a pattern of expressed genes. This pattern is called a phenotype. In Fig. 2.2, expressed genes are represented by filled circles and have a phenotypic value of 1, while non-expressed genes are represented by open circles and have a phenotypic value of 0.

genotype: Genes are related to each other. These mutual relations are defined by a gene regulatory network. In Fig. 2.2 each solid arrow represents an activating relation from one gene to another, and each dashed arrow represents an inhibiting relation. $J_{ij}(= \{-1, 0, 1\})$ represents the activation/inhibition relation between gene i and gene j. When $J_{ij} = 1$, gene i receives an activation effect from gene j. When $J_{ij} = -1$, gene i receives an inhibition effect from gene j. When $J_{ij} = 0$, there is no relation between genes i and j. A matrix J with elements J_{ij} is a gene regulatory network and is called a genotype. J determines the gene expression dynamics.

fitness: Fitness represents the adaptability to the present environment or condition of the system, and is calculated by a function $F(phenotype)$. That is, the fitness value is determined by the pattern of gene expression, which is governed by the genotype. From a biological perspective, a typical example of the function F is the number of expressed target genes. We select the target genes for this example from the perspective of a biological context. $F(phenotype)$ becomes the highest when the expression pattern of target genes consists of all 1s. From a network design perspective, introducing target genes is not necessary. We simply use traditional performance metrics to calculate the fitness value. In this chapter, we will use the average link utilization of the VNT for calculating the fitness.

The dynamics of gene expression levels is then described by the following equation,

$$dx_i/dt = \gamma\left\{ f\left(\sum_j^M J_{ij}x_j\right) - x_i \right\} + \sigma\eta_i, \tag{2.2}$$

where the first term represents the deterministic behavior driven by the gene regulatory network J_{ij},

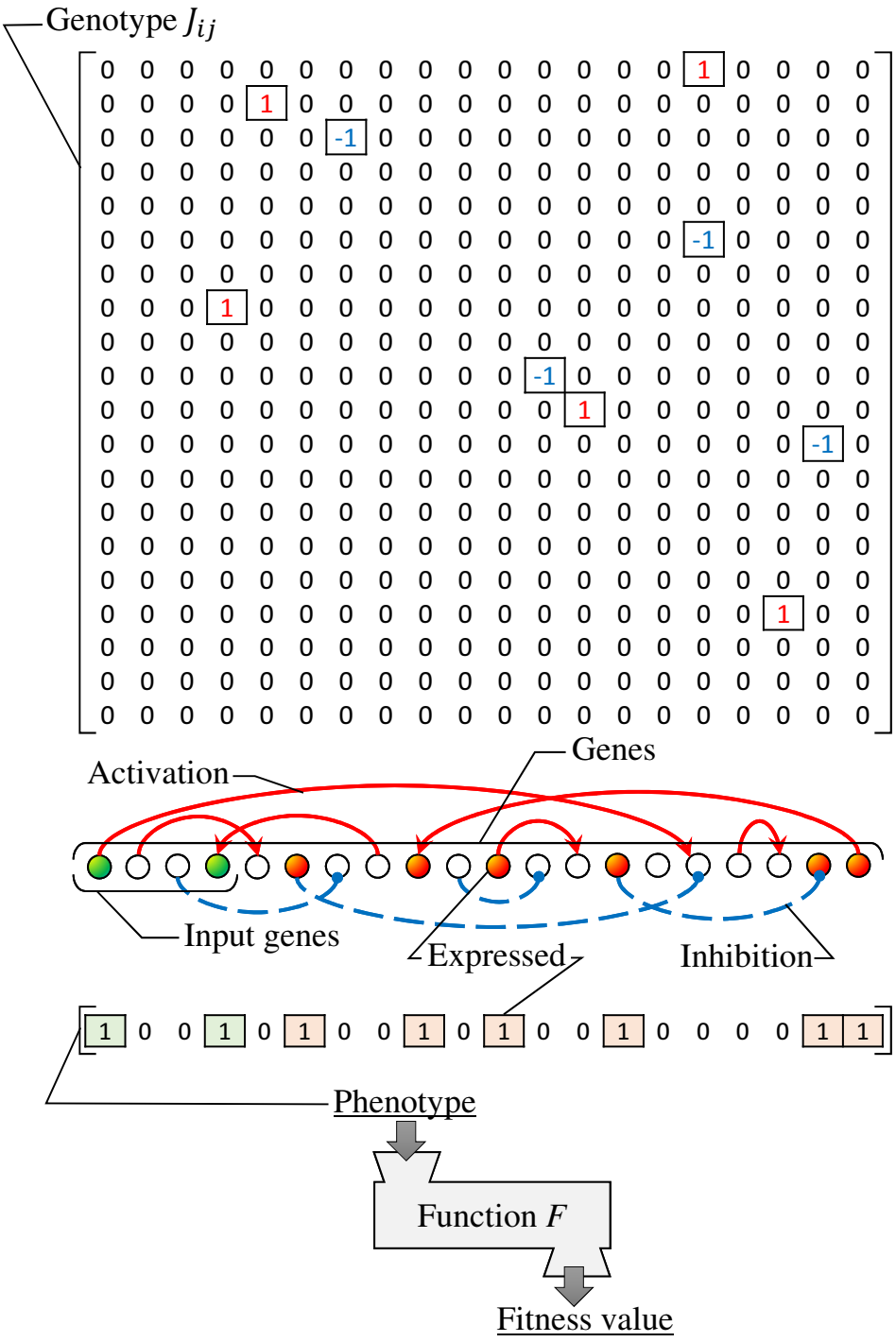


Figure 2.2: Genetic model for $M = 20, k_{inp} = 4$

γ is a constant. Here, $f(z)$ is a sigmoid function defined by

$$f(z) = \frac{1}{1 + \exp^{-\beta(z-\theta_i)}} + \delta, \tag{2.3}$$

where β is a parameter that determines the gradient in the neighborhood of the threshold θ_i and δ is a small positive number that represents a spontaneous expression level. The second term of Eq. (2.2) represents stochastic behavior caused by noise from the environment. In this, η_i is a random value that follows a normal distribution with a mean of zero and variance of σ^2.

The evolution model repeats a selection-mutation process for each generation. We start with K individuals, each of which has slightly different gene regulatory networks $\{J_{ij}^1, \ldots, J_{ij}^K\}$. In each generation, each individual updates its gene expression levels, x_i, by calculating the differential equation (2.2) for its own J_{ij}. The pattern of gene expression levels (i.e., the phenotype) determines the value of the fitness, $F(phenotype)$. That is, we obtain K fitness values that depend on the gene regulatory networks. The selection-mutation process is then applied to the K gene regulatory networks. Among the K gene regulatory networks, the K_s gene regulatory networks that show the highest fitness values are selected and kept for the next generation. The unselected gene regulatory networks are excluded from further calculations. K_s is a tunable parameter which we set to $K/4$ in the following. Each of the selected K_s gene regulatory networks is then mutated into 4 individuals by randomly choosing a few components in the matrices and changing the values to a random value from $\{-1, 0, 1\}$. This calculation of gene expression dynamics and selection-mutation process are repeated over many generations.

We can now explain how biological systems exhibit both robustness and plasticity. When the environment changes, that is, when the expression levels of the input genes change, the biological system first reacts through an increase in phenotypic variance. This reaction gives the biological system plasticity, which represents changeability in response to environmental changes. Robustness is obtained through the selection-mutation process. Once a genotype that produces a phenotype with higher fitness is found, its progeny will account for a large majority of individuals. The phenotypic variance thus decreases again.

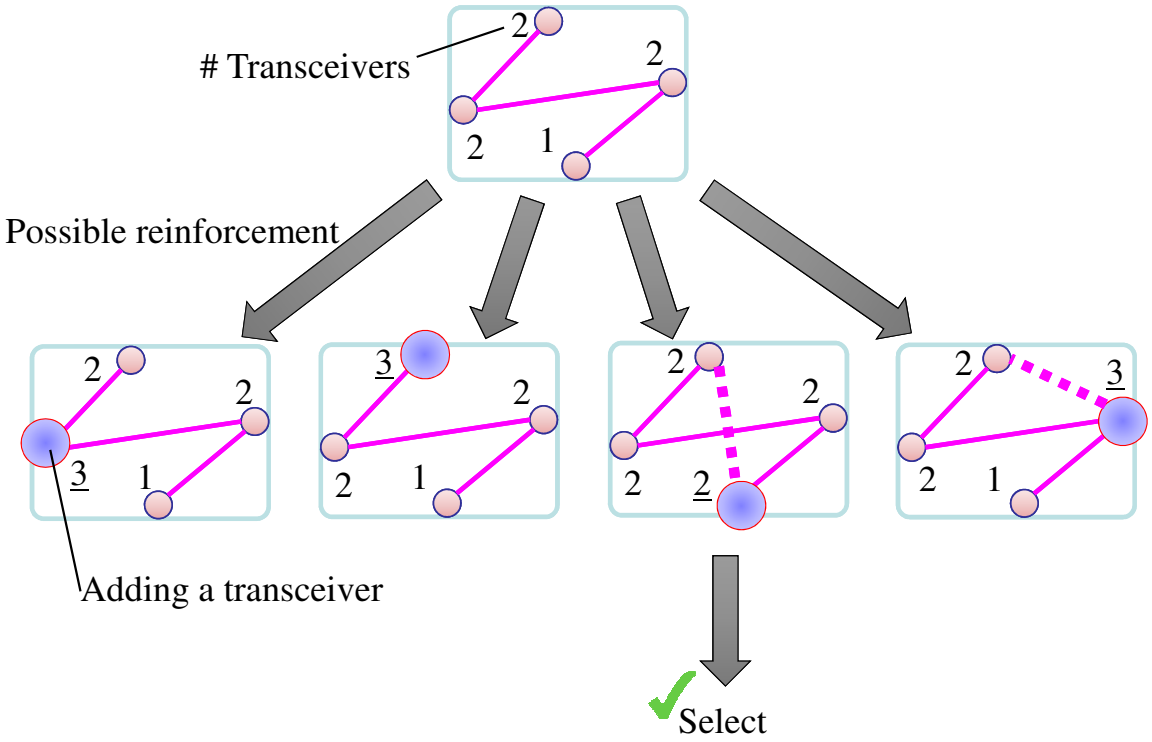


Figure 2.3: Example of applying our model to a WDM network

2.3.2 Applying our method to add transceivers

In this chapter, we consider optical transceivers as a target device for increasing the resources in a WDM network. Figure 2.3 shows a simple example of our application. A lightpath can be established only when transceivers are present at both end-nodes. Adding a transceiver may then result in making a new lightpath available. In this situation, the key is the selection of nodes to which we should add transceivers. Our proposed method determines the set of nodes (IP routers) to which transceivers should be added in order to give plasticity to the network by applying the biological evolution model.

Applying the biological evolution model to WDM network design

Table 2.1 shows the correspondence between the genetic evolution model and the design method for WDM networks. When the number of nodes in the WDM network topology is N, the number

Table 2.1: Correspondence between evolution model and WDM network

Biological evolution	WDM network
Dynamics of gene expression level	VNT control
Phenotype	VNT
Genotype	Regulatory matrix
Fitness	Average link utilization
Environmental change	Change in traffic demand

of candidates for lightpaths is equal to the number of node-pairs, N^2. Each gene i corresponds with a lightpath l_i, where $i = 1, 2, .., N^2$, and this correspondence is one-to-one. In each generation, the gene expression levels x_i are determined from the results of the expression dynamics (2.2). In the phenotype, that is, the pattern of gene expression levels that determines the VNT, the lightpath l_i is switched on (established) if x_i exceeds the threshold θ_i, and otherwise the lightpath l_i is switched off. Some constraints, such as wavelength-continuity constraints, that restrict the lightpath establishment can be easily incorporated by restricting this phenotype-to-VNT conversion. In this chapter, we will establish a lightpath only when there are available transceivers at the both source and destination IP routers. Note that, x_i where i equals to $n_*^2(n_* = 1, .., N)$, which represents a lightpath from one node to itself, is fixed to 0 to avoid a self-loop.

We use the average link utilization of the VNT to characterize the fitness. In the biological model, fitness is calculated on basis of the expression pattern of some of the genes. In our model, we instead substitute the average link utilization for the value of fitness. Note that lower values of average link utilization are more desirable. We therefore define fitness as the multiplicative inverse of average link utilization.

We treat changes to the physical network as an environmental change. In the biological model, the environment is represented by the expression of input genes with environmental changes given by modifying the values of input-gene expression levels. In our WDM design method, we assign the progress of adding transceivers as the values of the input genes. The number of input genes is equal to the number of WDM nodes, N. Therefore, there are N^2 ordinary genes ($i = 1, 2, .., N^2$) and N input genes ($i = N^2 + 1, N^2 + 2, .., N^2 + N$), giving $N^2 + N$ genes in total. The gene $N^2 + i$ represents the node i. Initially, the expression levels of all input genes are zero. Each

time a transceiver is added to node i, the expression level of gene $N^2 + i$ is incremented by 1 to express the effect of the physical network change, even though this may violate the allowable range of expression levels. This is one way to take changes to the WDM network into account in terms of the effect on expression dynamics and the way the VNT is constructed.

Evaluation of the plasticity of the WDM network

Our proposed method aims to determine the set of nodes (IP routers) to which transceivers should be added in order to give plasticity to the network. For this purpose, the degree of plasticity of a physical network needs to be evaluated. We thus examine the evolution process via the following steps.

Step 1 Observe the traffic demand.

Step 2 Repeat the selection-mutation process over $G(= 15)$ generations. In each generation, determine a VNT by using Eq. (2.2). The fitness value is then calculated given the observed traffic demand.

Step 3 Execute the following sub-steps L times.

Step 3.1 Change the traffic demand.

Step 3.2 Repeat the selection-mutation process over G generations. Calculate the fitness with the changed traffic demand.

Step 4 Calculate the degree of plasticity by using the L fitness values obtained in Step 3.

At the beginning of reinforcement, we first obtain the traffic demand (Step 1). In Step 2, we examine the selection-mutation process for the observed traffic demand and obtain a set of gene regulatory networks J_{ij} that are suitable for the observed traffic demand. In Step 3, we examine various patterns of traffic fluctuations in a random manner. Note that a single pattern of traffic fluctuation is not sufficient for estimating the plasticity. We obtain $L(= 16)$ fitness values as a result of Step 3. In this chapter, the degree of plasticity is chosen as the median fitness value.

Proposed design method

Our aim is to give plasticity to a WDM network as a result of adding transceivers. We evaluate the plasticity by computational simulation in which some transceivers are added to a certain set of nodes. However, it is difficult to estimate the plasticity in order to select the locations for the transceivers since the number of possible combinations of locations increases exponentially as the number of transceivers increases. We therefore apply a simple heuristic, called the ADD algorithm [46], to determine the locations for the transceivers. Given the number of transceivers to add, the ADD algorithm works as follows.

Step 1 Select a node at which to add a transceiver by calculating the plasticity when a transceiver is added to the node.

Step 1.1 Temporarily add a transceiver to each node.

Step 1.2 Evaluate the plasticity of the WDM network as explained in Sec. 2.3.2.

Step 1.3 Select the node that gives the highest value of plasticity in Step 1.2.

Step 2 Add a transceiver to the selected node. If there are more transceivers to add, go back to Step 1.

In this chapter, we consider a situation in which traffic demands keep on increasing, and apply the ADD algorithm to decide where to add transceivers. In practice, the traffic demands may possibly be decreased, but we can again apply our method to decide which transceivers should be removed by replacing “add” with “remove” in the algorithm.

2.3.3 Time scale of VNT control and network reinforcement

The biological evolution model explains how organisms obtain plasticity. When we apply the biological evolution model to network design methods, the question arises of when transceivers should be added. Organisms may have their own cycle for applying the evolutionary processes discussed above. In the case of our network design problem, we assume that network reinforcement is performed when the VNT control method cannot find a good VNT. Note that we define the goodness of

a VNT according to link utilization under the current traffic demand. Thus, network reinforcement is performed when the VNT control cannot achieve a link utilization that is lower than a certain threshold.

Figure 2.4 illustrates the time scale of VNT control, network reinforcement, and traffic changes. In the figure, the horizontal axis represents the time step of traffic changes, and the volume of traffic demand increases at each time step. At each step, if necessary, the VNT control method tries to find a good VNT for the traffic demand. If the VNT control method finds a good VNT, then it keeps the existing VNT until the next time step (see time steps 0, 1, and 2 in the figure). If the VNT control method cannot find a good VNT at time-step t, then we treat the VNT control as having failed at t. Network reinforcement is then performed as soon as we know that the VNT control has failed, and the VNT control method is again applied at time-step $t + 1$. In this illustration, we assume that the solution of the reinforcement is calculated within a time-step for simplicity. When the algorithm takes several time-steps to find the solution, the network operator should execute the reinforcement method more conservatively so as not to take a high link utilization during the calculation. Note that VNT control method works every time-step even when our design algorithm is under calculation. Thus, it is necessary for our design approach to prevent lack of resources during the calculation.

2.3.4 Possible extension

In this chapter, we treat the transceivers in IP routers as physical network resources. Our basic idea can easily be extended to the deployment of other network resources such as physical links and/or nodes. For example, when considering the deployment of physical links, we could assign input genes to the node-pairs between which links can be connected instead of the nodes to which transceivers can be added.

2.4 Evaluation

We evaluated the performance of the proposed method by computer simulation. The performance is measured in terms of the adaptability of the attractor-based VNT control method [33] on a WDM network that has been reinforced by the design method.

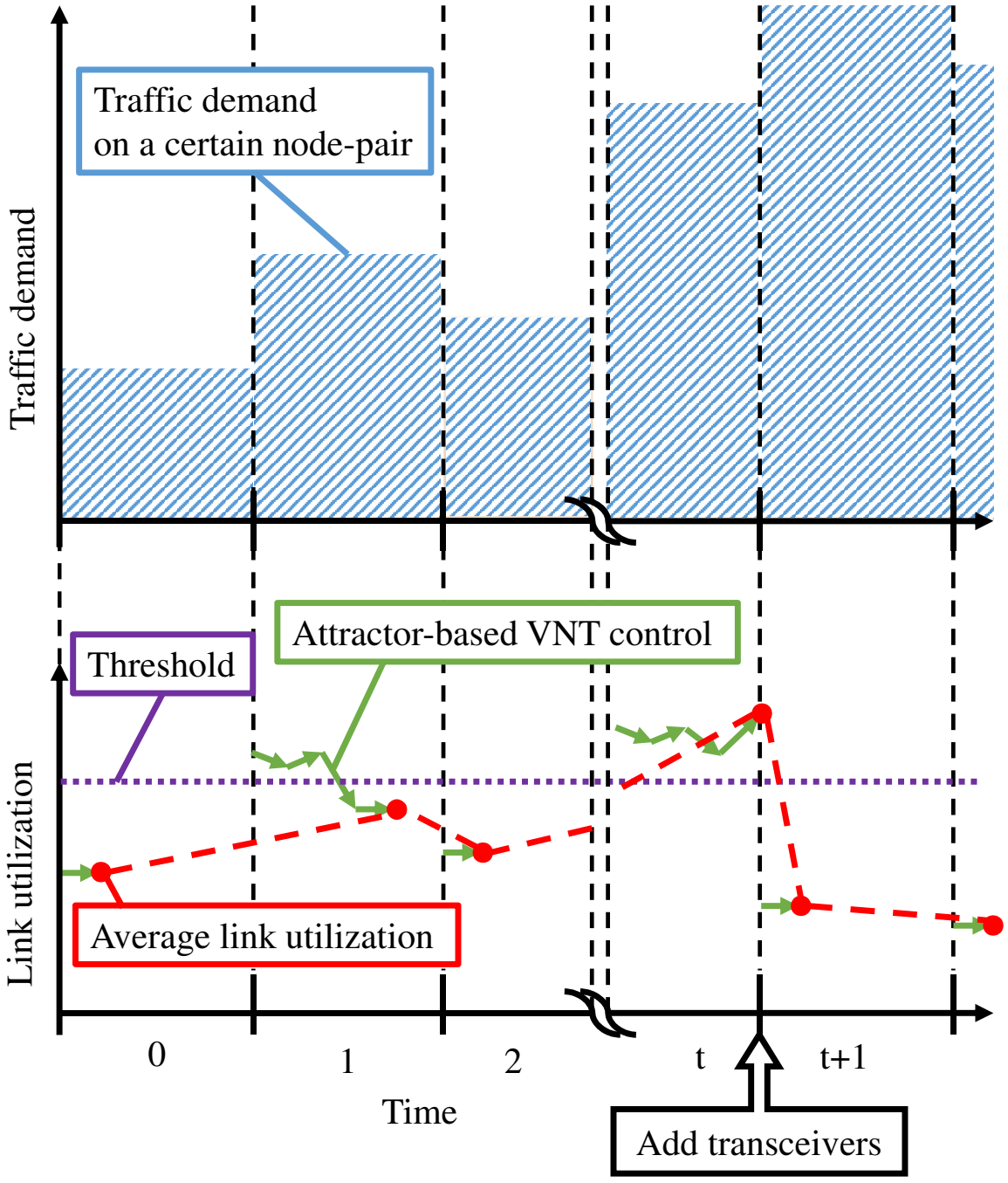


Figure 2.4: Relation between VNT control and network reinforcement

2.4.1 Methods for comparison

We consider an ad-hoc design approach for comparison purposes. Constructing a general design method as a method for comparison is unreasonable. This is because the design principle depends on the situation, such as the business scenario, available traffic information, or user demand.

The ad-hoc design approach here is intended to decide an effective placement of resources to achieve the best VNT performance. To do this, a VNT configuration method which intends to minimize the link utilization is applied, supposing resource reinforcement was done at a certain place. The above process is then repeated until reinforcement at all possible places is examined. For the VNT configuration method, mixed integer linear programming (MILP) methods of lightpath assignment [47] have been used, and heuristic methods have also been used. These methods collect the present traffic demand information, or predict future traffic demand in some cased, and then attempt to minimize the network load as characterized here by link utilization.

We construct the ad-hoc design by both MILP and heuristic methods. The formulation of the MILP is introduced in Sec. 2.4.1. The heuristic method, I-MLTDA, is introduced in Sec. 2.4.1. The flow of the ad-hoc design process is as follows.

Step 1 For each candidate, do the following sub-steps.

Step 1.1 Temporarily add transceivers to the node.

Step 1.2 Execute $\{\text{MILP}|\text{Heuristic}\}$ against the traffic demands at the time of reinforcement.

Step 1.3 Evaluate the average link utilization for the VNT obtained in Step 1.2.

Step 2 Determine a node at which to add transceivers. We select the node that shows the lowest value of average link utilization and then add transceivers to that node. Go back to Step 1 if there are more transceivers to add.

These design approaches are expected to show good performance in cases where the environment changes slowly and moderately. Traffic prediction may also help the designs. However, they are expected to show a severe degradation on performance in cases where traffic demand changes drastically and traffic prediction is not feasible. Our proposed design aims to accommodate various types of traffic change rather than to minimize the present link utilization.

MILP

We use the following formulation of MILP to obtain lightpath assignment that minimizes the link utilization.

Notation:

$\boldsymbol{V}$: Set of physical nodes.

N: Number of physical nodes. $N = |\boldsymbol{V}|$.

u, v, s, d: Node ID.

Given:

d_u: Number of transmitters and receivers at node u.

T_{uv}: Traffic demand from node u to node v, collected in some way and assumed to be known.

Variables:

x_{uv}: Binary variable that takes the value 1 if a lightpath is established from node u to node v, and 0 otherwise.

f_d^{uv}: Amount of traffic demand toward node d via the lightpath from node u to node v.

Constraint 1: The number of transceivers on each node limits the number of lightpaths that can be established.

$$\sum_v x_{uv} \leq d_u \qquad \forall u \in \boldsymbol{V} \tag{2.4}$$

$$\sum_u x_{uv} \leq d_v \qquad \forall v \in \boldsymbol{V} \tag{2.5}$$

Constraint 2: A lightpath from node u to node v must be established if there is some traffic demand to go through it. Note that traffic demand values are scaled so that the sum does not exceed 1.0.

$$x_{uv} \geq \sum_d f_d^{uv} \qquad \forall u, v \in \boldsymbol{V} \tag{2.6}$$

Constraint 3: Consistency of traffic accommodation and injection.

$$\sum_u f_d^{ud} = \sum_s T_{sd} \qquad \forall d \in \boldsymbol{V} \tag{2.7}$$

$$\sum_v f_d^{kv} = \sum_u f_d^{uk} + T_{kd} \qquad \forall k, d \in \boldsymbol{V} (k \neq d) \tag{2.8}$$

Constraint 4: Connectivity of physical topology.

$$x_{uv} = 1 \qquad \forall (u, v) \quad s.t. \quad u \text{ and } v \text{ are physically connected.} \tag{2.9}$$

Objective: Minimizing traffic load on links of VNT.

$$\textbf{minimize} \quad \sum_d \sum_u \sum_v f_d^{uv} \tag{2.10}$$

Heuristic method (I-MLTDA)

The increasing multi-hop logical topology design algorithm (I-MLTDA) [41] is a heuristic method of designing a quasi-optimal VNT by using traffic demand and hop lengths. I-MLTDA establishes lightpaths between node-pairs (s, d) in order from those that show the largest values of $\Delta^{sd} \times (H^{sd} - 1)$, where Δ^{sd} is the traffic demand from node s to node d and H^{sd} is the hop length along the shortest path from s to d. The details of I-MLTDA are as follows.

Step 1 Establish a lightpath between every node-pair that has a physical connection. Go to Step 2.

Step 2 Calculate the shortest path and determine the value of H^{sd} for each s and d. Go to Step 3.

Step 3 Determine the node-pair (s, d) that exhibits the maximum value of $\Delta^{sd} \times (H^{sd} - 1)$. If $\Delta^{sd} \times (H^{sd} - 1)$ is 0, stop. Otherwise, go to Step 4.

Step 4 If available transceivers remain on node s and node d, then establish a lightpath from node s to node d. Otherwise, set the value of Δ^{sd} to 0. Go to Step 2.

In this chapter, we use I-MLTDA for the heuristic algorithm. Although various design methods have been proposed, our purpose for introducing the ad-hoc design method is to examine the failure of design methods that are optimized and specialized to an environment as it exists at one point in time. We believe that our results in Sec. 2.4.3 are also valid for other heuristic algorithms.

2.4.2 Simulation environments

This section explains the environments used in our simulation.

Topology

We evaluated our proposed method on three physical topologies: the European Optical Network (EON) model [45], the USNET model [48], and the Japan telecommunication network model (which we call the JBN model) [49]. Figure 2.5, 2.6 and 2.7 show these topologies, and Table 2.2 shows the number of nodes and links in each topology. Each node is composed of an IP router and an OXC. Each OXC is connected to other OXCs by links, as shown in Fig. 2.5, 2.6 and 2.7. Each link is a single optical fiber. Establishing lightpath between an IP router and another IP router uses one transceiver of the source node and one transceiver of the destination node. A lightpath can be established when there are available transceivers at both source and destination routers. The initial number of transceivers at each node is set to 2 plus the degree of the node in the physical topology.

We execute computer simulations on the three physical topologies. For the EON topology, we compare our proposed method with the MILP-based method and the heuristic-based method. For the USNET topology and the JBN topology, we perform comparisons with only the heuristic-based method because the computation time needed for MILP becomes enormous for large topologies. However, from the results obtained for the EON topology, the distribution of the average link utilization in the MILP-based design is much the same as that in the heuristic-based design. Similar trends are also expected to be obtained for the USNET and JBN topologies.

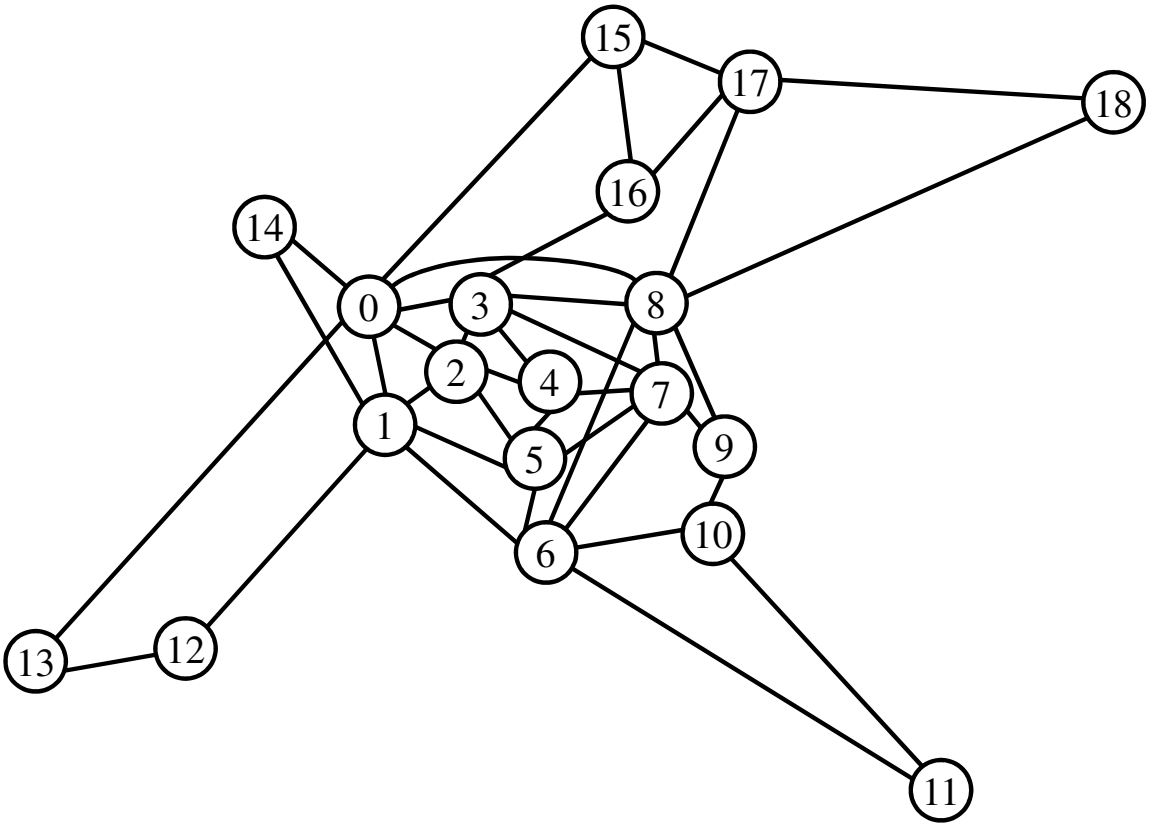


Figure 2.5: Topology used in the computer simulation: EON

Traffic demand model

Each node-pair has its own traffic demand. The initial values follow a lognormal distribution according to [50]; specifically, each traffic demand is set to a random number following $LN(\mu = 1, \sigma^2 = 0.5^2)$. The traffic demand is then increased or decreased at each time-step. Taking $T_{act}^{i,j}(t)$ to represent the traffic demand from node i to node j at time-step t, the traffic demand model [51] is defined by the following expression

$$T_{act}^{i,j}(t) = T_{exp}^{i,j}(t) + N(0, (\sigma_{noise} \times T_{exp}^{i,j}(t))^2), \tag{2.11}$$

where $T_{exp}^{i,j}(t)$ is the expected value of traffic demand from node i to node j at time-step t. The second term represents unexpected traffic fluctuations and is set to a random value, following a

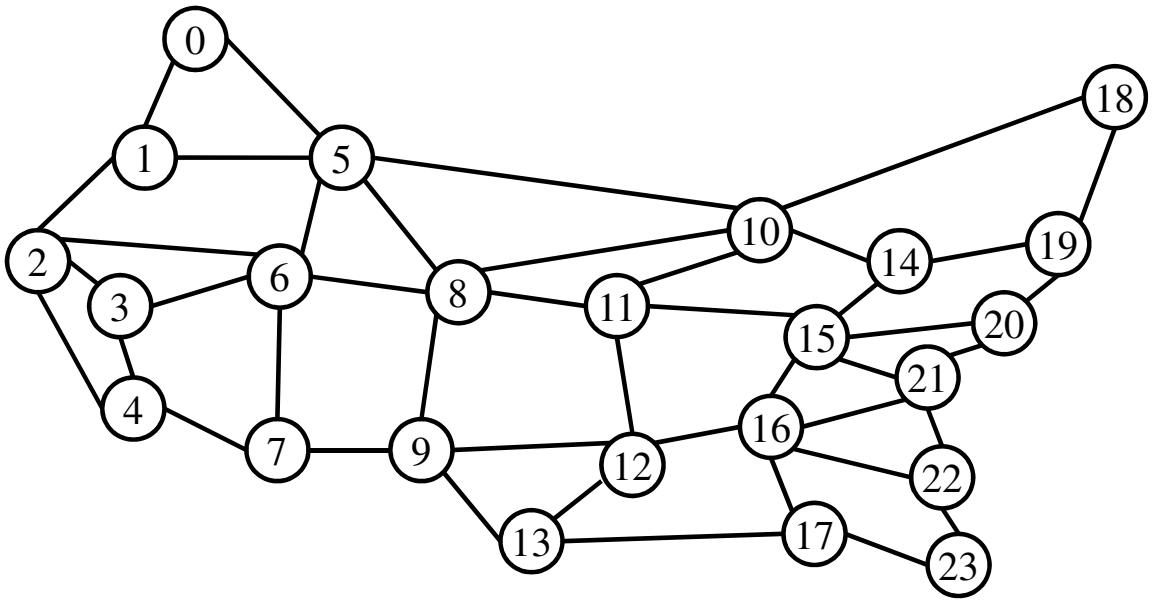


Figure 2.6: Topology used in the computer simulation: USNET

Table 2.2: Numbers of nodes and links

Topology	# of nodes	# of links
EON	19	39
USNET	24	42
JBN	49	91

normal distribution $N(0, (\sigma_{noise} \times T^{i,j}_{exp}(t))^2)$. Higher values of σ_{noise} represent sharper changes in traffic demand. In contrast, when σ_{noise} takes a lower value, the noise term has less effect and $T^{i,j}_{act}(t)$ is close to $T^{i,j}_{exp}(t)$. The values $T^{i,j}_{exp}(t)$ are calculated from the following recurrence formula:

$$T^{i,j}_{exp}(t) = m + T^{i,j}_{act}(t-1). \tag{2.12}$$

The expected value increases by m at each time step. Therefore, traffic demands continue to increase on average, but the trends in traffic fluctuation are different for each node-pair. The traffic in the VNT is assumed to be forwarded along the path with the minimum-hop path.

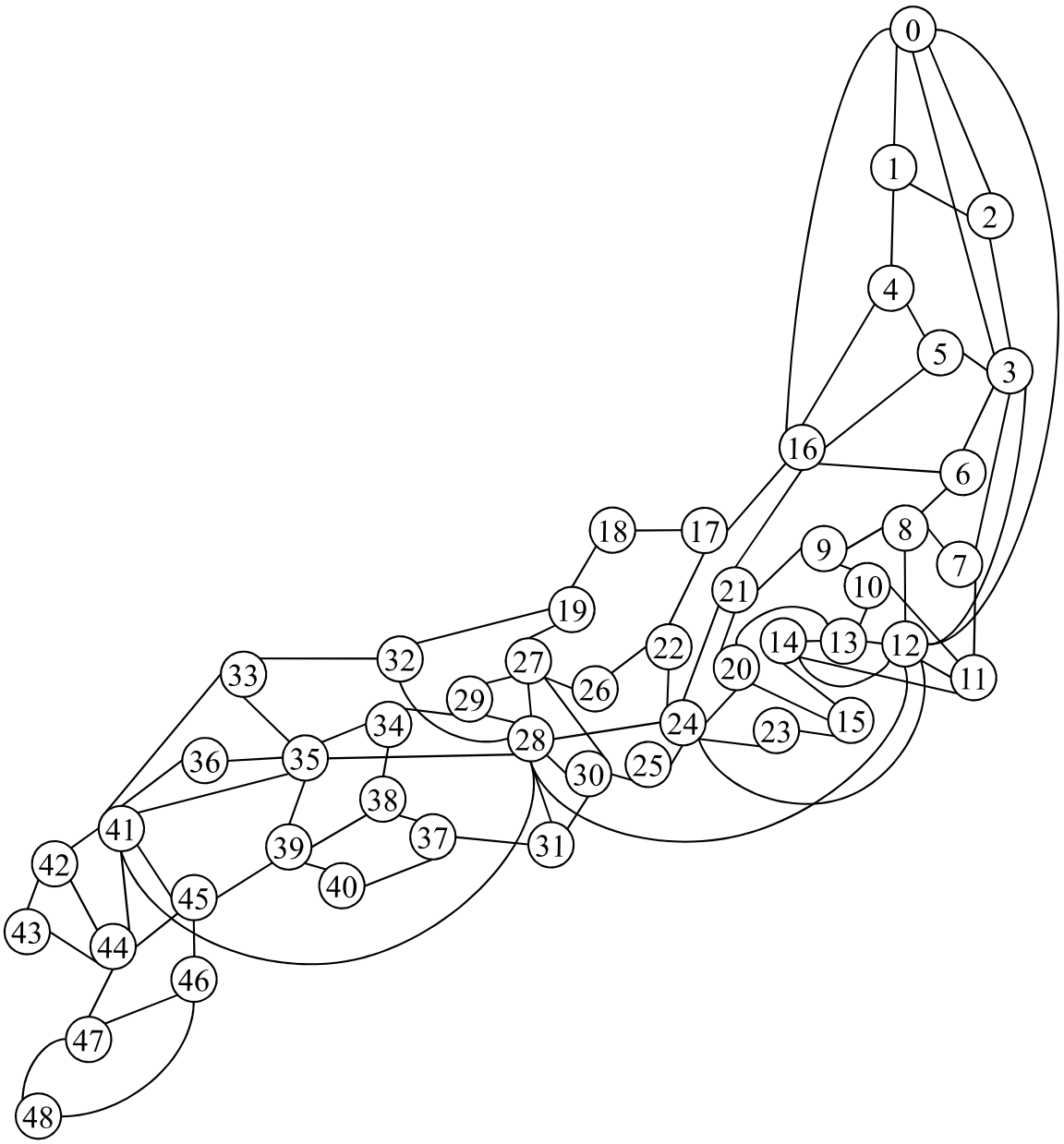


Figure 2.7: Topology used in the computer simulation: JBN

Attractor-based VNT control method

We use the attractor-based VNT control method in the evaluation. We again apply I-MLTDA as the VNT control method as well. However, our primary purpose is to design a WDM network that

maximizes the adaptability of VNT control, and so we also use our attractor-based VNT control method for evaluation.

Our VNT control method is driven by activity, which is a feedback of network status. When activity is low, the random behavior tries to seek a better VNT. In this chapter, the activity is given by the following equation

$$activity = \frac{\gamma}{1 + e^{\delta(L_{average} - \theta)}}, \tag{2.13}$$

where $L_{average}$ is the average link utilization and other literals are parameters. With this definition, the activity rapidly approaches zero when the average link utilization exceeds θ. That is, the VNT control method attempts to reduce the average link utilization to less than the threshold. The activity value is always in the range $(0, 1)$ since we set $\gamma = 1$. The condition of the IP network is assumed to be poor whenever the average link utilization is greater than θ. In this work, we set the threshold θ to 0.25 for the EON topology and to 0.50 for the USNET and JBN topologies. The gradient δ of the activity function is set to 50, following Ref. [33].

2.4.3 Simulation results

Evaluation against future traffic changes

We set σ_{noise} in Eq. (2.11) to 0 at the beginning of the simulation and apply attractor-based VNT control. Following Eqs. (2.11) and (2.12), the traffic demand eventually increases over time. At this point, we set m to 0.01. At time-step $t_{\text{reinforce}}$, the VNT control method fails to find a good VNT among 400 reconfigurations. Our design method and the ad-hoc design method then calculate the node at which to add transceivers. The methods select three nodes to reinforce transceivers, and 4 transceivers are added to each selected node. Table 2.3 shows the results of the calculations for each topology. For example, in the EON topology, the VNT control method fails at time-step 140, and our proposed method then adds transceivers to nodes $\{6, 6, 11\}$, where the repetition of node 6 indicates it is chosen multiple times.

Since the proposed method has higher computational complexity than the heuristic-based method does, it takes much longer to execute the simulation with this method. However, the computational

Table 2.3: Calculation results (transceivers are added to the following nodes)

Topology	$t_{\text{reinforce}}$	Proposed	MILP-based	Heuristic-based
EON	140	{6, 6, 11}	{13, 6, 18}	{11, 12, 18}
USNET	214	{6, 6, 6}	-	{18, 23 ,3}
JBN	118	{24 ,24 ,24}	-	{37 ,46 ,19}

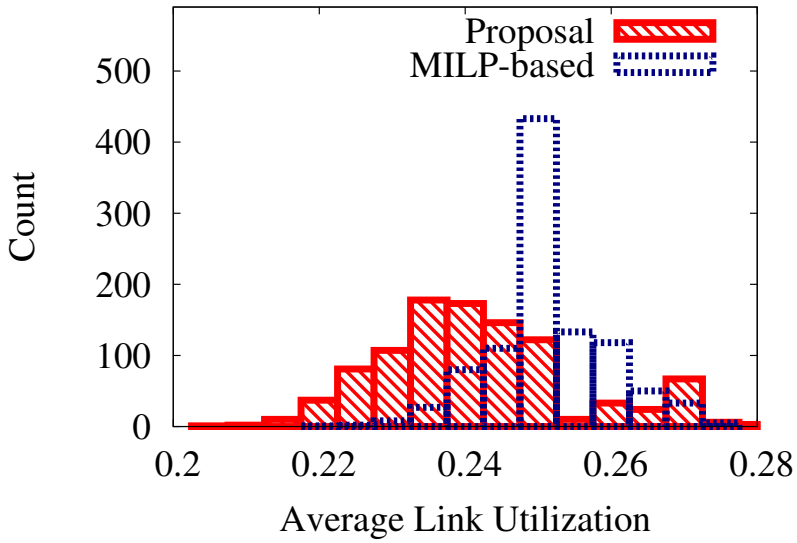


Figure 2.8: Distribution of average link utilization: Histogram on EON (Proposal - MILP)

time is not a significant problem here because physical network designs are not expected to be executed over short intervals but, instead, over the long term. Note that we stop the MILP execution of Step 1.2 in Sec. 2.4.1 after 10 minutes and then use the results obtained by then as an approximate solution.

We evaluate the adaptability of the reinforced WDM network against unexpected traffic increases. After the reinforcement, we set the parameter σ_{noise} to 0.10 and examine the various patterns of traffic fluctuation to check whether the attractor-based VNT control method can find a good VNT.

Figure 2.8 and 2.9 show the distribution of average link utilization at time-step t_{evaluate} for 1000 patterns of traffic fluctuation on the EON topology. We define $t_{\text{evaluate}} = t_{\text{reinforce}} + 70$. Note that the traffic increases and fluctuates in different ways from time-step $t_{\text{reinforce}}$ to t_{evaluate}.

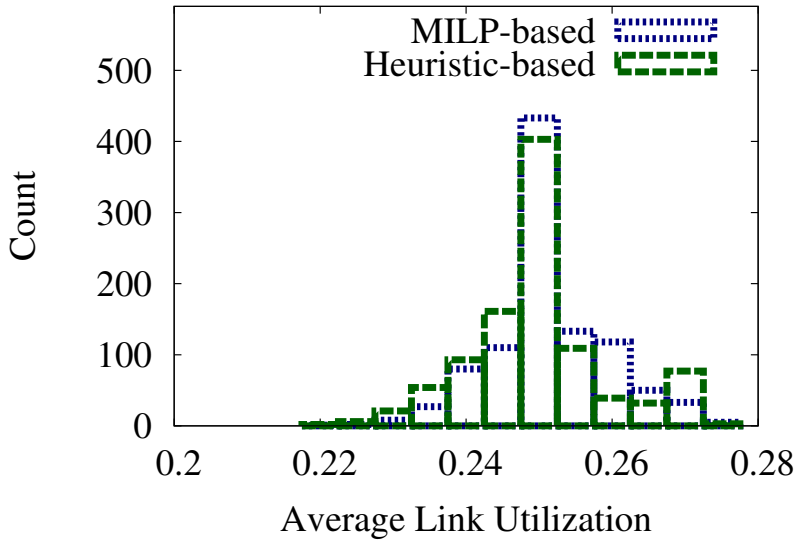


Figure 2.9: Distribution of average link utilization: Histogram on EON (MILP - Heuristic)

Therefore, Fig. 2.8 and 2.9 show the performance of the VNT control method against various patterns of traffic fluctuation. The threshold of the activity is set to 0.25, that is, the VNT control method is assumed to be successful if the average link utilization is less than 0.25. First, Fig. 2.8 shows the comparison between the proposed design and the MILP-based design. Both the proposed design and the MILP-based design succeed in accommodating most traffic patterns with admissible values. However, more traffic patterns are accommodated with lower link utilization by the proposed method than are accommodated by the MILP-based design method. In addition, the proposed design fails for fewer traffic patterns than the MILP-based design does. Second, Fig. 2.9 shows the comparison between the MILP-based design and the heuristic-based design. The heuristic-based design for the EON topology adds transceivers to nodes $\{11, 12, 18\}$, different from the decision of the MILP-based design. However, looking at Fig. 2.9, the distributions of link utilization by both of the ad-hoc designs are almost the same. Thus, we use the heuristic-based design as an ad-hoc design for the examination on the USNET and JBN topologies, since they are large topologies and are difficult for the MILP calculation.

Figure 2.10 and 2.11 show the results on the USNET topology and the JBN topology, respectively. In Fig. 2.10 and 2.11, the threshold of the activity is set to 0.5. In these situations, the VNT

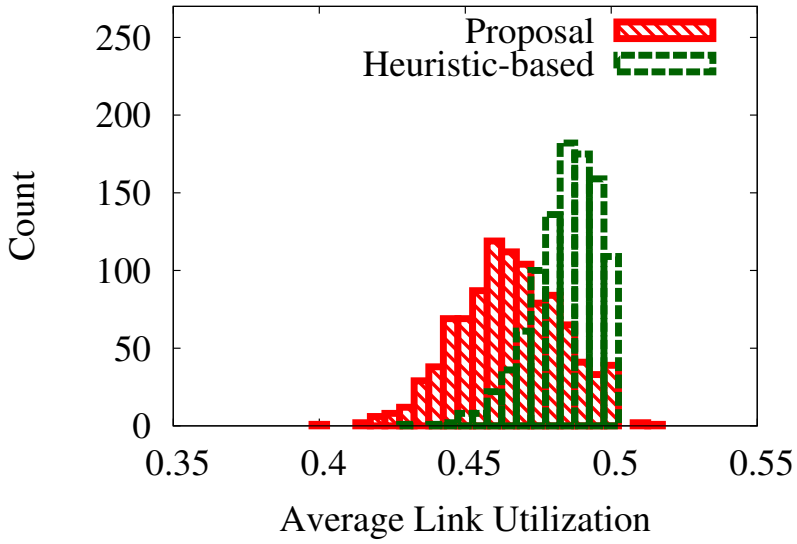


Figure 2.10: Distribution of average link utilization: Histogram on USNET

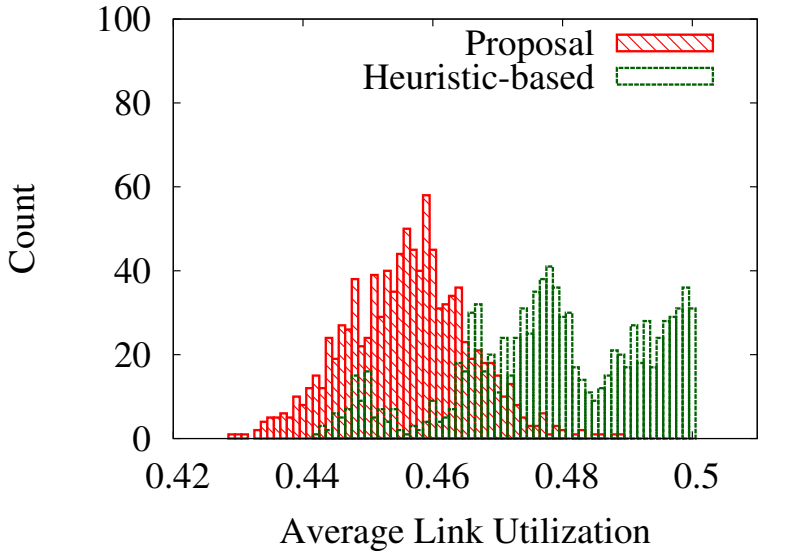


Figure 2.11: Distribution of average link utilization: Histogram on JBN

control succeeds for almost all traffic patterns, and more traffic patterns are accommodated with lower link utilization by the proposed design as with the EON topology. We can conclude that the proposed method makes the optical network more flexible, that is, our method improves the ability

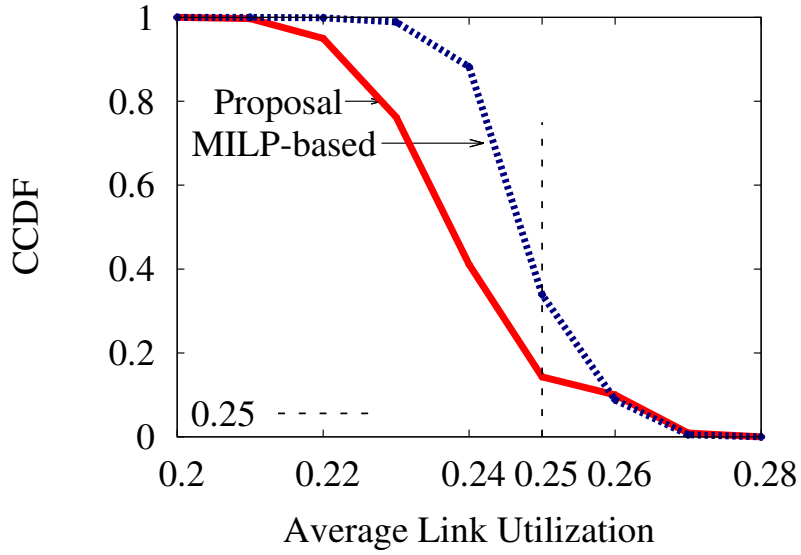


Figure 2.12: Distribution of average link utilization: complementary cumulative distribution function (CCDF) on EON

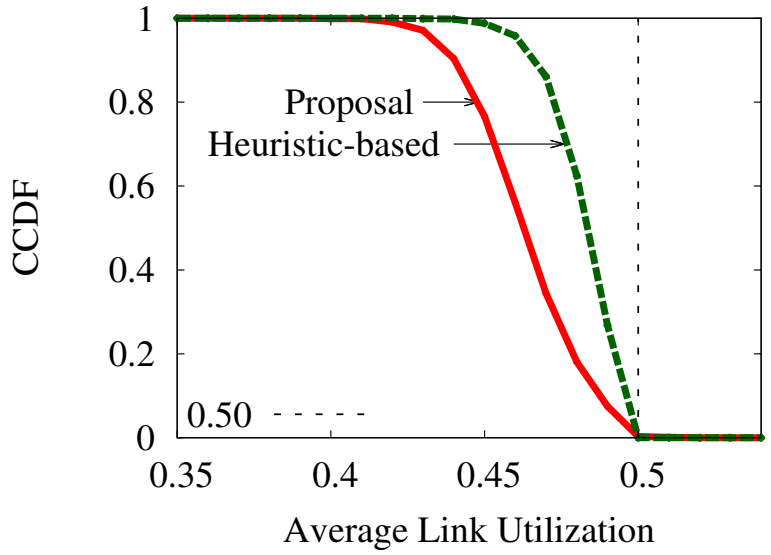


Figure 2.13: Distribution of average link utilization: complementary cumulative distribution function (CCDF) on USNET

to accommodate various traffic fluctuations with lower link utilization.

Figure 2.12, 2.13 and 2.14 show the complementary cumulative distribution function (CCDF)

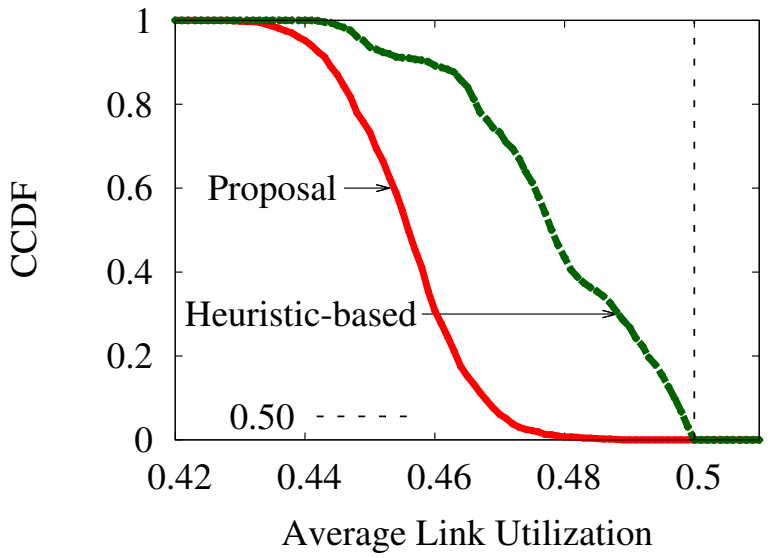


Figure 2.14: Distribution of average link utilization: complementary cumulative distribution function (CCDF) on JBN

for each network, using the same data as before. In Fig. 2.12, the CCDF of the intersection points with the vertical line at 0.25 are 0.143 and 0.339 respectively. This indicates that the proposed method accommodates 857 patterns of traffic fluctuation, the MILP based design method accommodates 661 on the EON topology. Consequently, the proposed method raises the success rate, which is a rate of the traffic patterns where the attracted-based VNT control method makes the average link utilization lower than the threshold, by about 29% ($0.857/0.661 \approx 1.297$) compared with the ad-hoc design methods.

Evaluation with respect to noise strength

We evaluated additional simulations on the EON topology by changing the noise strength σ_{noise} of traffic fluctuations to see whether the network suggested by our design method can accommodate various patterns of traffic fluctuation. Figure 2.15 shows the success rate of the VNT against σ_{noise}. We examine 100 patterns of traffic fluctuation and calculate the success rate for each σ_{noise}, and the average/minimum/maximum of success rates over 10 examinations are plotted. We observe that the proposed design improves the success rate when traffic fluctuates strongly. When the noise level

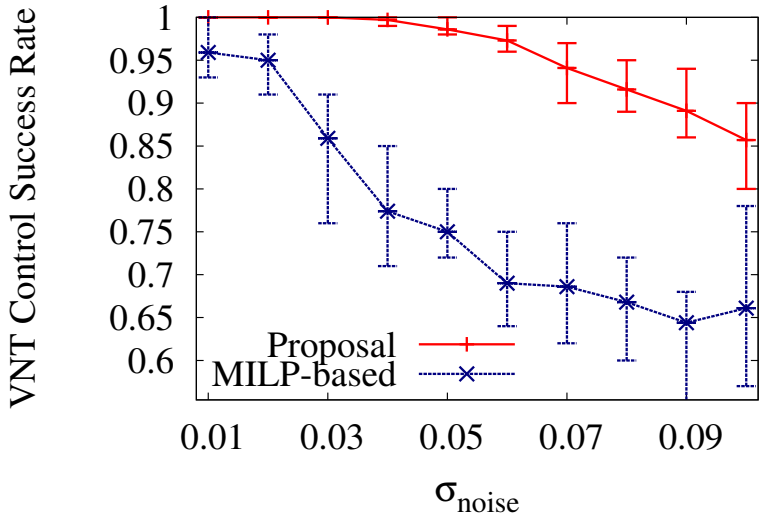


Figure 2.15: VNT control success rate

is low, both the proposed design achieve an almost 100% success rate, and the MILP-based design also achieves a success rate of over 95%. This is because the networks from the ad-hoc design are optimized and specialized to the traffic-demand matrix at the time of reinforcement, and the traffic demand matrix does not change drastically with lower levels of σ_{noise}. However, as the noise level increases, the traffic changes more drastically and the VNT control using the ad-hoc design cannot handle the traffic fluctuations. In comparison, the proposed design with VNT control can accommodate more traffic patterns, even when σ_{noise} is high.

Figure 2.16 shows the distribution of average link utilization against different σ_{noise}. The upper bar indicates the maximum value of average link utilization against 1000 patterns of traffic fluctuation. In the same way, the lower bar indicates the minimum value and the center bar indicates the median value. The box region indicates the range of the average link utilizations for 80% traffic patterns, which excludes the worst 10% and the best 10%. Focusing on the upper bound of the box for each value of σ_{noise}, it can be seen that the proposed design keeps the values smaller than the ad-hoc design does. Therefore, the proposed design is able to maintain the adaptability of the VNT control against most traffic fluctuations. However, the box ranges of the proposed design are larger, and the maximum value sometimes exceeds that obtained with the ad-hoc design. This is caused

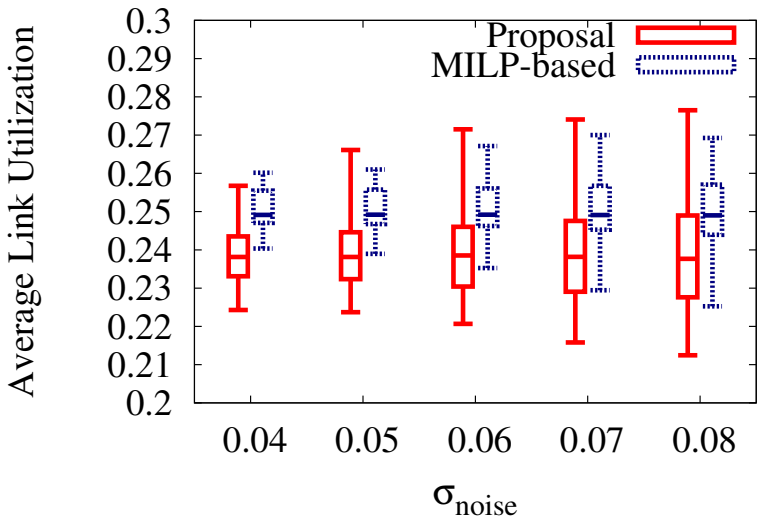


Figure 2.16: Distribution of average link utilization against different σ_{noise}

by the stochastic behavior of the attractor-based VNT control. When the average link utilization exceeds the threshold 0.25 (or 0.5) and the activity is reduced to 0, the system continues to search for a new VNT due to the stochastic term η. Hence, the final VNT after 400 iterations may be worse. However, this problem is not particularly critical and can also be mitigated by dynamically reconfiguring the activity [52].

2.5 Conclusions

We proposed a design method for optical networks with a concept of plasticity. The method determines the set of nodes where transceivers should be added and is inspired by biological evolution with the aim of network plasticity. Computer simulation for some WDM networks showed that our method makes attractor-based VNT control methods more adaptive to unexpected traffic fluctuations and reduces degradation of the adaptability under strong traffic fluctuations.

In the future, we intend to extend our method so that it can not only add transceivers to nodes but also add links between nodes.

Chapter 3

Noise-induced VNE Method for Software-defined Infrastructure with Uncertain Delay Behaviors

3.1 Introduction

Information networks are faced with new emerging services, such as mobile services, cloud computing services, and social services. Software-defined infrastructure (SDI) enables rapid deployment of new services on information networks and/or information systems by providing virtualized infrastructure to customers by slicing computing resources and network resources.

In an SDI framework, thanks to the advance of virtualization technologies combined with software technology, customers order resource by making requests to service providers and the sliced virtualized resource is immediately assigned to the requesting customer.

A key to leveraging an SDI framework is network virtualization technologies and their control. Network virtualization technologies are in the research and development phase. In recent years, software-defined networking (SDN) and network-function virtualization (NFV) technologies have been expected to replace the conventional network management systems, and standardization of SDN/NFV technologies is being promoted. SDN/NFV technologies enable programmable and

automated network control, while conventional systems require the network operator to configure various kinds of network devices [11–15]. That is, SDI frameworks realized by SDN/NFV technologies have the potential to support rapid and flexible deployment of services, such as on-demand resource allocation, self-service provisioning, and secure cloud services [12].

Although SDN/NFV technologies and their standardization are important for deploying SDI, another important problem is to control the assignment of physical resources to a virtual network under changes in traffic demand and service demand. For this problem, the virtual network embedding (VNE) problem has been addressed [16–22]. The VNE problem is a placement problem in which virtual resources are to be allocated to the physical network with optimization of some performance objectives. In the VNE problem, service demands from customers are translated to virtual network (VN) requests. A VN consists of virtual nodes and virtual links. Each of the virtual nodes is hosted on a physical node as a form of virtual machine. Then, the virtual nodes are connected through a path of physical nodes, forming virtual links.

The VNE problem is divided into two sub-problems: virtual node mapping and virtual link mapping. Virtual node mapping decides the location of the physical node for each virtual node. Note that each virtual node must be allocated to a physical node supporting its "node attribute." The node attribute allows classification of nodes in ways defined by the supported operating system (OS), storage type, or node use (e.g., computing, storage, or packet switching). Virtual link mapping decides the path on the physical network for virtual links between virtual nodes.

In [19–22], a centralized calculation was assumed to solve virtual node mapping and virtual link mapping. That is, a centralized component gathers traffic information and resource utilization for each VN and identifies the current situation (i.e., the current traffic demand and/or the current service demand) of the networks. Then, the component solves the optimization problem that optimizes some metric, such as maximizing revenue or minimizing resource utilization. However, when the network size gets larger and the number of multiplexed VNs increases, the identification of the current situation becomes complicated by the enormous amount of network information. As the network operators want to know the current situation more accurately and precisely, more information is necessary to collect. This will lead to increased used of link bandwidth, increased delay, and a bottleneck on network scalability [15]. There is also proposed an on-demand VNE

control with using predicted demands [53]. Although the method achieves high prediction accuracy of future service chain requests based on deep learning, adaptability for unexpected situation is not concerned. Note that the calculation time to obtain a solution of the optimization problem also gets larger. However, the calculation time is not crucial because it may be relaxed by some heuristic algorithms with some sacrifice of the quality of the solution. Our concern in adopting the centralized approach is the overhead of collecting information, and this overhead gets larger as the size of the infrastructure and number of VN requests increase. Moreover, the environments surrounding the Internet today are continuously changing, thus, adaptive control of VNE is required to handle uncertain changes in the environments. Although precise modeling of the end-to-end delay in SDI environment is difficult, it would be required to suppress the maximum delay in order to guarantee a specific quality of experience (QoE) for applications on VNs. There are several models of network delay proposed, which are constructed generally and disregard the data contexts of packets [23]. However, the processing delay on servers depends on multiple factors, including server specification; CPU and memory utilization (on virtual machines); and details of processing, which depend on the context of the data.

In this chapter, we present a VNE method that works with only a little information for large, complicated, and uncertain SDI frameworks. To achieve this, the proposed method applies the biological "Yuragi" principle. Yuragi is a Japanese word whose English translation is a small perturbation, both externally and internally generated, to the system. Yuragi is a mechanism that provides adaptability to organisms and is often expressed as an attractor selection model. Our research group has developed a virtual network control based on attractor selection for optical networks. Our results showed that our control mechanism has high adaptability to environmental fluctuations with restricted information. Unlike a virtual network on an optical network, a virtual network on an SDI framework has to consider various matters such as node attribute, computational performance of servers, and VN multiplexing. Therefore, this chapter develops a Yuragi-based VNE method that deals with node attributes, has the generality to set a performance objective, and runs in multi-slice environments. One process of the method is executed for each VN slice, and each process needs information about only its own VN requests. Each of the processes behaves so as to improve its own performance function, considering other VNs as a part of an external perturbation (i.e., Yuragi). We

have presented a preliminary version of this work in [27] and have demonstrated the basic behavior of the Yuragi-based VNE method with a simple queueing model for delay behavior. However, delay behavior is more complicated and difficult to identify in SDI. Thus, the system needs to operate under uncertain situations. In this chapter, we examine a more complicated model of end-to-end delay and show that the proposed method can sustain its adaptability when several delay behaviors are present.

The rest of this chapter is organized as follows. In Sec. 3.2, a service model for SDI frameworks is introduced and related works on VNE are referenced. The method based on the Yuragi principle is proposed in Sec. 3.3, and the results of performance evaluation are shown in Sec. 3.4. Finally, the conclusion of this chapter and future work are presented in Sec. 3.5.

3.2 Virtual Network Services in SDI Frameworks

In this section, we describe SDN frameworks and explain a service model for SDI frameworks. First, a whole system of the virtual network service is explained. Next, VNE, one of the important problems for SDI service, is described.

3.2.1 SDI

Figure 3.1 shows a service model of SDI frameworks. In the model, customers request a VN from their service providers. The VN request includes topology information, which is a set of virtual nodes and virtual links. Then, the provider assigns computing resources for the virtual nodes by preparing virtual machines. Then, the provider configures the packet forwarding rules on the network switches via SDN controller to form virtual links.

The customers can specify the performance and capacity requirements, such as the CPU power of a virtual node and the bandwidth of a virtual link. They may also specify memory capacity (RAM), storage capacity (HDD), and in some cases, specify the detail of restrictions: the operating system (OS) of the virtual machine, the RAID type of storage, and the RAM type of a switching device. We call these specifications of virtual nodes the “node attributes”. Note that node attributes do not correspond one-to-one to server resources but do correspond one-to-one to a combination

of server resources and some specific constraints. For example, node attribute A might express a requirement for a high-performance computing server (with a high number of CPUs and a large amount of memory), attribute B might express the need for a cloud file server (with big storage disks), and attribute C might express the need for several kind of servers with other specific constraints (e.g., required some geographical restriction).

The service provider has a network manager to handle VN requests. The network manager plays three roles. First, the network manager receives VN requests from customers and pushes them into a queue. Second, the network manager executes a certain VNE algorithm for each VN request in the queue in first-in-first-out (FIFO) order. The VNE algorithm decides a VN mapping (i.e., a virtual node mapping and virtual link mapping). Virtual node mapping decides the location of the physical node for each virtual node. Then, the virtual node is hosted on the physical node as a virtual machine. Virtual link mapping decides the path on the physical network for virtual links between virtual nodes. Then, the virtual nodes are connected through the path. When the VNE algorithm fails to find a VN mapping due to a shortage of physical resources, the VN request is rejected. Next, the network manager offers the mapping request to the SDN controller. Note that the SDN controller might be managed by other organizations, such as infrastructure providers, rather than the service provider. Then, the service provider installs virtual machines into physical servers and allocates the requested computing resources. Then, the SDN controller accesses the substrate nodes via some protocol (such as OpenFlow) and reconfigures the forwarding rules to establish the virtual links.

3.2.2 The Virtual Network Embedding Problem

VNE is one of the important problems in allocating physical resources in response to a VN request. The physical resources, including resources of the physical network and resources of physical servers, form a substrate network. OpenStack, which is one of the most general infrastructure-as-a-service (IaaS) frameworks, defines virtualized resource components [20]. The substrate node is classified into three types: computing servers, network switches, and storage. Each virtual node may have individual features, such as supported OS, protocols, and storage types. It is necessary

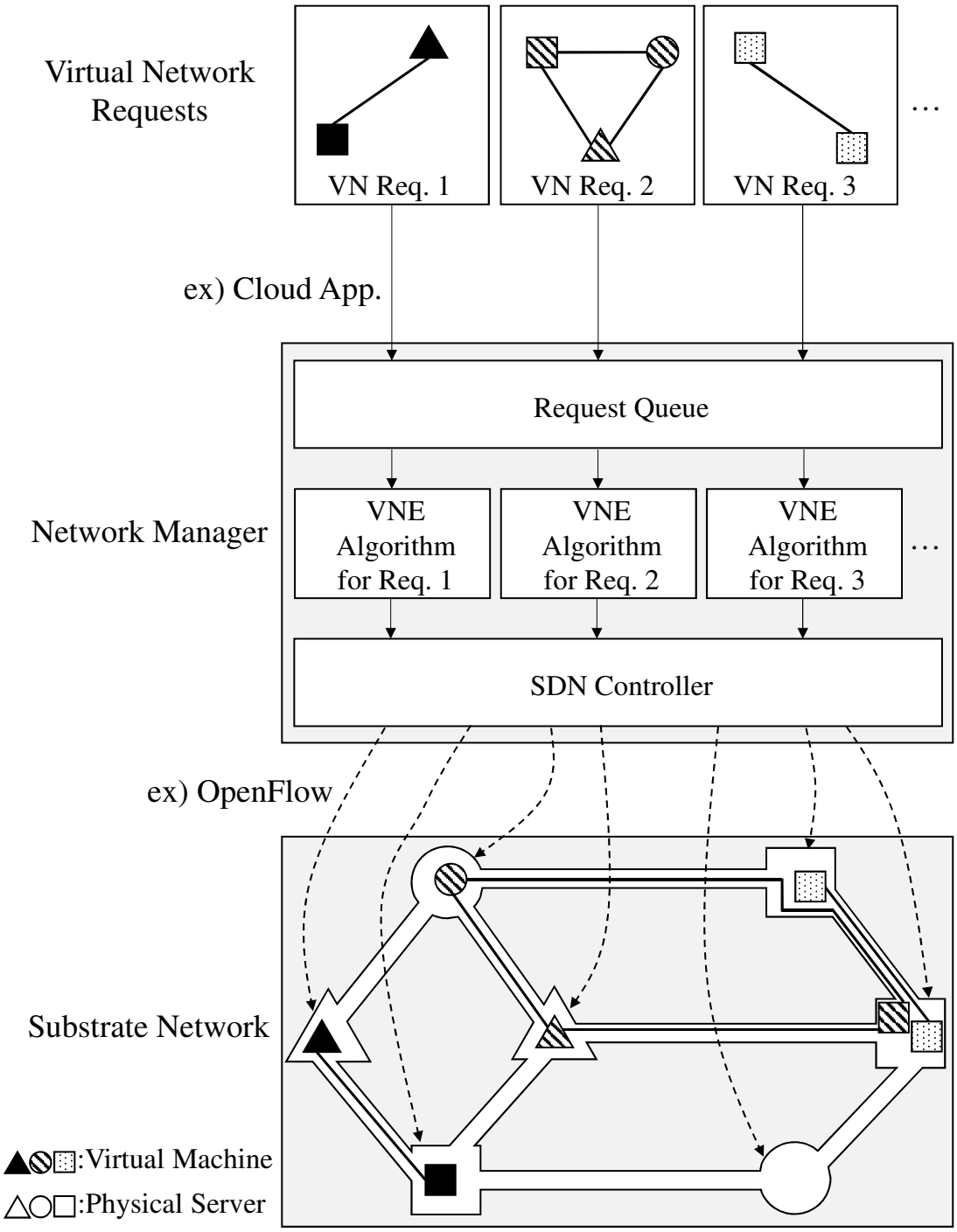
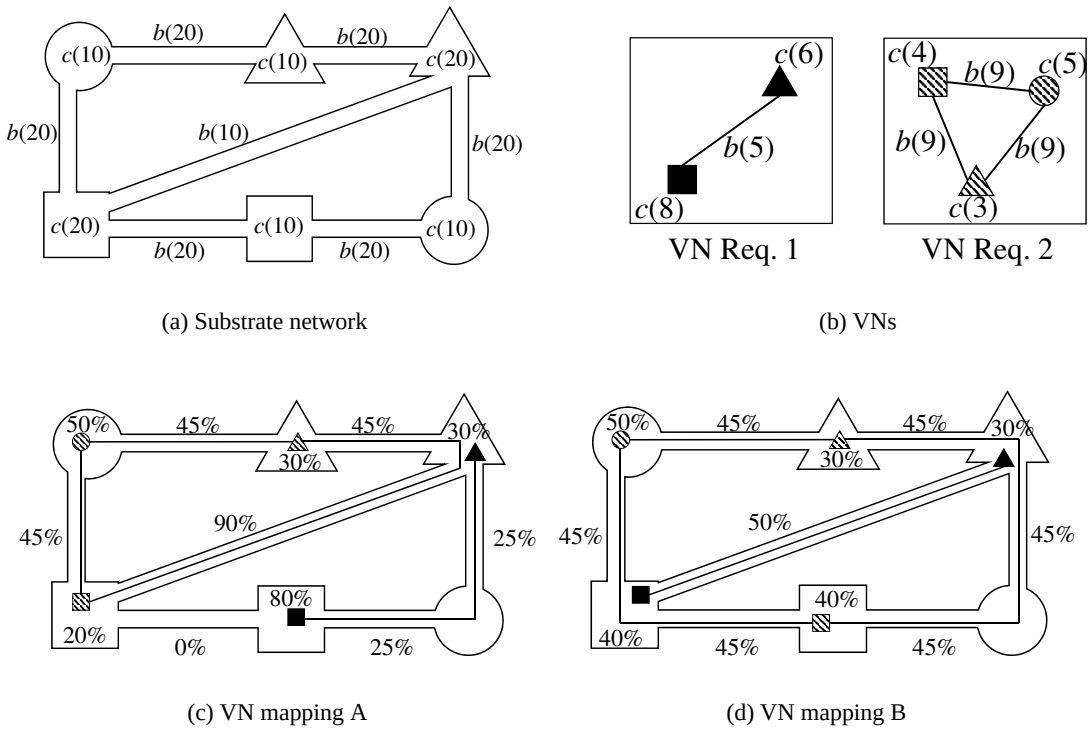


Figure 3.1: Service model in software-defined infrastructure

to strictly check the consistency of the node features when embedding a virtual node to a substrate node. That is, the requested features of the virtual node must be supported by the substrate node. To simplify the service model, this chapter abstracts the classifications of features of OpenStack into "node attributes."

The mapping of the virtual network has an effect on many aspects, such as resource utilization, blocking rate, revenue, QoE, energy efficiency, and migration cost. That is why the VNE problem deserves consideration. Figure 3.2 shows an illustrative example of how the experienced delay of VNs differs depending on the mapping of the virtual network. Figure 3.2(a) shows a substrate network including resource capacities. Figure 3.2(b) shows VN requests including resource requirements. The numerical values $c(\cdot)$ and $d(\cdot)$ in the figure represent the number of CPUs on the node and the bandwidth of the link, respectively. Figures 3.2(c) and 3.2(d) show two patterns of VN mapping, denoted as mapping A and B, respectively. In general, the delay of a server is longer when the CPU utilization is higher, and the delay of a link is longer when the link utilization is higher. In the case of mapping A, the CPU utilization on one of the substrate nodes reaches 80% and the calculation delay gets longer. However, in the case of mapping B, the CPU utilizations are at most 50%. As for the delay on a link, the maximum link utilization of the substrate link is 90% in the case of mapping A. In the case of mapping B, the link utilizations of the substrate links are low, and so no additional delay will be introduced. Therefore, the experienced delay under mapping B is expected to be shorter than that under mapping A. Thus, between the two mappings, mapping B is the preferred solution of the VNE problem.

Generally, the VNE problem is divided into two phases: virtual node mapping (VNoM) and virtual link mapping (VLiM). The goal of VNoM is to obtain a matching between virtual nodes and substrate nodes under the constraint that the substrate node must support the node attribute of the matched virtual node. The goal of VLiM is to obtain a set of links in the substrate network that connects one virtual node to another virtual node.



(a) Substrate network

(b) VNs

(c) VN mapping A

(d) VN mapping B

Figure 3.2: Comprehension of VNE problem with a simple example

3.2.3 Centralized Approaches for VNE

A number of approaches to coping with VNE problem have been proposed. Most of them try to formulate and solve optimization problems and maximize/minimize some performance objectives. However, existing VNE formulations typically use integer linear programing (ILP), and the VNE problem is known to be an $\mathcal{NP}$-hard problem. Thus, some heuristic methods are also developed. Note that both the ILP methods and heuristic methods assume information of the network is collected in advance.

Chowdhury et al. deal with VNE problem of embedding multiple VN requests onto a substrate network [19]. They give a formulation as mixed integer linear programing (MILP) to minimize embedding cost while achieving a balance of resource utilization. Guerzoni et al. formulate a MILP that considers node attributes to maximize the revenue while minimizing resource utilization [20]. Chen et al. present a virtual node mapping method to optimize energy efficiency, and also

propose a heuristic algorithm for this [21]. Fajjari et al. minimize the running cost of the network infrastructure by releasing the unused bandwidth of a VN for other VNs [22].

To handle the VNE problem, it has been widely considered to take optimization approaches such as ILP and its heuristic methods. It is expected that those methods will give the solution with the best objective function value. However, to compute the best performance, these optimization methods examine the detailed situation of the whole infrastructure, and in the worst cases, the network will be congested with an increasing volume of traffic related to control messages for collecting the details of the situation [15]. The overhead of gathering such global information becomes a fundamental limitation to adopting the optimization approach in SDI because the orchestrater needs to manage a huge number of multiplexed VNs and highly dynamic requests. To avoid this problem, control methods driven by a small amount of knowledge of the situation are required.

3.3 Yuragi-based Virtual Network Embedding Method

This section proposes a Yuragi-based VNE method for SDI frameworks. The Yuragi principle, which is often called an attractor selection model, explains the biological adaptability. The key concept of attractor selection models is that systemic behavior is governed by a single value, called "activity," and a small perturbation, which we call "Yuragi". The activity is a kind of "comfortableness" for the system, and via feedback of the activity and small perturbations, the control state of the system falls into a comfortable state. When activity is high, the control state of the system is in a good condition and stays in that state. Such an equilibrium point is called an "attractor". When activity becomes low or the condition becomes uncomfortable due to environmental changes, the system gets out of the attractor, i.e., escapes the basin of attraction (hereinafter, the attractor structure), and then looks for another attractor via feedback of the activity and small perturbations.

The proposed VNE method is expected to enjoy the adaptability of Yuragi to environmental changes. That is, VN migrations are driven according to experienced performance and the new VN mapping is obtained by means of attractor selection. A process of the Yuragi-based method is executed for each VN request. Thus, multiple processes are executed in parallel to deal with multiple VN slices. Different from optimizing problems and related heuristics, the Yuragi-based

method can avoid the necessity of collecting detailed information about the entire network. The process for a VN request needs only enough information for comfortableness and does not need any information related to other VN requests.

3.3.1 Yuragi Principle

The Yuragi principle is the principle that biological organisms use to adapt to environmental fluctuations. Attractor selection is a model that represents the Yuragi principle. The model describes the dynamics of state variables x_i $(i = 1, 2, \ldots, n)$ through environmental fluctuations as

$$\frac{d\mathbf{x}}{dt} = \alpha \times f(\mathbf{x}) + \eta, \tag{3.1}$$

where $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$ represents the system state, activity α is the comfortableness of the present system state, $f(\mathbf{x})$ defines deterministic behavior governed by the attractor structure, and η represents stochastic behavior. When the system is in a comfortable state, and hence activity α is high, the deterministic term $f(\mathbf{x})$ controls the dynamics while the noise η is almost negligible. When the system condition gets worse and α gets close to zero, $f(\mathbf{x})$ is no longer influential and the stochastic term η becomes relatively dominant. Therefore, the system changes its state at random and searches for another attractor. Once the system reaches an attractor with a comfortable activity level (though not necessarily the best possible activity level), the system will stay in the new good state. When the system reaches a state with a high activity that has not been defined by the attractor structure $f(\mathbf{x})$, the system also stays in the state and $f(\mathbf{x})$ is reconstructed to register the state as a new attractor.

A system driven by the Yuragi principle achieves adaptability to environmental changes. The adaptability has two aspects. First, the system is robust to small fluctuations in the surrounding environment. As long as activity remains higher than a certain level, the system keeps staying at an equilibrium point even though the noise term is still present (see the right-hand side of Fig. 3.3). Second, the system has flexibility in responding to drastic changes in the environment. When the system falls into an uncomfortable state, the activity decreases immediately and the dynamics of the system behavior escapes from the attractor structure (see the left-hand side of Fig. 3.3).

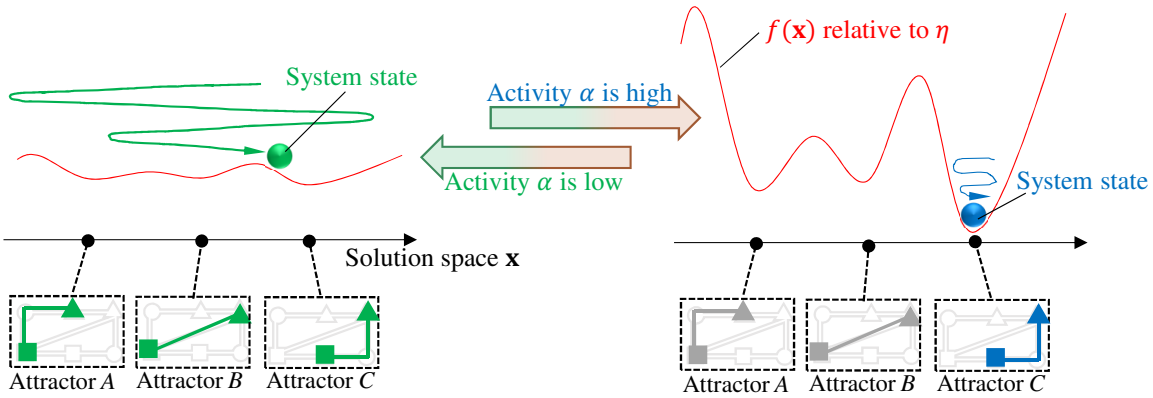


Figure 3.3: An illustration of the Yuragi mechanism

3.3.2 Performance Objectives

We can select various definitions of the activity when the Yuragi principle is applied to the VNE problem. The Yuragi-based VNE method tries to find a system state that maintains high activity. The high activity should be designed so that the performance objective does not violate a required threshold.

Conventional works usually consider link utilization [54] and/or energy consumption [55] as performance objectives because these can be described as a linear function of traffic load. Note that linearity in a mathematical sense is one of the key factors to solving the optimization problem. In this chapter, we focus on experienced delay, which is the end-to-end delay on the VNs, consisting of the communication delays between VMs and processing delays on VMs. Experienced delay is thus not necessarily a linear function of the performance objective (comfortability) because experienced delay is one of the simplest and most fundamental performance objectives in networking. It is true that link utilization is often used as the performance objective of VN control. However, experienced delay is a more important measure in networking and is especially important for SDI frameworks. A longer delay can cause considerable degradation in QoE of an application running on the VN. Thus, customers of SDI services want to require a low delay to the infrastructure provider. Nevertheless, conventional approaches usually have to minimize utilization or workload instead of delay for the objective function because of the difficulty of modeling the experienced end-to-end delay, which is composed of complicated factors. Moreover, under virtualization environments, the delay is caused

by not only utilization of the network bandwidth but also workloads on the VMs, and the delay becomes extremely long under heavy workloads [56, 57]. The end-to-end delay in SDI frameworks comprises delays in networks and delays on servers. The processing delays on servers depend on multiple factors. Thus, exact analyses and estimation of the software processes are indispensable for calculating optimization problems, but these are difficult in general [58]. Even when we have a good model, the network manager may deal with non-linear optimization problems that are tough to solve even by offline computation. Therefore, it is difficult to deal with delay requirements in conventional approaches. Instead of applying optimization with some sort of delay model, an online control approach is needed. The online approach measures the actual delay continuously. When the measured delay does not satisfy requirements, the network manager reconstructs the VN mapping immediately. In this way, the online approach avoids calculation of complicated optimization problems. Of course, the network manager must obtain a VN mapping solution quickly enough to control the VN. In this chapter, we consider the end-to-end delay with applying the Yuragi principle, and confirm by simulation (with a topology of 50 nodes) that the calculation of the proposed Yuragi-based method terminates within a few seconds.

3.3.3 Yuragi-based VNE Method

This section explains our Yuragi-based VNE method. Our proposed method consists of two phases: attribute-aware virtual node mapping and shortest-path virtual link mapping. The relation between state variables in the Yuragi principle and the VNE problem are explained first.

Our method decides where to allocate a virtual node with attribute a. In other words, the method finds a coupling between attribute a and physical node n. Let the number of attributes be A and the number of physical nodes be N. We prepare variables $\mathbf{x} = (x_1, \ldots, x_{an}, \ldots, x_{AN})$. A variable x_{an} is a decision variable that designates whether physical node n is a candidate for virtual node with attribute a. Then, the dynamics of each x_i $(i = 1, 2, \ldots, AN)$ is described as

$$\frac{dx_i}{dt} = \alpha \left\{ \varsigma \left(\sum_j W_{ij} x_j \right) - x_i \right\} + \eta, \tag{3.2}$$

where $\varsigma\left(\sum_j W_{ij}x_j\right) - x_i$ represents a deterministic term and η is a stochastic term. In the first term, the matrix $\mathbf{W}$ represents an attractor structure (discussed later). The function $\varsigma(z)$ is a sigmoid function defined as

$$\varsigma(z) = \tanh(\frac{\mu}{2}z), \tag{3.3}$$

where μ represents the gradient in the vicinity of the threshold. Here, the threshold is 0, and the output value of $\varsigma(z)$ gets close to 1 or -1. Note that the range of x_i is $[-1, 1]$. The second term η in Eq. (3.2) is a random value following a normal distribution. If $x_i > 0$ and i's corresponding node (resp., attribute) is n (resp., a), then physical node n is a candidate for a virtual node with attribute a. If $x_i\ (= x_{an}) < 0$, the virtual node with attribute a is not embedded to physical node n. Each of the virtual nodes with attribute a is allocated onto one of the candidate nodes in descending order of x_{a*} values. Note that, when physical node n is not compatible with attribute a due to the attribute restriction, x_{an} is set to 0 without calculating the differential equation (3.2).

Finally, our method assigns the shortest path for each virtual link request. In this chapter, we consider shortest-path routing to minimize hop length on the physical topology. Other routing policies can be applied, but this is not examined in the evaluation in Sec. 3.4.

Activity Function with Performance Profile

Activity α is feedback from the system and reflects the comfortableness of the VN. Let p be an objective metric, expected to be small. Activity is described as,

$$\alpha = \frac{\gamma}{1 + \exp(\delta(p - \theta))}, \tag{3.4}$$

where γ represents the scale of the activity value and δ represents the gradient around the threshold θ. Let γ be 1, to which the activity value gets close if $p < \theta$. Otherwise, the activity becomes 0. Note that the activity is subject to be reduced to 0 regardless of Eq. (3.4). Letting V_a be the number of virtual node requests with attribute a, the activity α is reset to be 0 when the number of candidates $|x_{a*}|\ (s.t.\ x_{a*} > 0)$ is less than V_a. This is necessary because the system state found by Yuragi does not have a sufficient number of candidate nodes. Also, when the available capacity of

a physical resource is not enough to embed the found system state, α is forced to 0.

In our method, the objective metric p can be directly monitored. However, when the monitoring incurs some overhead or it is difficult to monitor p directly, the activity should be calculated by estimating p rather than finding p exactly. For the estimation, we consider making use of a performance profile. The profile database consists of the correspondences between delay and resource utilization based on a history and is maintained in some form (typically, as a table).

Attractor Structure

The matrix $\mathbf{W}$ in Eq. (3.2) represents an attractor structure. It stores some equilibrium points of a virtual node mapping, and the equilibrium point is called an attractor. Each attractor is defined as $\mathbf{y} = (y_1, \ldots, y_i, \ldots, y_{AN})$, where $y_i \in \{$-1, 0, 1$\}$. If physical node n is one of the candidates for a virtual node with attribute a, then y_{an} is set to 1. If node n cannot allocate attribute a due to the node attribute restriction, then y_{an} is set to 0. Otherwise, y_{an} is set to -1. Letting M be the number of attractors stored in $\mathbf{W}$, a set of attractors $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M)$ can be stored by

$$\mathbf{W} = \mathbf{Y}^{+}\mathbf{Y}, \tag{3.5}$$

where $\mathbf{Y}^{+}$ is the pseudo inverse matrix of $\mathbf{Y}$. This way of storing attractors uses the knowledge of Hopfield neural network of associative memory [59]. When the present state is in one of the attractors, $d\mathbf{x}/dt$ in Eq. (3.2) becomes close to 0 and stays in the attractor.

3.3.4 VN calculation

When the activity gets extremely low, that is, when the observed end-to-end delay exceeds the performance objective value θ, the network manager executes then Yuragi-based VNE method in an offline calculation by using a performance profile. The performance profile enables estimating performance without running the services on actual infrastructure. Thus, the service continues to run with the extant VN while calculating a new VN. Once the Yuragi system converges to a good VN mapping, the network manager is ready to enter the VN migration phase.

3.3.5 VN migration

The network manager migrates each VM that needs to be transferred for a new VN mapping. The process is executed according to the "make-before-break" principle to reduce service downtime:

Step 1 Copy the VM image from the source node to the destination node. Note that the service is still running on the source node.

Step 2 Boot a VM on the destination node.

Step 3 Copy the state differences between VMs (typically implemented as "dirty pages") to the destination node recurrently.

Step 4 Suspend the VM at the source node and copy the remaining state differences from source to destination.

Step 5 Switch the traffic flow to the destination node and resume the VM on the destination node. Then, the service is running on the destination node.

Step 6 Delete the VM on the source node.

The service may be suspended during the time to copy the state differences in Step 4. Thus, the service downtime is shortened to only hundreds of milliseconds [60]. Note that, the system needs to make particular provision to guarantee the user experience of specific types of applications, such as more real-time oriented services (e.g., as voice or video).

3.4 Evaluation by Computer Simulation

This section presents the results of evaluating the Yuragi-based VNE method by computer simulation.

3.4.1 Simulation Environment

The substrate network consists of physical servers and links. The number of physical servers (physical nodes) is 50. Each node has the capability to host virtual nodes with one of the node attributes.

In this environment, each node has three kinds of resource capacities for required virtual machines: CPU, memory and storage capacity. These are determined uniformly within $[50, 100]$ for each node. For each pair of physical nodes, a physical link is randomly established between the nodes with probability 50%. As a result, we obtained a physical topology with 50 nodes and 617 links. The (integer) capacity of physical links is determined uniformly randomly within $[50, 100]$. During the simulation, the substrate network is fixed.

Several requests of virtual network are generated and arrive. During the simulation, the number of VN requests is set to 20 and the number of node attributes A is set to 4. Each VN request is generated as follows. The number of virtual machines (virtual nodes) is determined uniformly randomly within $[2, 5]$. Each virtual node belongs to an attribute, and each virtual node requires capacities for CPU, memory, and storage. Each of the required capacities is determined uniformly randomly within $[1, 10]$. Virtual links are undirected, and each pair of virtual nodes is randomly connected through a virtual link with probability 50%. The number of virtual links is within $[1, 10]$ because the number of virtual nodes is 2 to 5. Each virtual link has a required bandwidth, and the required capacity is determined uniformly randomly within $[1, 25]$.

Every 100 time steps, all 20 VN requests are regenerated in the same way as described above. In addition, at every 10 time steps, each VN request fluctuates with relatively small changes: we change the requested capacities by a random integer, which is obtained by rounding a value following the normal distribution with $\mu = 0$ and $\sigma^2 = 1$ (see Fig. 3.4). The service downtime caused by VN migration is regarded as negligible in the following simulation.

Table 3.1 summarizes the parameters in the simulation environment.

3.4.2 Delay Profile

We use end-to-end delay as the objective metric. In an actual environment, the experienced delay may be available by monitoring of packet arrivals. However, when the monitoring incurs some overhead or it is difficult to monitor directly for some reason, the activity should be calculated by an estimated p rather than actually measuring p. For the estimation, we consider the use of a performance profile. For the performance profile, we prepare several delay models as a function

Table 3.1: List of variables and values in the simulation

Variable	Value
Substrate network:	
- Number of nodes N	50
– Number of attribute A	4
– CPU number	[50, 100]
– Memory capacity	[50, 100]
– Storage capacity	[50, 100]
- Number of links	617
– Link capacity	[50, 100]
VN requests:	
- Number of VN	20
Each VN request:	
- Number of nodes	[2, 5]
– CPU number	[1, 10]
– Memory capacity	[1, 10]
– Storage capacity	[1, 10]
- Number of links	[1, 10]
– Link capacity	[1, 25]
Delay weight:	
- (w_c, w_m, w_s, w_b)	(0.25, 0.25, 0.25, 0.25) in §3.4.4. {(0.6, 0.2, 0.0, 0.20), (0.2, 0.6, 0.0, 0.20), (0.2, 0.2, 0.0, 0.60)} in §3.4.4.
Yuragi parameters:	
- $(\mu, \gamma, \delta, \theta)$	(20.0, 1.0, 2.0, 5.0)

of resource utilization, referring to bandwidth on links and the tuple (memory, CPU, storage) on servers. Figure 3.5 shows the delay models. The first model simulates delay caused by bandwidth utilization in a link. A basic M/M/1-based model of delay in networks is used. The second delay model simulates memory utilization on the server and imitates the response time of an Apache web server. As shown in Fig. 3.5, the second model is characterized by the multi-stage elevations of delay. In the case of web service, such elevations are caused by swapping memory pages and storage disks. The last model simulates CPU utilization. The delay increases linearly as the resource utilization becomes high, except at extremely high utilization, where the delay increases rapidly. Such an increase in delay is caused by conflicts among VMs demanding more calculation power

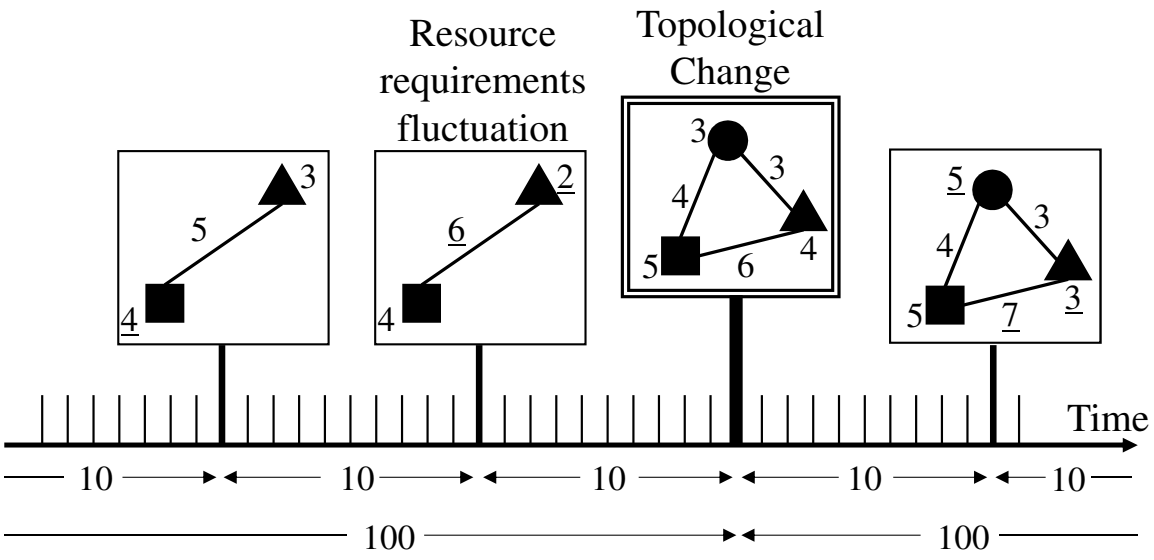


Figure 3.4: Environmental fluctuations over elapsed time

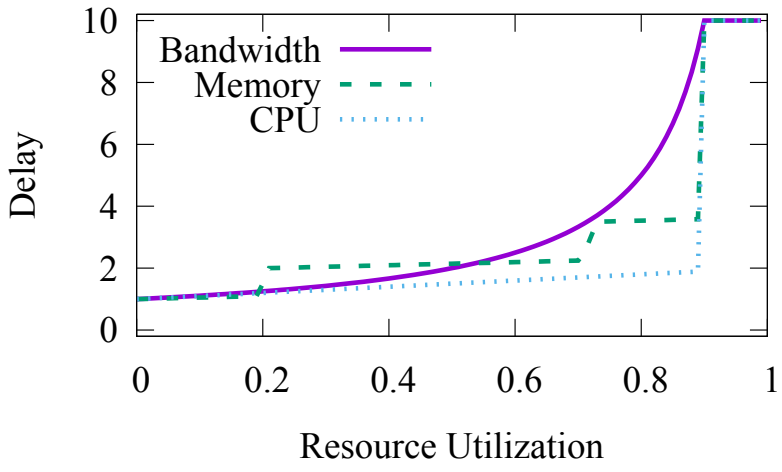


Figure 3.5: Delay models used for computer simulation

than the CPUs can provide.

In the following simulation, d_{ij}, which represents the delay from virtual node i to virtual node j, is calculated as,

$$d_{ij} = w_c \sum_{n \in R_{ij}} d_n^c + w_m \sum_{n \in R_{ij}} d_n^m + w_s \sum_{n \in R_{ij}} d_n^s + w_b \sum_{l \in L_{ij}} d_l^b, \tag{3.6}$$

where the set R_{ij} consists of the physical nodes along the route from i to j, and the set L_{ij} consists of the physical links. Then d_n^c, d_n^m, and d_n^s are the computing delay in virtual machine n according to CPU, memory and storage, respectively. In this, d_l^b is the delay through physical link l, and w_c, w_m, w_s, and w_b are weight parameters. Each d_n^* and d_l^b follows the delay model and has a value calculated by its own utilization of physical resources.

Network managers maintain delay profiles in which correspondences between resource utilization and actually measured delays are recorded. Referring to a delay profile makes it possible for the manager to estimate delays in the VN when consider a VN request to be embedded. In the simulation environment, delays are calculated with the same delay models as the delay profile. Note that, in actual usage, the delay can be easily obtained by referring to the timestamps of packets.

3.4.3 Heuristic Method for Comparison

As for the benchmark of our method, a heuristic VNE method is also simulated in the same environments. The heuristic method has two phases: virtual node mapping based on a greedy algorithm [61] and virtual link mapping on shortest paths. Note that we do not aim to obtain a better end-to-end delay than that provided by the greedy algorithm. We intend to obtain a reference delay to confirm that our method can find a comfortable state by using the noise-induced search, rather than simply by using low-traffic settings. The VNE method executes the following algorithm for VN requests, acting sequentially.

(1) Execute the following processes for each virtual node v.

- **(1.1)** Find the set $P(v)$ of physical nodes that accept the attribute of v and have enough unreserved resource capacities to embed v. When $P(v)$ is null, reject the VN request and finish.
- **(1.2)** Find the physical node that indicates the highest value of I among $P(v)$, where I is defined as Eq. (3.7). Then reserve the resources of that physical node.

(2) Embed the virtual nodes according to the reservation taken in (1).

(3) For each virtual link between virtual nodes embedded in (2), find a path that is the minimum hop in physical topology. Embed the virtual links onto the paths. When a shortage of link bandwidth occurs, reject the VN request.

The greedy method aims to minimize the utilization of node and link resources. The heuristic method calculates an available resource indicator I for each physical node n, defined as

$$I(n) = C_n \times M_n \times S_n \times \sum_{l \in L(n)} B_l, \tag{3.7}$$

and avoids embedding a virtual node onto bottleneck resources. The values C_n, M_n, and S_n represent the available capacity of CPU, memory, and storage, respectively, on physical node n. The set $L(n)$ represents a set of physical links attached to node n, and B_l represents the available capacity of physical link l. The computational complexity is $\mathcal{O}(n \log n)$ for sorting $L(n)$, assuming the shortest path between every node pairs is available in advance.

3.4.4 Simulation Results

In the simulation, the Yuragi-based method calculates the VN mapping at each time step and migrates the VN until the system state converges to an attractor. The threshold of activity θ in Eq. (3.4) is set to 5.0, regarding the metric p as the maximum of d_{ij} for every pair of virtual nodes i and j. The greedy method executes VN migration according to the demand changes at every 10 time steps.

Performance of Yuragi-based VNE method

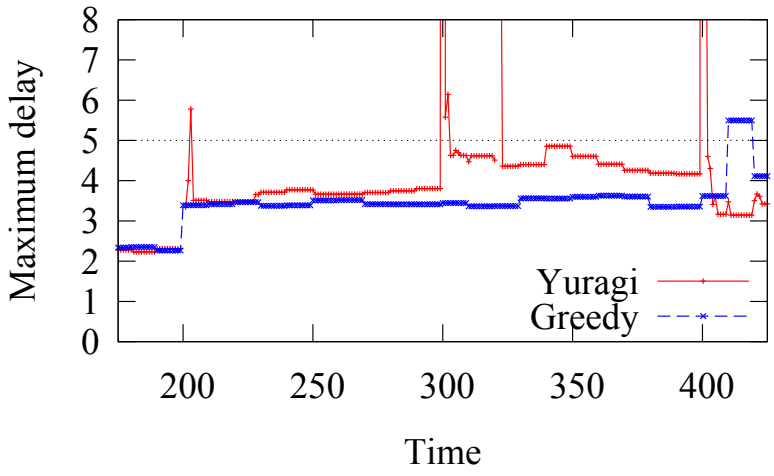
We first show the performance of Yuragi-based VNE method with a simple M/M/1-based delay model. We explorer its adaptability to fluctuations of VN requests by evaluating the end-to-end delay, the embedding ratio, and the number of VN migrations. Here, the weight values in Eq. (3.6) are set as $w_c = w_m = w_s = w_b = 0.25$ with an M/M/1-based delay model. We will examine other weight values in Sec. 3.4.4.

Figure 3.6(a) shows the maximum delay on a VN request out of the 20 requests. The activity

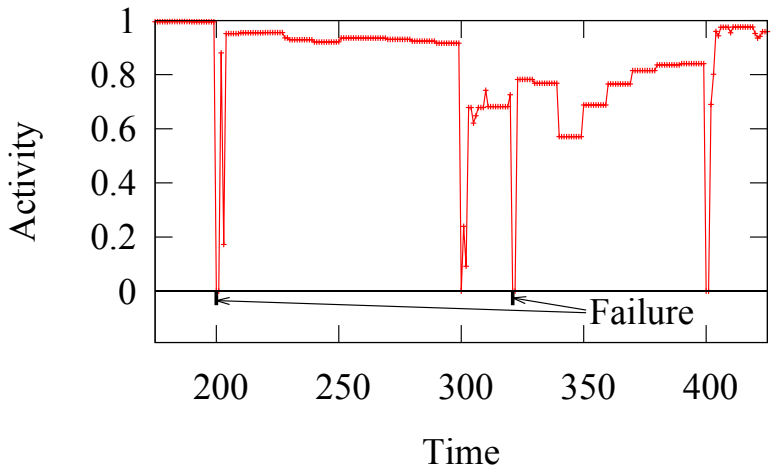
of each VN is also shown in Fig. 3.6(b). In the figure, the region denoted as "Failure" represents a failure of embedding the VN caused by a shortage of physical resources or violation of other restrictions. Note that the demands of VN requests fluctuate with relatively small changes at every 10 time steps and the demands fluctuate greatly at every 100 time steps. Thus, the maximum delays for the Yuragi-based method exceed the threshold drastically at every 100 time steps owing to the topological changes in VN requests. The activities drop sharply, and then the VN migration starts. Within a few steps, the activities are recovered and converge to another system state. Against a small fluctuation of required capacities, occurring at every 10 time steps, VN migrations occurs only if the activities decrease sharply as seen, for instance, in time step 320 in Fig. 3.6(a). Figure 3.7 shows the mean of the maximum delay of 20 VN requests. Considering the mean of maximum delay of the 20 VN requests, the Yuragi-based method does not achieve a delay as short as that obtained by the greedy method in general. This is because the Yuragi-based method does not aim to minimize the delay or the resource utilization but, rather, to keep them smaller than a certain threshold. Making the threshold smaller might achieve a smaller delay but will result in a longer convergence time for finding an attractor.

Figure 3.8 shows the embedding ratio, indicating how many VN requests are accepted out of the 20 requests. The topological changes occurring at every 100 time steps cause a temporary decrease in the embedding ratio, but both of the methods keep almost 95% to 100% acceptances outside those periods.

Figure 3.9 shows the number of VN migrations, defined as the number of VNs whose location has been changed from the previous operation. Note that the operations are performed at every step by the Yuragi-based method and at every 10 steps by the greedy method. When the VN requests are regenerated at every 100 steps, almost all VNs are migrated for both methods. The simulation result shows that the Yuragi-based method takes fewer VN migrations in response to small fluctuations to maintain the performance objective, and therefore our method costs less in terms of VM migration. The greedy method migrates 2 to 11 VNs at each change to maintain the required capacity. This is because the greedy method tries to achieve better objective values, even when the improvement in delay is marginal. Note that we may develop a greedy method that requires fewer VN migrations with some additional constraints or considerations. The key point is that the



(a) Maximum delay



(b) Activity

Figure 3.6: Maximum delay and activity on a VN

greedy method makes drastic changes due to the nature of the optimization, whereas the Yuragi-based method does not. The total number of VN migrations required by the Yuragi-based method is 153 for the 250 time steps of simulation, and 215 by the greedy method. This result indicates that the Yuragi-based method adapts to demand fluctuations with about 29% fewer VN migrations than

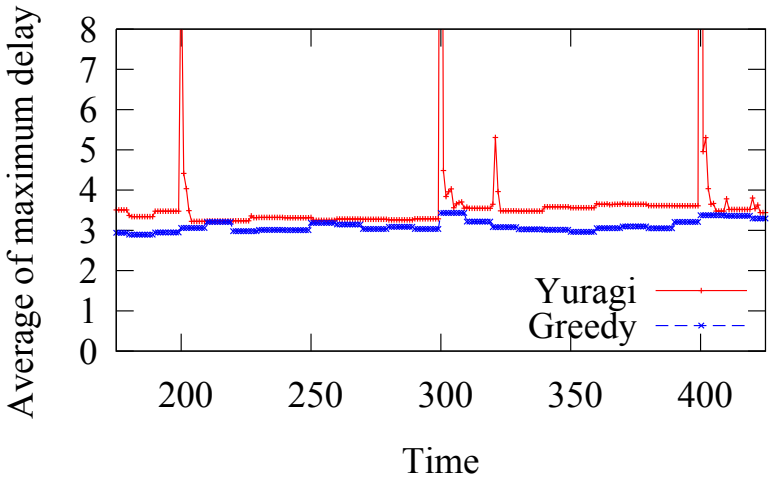


Figure 3.7: Average of maximum delay for 20 VNs

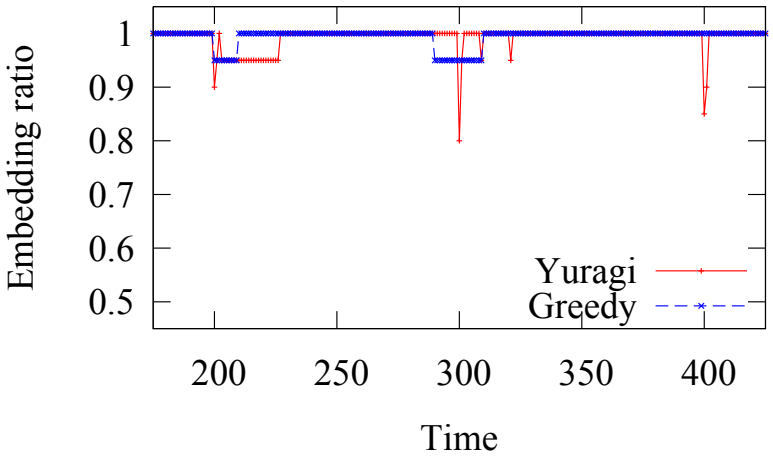


Figure 3.8: Embedding ratio of VN requests

the greedy method. For the small fluctuations occurring at every 10 time steps, the number of VN migrations with the Yuragi-based method is 33, against 156 with the greedy method.

Adaptability to different delay behaviors

In the previous section, we used the simple M/M/1 delay model where the delays on CPU, memory, and storage are all estimated by their resource utilization. However, in actual SDI environments,

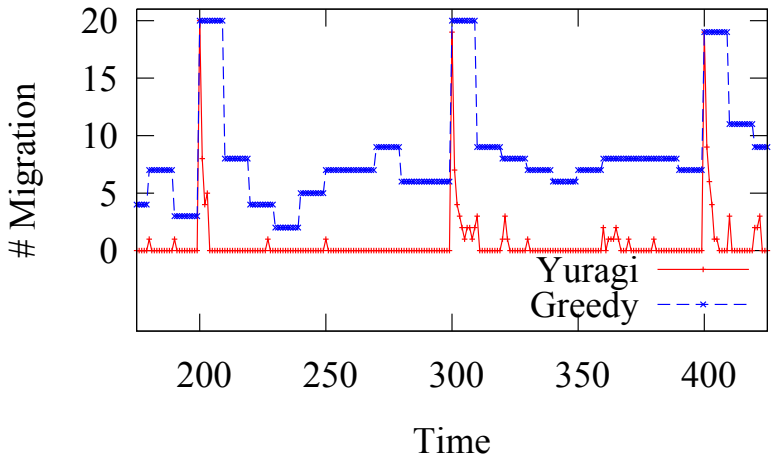


Figure 3.9: The number of VN migrations

the end-to-end delay behaves in a more complicated way, depending on multiple factors such as the processing delay (which also depends on the server specification) and memory utilization on virtual machines. Here, we demonstrate that the Yuragi-based VNE method has adaptability under various types of delay behaviors. We run computer simulations with distinct sets of weight parameters (w_c, w_m, w_s, w_b). The simulation results show that our proposed VNE method can achieve its performance objective even in a situation where the heuristic method fails to obtain acceptable performance.

Figures 3.10 - 3.12, 3.13 - 3.15, and 3.16 - 3.18 correspond to the simulation results for parameter sets $(w_c, w_m, w_s, w_b) = (0.6, 0.2, 0.0, 0.2)$, $(0.2, 0.6, 0.0, 0.2)$, and $(0.2, 0.2, 0.0, 0.6)$, respectively. Each figure shows the maximum delays on 3 VN requests out of the 20 requests because similar tendencies are observed on the other VN requests. In Figs. 3.10 - 3.12, 3.13 - 3.15, and 3.16 - 3.18, the Yuragi-based method mostly keeps the maximum end-to-end delay lower than the threshold of 5.0. Note that sharp increases of the end-to-end delay are observed at time steps 200, 300, and 400, but these delays are not crucial because they are caused by the change of VN request. The Yuragi-based VNE method gradually adapts to the VN requests and soon finds a good VN mapping.

The greedy method sometimes violates the performance threshold in response to some VN

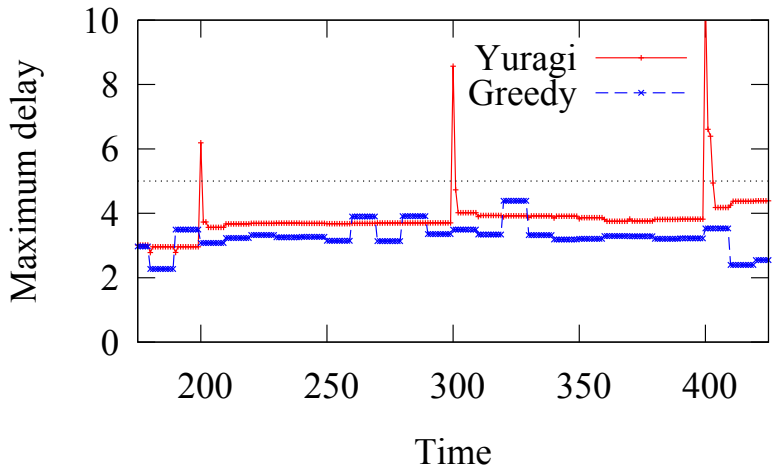


Figure 3.10: Maximum delay ($w_c = 0.6, w_m = 0.2, w_s = 0, w_b = 0.2$): VN request 1

request fluctuations (see, for example, at time step 320 on VN request 1 in Figs. 3.16 - 3.18). The greedy method, a heuristic for optimization, does not always achieve the lowest end-to-end delay. More importantly, the violation cannot be solved and may continue for a while because the greedy method has already optimized its objective and has no way to improve the performance. Those violations of the threshold occur due to a gap between estimated delays and actual delays (i.e., due to the lack of a precise delay model). Note that optimization approaches (including heuristic approaches) will always have such gaps unless they use a precise delay model. Especially in an SDI framework, defining a delay model to decrease the gap gets more difficult because the model depends on complicated factors.

As for the Yuragi-based method, it shows its adaptability under uncertain delay behaviors. An advantage of the method is that it finds a VNE solution with direct measurements of end-to-end delay, where the models of delay behaviors are not used in our method and thus no longer necessary. The delay models used in the simulations may not completely imitate actual delay profiles, but we believe that the Yuragi-based method is feasible even when the actual end-to-end delay behaves in more complicated or non-deterministic manner.

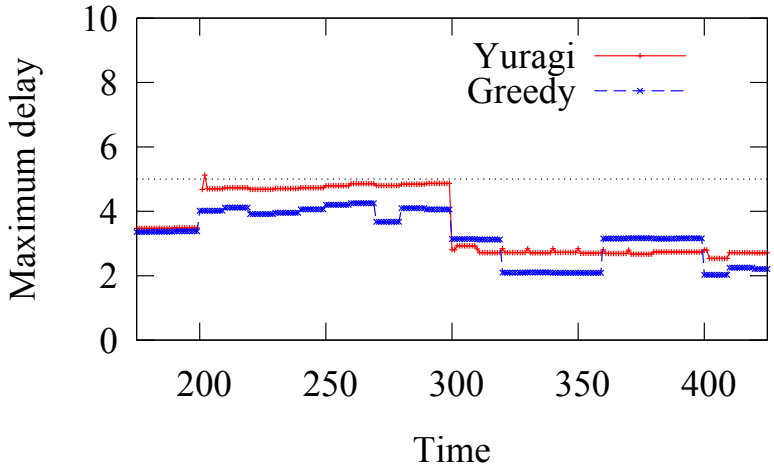


Figure 3.11: Maximum delay ($w_c = 0.6, w_m = 0.2, w_s = 0, w_b = 0.2$): VN request 2

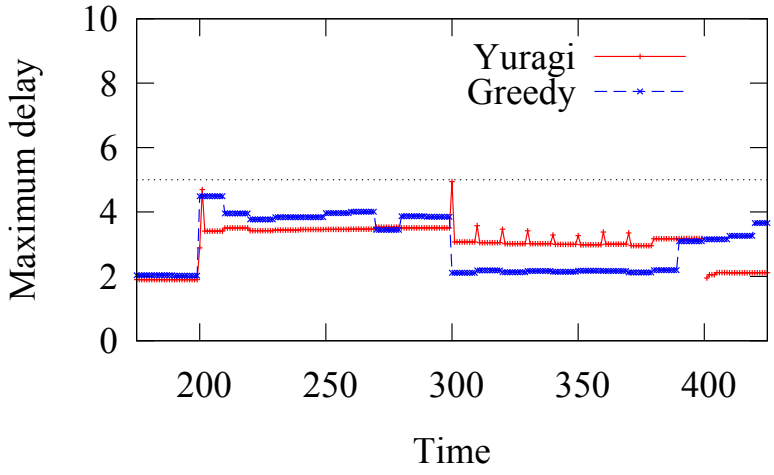


Figure 3.12: Maximum delay ($w_c = 0.6, w_m = 0.2, w_s = 0, w_b = 0.2$): VN request 3

3.5 Conclusion

This chapter presented a VNE method based on the Yuragi principle as applied to SDI frameworks. A system driven by the Yuragi principle achieves adaptability to environmental changes, and the dynamics is described as an attractor selection model. In attractor selection models, the system behavior is governed by an activity measure and small perturbations. When activity is high, the

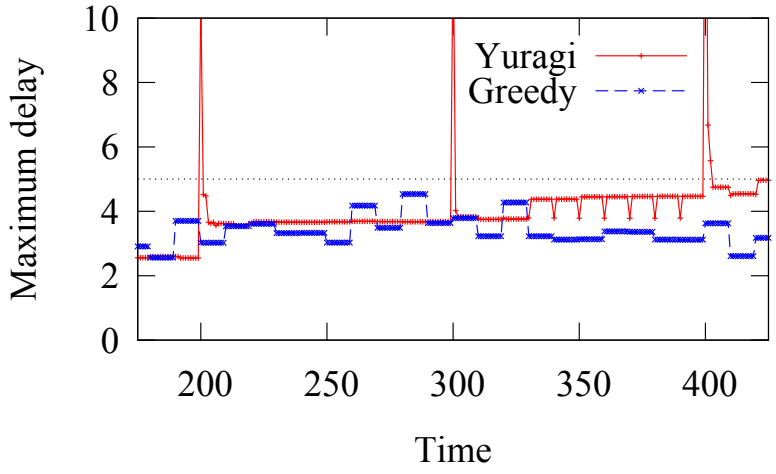


Figure 3.13: Maximum delay ($w_c = 0.2, w_m = 0.6, w_s = 0, w_b = 0.2$): VN request 1

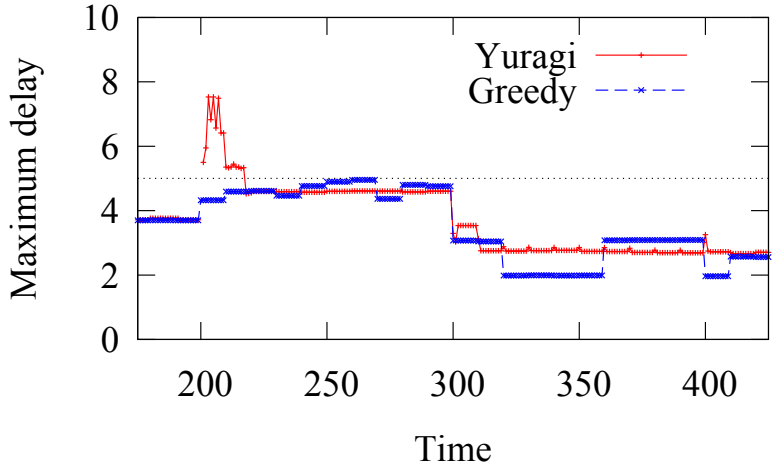


Figure 3.14: Maximum delay ($w_c = 0.2, w_m = 0.6, w_s = 0, w_b = 0.2$): VN request 2

control state of the system is in a good condition and stays in that state. When activity becomes low or the condition becomes uncomfortable due to environmental changes, the system looks for another stable state. The Yuragi-based VNE method decides the mapping of virtual nodes by means of attractor selection, where the network mapping is regarded as the system state and the activity is defined as a certain performance objective. The end-to-end delay in SDI frameworks depends on application processes and other factors. That makes it difficult to pre-estimate experienced

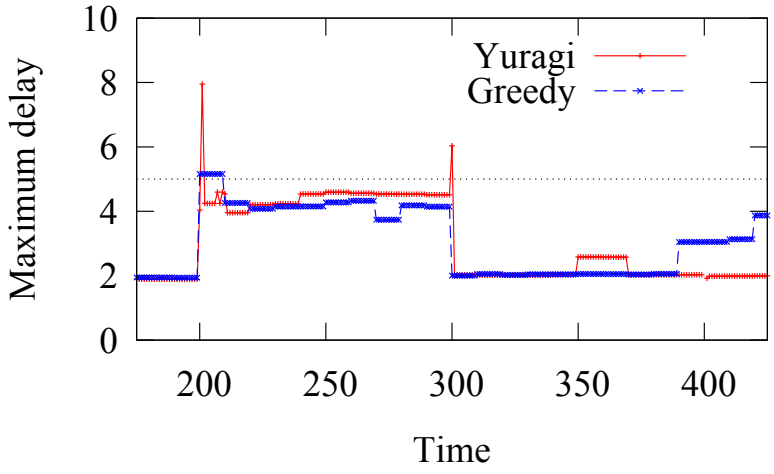


Figure 3.15: Maximum delay ($w_c = 0.2, w_m = 0.6, w_s = 0, w_b = 0.2$): VN request 3

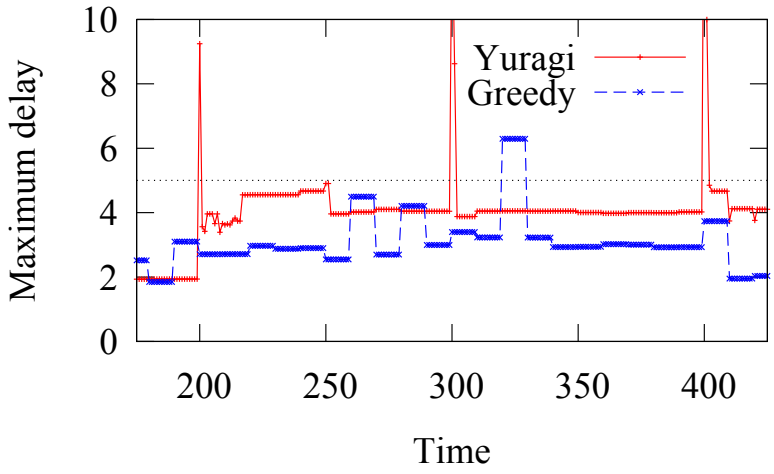


Figure 3.16: Maximum delay ($w_c = 0.2, w_m = 0.2, w_s = 0, w_b = 0.6$): VN request 1

delay accurately and causes degradation of VNE control performance. Nevertheless, our Yuragi-based method shows its adaptability under such uncertain delay conditions. In the evaluation, we considered the end-to-end delay as the activity. Simulation results show that the method provides shorter delays and adapts to the request fluctuations by rearranging the VN mapping in response to drastic changes in environments. The Yuragi-based method decreases VN migrations by about 29% relative to a heuristic method to adapt to fluctuations in required resource capacities.

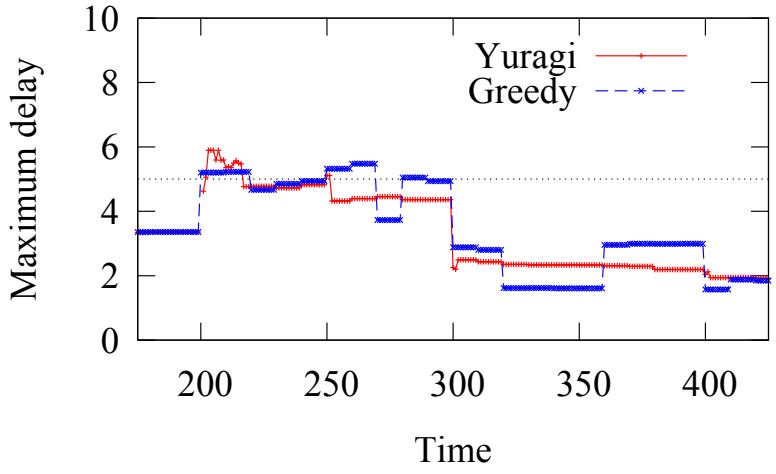


Figure 3.17: Maximum delay ($w_c = 0.2, w_m = 0.2, w_s = 0, w_b = 0.6$): VN request 2

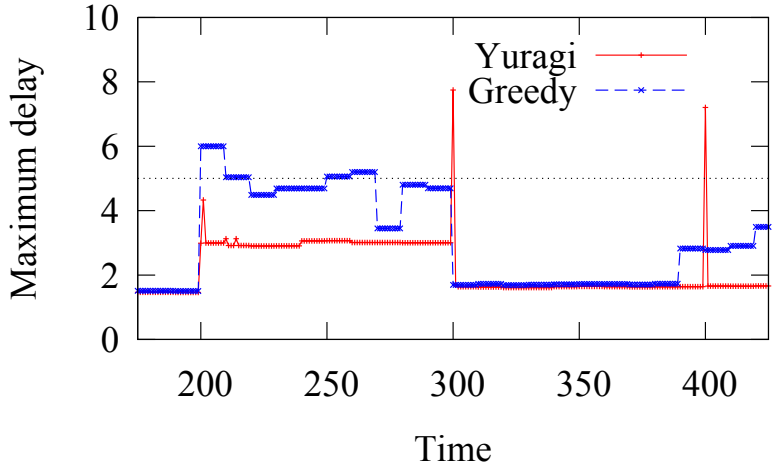


Figure 3.18: Maximum delay ($w_c = 0.2, w_m = 0.2, w_s = 0, w_b = 0.6$): VN request 3

In future work, we will investigate a method of constructing the attractor structure to improve the convergence time or some other performance measure. We suppose that our proposed method is performed in a centralized SDN controller. Recently, distributed controllers for a single infrastructure are being studied toward wide-area SDN and large-scale SDN. It is worth studying how our noise-induced method can be extended to account for mutually interfering situations. We should also demonstrate the behavior of our proposed method in real implementation. Our method will

cause a delay in the SDN controller, which is not included in the computer simulations. It is worth analyzing the impact of executing our method.

Chapter 4

Network resource planning for evolvability in software-defined infrastructure

4.1 Introduction

Building software-defined infrastructure (SDI) frameworks that can flexibly manage information network systems at low cost would make it possible to construct virtual networks with finer time granularity than before. This improvement is expected to allow more immediate response to customer requests to increase or decrease the amount of available resources. In contrast with conventional control, in which a network administrator manually configures the settings of various network devices, software-defined networking (SDN) and network-function virtualization (NFV) technologies enable flexible and responsive services (e.g., on-demand infrastructure supply and user-provisioning services) [11–15].

Virtual network embedding (VNE) control is expected to allow properly configuring virtual resource allocation in response to environmental fluctuations, such as changes in virtual resource demands, but a VNE control may not result in good virtual network performance. Such failure is caused mainly by two factors. The first factor is the VNE algorithm itself, and many

VNE algorithms have been studied with the aim of achieving better allocation of virtual resources [18–20, 22, 24–27]. The second factor is related to the physical resource design. When resource utilization levels become high, processing delays and data transfer delays will increase, resulting in worse performance of services running on a virtual network. Despite the extensive research on VNE algorithms, the design of physical infrastructure for SDN/NFV applications has been scarcely considered to date.

Although physical network resource designs have been considered in traditional communication systems, such systems aim to have adequate capacity for future states as predicted from long-term traffic observation. Indeed, physical network designs have been studied to optimize performance on the basis of current demand or a predicted future demand. For example, in IP-optical networks, ref. [28] describes the design of an optical-cross-connect topology in which the number of distinct wavelengths is minimized by knowledge of the optical path demands. Reference [29] describes a design for an optical layer network with the capacity to accommodate a predefined IP-layer topology, decided on the basis of predicted future traffic and possible failure scenarios. However, such conventional methods of designing for capacity are unsuited to SDI frameworks. A fundamental difference between capacity planning in conventional frameworks and in the SDI framework is the time granularity of changes in demand. That is, with SDI, the resource demands from various users may change over short periods. This is inherent to SDI frameworks, where virtual network configurations are executed by a softwarized control instead of by conventional manual operation. Because of this, adaptation by a softwarized VNE control becomes more important in SDI frameworks for achieving rapid provisioning of resources to meet fluctuating demands, and physical resource design is an important factor in the adaptability of VNE control. As mentioned above, algorithms for finding better VNE solutions under given resource constraints have been considered, but a strategy for choosing a physical network design that promotes VNE adaptability has not been discussed.

In contrast with virtual resource allocation, which is nearly instant, installing physical resources in an SDI framework takes considerable time and manual work. It is thus not practical to adjust the physical resources in response to every demand fluctuation. Physical resource planning requires that short-term fluctuations be managed by a dynamic VNE control. Note that drastic fluctuations should be expected to occur in the future for SDI because user requests frequently arrive through

user-friendly interfaces (e.g., graphical user interfaces) and applications are customized to be suitable for their intended purpose. A promising way to enhance the ability of the VNE control to adapt to unexpected fluctuations is to reinforce the physical resources so that the VNE control can draw on this more robust infrastructure, which makes a higher number of VNE solutions feasible. Therefore, we consider which physical resource designs will increase the diversity of feasible solutions considered by a VNE control. It is expected that providing more varied candidates for VNE solutions will enhance the robustness of VNE control against environmental fluctuations (i.e., will enhance its adaptability). Even in situations where it is difficult to predict demand changes, our design strategy aims to deploy physical resources such that the VNE control can accommodate various fluctuations in future demand.

Our design strategy for SDI system aims to achieve adaptability in the face of environmental changes by increasing the diversity of considered VNE states. As a successful biological model, we consider the evolution of populations of organisms to better fit changing environments. One key to obtaining evolutionary adaptability is to increase genotypic evolvability (i.e., the phenotypic diversity that can arise from a genetic distribution) [34, 38]. Even when the environment drastically changes, genotypic evolvability lets the system produce phenotypes that are much different from the previously dominant phenotype and ultimately settle on a phenotype that is suitable for the changed environment. In this chapter, we propose an SDI resource design strategy that increases VNE solution diversity, which originates from control system variation under demand fluctuations. The strategy imitates the evolvability of biological populations, adopting an evolutionary model that treats each VNE solution as characterizing a biological phenotype. We use this method to construct a method for reinforcing node computational capacity and conduct experiments by computer simulation to demonstrate that the adaptiveness of the VNE control will improve.

In an earlier chapter, we considered a method, based on biological evolution, that can increase the number of transceivers of IP routers in a wavelength-division multiplexing (WDM) network [30]. That method defines a correspondence between an evolution model and a WDM network and simulates the process of biological evolution (i.e., mutation and selection of gene regulatory networks through generations) with transceiver arrangement reflected in differences in

the gene regulatory network. Fitness is measured according to the performance of the virtual network control [33] (i.e., by average link utilization rate). However, the method given in our earlier chapter, which incorporates the state of resource reinforcement into the gene regulatory network, is specific to the combination of a virtual network control method and increases in the number of IP transceivers in a WDM network. In addition, that method does not consider a diverse set of potential virtual networks, so evolvability is not obtained. In the present chapter, we instead focus on improving evolvability (in the form of phenotype diversity caused by genetic mutation) and thereby contributing to improvement of environmental adaptability, analogous with biological evolution. For this, we develop an evolvability index to characterize the diversity of a VNE solution set in an SDI framework. This index is independent of the type of resource to be reinforced (e.g., node resources and link resources are treated the same), thus constructing a more general method of resource design.

In brief, this chapter contributes the following. We apply the adaptation strategy of biological evolution to SDI resource design strategy and conduct a simulation to verify its utility. The proposed reinforcement method has the following properties.

- Prediction of demand is not needed when determining the reinforcement plan. This makes it a suitable design strategy for situations in which environmental fluctuations occur frequently.
- The "evolvability" of the target network will improve. That is, the diversity of VNE solutions that can be constructed by the virtual resource control will increase.
- As a benefit of improved evolvability, the VNE control has a higher probability of adapting (via reconfiguration) to new states after drastic fluctuations in demand.

The rest of this chapter is organized as follows. In Section 4.2, we introduce a service model and the physical resource design problem in an SDI framework. In Section 4.3, we then discuss strategies for adapting to environmental changes in SDI and propose an implementation that applies biological evolutionary adaptability theory. In Section 4.4, we construct a method that reinforces the computational resources of nodes to increasing evolvability, and in Section 4.5 we evaluate its performance. Finally, we summarize the chapter in Section 4.6.

4.2 Physical resource design in SDI

4.2.1 Virtual network service in an SDI framework

Figure 4.1 shows a service model of SDI frameworks. In the model, customers request a VN from their network manager via an interface, such as a web application. The network manager executes a certain VNE algorithm for VN requests. The VNE algorithm decides on a VN mapping (i.e., a virtual node mapping and a virtual link mapping). Then, the manager installs virtual machines within the chosen physical servers and allocates the requested computational resources. An SDN controller accesses the substrate nodes via some protocol (such as OpenFlow) and configures forwarding rules to establish the virtual links.

The physical resources, including the physical nodes and physical links, form the substrate network. The physical nodes are furnished with computational resources (CPU capacity, memory capacity, storage resource quantity). The physical links are furnished with link resources (bandwidth).

A VN request includes topology information, which is a set of virtual nodes and virtual links. Virtual node mapping assigns each virtual node to a physical node. Then, each virtual node is hosted on its assigned physical node as a virtual machine. Virtual link mapping decides the path through the physical network to be used by virtual links between virtual nodes. The virtual nodes are connected through these paths.

4.2.2 Physical resource planning problem

Physical nodes and links have limited capacities. Thus, in allocating virtual resources to VNs, constraints on resource utilization at physical nodes and physical links leads to degradation in the throughput and service performance of VNs. It may also happen that some VNs cannot be accommodated because not enough resources are free. To avoid such situations, it is important to have appropriate amounts of physical resource capacity at appropriate places in the substrate network. Physical resource reinforcement (i.e., adding additional computational resources or bandwidth) is a

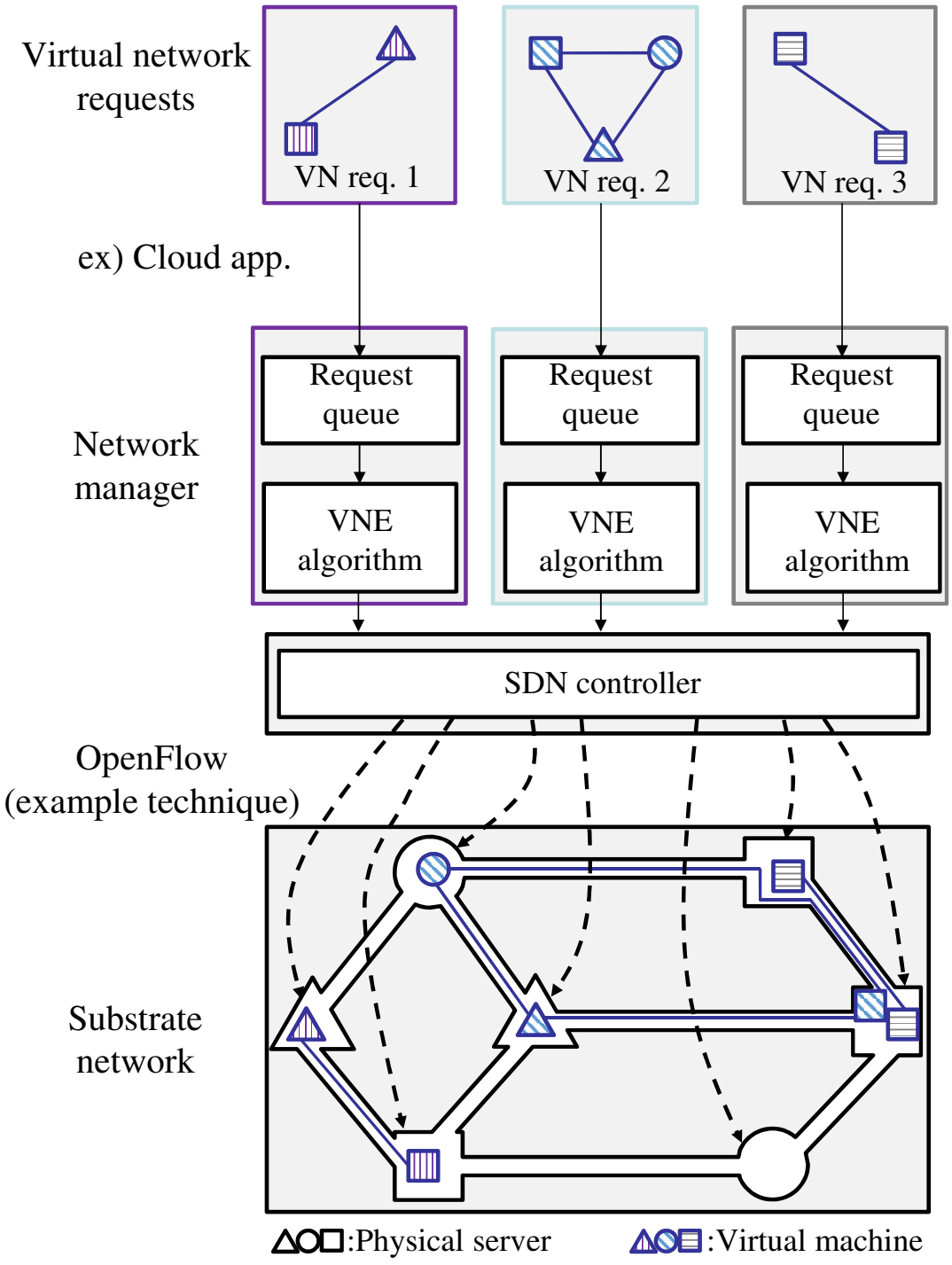


Figure 4.1: Service model in SDI

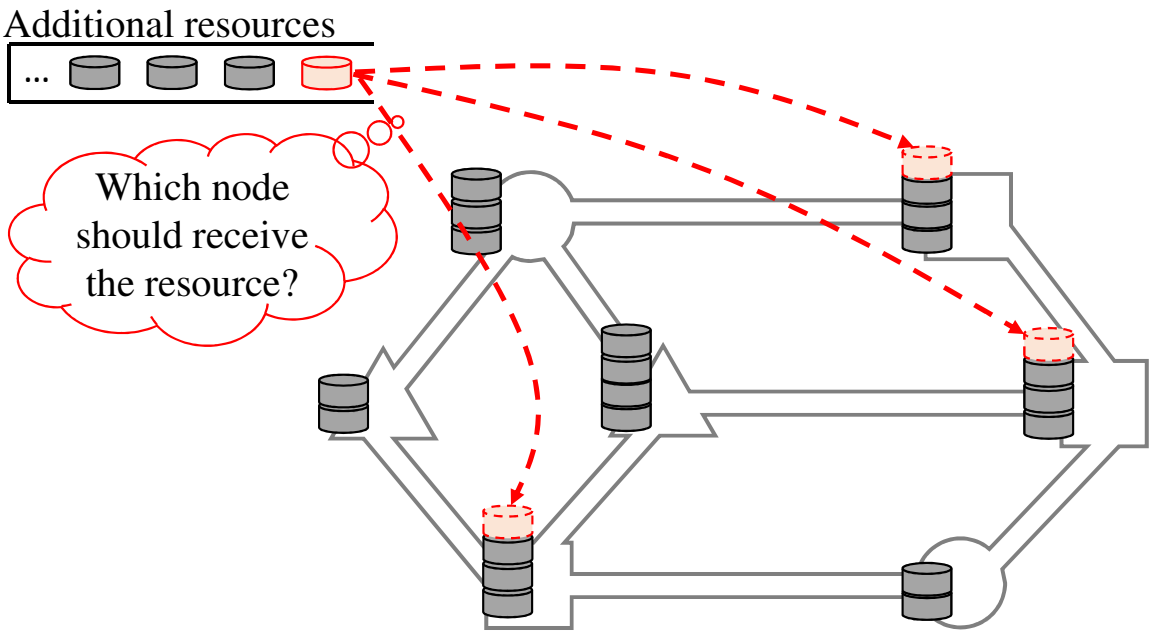


Figure 4.2: Problem of planning resource reinforcement

necessary part of responding to increases in demand. Figure 4.2 shows a resource planning problem of the type considered in this chapter. The objective of solutions is to identify where resources should be reinforced. From the viewpoint of CAPEX, it is necessary to select target nodes in a way that satisfies constraints on the infrastructure administrator in the form of limits on the amount of resources that can be newly added. The figure depicts the reinforcement of computational resources on nodes, but the addition of links and the expansion of bandwidth can be handled in an analogous way.

General network resource planning attempts to optimize some performance metric for a future demand level predicted from the current demand level. For example, suppose a performance optimization method of the VNE problem (e.g., [21]) is applied against predicted demand. The set of places to be reinforced will be decided such that the best performance is obtained, subject to future demand matching predicted demand. Such policies cannot guarantee performance when the actual demand deviates significantly from the predicted demand. Therefore, it is more practical to reinforce resources such that the average performance is optimized, considering several patterns of potential demand and using some lightweight heuristic VNE method (e.g, [61]). This stochastic

optimization reinforcement policy will produce better (average) VN performance even when some demand fluctuation occurs.

4.3 Resource planning strategy to increase evolvability

4.3.1 Resource planning strategy required in SDI

In an SDI framework, customized service must be provisioned to an enormous number of users on demand. In such a situation, resource demand can fluctuate over a short period as a result of various factors (e.g., increase or decrease in traffic demand, entry of a new VN slice, and withdrawal of an existing VN). It is necessary to reinforce the substrate network to improve the adaptability of VNE control under the assumption that unexpected environmental changes will frequently occur. When appropriate physical resources are provided in appropriate places, a VNE control can immediately adapt to minor fluctuations, and VNs can be flexibly constructed even in the face of unpredictable fluctuations.

For that purpose, a promising strategy is to reinforce resources in a way that allows a VNE control to construct a diverse set of solutions. This is expected to increase the possibility that the VNE control can adapt to future environmental fluctuations and cope with more diverse patterns of demand fluctuation.

4.3.2 Adaptation strategy for environmental change by biological evolution

A phenotype, which is a pattern of expressed traits in an organism, is derived by developmental dynamics from an underlying genotype (gene regulatory network) and environmental noise. Fitness, which indicates adaptability to the environment, is defined by a function taking the phenotype as an argument. Even among organisms with the same genotype, the phenotypical expression may differ because of environmental difference, and organisms with a phenotype more suitable to the environment are said to be more fit. After drastic environmental change, one genotype may not be able to express a phenotype with good fitness, but an organism with better fit can be recovered through an evolutionary process (i.e., mutation and selection of the genotype). Genotypic mutation

is carried out by random changes in small parts of the gene regulatory networks, and this will be accompanied by a transformation of the developmental dynamics. As a result, previously unexpressed phenotypes will appear. During selection, genotypes that give rise to phenotypes more suitable for the current environment will survive, and other genotypes will be weeded out.

In biological evolution, evolutionary adaptation is promoted (e.g., fitness recovers more rapidly and reliably) when phenotypic variance increases immediately after a drastic environmental change [34]. Phenotypic variance is a result of two factors: genotypic mutation and environmental noise. In particular, a diversity of phenotypes can be induced by genotype mutation, which is called "(genetic) evolvability" [38]. When evolvability is sufficiently high, various phenotypes can appear through developmental dynamics from genotype mutation, and thus the possibility of expressing a phenotype suitable for the new environment increases. Conversely, when evolvability is small, it will be difficult to recover fitness within the new environment. Therefore, evolvability indicates the degree of potential to adapt to drastic environmental changes.

An evolvability index is defined as a variance form (e.g., $V_g(i)$ and V_g as defined below) in a biological evolution model [34]. Each of K individuals is composed of M genes, and each gene $i(\in \{1 \dots M\})$ has an expression level x_i. Letting $x_i^{k,l}$ indicate the gene expression level of gene i for individual k in environment l, the vector of gene expression levels, $\boldsymbol{x}^{k,l}$, is determined by the developmental dynamics governed by the genotype J_{ij}^k and the environmental noise η^l (See Fig. 4.3). The gene i is expressed if and only if its expression level x_i is greater than a threshold θ_i. The phenotype is given by the expression pattern of the genes $\boldsymbol{X}^{k,l} = \{X_i^{k,l} | X_i^{k,l} = Sign(x_i^{k,l} - \theta_i), i = 1, \dots, M\}$, where $Sign(x)$ is the signum function, which returns 1 when $x > 0$, 0 when $x = 0$, and -1 when $x < 0$. The fitness value is obtained by a function $F(\boldsymbol{X}^{k,l})$ that takes a phenotype as input.

The variance of the expression of gene $V_g(i)$ due to the genotype mutation is defined as

$$V_g(i) = \frac{1}{K} \sum_{k=1}^{K} \left(\overline{X_i^k} - \langle \overline{X_i} \rangle \right)^2, \tag{4.1}$$

where $\overline{X_i^k} = \frac{1}{L} \sum_{l=1}^{L} X_i^{k,l}$ is the average state of mutant k over L environments and $\langle \overline{X_i} \rangle =$

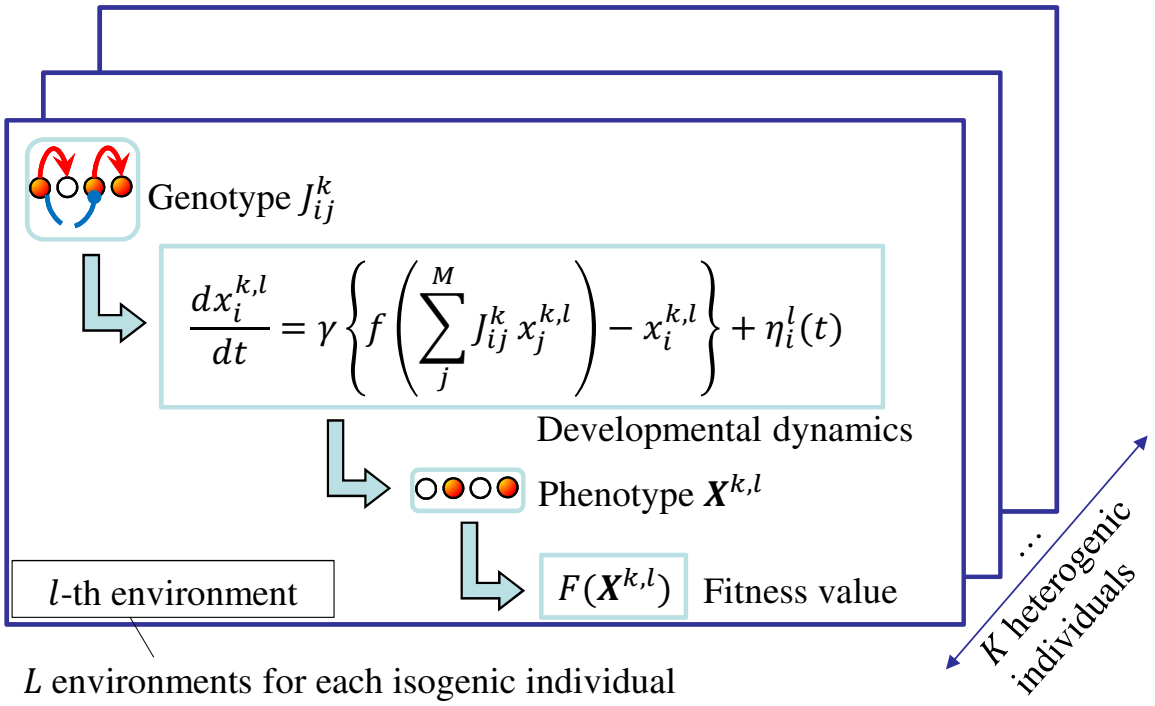


Figure 4.3: Biological evolution model at each generation

$\frac{1}{K}\sum_{k=1}^{K}\overline{X_i^k}$ is the average over K mutants, removing the variance due to noise. The variance of the fitness value V_g is defined in the same way, as

$$V_g = \frac{1}{K}\sum_{k=1}^{K}\left(\overline{F(\boldsymbol{X}^k)} - \langle\overline{F(\boldsymbol{X})}\rangle\right)^2. \tag{4.2}$$

4.4 Computational resource reinforcement method for increasing evolvability

In this chapter, we deal with a planning method to reinforce node computational resources in a substrate network. Here, we describe the problem design, considering generalized computational resources. In our specific case, the computational resources are the number of CPU cores, the memory capacity, and the disk storage capacity. The initial resource levels provided for the substrate network $G(V^s, E^s)$ are described by the computational resource capacity $\{C_{v^s}|v^s \in V^s\}$, and the

link capacity by $\{C_{e^s}|e^s \in E^s\}$. Let the unit of computational resources to be added be Δ, and the total number of nodes to be reinforced be T, where duplicate selection of nodes is allowed and indicates reinforcing that node multiple times. The solution to the reinforcement planning problem of node computational resource is to obtain the set of nodes $\{v_t^s\}_{t=1...T}(v_t^s \in V^s)$ to which to add computational resources.

The ultimate goal of resource reinforcement by the proposed method is to simulate evolutionary adaptability (robustness and plasticity). To achieve this, it is important to increase evolvability, following the strategy of biological evolution. Therefore, we propose an index $H(G)$ that characterizes the evolvability of an SDI substrate network. This is used for selecting the node to be reinforced by calculating the index $H(G)$ when a computational resource is added. In this section, we first outline our proposed method, which is based on the ADD algorithm [46]. Next, the formulation of the evolvability index $H(G)$ used in the ADD algorithm will be explained. We also introduce a VNE method that uses a genotype–phenotype structure to characterize evolvability.

4.4.1 ADD algorithm

The offline simulation procedure based on the ADD algorithm [46] is shown below. The output of the procedure is the set of nodes to which resources should be added.

Step 1

Let $t = 1$.

Step 2

Perform the following procedure for each node in the substrate network $\forall v^s \in \{0, 1, \ldots, |V^s| - 1\}$. Calculate the evolvability index $H(G_{v^s})$ after trial adding to the node v^s, then choose the best node from those trials.

Step 2.1

Add the computational resource to node v^s temporarily. Let the node resource capacity be $C_{v^s} = C_{v^s} + \Delta$.

Step 2.2

Calculate the evolvability index $H(G_{v^s})$ by the evaluation method explained below in detail.

Step 3

Select the node satisfying $v_t^s = \arg\max H(G_{v^s})$, and add the resource amount of Δ to the node v_t^s.

Step 4

If $t \geq T$, then terminate. Otherwise, let $t = t + 1$ and return to **Step 2**.

Note that the method can be converted in the obvious way to network resource reinforcement, by replacing node computational resources with link bandwidth in Step 2. In this chapter, the method described above is limited to the problem of capacity reinforcement for node resources as a first step in designing an SDI substrate network of resources, based on the following idea. A VNE control consists of two phases: virtual node mapping (VNoM) and virtual link mapping (VLiM). The VNE solution largely depends on the first phase VNoM, so we regard relaxing the upper limit of node capacity to be more acceptable than relaxing the link bandwidth when trying to increase the diversity of VNE solutions.

4.4.2 How to calculate the evolvability index

Evolvability of organisms is defined as the phenotypic diversity that genetic mutation brings. In biological systems, the phenotypic expression of traits is determined by dynamics ruled by the genotype, which is not directly expressed, and by environmental noise. This two-stage structure enables the system to express a phenotype that fits the current environment while maintaining adaptability to other environments by permitting expression of other phenotypes from the same genotype. For small environmental noise, the system adapts instantaneously, providing a phenotype that the single genotype can express. For drastic environmental changes, adaptation occurs by providing a broader phenotype through mutation of the genotype via the evolutionary process.

In an SDI framework, we also aim to provide evolutionary adaptability by using such a two-stage structure mechanism. Here, a VNE solution is regarded as a phenotype in SDI. Then, a

control mechanism of VNE is regarded as a genotype, and VN demand fluctuations are regarded as environmental changes. We formulate the evolvability index on the basis of the VNE control method [27], which has the mechanism above.

VNE control method based on genotype–phenotype system [27]

This VNE method outputs a node mapping $f : V^r \mapsto V^s$ from an input of a substrate network $G(V^s, E^s)$ and a virtual network request $G(V^r, E^r)$. Let the requested virtual node be $v^r (\in V^r)$, and the substrate node be $v^s (\in V^s)$. Then, define the gene expression level $\mathrm{x} = \{x_{|V^r|,|V^s|}\}$, and the function

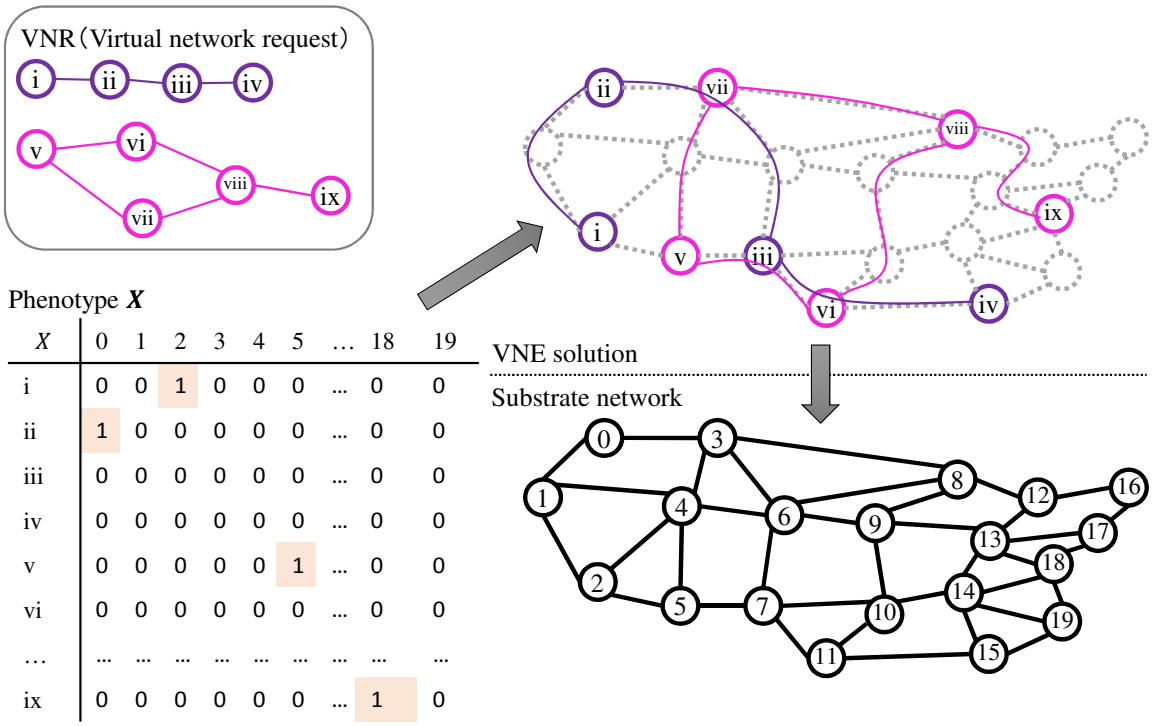
$$f(v^r) \triangleq \begin{cases} \arg\max_{v^s} x_{v^r,v^s} & (\exists x_{v^r,v^s} \geq 0) \\ \mathrm{null} & (otherwise). \end{cases} \tag{4.3}$$

This method determines a node mapping by x, derived from the stochastic differential equation, which is a mathematical model of attractor selection of the biological gene expression dynamics. The value of x_{v^r,v^s} is in the range $[-1, 1]$. The gene is expressed when $x_{v^r,v^s} \geq 0$, and the corresponding $v^r \mapsto v^s$ becomes a candidate for the node mapping. The stochastic differential equation is regarded as having converged when the system reaches a state in which enough genes have been expressed that the value of the function f can be found. The expression / non-expression state is called a solution. The phenotype $\boldsymbol{X} = \{X_{v^r,v^s} | X_{v^r,v^s} = Sign(x_{v^r,v^s})\}$ is obtained as a result. For virtual link mapping (routing), the flow for each virtual link request is set to minimize hop length for each route. An example of a VNE solution based on an obtained phenotype is shown in Fig. 4.4.

The stochastic differential equation of the attractor selection model,

$$\frac{dx_i}{dt} = \alpha \left\{ \varsigma \left(\sum_j W_{ij} x_j \right) - x_i \right\} + \eta, \tag{4.4}$$

is ruled by the control matrix $\mathbf{W}$, which is defined by an attractor structure as the genotype. Activity α, which takes a value in $[0, 1)$, indicates feedback from the system about the comfortableness of the system state. Here, the index i in x_i indicates the pair (v^r, v^s) where $i = |V^s| * v^r + v^s)$,



X	0	1	2	3	4	5	…	18	19
i	0	0	1	0	0	0	...	0	0
ii	1	0	0	0	0	0	...	0	0
iii	0	0	0	0	0	0	...	0	0
iv	0	0	0	0	0	0	...	0	0
v	0	0	0	0	0	1	...	0	0
vi	0	0	0	0	0	0	...	0	0
…	...	...	...	...	...	...	...	...	...
ix	0	0	0	0	0	0	...	1	0

Figure 4.4: An example of VNE solution by phenotype $\boldsymbol{X}$

$\{v^r|0,1,2,\ldots,|V^r|-1\}$ and $\{v^s|0,1,2,\ldots,|V^s|-1\}$. In the following, both x_{v^r,v^s} and x_i as convenient. In the first term of Eq. (4.4), $\varsigma\left(\sum_j W_{ij}x_j\right) - x_i$ represents the attractor structure. The function $\varsigma(z)$ is a sigmoid function defined as,

$$\varsigma(z) = \tanh(\frac{\mu}{2}z), \tag{4.5}$$

where μ represents the gradient in the vicinity of the threshold. This attractor structure is derived from the genetic interaction, that is, from the genotype. The second term η in Eq. (4.4) is a random value.

The matrix $\mathbf{W}$ in Eq. (4.4) represents an attractor structure. It stores some equilibrium points of the virtual node mapping; these equilibrium points are called attractors. Each attractor is defined as $\mathbf{y} = (y_1, \ldots, y_i, \ldots, y_{|V^r||V^s|})$, where $y_i \in \{-1, 1\}$. If physical node v^s is one of the candidates for a virtual node v^r, then y_{v^r,v^s} is set to 1. Otherwise, $y_{v^r,v-s}$ is set to -1. Let M be the number

of attractors stored in $\mathbf{W}$. A set of attractors $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M)$ can be stored by letting

$$\mathbf{W} = \mathbf{Y}^{+}\mathbf{Y}, \tag{4.6}$$

where $\mathbf{Y}^{+}$ is the pseudoinverse matrix of $\mathbf{Y}$. In Eq. (4.4), this matrix governs the state transition so that the variables $\mathbf{x}$ will be attracted to one of the retained attractors when the activity level is sufficiently high. When the present state is at one of the attractors, $d\mathbf{x}/dt$ in Eq. (4.4) becomes close to 0 and stays at the attractor.

Formulation of evolvability

In evaluating the evolvability of an SDI substrate network, we interpret the evolvability in the SDI framework as follows: VNE control expresses various VNEs (phenotypes) by mutation operation of the control matrix (genotype). Figure 4.5 depicts an image of the diversity of VNE solutions, showing the appearance probability distribution for obtaining VNE solutions through the VNE control. Even without any reinforcement, the number of VNE solutions can be increase by repeating the control matrix mutation operations. By incrementally applying resource reinforcements, an increasing variety of VNE solutions can be obtained, with a wider search range.

Figure 4.6 shows the procedure of calculating the entropy $H(G_{v^s})$.

Step 1

Generate control matrices $W_1, \ldots, W_K$ by the mutation operation of the control matrix W. An example of the mutation operation is shown in Fig. 4.7. First, randomly select an attractor from the set of the attractors retained in the control matrix. Second, randomly select a virtual node from within the selected attractor. Third, randomly select a substrate node and modify the candidate node for the virtual node to the selected substrate node. Finally, reconstruct the matrix with the revised attractor.

Step 2

Generate VNR demand $D_1, \ldots, D_L$ in a random fashion. The demand is given in terms of memory amount for the virtual nodes and bandwidth for the virtual links.

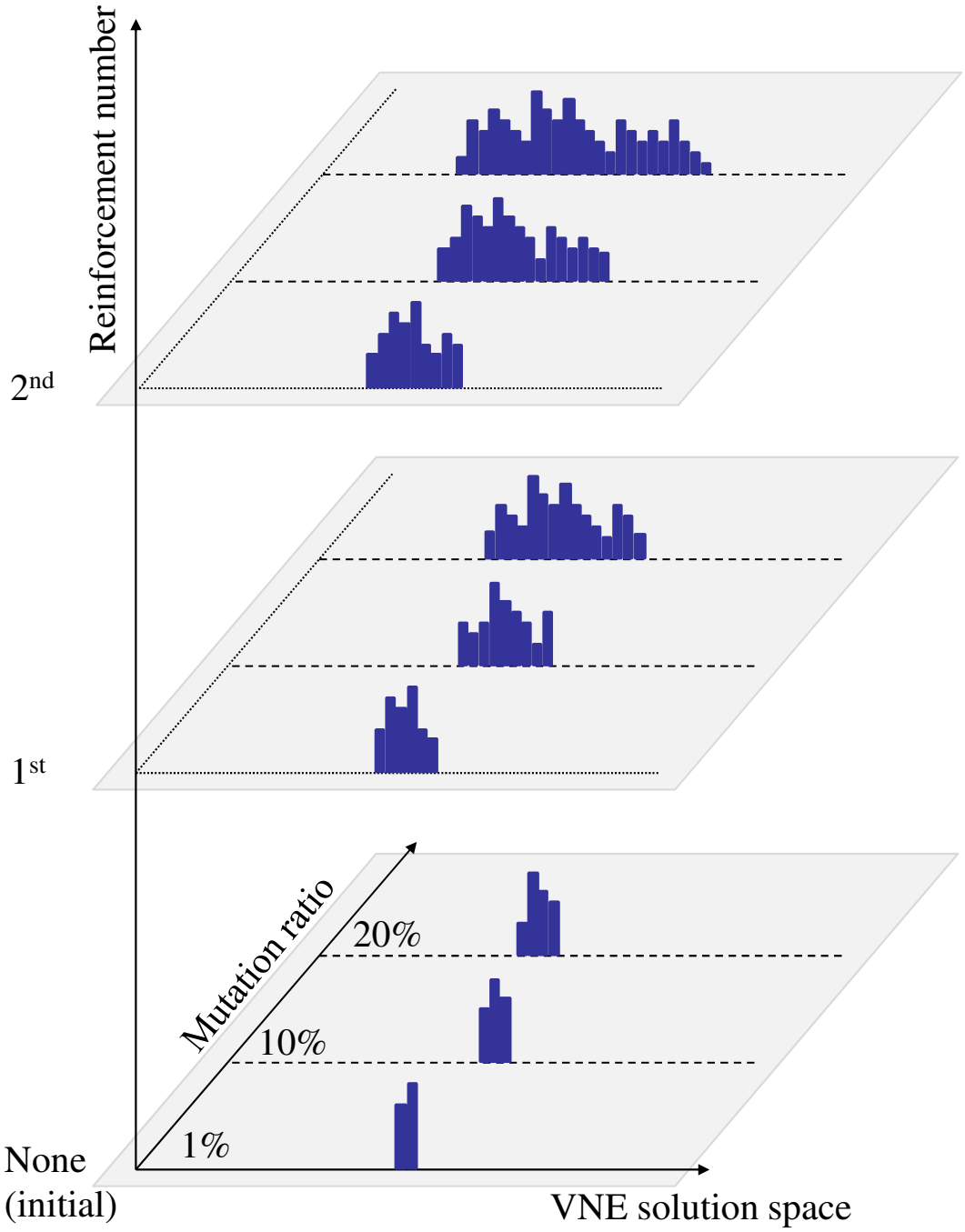


Figure 4.5: Concept of evolvability: Appearance probability distribution for VNE solutions at each resource reinforcement stage

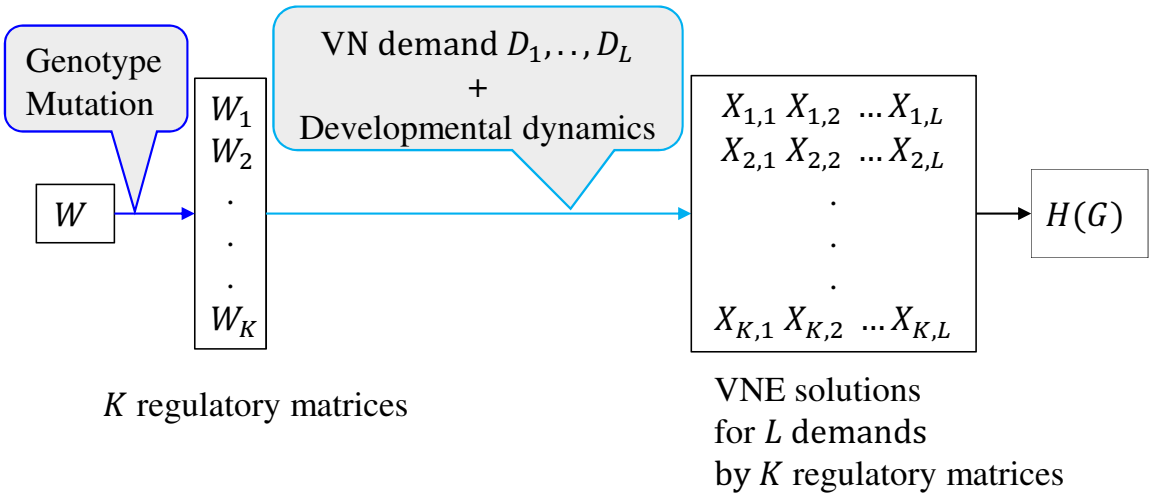


Figure 4.6: Procedure of calculating the evolvability index

Step 3

Let $\boldsymbol{X}_{k,l}$ be the solution obtained at convergence through the gene expression dynamics given by the control matrix W_k for the demand D_l. Then, calculate the entropy of appearance probability $H(G_{v^s}) = -\sum_X p(\boldsymbol{X}) \log p(\boldsymbol{X})$, which will explained below in detail.

The VNE method attempts to adapt to a new environment by reconfiguration (mutation) of the control matrix when it is difficult to adapt with the initial control matrix. Designing the substrate network so that more types of VNE solutions can be reached by minor changes to the control matrix makes it easier to adapt to such situations. Also, according to ref. [34], when diverse solutions can be obtained by minor updates, plasticity is achieved in the evolution process of repeating genotype mutations. This chapter introduces a phenotypic diversity index related to genetic mutation in an SDI framework. However, simply applying the indexes $V_g(i)$ or V_g, which are defined in the biological background [34], is not suitable for evaluation of the VNE solution diversity because those indexes will just reflect the variance of control variables or VN performance value (e.g., end-to-end delay). Therefore, we define the entropy expression $H(G)$, which characterizes the various kinds of VNE solution that can be obtained.

Let control variable vectors be $\boldsymbol{X}^{k,l} = \{X_i^{k,l} | i = 1 \ldots |V^r||V^s|\}$, obtained by offline simulation with control matrix $\{W_k | k = 1 \ldots K\}$ for VN demand $\{D_l | l = 1 \ldots L\}$. In the simulation,

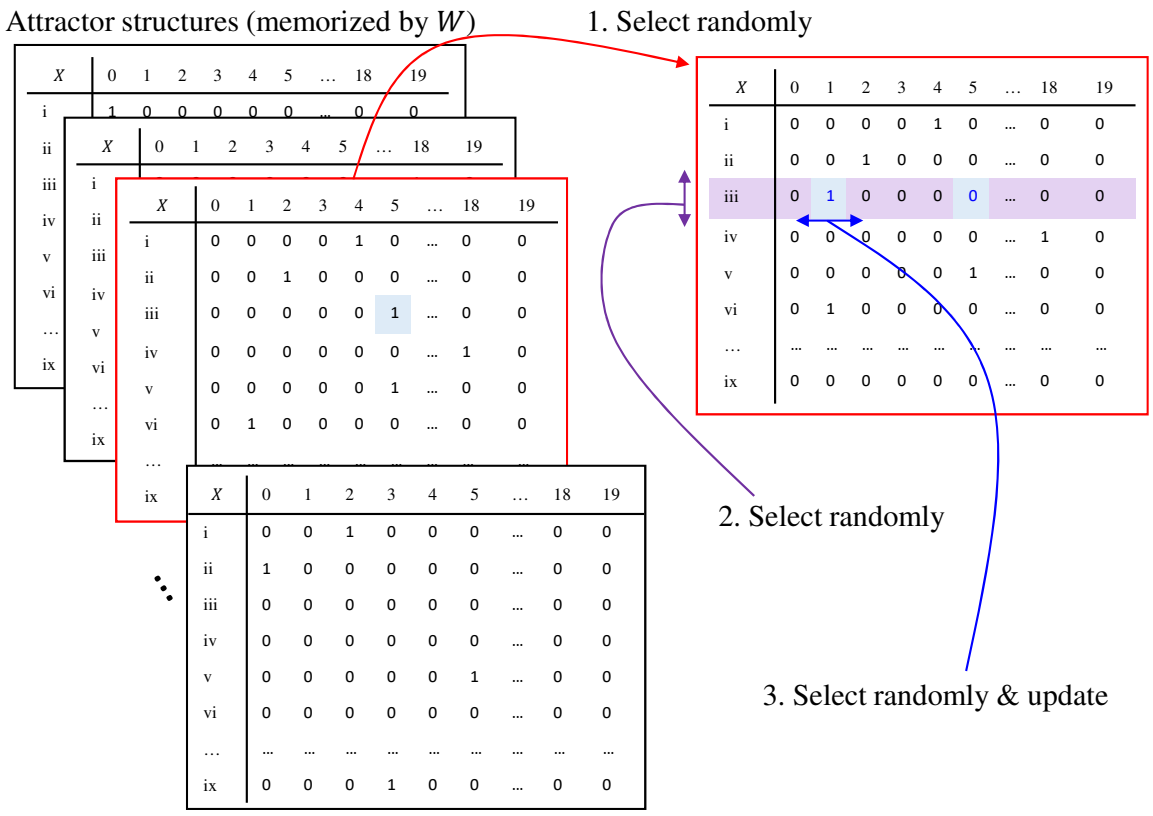


Figure 4.7: An example of a mutation operation for the control matrix

the initial component values of $\boldsymbol{x}_{k,l}$ are all set to -1 (unexpressed), and the activity level is fixed to 0.1 so that the noise-induced search will converge to a solution more likely related to results from genetic mutation than from an efficient search with activity feedback. Let the number of pairs (k, l) be $n(\boldsymbol{X})$ among KL simulations where the control vector is converged to $\boldsymbol{X}_{k,l} = \boldsymbol{X}$. For an evolvability index, the entropy $H(G)$ of appearance probability is defined as

$$H(G) = -\sum_{X} p(\boldsymbol{X}) \log p(\boldsymbol{X}), \tag{4.7}$$

where the appearance probability $p(\boldsymbol{X}) = \frac{n(\boldsymbol{X})}{KL}$.

4.5 Performance evaluation

We used computer simulation to evaluate the effectiveness of our proposed method. First, we verified that the potential of the VNE control to discover solutions is improved by applying the resource reinforcements suggested via our method. Then, we compared the solution-discovering potential with resource reinforcement as given by a general stochastic optimization method. We further demonstrated that the proposed reinforcement improved the convergence probability of the VNE control when demand was highly variable.

In this simulation, we observe with the end-to-end delay between virtual nodes as the service performance of a VN, and reinforce (expand) the memory of a certain server as a computational resource. The aim of reinforcing memory capacities is to suppress increases in delay time by lowering the rise of the memory utilization rate. When the memory utilization rate is high, virtual memory will be used by the host OS, which will cause some of the storage disk to be used for swapping memory pages. This causes a calculation delay since the speed of access to the storage is much slower than that of access to the memory. Thus, reinforcing the memory capacity is expected to reduce the end-to-end delay, assuming some calculations are needed on transit servers. By applying memory enhancement to appropriate nodes, it is possible to cope with environmental changes such as fluctuation of computing resource or virtual link requests. That is, the reinforcement promotes the adaptability of the VNE control, which will more easily find a good solution with a comfortable delay under the new environment than without reinforcement.

4.5.1 Simulation environment

Substrate network and virtual network requests

The topology of the substrate network $G(V^s, E^s)$ used is shown in Fig. 4.4. The number of nodes is 20. The initial computational resource amount available at each node before applying reinforcement is set as follows. For evaluation of improvement in solution-discovering potential, as in Sec. 4.5.3, the computational resource (memory) amount C_{v^s} of each node is uniformly set to 100. Also, for evaluation of the VNE control convergence probability, as in Sec. 4.5.3, the initial value of memory equipped on each node is set to a value chosen uniformly randomly from $[60, 140]$. When applying

the resource reinforcement method, the additional memory amount Δ for each round in the ADD algorithm is set to 20. The physical link bandwidth C_{e^s} is set to 1000 for each link.

A virtual network request $G(V^r, E^r)$ consists of 10 virtual nodes. When the VN demand is generated, the requested memory amount C_{v^r} of each virtual node is determined by a number chosen uniformly randomly from $[20, 80]$. For each pair of virtual nodes, a virtual link is established between the virtual nodes with probability 50%. When a virtual link exists between the components of a virtual node pair (v_i^r, v_j^r), the required bandwidth $C_{e_{i,j}^r}$ of that virtual link is taken to be $C_{e_{i,j}^r} = \frac{C_{v_i^r} + C_{v_j^r}}{2}$. The virtual link $e_{i,j}^r$ is allocated to the shortest path in the substrate network from the physical node hosting v_i^r to the one hosting v_j^r, and the amount $C_{e_{i,j}^r}$ is reserved from the bandwidth of the physical links along the path. If one or more physical links along the path has an over-subscribed bandwidth capacity, the VN request cannot be accommodated and the VNE control fails.

VNE control method

In this evaluation, the VNE control method (4.4.2) is used. As a threshold for activity, let the maximum end-to-end delay (the largest delay among the end-to-end delays of each virtual link) be θ. When the maximum end-to-end delay is suppressed under θ, the activity becomes higher. Otherwise, when the maximum end-to-end delay exceeds θ, the activity becomes close to zero. The end-to-end delay $d_{e_{i,j}^r}$ of the virtual link $e_{i,j}^r$ is calculated according to the memory resource utilization rate of the physical nodes along the path, as

$$d_{e_{i,j}^r} = \sum_{v^s \in V_{e_{i,j}^r}^s} d_{v^s}, \tag{4.8}$$

where $V_{e_{i,j}^r}^s$ is the set of physical nodes along the path from the virtual node v_i^r to v_j^r, and d_{v^s} is the delay caused by the memory utilization on the physical node v^s. In this chapter, the delay d_{v^s} of the physical node v^s related to its memory resource utilization $u_{v^s} = \frac{\sum_{v^r | f(v^r) = v^s} C_{v^r}}{C_{v^s}}$ is defined by the delay profile shown in Fig. 4.8.

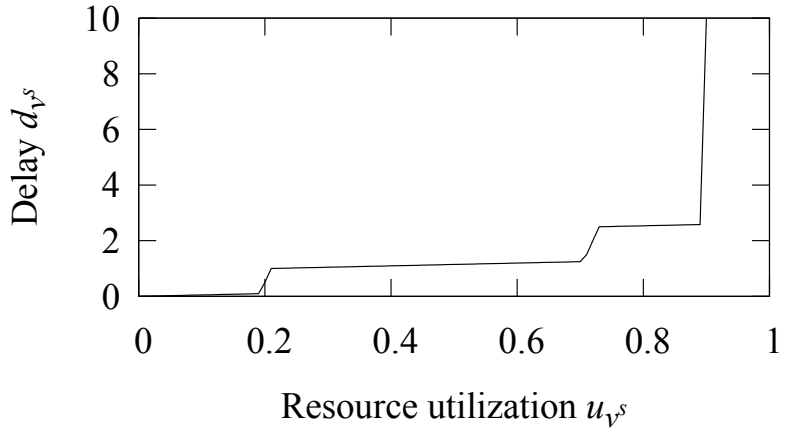


Figure 4.8: Delay profile

The activity α is defined as

$$\alpha = \frac{1}{1 + \exp(\delta(d^{max}(f) - \theta))}, \tag{4.9}$$

where $d_{max}(f) = \max_{e^r_{i,j} \in E^r} d_{e^r_{i,j}}$ is the maximum end-to-end delay of virtual node mapping f. A converged state with $\alpha \geq 0.5$ is regarded as a VNE solution. For the parameters, we use $\delta = 1.5$ and $\theta = 6$ in this chapter.

The number of kinds of solutions converged to by the VNE control is characterized by the parameter of a mutation ratio of the control matrix. A single mutation operation is defined as follows. Select one node randomly from among the virtual node mapping candidates retained as attractors, randomly select one virtual node from the virtual network, and randomly select one of the physical nodes where the virtual node is to be hosted. Replace the original attractor by the randomly revised virtual node mapping candidate and reconstruct the control matrix. The mutation rate of the control matrix is defined as $\frac{n}{M|V^r||V^s|}$, where n is the number of mutation operations applied, M is the number of attractors retained by the control matrix, $|V^r|$ is the number of virtual nodes, and $|V^s|$ is the number of physical nodes. Note that $|V^r||V^s|$ is the number of dimensions of the control variable, and $M|V^r||V^s|$ is the amount of information stored in the control matrix.

The VNE control applies the mutation operation periodically when the solution search does not converge immediately, thereby spreading the search range to discover a good solution for the current VN demand. Increasing the mutation rate enables the control method to search for a wider solution space, although it may cause degradation of the immediate adaptiveness against minor fluctuations.

4.5.2 Basis method for comparison

We construct a basis reinforcement method for comparison with our proposed method. The basis method aims to improve the average performance over predicted demand fluctuations. Here, we use a heuristic VNE control [61] that avoids increases in resource utilization on a certain node by considering the VN demand and the unclaimed resources of the substrate network. The basis reinforcement method follows the ADD algorithm, given below.

Step 1

Let $t = 1$.

Step 2

Perform the following procedure for each node in the substrate network $\forall v^s \in \{0, 1, \ldots, |V^s| - 1\}$. Calculate the average value of the maximum end-to-end delay $\overline{d_{max}^{(v^s)}}$ when the resource is trial added to the node v^s, then choose the node resulting in the best of those values.

Step 2.1

In the simulation, add the computational resource to node v^s temporarily. Let the node resource capacity be $C_{v^s} = C_{v^s} + \Delta$.

Step 2.2

Calculate $\overline{d_{max}^{(v^s)}}$ through the following procedure.

Step 2.2.1

Obtain the maximum end-to-end delay $d_{max}^{(v^s)}(l)$ by the heuristic VNE control against VN demand $D_1, \ldots, D_L$.

Step 2.2.2

Calculate the average value of the maximum end-to-end delays $\overline{d_{max}^{(v^s)}} = \sum_{l=1}^{L} d_{max}^{(v^s)}(l)/L$ as a performance index.

Step 3

Select a node $v_t^s = \arg\min \overline{d_{max}^{(v^s)}}$ and add the resource amount of Δ to the node v_t^s.

Step 4

If $t \geq T$, then terminate. Otherwise, let $t = t + 1$ and return to **Step 2**.

4.5.3 Simulation results

Improving the potential to discover solutions with the VNE control

We implemented the proposed reinforcement method for $T = 3$ stages with $K = L = 100$, and the nodes to be reinforced are determined as $\{$1st: node 11; 2nd: node 8; 3rd: node 3$\}$.

In the substrate network of each reinforcement stage, 1000 VN demands are given. Figure 4.9 shows the number of VNE solutions obtained through the VNE control simulation against the 1000 patterns of VN demand. The number of VNE solutions is plotted on the horizontal axis and the mutation rate of the control matrix is plotted on the vertical axis. Even before reinforcement (at "Init"), the number of solutions increases as the mutation rate increases, but the number of solutions is increased more rapidly by applying reinforcements. Comparing the third-stage reinforcement with the substrate before-reinforcement, the number of solutions to be found increased by about $[36 - -54]\%$ at a mutation rate of 1% or less, showing that the solution discovery potential of the VNE control is improved by the proposed resource reinforcements.

Next, for comparative evaluation, we implemented the basis reinforcement method for $T = 3$ stages with $L = 1000$. The total amount of additional resources installed by both the proposed and the basis method is same. The nodes to be reinforced selected by the basis method are nodes 6, 9, and 14. The number of discovered VNE solutions is shown in Fig. 4.10. When applying reinforcement by the proposed method (H(G)), the number of VNE solutions increases with increasing the mutation rate, and more solutions are found than before reinforcement (Init) with a higher mutation

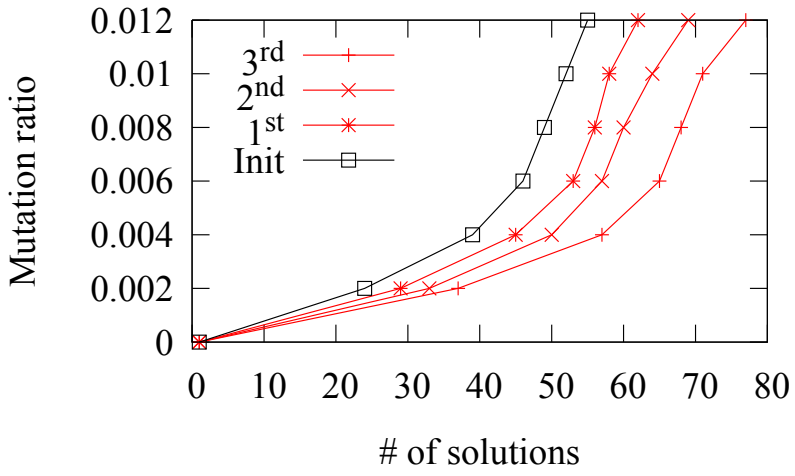


Figure 4.9: The number of VNE solutions reached by the VNE control

ratio. In the basis method (Basis), the number of solutions increases to only the same level as before reinforcement (Init), even when the mutation rate was raised. The reinforcement by the proposed method thus encourages the discovery of a wider variety of VNE solutions than does the basis reinforcement, even with the same amount of resources added. The increase in solution diversity is promoted by the mutation operations of the control matrix, that means evolvability is increased. Since the proposed reinforcement makes it possible to accept more diverse VNE solutions for the given substrate network, improvement of the adaptability of the VNE control against environmental changes is expected.

Improving convergence probability of VNE control

Here, we evaluate whether the convergence probability of the VNE control is improved by reinforcing resources. The convergence probability of the VNE control against VN demand variation is simulated in the initial stage and in reinforcement stages 1 through 10. In the simulation, the search duration of the VNE control is limited to 1000 control steps. If it does not converge within 50 control steps, the control method applies a mutation operation to the control matrix every 10 steps thereafter. The convergence probability of the VNE control is shown in Fig. 4.11. The convergence

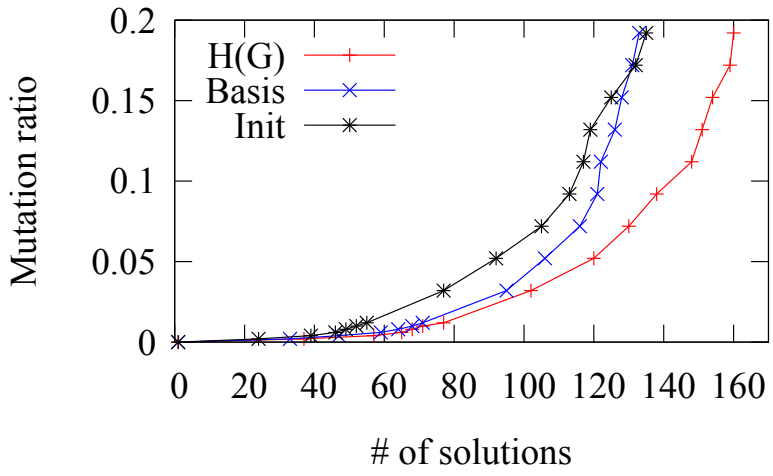


Figure 4.10: Number of VNE solutions: comparison with the basis reinforcement

probability is calculated by simulating the VNE control for 1000 patterns of VN demand. The convergence probability before reinforcement (Init) is 70%, and this rate improves as reinforcements are made, in both cases (by the proposed method and the basis method). The convergence probability was improved to 94% by the proposed method at the tenth reinforcement stage, while it was improved to only 75% by the basis method with the same amount of additional resources. In the basis method, the substrate network may become specialized at avoiding delay against the predicted demand (of $L = 1000$ patterns), but the proposed reinforcement improves the adaptability of the VNE control against a wider variety of demand patterns than the set considered in the planning phase (of $L = 100$ patterns).

4.6 Conclusions

In this chapter, we consider a problem of planning the capacities of physical network resources in an SDI framework to deal with difficult-to-predict demand fluctuation. We have proposed a method of deciding computational resource reinforcement by applying knowledge on evolvability. The method increases the adaptability against unknown environmental fluctuations, reconstructing an attractor structure in the VNE control after large changes. The results obtained by computational

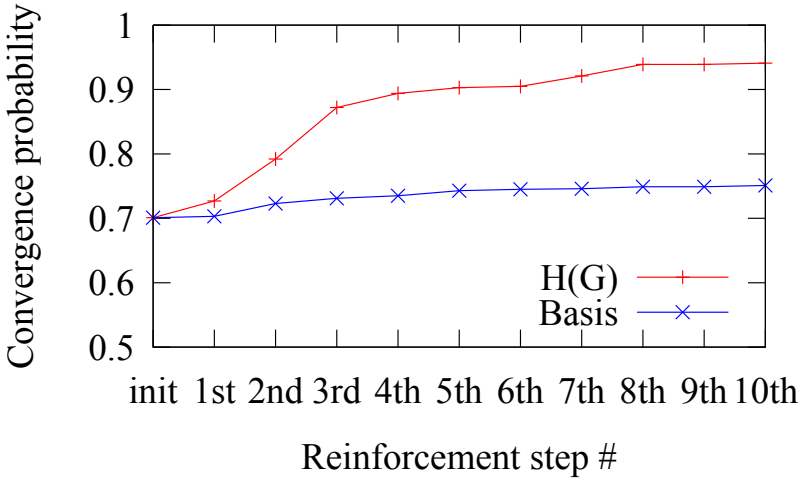


Figure 4.11: Convergence probability of the VNE control

simulation show that when memory reinforcement of the substrate network is done according to the proposed strategy, the adaptability in terms of solution discovery by a VNE control in the face of demand fluctuation is improved relative to a basis reinforcement, which optimizes a expected performance under predicted demand fluctuations, by the same amount of resources.

Future work remains in order to improve the proposed strategy. We will further examine the adaptability of the proposed method against additional environmental changes (e.g., faults on node equipment or links), rather than only demand fluctuation.

Chapter 5

Conclusion

SDI is a promising framework to enable flexible and rapid deployment of new services on information networks by providing virtualized infrastructure to customers by slicing computing resources and network resources. In this thesis, we present an adaptive VNE method that works with only a little information for large, complicated, and uncertain SDI frameworks. Also, we consider physical resource design to increase a diversity of solutions reached by a VNE control.

First, we proposed a physical resource design method for optical networks, e.g., wavelength division multiplexing (WDM) networks, as a prior inspection for physical resource design method in SDI frameworks. We propose a design method for adding transceivers to IP routers in IP-over-WDM networks. The method defines correspondence between an evolution model and a WDM network, and simulates a process of biological evolution (i.e., mutation and selection of gene regulatory networks through generations) where transceivers arrangement is reflected by modifying the gene regulatory network. Then it measures performance of the VNT control method (i.e., average link utilization rate). Computer simulation for some WDM networks showed that our method makes attractor-based VNT control methods more adaptive to unexpected traffic fluctuations and reduces degradation of the adaptability under strong traffic fluctuations, that is, our method accommodates more patterns of traffic fluctuation with lower link utilization than ad-hoc design methods do. Thus we confirm the bio-inspired approach is promising for physical resource design. Despite the effectiveness, the method does not consider a diverse set of potential virtual networks, so evolvability

will not be obtained. Our physical resource design strategy could stand further improvement by adopting the essence of biological evolvability, which we consider in the physical resource design method for SDI frameworks.

Second, we presented a VNE method based on the Yuragi principle as applied to SDI frameworks. A system driven by the Yuragi principle achieves adaptability to environmental changes, and the dynamics is described as an attractor selection model. In attractor selection models, the system behavior is governed by an activity measure and small perturbations. When activity is high, the control state of the system is in a good condition and stays in that state. When activity becomes low or the condition becomes uncomfortable due to environmental changes, the system looks for another stable state. The Yuragi-based VNE method decides the mapping of virtual nodes by means of attractor selection, where the network mapping is regarded as the system state and the activity is defined as a certain performance objective. The end-to-end delay in SDI frameworks depends on application processes and other factors. That makes it difficult to pre-estimate experienced delay accurately and causes degradation of VNE control performance. Nevertheless, our Yuragi-based method shows its adaptability under such uncertain delay conditions. In the evaluation, we considered the end-to-end delay as the activity. Simulation results show that the method provides shorter delays and adapts to the request fluctuations by rearranging the VN mapping in response to drastic changes in environments. The Yuragi-based method decreases VN migrations by about 29% relative to a heuristic method to adapt to fluctuations in required resource capacities.

Finally, we proposed an SDI resource design strategy that increases VNE solution diversity, which is derived by a control system variation under demand fluctuations. Our design strategy for SDI system aims to achieve an adaptable characteristic against environmental changes by increasing diversity of VNE states. As a successful model, an evolution system of organisms takes a similar strategy, where they can evolve fitting to environmental changes. We propose an SDI resource design strategy that increases VNE solution diversity, which is derived by a control system variation under demand fluctuations. The strategy imitates evolvability of a biological evolutionary adaptation model with regarding a VNE solution as a biological phenotype. Then we construct a node computational capacity reinforcement method in accordance with the proposed strategy, and

conduct experiments by computer simulations demonstrating that the adaptiveness of a VNE control will improve. The result shows convergence probability of a VNE control is improved after our proposed physical resource reinforcement with an up to 19% gain compared with an ad-hoc reinforcement.

In summary, we constructed a VNE control method which can immediately response to demand fluctuations in SDI frameworks where short-term requests must be managed. We also constructed a strategy for physical resource planning to promote the VNE control adaptability against drastic demand changes. Both of the methods take bio-inspired approaches. The proposed VNE method is expected to enjoy the adaptability of Yuragi to environmental changes. VN migrations are driven according to experienced performance and the new VN mapping is obtained by means of attractor selection. Different from optimizing problems and related heuristics, the Yuragi-based method can avoid the necessity of collecting detailed information about the entire network. The process for a VN request needs only enough information for comfortableness and does not need any information related to other VN requests. Also, the proposed physical resource reinforcement method is to acquire the nature of evolutionary adaptation (robustness and plasticity). To achieve that, it is important to increase evolvability, following the strategy of biological evolution. Therefore, we propose an index $H(G)$ which evaluate evolvability of a SDI substrate network. This is used for selecting the node to be reinforced, by calculating the index $H(G)$ when a computational resource is added.

Future work remains in constructing a more feasible scheme based on our proposed strategy in order to be deployed in practice. First, a method for VNE control is to be investigated that constructs the attractor structure to improve the convergence time or some other performance measure. There is also room for improvement in the random mutation operation strategy, which should be improved with adopting intentional attractor modification based on medium-term traffic observation. Second, despite our physical resource design strategy succeed in increasing an evolvability feature, some questions occurred: ”Is there a certain topological characteristic which brings evolvability to SDI?” or ”How much resources should be reinforced to maintain evolvability?”. Revealing these obscure traits and constructing a heuristic method will make the implementation of our bio-inspired strategy be much more systematic and practical. We believe that the whole discussion in this thesis

and the remained research topics above will contribute to SDI deployment for activating efforts in development of various ICT services, which bring a prosperous future.

Bibliography

[1] H. Farhangi, "The path of the smart grid," *IEEE Power and Energy Magazine*, vol. 8, pp. 18–28, Jan. 2010.

[2] P. Papadimitratos, A. D. L. Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Communications Magazine*, vol. 47, pp. 84–95, Nov. 2009.

[3] S. Namiki, T. Kurosu, K. Tanizawa, J. Kurumida, T. Hasama, H. Ishikawa, T. Nakatogawa, M. Nakamura, and K. Oyamada, "Ultrahigh-definition video transmission and extremely green optical networks for future," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 17, pp. 446–457, Mar. 2011.

[4] R. S. Weinstein, A. M. Lopez, B. A. Joseph, K. A. Erps, M. Holcomb, G. P. Barker, and E. A. Krupinski, "Telemedicine, telehealth, and mobile health applications that work: Opportunities and barriers," *The American Journal of Medicine*, vol. 127, pp. 183–187, Mar. 2014.

[5] D. Haluza and D. Jungwirth, "ICT and the future of health care: aspects of health promotion," *International Journal of Medical Informatics*, vol. 84, pp. 48–57, Jan. 2015.

[6] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, pp. 30–39, Jan. 2017.

[7] G. Kandiraju, H. Franke, M. D. Williams, M. Steinder, and S. M. Black, "Software defined infrastructures," *IBM Journal of Research and Development*, vol. 58, pp. 2:1–2:13, March 2014.

[8] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, "Software defined cloud: Survey, system and evaluation," *Future Generation Computer Systems*, vol. 58, pp. 56–74, May 2016.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 69–74, Mar. 2008.

[10] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, pp. 14–76, Dec. 2014.

[11] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing architecture," *IETF Request for Comments: 8402*, July 2018.

[12] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future Internet," *Computer Networks*, vol. 75, Part A, pp. 453–471, Dec. 2014.

[13] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 1617–1634, Feb. 2014.

[14] P. Bhaumik, S. Zhang, P. Chowdhury, S. S. Lee, J. Lee, and B. Mukherjee, "Software-defined optical networks (SDONs): A survey," *Photonic Network Communications*, vol. 28, pp. 4–18, June 2014.

[15] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, pp. 36–43, July 2013.

[16] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 38–47, Apr. 2011.

[17] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, pp. 81–88, Aug. 2009.

[18] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, pp. 1888–1906, Feb. 2013.

[19] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proceedings of IEEE INFOCOM*, pp. 783–791, Apr. 2009.

[20] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, "A novel approach to virtual networks embedding for SDN management and orchestration," in *Proceedings of IEEE NOMS*, pp. 1–7, May 2014.

[21] X. Chen, C. Li, and Y. Jiang, "Optimization model and algorithm for energy efficient virtual node embedding," *IEEE Communications Letters*, vol. 19, pp. 1327–1330, Aug. 2015.

[22] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Adaptive-VNE: A flexible resource allocation for virtual network embedding algorithm," in *Proceedings of IEEE GLOBECOM*, pp. 2640–2646, Dec. 2012.

[23] L.-S. Peh and W. J. Dally, "A delay model for router micro-architectures," *IEEE Micro*, vol. 21, pp. 26–34, Jan. 2001.

[24] S. Haeri and L. Trajković, "Virtual network embedding via monte carlo tree search," *IEEE Transactions on Cybernetics*, vol. 48, pp. 510–521, Feb. 2018.

[25] N. Shahriar, R. Ahmed, S. R. Chowdhury, A. Khan, R. Boutaba, and J. Mitra, "Generalized recovery from node failure in virtual network embedding," *IEEE Transactions on Network and Service Management*, vol. 14, pp. 261–274, June 2017.

[26] H. Zhang, X. Zheng, J. Tian, and Q. Xue, "A virtual network embedding algorithm based on RBF neural network," in *Proceedings of IEEE CSE 2017 and IEEE EUC 2017*, vol. 1, pp. 393–396, July 2017.

[27] K. Inoue, S. Arakawa, S. Imai, T. Katagiri, and M. Murata, "Adaptive VNE method based on Yuragi principle for software defined infrastructure," in *Proceedings of IEEE HPSR*, pp. 191–196, June 2016.

[28] Y. Xin, G. N. Rouskas, and H. G. Perros, "On the physical and logical topology design of large-scale optical networks," *IEEE Journal of Lightwave Technology*, vol. 21, pp. 904–915, Apr. 2003.

[29] O. Gerstel, C. Filsfils, T. Telkamp, M. Gunkel, M. Horneffer, V. Lopez, and A. Mayoral, "Multi-layer capacity planning for IP-optical networks," *IEEE Communications Magazine*, vol. 52, pp. 44–51, Jan. 2014.

[30] K. Inoue, S. Arakawa, and M. Murata, "A biological approach to physical topology design for plasticity in optical networks," *Optical Switching and Networking*, vol. 25, pp. 124–132, July 2017.

[31] K. Inoue, S. Arakawa, and M. Murata, "Achieving plasticity in WDM networks: Application of biological evolutionary model to network design," in *Proceedings of IEEE GLOBECOM*, pp. 1–7, Dec. 2015.

[32] K. Inoue, S. Arakawa, and M. Murata, "A design method of WDM networks based on biological evolution model," *Technical Report of IEICE(PN2014-8)*, vol. 114, pp. 41–46, June 2014.

[33] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Adaptive virtual network topology control based on attractor selection," *IEEE Journal of Lightwave Technology*, vol. 28, pp. 1720–1731, June 2010.

[34] K. Kaneko, "Evolution of robustness and plasticity under environmental fluctuation: Formulation in terms of phenotypic variances," *Journal of Statistical Physics*, vol. 148, pp. 687–705, Sept. 2012.

[35] K. Inoue, S. Arakawa, S. Imai, T. Katagiri, and M. Murata, "Noise-induced VNE method for software-defined infrastructure with uncertain delay behaviors," *Computer Networks*, vol. 145, pp. 118–127, Nov. 2018.

[36] K. Inoue, S. Arakawa, S. Imai, T. Katagiri, M. Sekiya, and M. Murata, "Yuragi-based approach with delay profile for virtual network embedding in software defined infrastructure," *Technical Report of IEICE(IN2015-148)*, vol. 115, pp. 235–240, Mar. 2016.

[37] K. Inoue, S. Arakawa, and M. Murata, "An evolvable network resource planning for adaptive virtual network control in software defined infrastructure," *Technical Report of IEICE(NS2017-160)*, vol. 117, pp. 93–98, Jan. 2018.

[38] A. Wagner, "Robustness and evolvability: a paradox resolved," in *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 275, pp. 91–100, Jan. 2008.

[39] N. Ghani, S. Dixit, and T. Wang, "On IP-over-WDM integration," *IEEE Communications Magazine*, vol. 38, pp. 72–84, May 2000.

[40] A. Kadohata, A. Hirano, F. Inuzuka, A. Watanabe, and O. Ishida, "Wavelength path reconfiguration design in transparent optical WDM networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 5, pp. 751–761, July 2013.

[41] D. Banerjee and B. Mukherjee, "Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 598–607, Oct. 2000.

[42] A. Narula-Tam, E. Modiano, and A. Brzezinski, "Physical topology design for survivable routing of logical rings in WDM-based networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 1525–1538, Oct. 2004.

[43] C. Meusburger, D. Schupke, and J. Eberspacher, "Multiperiod planning for optical networks - approaches based on cost optimization and limited budget," in *Proceedings of IEEE ICC*, pp. 5390–5395, May 2008.

[44] A. Nag, M. Tornatore, and B. Mukherjee, "Optical network design with mixed line rates and multiple modulation formats," *Journal of Lightwave Technology*, vol. 28, pp. 466–475, Feb. 2010.

[45] M. Klinkowski, F. Herrero, D. Careglio, and J. Solé-Pareta, "Adaptive routing algorithms for optical packet switching networks," in *Proceedings of IEEE ONDM*, pp. 235–241, Feb. 2005.

[46] A. A. Kuehn and M. J. Hamburger, "A heuristic program for locating warehouses," *Management Science*, vol. 9, pp. 643–666, July 1963.

[47] E. Leonardi, M. Mellia, and M. A. Marsan, "Algorithms for the logical topology design in WDM all-optical networks," *Optical Networks Magazine*, vol. 1, pp. 35–46, Jan. 2000.

[48] S. Gieselman, N. Singhal, and B. Mukherjee, "Minimum-cost virtual-topology adaptation for optical WDM mesh networks," in *Proceedings of IEEE ICC*, vol. 3, pp. 1787–1791, June 2005.

[49] S. Arakawa, T. Sakano, Y. Tsukishima, H. Hasegawa, T. Tsuritani, Y. Hirota, and H. Tode, "Topological characteristic of Japan photonic network model," *Technical Report of IEICE(PN2013-2)*, vol. 113, pp. 7–12, June 2013.

[50] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating IP traffic matrices: Initial recommendations," *SIGCOMM Computer Communication Review*, vol. 35, pp. 19–32, July 2005.

[51] P. Hegyi, T. Cinkler, N. Sengezer, and E. Karasan, "Traffic engineering in case of interconnected and integrated layers," in *Proceedings of 13th International Telecommunications Network Strategy and Planning Symposium (NETWORKS 2008)*, pp. 1–8, Oct. 2008.

[52] S. Kamamura, Y. Koizumi, D. Shimazaki, T. Miyamura, S. Arakawa, K. Shiomoto, A. Hiramatsu, and M. Murata, "Attractor selection-based virtual network topology control with dynamic threshold reconfiguration for managed self-organization network," in *Proceedings of the 24th International Teletraffic Congress*, pp. 1–6, Sept. 2012.

[53] B. Li, W. Lu, S. Liu, and Z. Zhu, "Deep-learning-assisted network orchestration for on-demand and cost-effective vNF service chaining in inter-DC elastic optical networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, pp. D29–D41, Oct. 2018.

[54] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Proceedings of IEEE CloudNet*, pp. 171–177, Oct. 2015.

[55] L. Nonde, T. El-Gorashi, and J. Elmirghani, "Energy efficient virtual network embedding for cloud networks," *IEEE Journal of Lightwave Technology*, vol. 33, pp. 1828–1849, May 2015.

[56] J. Whiteaker, F. Schneider, and R. Teixeira, "Explaining packet delays under virtualization," *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 38–44, Jan. 2011.

[57] G. Wang and T. Ng, "The impact of virtualization on network performance of Amazon EC2 data center," in *Proceedings of IEEE INFOCOM*, pp. 1–9, Mar. 2010.

[58] S. Farokhi, E. B. Lakew, C. Klein, I. Brandic, and E. Elmroth, "Coordinating CPU and memory elasticity controllers to meet service response time constraints," in *Proceedings of IEEE ICCAC2015*, pp. 69–80, Sept. 2015.

[59] Y. Baram, "Orthogonal patterns in binary neural networks," *NASA Technical Memorandum No. 100060*, Mar. 1988.

[60] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. D. Laat, J. Mambretti, I. Monga, B. V. Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Generation Computer Systems*, vol. 22, pp. 901–907, Oct. 2006.

[61] M. Yu, Y. Yi, J. Rexford, and M. Chiang, “Rethinking virtual network embedding: Substrate support for path splitting and migration,” *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 17–29, Mar. 2008.