# Managed Self-organizing Control Method Inspired by Adaptive Biological Behavior

Submitted to

Graduate School of Information Science and Technology

Osaka University

January 2016

Naomi KUZE

# List of publication

## Journal papers

1. Naomi Kuze, Daichi Kominami, and Masayuki Murata, "A predictive mechanism for enhancing adaptability of self-organised routing," *International Journal of Bio-Inspired Computation*, vol. 6, no. 6, pp. 384–396, January 2015.

2. Naomi Kuze, Daichi Kominami, Kenji Kashima, Tomoaki Hashimoto, and Masayuki Murata, "Controlling large-scale self-organized networks with lightweight cost for fast adaptation to changing environments," *to appear in ACM Transaction on Autonomous and Adaptive Systems*, 2015.

## Refereed Conference Papers

1. Naomi Kuze, Naoki Wakamiya, and Masayuki Murata, "Proposal and evaluation of ant-based routing with prediction," in *Proceedings of the Fifth International Workshop on Guided Self-Organization (GSO-2012)*, September 2012.

2. Naomi Kuze, Naoki Wakamiya, and Masayuki Murata, "Proposal and evaluation of ant-based routing with autonomous zoning for convergence improvement," in *Proceedings of The 15th International Conference on Network-Based Information Systems (NBiS-2012)*, pp.290–297, September 2012.

3. Naomi Kuze, Naoki Wakamiya, Daichi Kominami and Masayuki Murata, “Proposal and evaluation of a predictive mechanism for ant-based routing,” in *Proceedings of International Conference on Emerging Network Intelligence (EMERGING 2013)*, pp. 7–12, September 2013.

4. Naomi Kuze, Daichi Kominami, Kenji Kashima, Tomoaki Hashimoto, and Masayuki Murata, “Enhancing convergence with optimal feedback for controlled self-organizing network,” in *Proceedings of The 2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, pp. 1–7, September 2014.

5. Naomi Kuze, Daichi Kominami, Kenji Kashima, Tomoaki Hashimoto, and Masayuki Murata, “Hierarchical optimal control method for controlling self-organized networks with lightweight cost,” in *Proceedings of IEEE Global Communications Conference (GLOBECOM 2015)*, December 2015.

6. Naomi Kuze, Shu Ishikura, Takeshi Yagi, Daiki Chiba, and Masayuki Murata, “Crawler Classification using Ant-based Clustering Scheme,” in *Proceedings of International Conference for Internet Technology and Secured Transactions (ICITST-2015)*, December 2015.

## Non-Refereed Technical Papers

1. Naomi Kuze, Naoki Wakamiya, and Masayuki Murata, “Proposal and evaluation of ant-based routing with autonomous zoning,” *Technical Report of IEICE (IN2011-196)*, vol. 111, no. 469, pp. 353–358, March 2012 (in Japanese).

2. Naomi Kuze, Daichi Kominami, Kenji Kashima, Tomoaki Hashimoto, and Masayuki Murata, “Robustness of potential-based routing with model predictive control,” *Technical Report of IEICE (IN2013-195)*, vol. 113, no. 473, pp. 305–310, March 2014 (in Japanese).

3. Naomi Kuze, Daichi Kominami, Kenji Kashima, Tomoaki Hashimoto, and Masayuki Murata,

"Potential-based routing with optimal feedback using reduced order model for controlled self-organizing networks," *Technical Report of IEICE (IN2014-32)*, vol. 114, no. 139, pp. 13–18, July 2014 (in Japanese).

4. Naomi Kuze, Shu Ishikura, Takeshi Yagi, Daiki Chiba, and Masayuki Murata, "Website vulnerability scanning detection inspired by biological adaptation toward diversifying communication services," *Computer Security Symposium 2015 (CSS 2015)*, October 2015 (in Japanese).

# Preface

The Internet originated from the ARPANET becomes an increasingly important infrastructure with the growth of device and communication technologies. However, information networks including the Internet remain to be based on the conventional architecture which is designed for limited and simple conventional networks. Temporary expansions of systems according to additional requirements for new services and applications are insufficient for retaining the durability and controllability of systems.

For future evolution of information networks, therefore, we need to develop a new architecture which has high scalability, adaptabiity, robustness, and flexibility. For such a purpose, self-organization that is natural phenomenon are focused on. Specifically, biological systems can evolve according to their habitats, and moreover, can adapt flexibly to environmental changes including unexpected ones. Adopting such bio-inspired mechanisms, network systems are therefore highly expected to work even in unexpected situations and adapt to dynamical changing environment.

In spite of such benefits, it is pointed out that such systems have several limitations that complicate practical use in industrial and business systems. Since a global behavior or pattern emerges as a result of local interactions among components, (1) the global optimality is not guaranteed, (2) it takes a lot of times for a global behavior or pattern to emerge, and (3) the adapation to environmental changes is slow, especially in large-scale systems. Such disadvantages brought about the idea of managed self-organizing systems, where self-organizing systems are managed through

some constraints.

In this thesis, we study managed self-organizing control methods inspired by adaptive biological behavior for large-scale networks. We first introduce a predictive mechanism to self-organizing systems, as a means of managing self-organization, for faster adaptation to envornmental changes. For this purpose, we take AntNet that is a self-organizing routing scheme inspired by foraging behavior of ants and extend it by newly introducing a predictive mechanism into AntNet. Then we show that faster convergence can be attained even in changing environments through managing the direction of the self-organizing system. In the predictive mechanism, each component predicts the future states, e.g., location of neighbors, with past behaviors of them and behaves in accordance with the predicted states. Such a predictive mechanism allows the whole system to achieve fast adaptation to dynamical changes of the environment. We adopt the predictive mechanism for a node to predict the future convergence from history of pheromone accumulation. In the original AntNet, control messages called ants explore the network in accordance with pheromones and lay down pheromones on their trails, i.e., paths from a source node to a destination node. A shorter path collects more pheromones than longer paths, and it attracts more ants which further deposit pheromones on the path. Therefore, the increase rate of pheromone implicitly indicates the goodness of a path. In our proposal, each node predicts a path which will obtain the largest amount of pheromone from historical information about pheromone accumulation. Then, it is likely to boosts pheromone accumulation on the predicted path to have faster convergence. Our proposal is shown to have a higher convergence property than the case without the predictive mechanism through simulation evaluation.

We next consider a bio-inspired clustering scheme to crawler classification and prove its advantages through simulation evaluation using data collected in a real network. Web attacks are increasing rapidly with the expansion of web services, and therefore it is necessary to collect and analyze web attacks in order to detect unknown malicious threats. Web honeypots are often deployed for that purpose, although they also collect many normal accesses such as those by search engine crawlers. It is essential to develop a means of identifying malicious accesses automatically from collected data. In this study, we focus on detecting accesses by crawlers whose behavior is similar to vulnerability scanning because they are difficult to be distinguished from vulnerability scanning. For this purpose, we use AntTree, a bio-inspired clustering scheme that has high scalability and adaptability, for crawler detection. Through evaluation using data collected in a real network, we show that AntTree can detect accesses by crawlers more precisely than a conventional scheme.

Based on studies mentioned above, we consider a more general control method which can be adopted in various self-organizing systems. For practical use of self-organizing systems, enhancing the convergence speed of self-organizing systems is a challenging task as mentioned above. Therefore, we introduce an optimal feedback scheme applicable to the self-organizing systems for faster convergence. In the optimal feedback scheme, an external controller which observes the states of a network, estimates the network dynamics, and provides feedback inputs to the network. We propose the managed control framework for the self-organizing systems, and through simulation evaluation, the convergence speed of potentials is extremely enhanced by the optimal feedback scheme by considering potential-based routing as an example.

Moreover, we propose a hierarchical optimal feedback scheme which is an improvement version of our proposed managed control framework. In the hierarchical optimal feedback scheme, a network is divided to several sub-networks, which are controlled in a hirarchical manner by two types of controllers, a central controller and several sub-controllers. A sub-controller observes states of a sub-network, estimates the sub-network dynamics, and provides feedback inputs to the sub-network. Since each sub-controller needs not to control the corresponding sub-network, i.e., only a

part of the entire network, the computation cost for a sub-controller is much smaller than that for the external controller of the previous scheme. Therefore, the convergence speed of self-organization can be enhanced with low cost by introducing the hierarchical optimal feedback scheme. This indicates that the hierarchical method can accelerate convergence even in the large-scale systems by keeping the inherent nature of the self-organizing systems. That is, the emergence is achieved mainly by local interactions, but at the same time, the problem of slow convergence is resolved by controlling the entire system to some extent. Through simulation evaluation, we prove that potentials can converge almost as fast as the previous (non-hierarchical) scheme with low computation cost by introducing the hierarchical optimal control method.

# Acknowledgments

This thesis could not have been accomplished without the assistance of many people, and I would like to acknowledge all of them.

First of all, I would like to express my great gratitude to my superviser, Professor Masayuki Murata, for his generous guidance and insightful comments throughout my Ph.D.

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.

Also, I would like to express my sincere appreciation for Assistant Professor Daichi Kominami of Graduate School of Economics, Osaka University. Without his continuous advices and support, I would not have entered the Ph.D. program. I would like to thank Associate Professor Kenji Kashima of Graduate School of Informatics, Kyoto University, and Lecturer Tomoaki Hashimoto of Faculty of Engineering, Osaka Institute of Technology for their specialized comments and support from a viewpoint of control system theory. I am thankful to Dr. Takeshi Yagi, and Mr. Daiki Chiba of NTT Secure Platform Laboratories for their support from the perspective of network security. I am very grateful to Professor Naoki Wakamiya of Graduate School of Information Science and Technology, Osaka University. He gave me a fundamental principle in my study.

Furthermore, I must acknowledge Associate Professor Shin'ichi Arakawa, Associate Professor

Go Hasegawa, Assistant Professor Yuichi Ohsita, Assistant Professor Yuki Koizumi of Graduate School of Information Science and Technology, Osaka University, and Dr. Kenji Leibnitz of National Institute of Information and Communications Technology for their valuable comments and suggestions on my study.

I express my appreciation 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 cannot conclude my acknowledgement without expressing my thanks to my parents and family. Thank you for your giving me invaluable supports throughout my life.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

Information networks have been and are growing rapidly in scale and complexity with the expansion of network-based services and the development of devices and communication technologies. This leads to diversification of network-based services and applications that make our lives more comfortable, productive, and safe. In our life environment, wireless sensing and actuation systems are embedded everywhere. Sensors monitor environmental information, not only temperatures and humidity, but also human condition, and actuators control environment using sensed information [3,4]. In the industrial field, information networks are used for managing infrastructures such as power grid, transportation, and water supply [5]. The Internet is being no longer a network of computers, but of various devices such as sensors, actuators, electric grid, smart-phones, vehicles, and electrical appliances. They communicate with each other with wired or wireless communications and construct massive (city or country scale) networks [6, 7]. As the scale and complexity of the information networks increase, it becomes more and more difficult to collect all information of such networks and configure the whole of them according to requirements of the network due to considerable overhead of computation and communication. This indicates that conventional network control methods and techniques based on the global information of the network are becoming unsuccessful and unfeasible. Therefore, we need to develop a new network architecture which has high scalability, adaptability, and robustness for future growth of information networks. As the size of networks increases, the *scalability*, the ability of systems to work regardless of the network size, is indispensable. The increasingly diversification of network-based services complicates the manual configuration of dynamically changing network systems according to various requirements. Therefore, it is essential for systems to configure themselves automatically. We call this property the *adaptability*. Moreover, since the stop of network-based services can occur not only the dissatisfaction of customers but also a serious trouble with our lives, systems need the *robustness*, the ability to keep working even when a part of systems fails.

For realizing a new network architecture, much attention is paid on self-organizing systems which have high scalability, adaptability, and robustness [8]. In self-organizing systems, each component behaves individually and autonomously with simple rules using only local information. A global pattern or behavior emerges in a macroscopic level through local interactions among components. In self-organizing systems, up-to-date information regarding the entire system or many other components is unnecessary, which considerably reduces computational cost and communication overhead for collecting global information. Therefore, the control overhead of each component in self-organizing systems is small regardless of the network scale, which results in high scalability of these systems. Moreover, the localized control leads to a capability of handling local failures and small environmental changes. Thus, self-organizing systems are expected to automatically recover from failures and adapt to environmental changes, without involving centralized control. This is a reason why self-organizing systems have high robustness and adaptability. Especially, biological systems are inherently self-organizing and network control inspired by these systems is promising idea for realiBecause ofzing scalable, adaptable and robust network systems. Because of these advantages, many researchers study autonomous and distributed systems based on a variety of self-organization models including bio-inspired ones [9,10], and furthermore, they have been applied to information networking such as routing, synchronization, and task assignment [11,12].

Although self-organization control without global knowledge of the current network state has various benefits, such control has critical limitations that complicate practical use in industrial and business systems [13]. It may take a long time for global patterns to emerge in large-scale systems because they appear as a consequence of interactions among autonomous components. This property also leads to slow adaptation to large environmental changes, which is difficult to solve solely by local interactions in self-organizing systems. Also, self-organizing systems that use only local information sometimes fall into local optima. On the other hand, although conventional systems using global information of the network can reach an optimal solution, the required computational cost to do so is often unrealistic. These limitations of self-organizing systems raised from real applications bring about the idea of managed self-organization[1], where the self-organizing system is controlled through some constraints [14–16].

The goal of this thesis is to study managed self-organizing systems inspired by biological behavior for future growth of information networks.

### Self-organizing Network Control

We can find various types of self-organizing systems in nature [17, 18]. Although most of such systems are extremely large in scale and have complex structures, local interactions among components result in a well-ordered pattern or behavior at a global level without any central control. Many researchers are trying to apply these mechanisms to artificial systems.

Specifically, biological systems are inherently self-organizing and the biology is one of mines of self-organization models that can be applied to systems and technologies in various fields such as

[1]It is sometimes called controlled or guided self-organizing systems in the literature.

chemistry, economics and engineering. In the field of information networks, many researchers focus on the scalability, diversity, and flexibility of biological systems. Swarms of such as social insects, birds, and fish are well-known self-organizing system called “Swarm Intelligence”. Ants, a typical species of social insects, construct routes between their nests and food spots in a self-organizing manner [19, 20]. Ant colony optimization (ACO), i.e., a heuristic algorithm inspired by foraging ants, is applied to many routing mechanisms [1, 21–24]. Adaptive response of a gene network to environmental changes brought attractor selection method [25]. This method is applied to various types of network control, e.g., routing [26, 27] and topology control in virtual networks [28]. Moreover, pulse coupled oscillator model [29] inspired by glow synchronization of fireflies is adopted to synchronization control for sensor nodes [30].

### Managed Self-organizing Network Control

Managed (controlled, guided) self-organization is a novel idea for realizing the implementation of self-organizing systems in industrial fields. In managed self-organizing systems, self-organizing systems are guided to achieve the desired state [14–16].

A typical scheme for managing self-organizing systems is to introduce an external observer/ controller which monitors systems and provides feedback inputs to them according to requirements. For example, the authors of [31, 32] use the concept of controlled self-organization, where an external observer/controller guides self-organizing optical network [31] and sensor network [32] systems through a feedback mechanism that leads them to a desired state. The self-organizing system can then be managed through fully observed information. However, enhancing the convergence speed to reach an optimal or semi-optimal solution with light-weight cost remains as an outstanding task.

## 1.2 Outline of Thesis

### A Predictive Mechanism for Enhancing Adaptability of Self-organized Routing [33–38]

In Chapter 2, we first introduce a predictive mechanism to self-organizing systems for faster adaptation to environmental changes, and examine the property and the limit of self-organizing systems without any centralized control. For this purpose, we take AntNet [1] that is a self-organizing routing scheme inspired by foraging behavior of ants, a typical species of social insects, and extend it by newly introducing a predictive mechanism into AntNet. Then we show that faster convergence can be attained even in changing environments through managing the direction of the self-organizing systems. Humans have the ability of predicting the future states using information of past occurrences which they perceived before [39, 40]. In [41], a predictive mechanism was proposed for faster consensus in flocking birds. In self-organized flocking with a predictive mechanism, each member predicts the future states, e.g., location of neighbors, with past behaviors of them and behaves in accordance with the predicted states. Such a predictive mechanism allows natural organisms to achieve fast adaptation to dynamical changes of the environment. We adopt the similar idea for a node to predict the future convergence from history of pheromone accumulation. In the original AntNet, control messages called ants starting from a source node initially explore the network randomly. Arriving at their destination node, they return to their source node while laying down pheromone trails. Following ants explore the network stochastically in accordance with pheromones. In details, they are likely to follow pheromone trails. Since a shorter path collects more pheromones than longer paths, it attracts more ants which further deposit pheromones on the path. Therefore, the increase rate of pheromone implicitly indicates the goodness of a path. In our proposal, each node predicts a path which will obtain the largest amount of pheromone from historical information about pheromone accumulation. Then, it is likely to boosts pheromone accumulation on the predicted path to have faster convergence. Our proposal is shown to have a higher convergence property than the case without the predictive mechanism through simulation evaluation.

### Website Vulnerability Scanning Detection Inspired by Biological Adaptation Toward Diversifying Communication Services [42, 43]

We next consider a bio-inspired clustering scheme to crawler classification and prove its advantages through simulation evaluation using data collected in a real network. Web attacks are increasing rapidly with the expansion of web services, and therefore it is necessary to collect and analyze web attacks in order to detect unknown malicious threats. Web honeypots are often deployed for that purpose, although they also collect many normal accesses such as those by search engine crawlers. It is essential to develop a means of identifying malicious accesses automatically from collected data. In this study, we focuse on detecting accesses by crawlers whose behavior is similar to vulnerability scanning because they are difficult to be distinguished from vulnerability scanning. For this purpose, we use AntTree, a bio-inspired clustering scheme that has high scalability and adaptability, for crawler detection. Through evaluation using data collected in a real network, we show that AntTree can detect accesses by crawlers more precisely than a conventional scheme.

### Controlling Large-scale Self-organized Networks with Lightweight Cost for Fast Adaptation to Changing Environment [44–46]

Based on studies mentioned above, we consider a more general control method which can be adopted in various self-organizing systems in Chapter 4. For practical use of self-organizing systems, enhancing the convergence speed of self-organizing systems is a challenging task as mentioned above. Therefore, we introduce an optimal feedback scheme applicable to the self-organizing systems for faster convergence. In the optimal feedback scheme, an *external controller* collects information regarding the network such as node states and network topology via a partial set of nodes directly monitored by the controller, and estimates system dynamics using a mathematical model that describes the network dynamics, then determines optimal control inputs based on robust control theory [47] for facilitating the convergence. Optimal feedback mechanisms for controlling dynamical systems have been researched for long years in the field of control theory [48]. A controller monitors a system and provides an optimal control feedback that minimizes the cost function [48] based on a mathematical model of the system. Note that there are various errors such as modeling errors and unexpected disturbances so that the optimal control feedback is not necessarily optimal at any time in real systems.

The contribution of this study is to converge self-organizing network systems with lightweight cost while retaining a high performance (such as high convergence and adaptability in our proposal) of robust control. To that end, we first regulate the area in which the external controller collects node states to reduce the cost for collecting information. If the controller collects all node states, the communication overhead is extremely large and traffic congestion would occur. Therefore, we limit the area from which the controller can collect information to nodes that can be reached within several hops from the monitored nodes. The controller can estimate the states of all nodes from only a part of them given the network topology, because the state of each node is determined in accordance with local interactions based on prescribed rules [47]. That is, the controller acquires network information with lower communication overhead, and then calculates optimal feedback inputs. In our scheme, a controller has an internal model estimating the actual network system. The order $(N_{dim})$ of that model is proportional to that of the actual network model, namely the number of nodes. Note that we need $O({N_{dim}}^2)$ for controller state updates at each time instance, and also $O({N_{dim}}^3)$ computation for controller re-design when the network topology changes. The computational cost is thus extremely large in large-scale networks when we use the original model, whose order is proportional to the number of nodes. We therefore reduce the order of the network model (consequently that of $N_{dim}$) to decrease the computational cost via a model reduction technique [49]. We propose the managed control framework for the self-organizing systems, and through simulation evaluation, the convergence speed of potentials is extremely enhanced by the optimal feedback scheme by considering potential-based routing as an example.

### Hierarchical Optimal Control Method for Controlling Large-scale Self-organizing Networks [50]

In Chapter 5, we propose a hierarchical optimal feedback scheme which is an improvement version of our proposed maneged control framework. In the previous scheme, the computational cost for designing the estimation model is roughly proportional to the cube of the number of nodes, which makes it increasingly difficult for the external controller to collect topology information and to obtain the estimation model of the whole network as the network size becomes larger. These problems become critical when topological changes occur, because the estimation model needs to be redesigned to include the latest topology information so that the external controller can guide the network to converge to a targeted state.

The basic idea of the hierarchical scheme is to divide a network to several sub-networks and to divide the external controller's roles to two types of controllers, i.e., a *central controller* and *sub-controllers*. Each sub-controller monitors a different sub-network, that is, only a part of the entire network, and provides optimal feedback inputs to the sub-network so that fast convergence can be achieved. Specifically, a sub-controller collects information regarding the corresponding sub-network, such as node states and network topology, via a partial set of nodes monitored by the sub-controller. Then, the sub-controller estimates the dynamics of the sub-network by using a mathematical model that describes the sub-network dynamics to calculate optimal feedback control inputs that facilitate the convergence speed of self-organizing systems. This is based on robust control theory [47]. In contrast, the role of a central controller is to guide sub-networks to achieve the identical global optimality. If each sub-controller determines the optimal inputs selfishly, the interactions among the sub-networks may cause network instability. This is because each individual sub-controller can monitor the node states of only its own sub-network, even though sub-networks actually interact with each other. A central controller obtains information about the network from sub-controllers, estimates the degrees of interactions among sub-networks, and then returns feedback inputs to the sub-controllers. Through simulation evaluation, we prove that potentials can converge almost as fast as the previous (non-hierarchical) scheme with low computation cost by introducing the hierarchical optimal control method. This indicates that the hierarchical method can accelerate convergence even in the large-scale systems by keeping the inherence nature of the self-organizing systems. That is, the emergence is achieved mainly by local interactions, but at the same time, the problem of slow convergence is resolved by controlling the entire system to some extent.

Finally, in Chapter 6, we present our conclusions and suggest areas for future work.

# Chapter 2

# A Predictive Mechanism for Enhancing Adaptability of Self-organized Routing

## 2.1 Introduction

Rapidly increasing network scale and complexity pose significant limitations for conventional network systems and technologies based on central or distributed control. As scale and complexity increase, information network systems adopting conventional control technologies in particular suffer from considerable overhead in managing up-to-date information to grasp changing conditions. There has been active research regarding problems likely to emerge in future networking, including GENI [51] and NSF FIA [52] in the United States, FP7 [53] in Europe, and the NWGN (New-Generation Network) Project [54] in Japan, in order to establish a novel network architecture and relevant technologies. Future network requirements such as scalability, adaptability, robustness, and sustainability, will require new methods of organising and controlling network systems in a fully distributed and *self-organising manner*. In particular, realisation of network systems that can quickly adapt to changing conditions will require systems that can consider their own future state.

Self-organisation is a natural phenomenon of distributed systems, where components behave individually and autonomously. In a self-organising system, each component follows simple rules using locally available information. Through direct or indirect interactions among components, global behaviour patterns emerge on a macroscopic level without central control. In a self-organising system, the cost of information management can be considerably reduced, because up-to-date information regarding the entire system or many other components is unnecessary. Moreover, local failures and small environmental change are handled locally and immediately by local components. Therefore, a self-organising system is expected to automatically recover from failures and adapt to environmental change, without involving centralised control. This property is quite important for realising future networks. In particular, biological systems are inherently self-organising, implementing self-organisation models that can be applied to information networking such as routing, synchronisation, and task assignment [11, 12, 55–57].

Although self-organisation has various benefits, such control has critical limitations that complicate implementation in industrial and business systems [13]. For example, it may take a long time for global patterns to emerge in large-scale systems, because they appear as a consequence of interactions between autonomous components. Also, self-organising systems that use only local information can fall into a local optima, while conventional systems using global information can often reach an optimal or semi-optimal solution. These disadvantages can lead to slow adaptation to environmental change in self-organising systems. Therefore, we introduce a predictive mechanism to self-organising systems in order to improve adaptation to environmental change, one challenging problem for implementation of self-organising control. Some bio-groups are known to have the ability of prediction, that is, each component predicts future behaviour of its neighbors from their past behaviour, and adapts movement to conform to the predicted behaviour. Zhang et al., proved that a predictive mechanism incorporated in a model of flocking birds made its convergence speed faster [41]. In self-organised flocking with a predictive mechanism, each component achieves faster self-adaptation to environmental changes. It is pointed out that *'historical local information* *is equivalent to current global information'*, and in this sense, a predictive mechanism can contribute to faster self-adaptation to environmental change with only local information when applied to self-organised behaviour of flocking birds. Montague et al., and Summerfield et al., also provide investigation of self-organisation with prediction in the field of biology [39,58], but the introduction of prediction to self-organised information network systems needs further discussion.

In this chapter, we show advantages obtained by introducing a predictive mechanism to self-organizanising network systems through inclusion of a predictive mechanism in AntNet [1], which is a self-organising routing mechanism based on ant colony optimisation (ACO) and does not have predictive feature inherently. ACO, a heuristic in the travelling salesman problem, is a mathematical model of foraging behaviour of ants [59–61], and many researchers have applied ACO to routing mechanisms in information networks due to similarities between the two systems [23, 62, 63]. Previous research shows that AntNet [1] is superior to conventional mechanisms regarding robustness against failure, control overhead, and communication performance [64]. However, the time required for path establishment to converge depends on the length of the path, defined as the minimum hop count from a source node to a destination node [65]. Moreover, the large amount of control messages generated during path establishment depletes network bandwidth, hindering data message transmission [1]. It is therefore essential to accelerate convergence in self-organising systems. There are various other ACO-based routing mechanisms, but they too have the same inherent problems as AntNet. AntHocNet [24] is one such example for wireless networks, the target environment in this study, but we use AntNet because it is one of the most typical ACO-based routing mechanisms, and is designed with simple rules. These features allow easy implementation to other ant-based routing mechanisms [66–68]. Of course, ACO and AntNet have tuning parameters for accelerating convergence, but fast convergence via parameters introduces further control overhead necessary for exchanging local information between adjacent nodes to move the entire system to a new stable state. Accordingly, we need another acceleration mechanism that introduces little overhead, one that performs only when it is likely that fast convergence to a new environment is actually necessary. For this purpose, we can easily introduce a predictive mechanism by which each component in a self-organising system behaves according to a future state predicted from past information. Unlike [41], we take an ant-based routing mechanism in a wireless network scenario.

In an ant-based routing mechanism, a shorter path collects more pheromones than do longer paths. The accumulated pheromones attract more ants, which further deposit pheromones on the path, and this positive feedback eventually leads to all ants following a single path; increased pheromone values implicitly indicate the goodness of a path. In our mechanism, each node predicts a path that will obtain a large amount of pheromones from historical information about pheromone accumulation. Nodes then boost pheromone increases on the predicted path for faster convergence. Self-organising control with a predictive mechanism is thus not only tolerant, but also highly adaptable to environmental change. We show that prediction promotes adaptation to environmental change through simulation experiments on grid and random networks, and that paths established by prediction are more optimal than those formed by the original AntNet. Moreover, we show that the control overhead of predictive ants can be reduced, because prediction reduces the control overhead of forward and backward ant transmissions.

The remainder of this chapter is organised as follows: First, we describe the original AntNet in Section 2.2, and propose and explain a predictive mechanism using increased pheromones for AntNet in Section 2.3. We then evaluate the adaptability of the proposed method through simulation, and give results and a discussion in Section 2.4. Finally, in Section 2.5 we present our conclusions and suggest areas for future work.

## 2.2 Self-organized Routing Method: AntNet

We consider AntNet as the basis of our investigation of self-organisation with prediction. In this section, we give a brief summary of the AntNet mechanism.

### 2.2.1 Overview

AntNet [1] is an adaptive best-effort routing algorithm in packet-switched networks based on the principles of ACO. AntNet introduces two types of control messages called 'ants,' *forward ants* and *backward ants*. A source node proactively launches mobile agents called forward ants at regular intervals. A forward ant stochastically selects a neighbour node to visit according to the amount of *pheromones*, which are laid by ants. On the way to a destination node, a forward ant records its path and the time of arrival at each node to evaluate the quality of the travelled path. When a forward ant arrives at a destination node, it changes into a backward ant. A backward ant returns to the source node on the loop-free reverse path of the forward ant, updating pheromone values along the way. When a path has better quality (smaller delay), a backward ant increases the pheromone value of the neighbour node it came from.

Each data packet is forwarded to a neighbour node as a next-hop node according to pheromone values that backward ants have updated. Since a neighbour node with a larger pheromone value is more likely to be selected, data packets reach destination nodes following a shorter path.

### 2.2.2 Path Establishment and Maintenance

In AntNet, each node has a pheromone table $\mathcal{T}^k$ for routing information. $\mathcal{T}^k = \{\mathcal{T}_d^k\}$, where $\mathcal{T}_d^k$ is a list of pheromone values $\tau_{nd}^k \in [0, 1]$ for all neighbour node $n \in N_k$ (a set of neighbour nodes of node $k$) regarding destination node $d$, $\mathcal{T}_d^k = \{\tau_{nd}^k\}$. Source node $s$ establishes and maintains a path to destination node $d$ by sending forward ants at regular intervals. A forward ant stochastically selects a next hop node to visit. The probability $p_{nd}$ that neighbour node $n \in N_k$ is selected as a next hop node of node $k$ for destination node $d$ is given as follows: If there is no pheromone information for destination node $d$ at node $k$, a next hop node is randomly chosen.

$$p_{nd} = \begin{cases} 1, & \text{if } |N_k| = 1 \\ \frac{1}{|N_k|-1}, & \text{if } |N_k| > 1 \wedge n \neq v_{i-1} \\ 0, & \text{otherwise} \end{cases}, \tag{2.1}$$

where $v_{i-1}$ is an identifier of the $(i-1)$-th node that the forward ant visited just before arriving at node $k$ at the $i$-th step. Otherwise, selection is performed based on the pheromone value $\tau_{nd}$:

$$p_{nd} = \begin{cases} 1, & \text{if } |N_k| = 1 \\ \frac{1}{|N_k|-1}, & \text{if } |N_k| > 1 \wedge {}^{\forall}n \in V_{s\to k} \wedge n \neq v_{i-1} \\ \frac{\tau^k_{nd}+\alpha l_n}{1+\alpha(|N_k|-1)}, & \text{if } |N_k| > 1 \wedge {}^{\exists}n \notin V_{s\to k} \\ 0, & \text{otherwise} \end{cases}, \tag{2.2}$$

where $V_{s\to k} = \{s, v_1, v_2, \cdots, v_{i-1}\}$ is a list of nodes that the forward ant visited before arriving at node $k$. $l_n$ is a variable indicating the degree of the congestion for neighbour node $n$ at node $k$, which is given by $1 - \frac{q_n}{\sum_{j\in N_k} q_j}$, and $q_n$ is the number of messages waiting in a sending buffer for neighbour node $n$. $\alpha \in [0, 1]$ is a coefficient. From Eq. (2.1), it is clear that a larger $\alpha$ allows forward ants to select a next hop node according to local traffic conditions. As a consequence, path convergence becomes difficult. In contrast, with $\alpha$ close to zero, a path traversing congested links would be established. A forward ant whose travelled hop count reaches the predetermined TTL is discarded at a node.

A forward ant changes to a backward ant when it reaches destination node $d$, and returns to source node $s$ following the loop-free path it traversed as a forward ant. While doing this, it updates pheromone values at visited nodes. The pheromone value $\tau^k_{nd}$ for neighbour node $n \in N_k$ at node $k$

is updated by Eq. (2.3):

$$\tau_{nd}^{k} \leftarrow \begin{cases} \tau_{nd}^{k} + r(1-\tau_{nd}^{k}), & \text{if } n = f \\ \tau_{nd}^{k} - r\tau_{nd}^{k}, & \text{otherwise} \end{cases}, \tag{2.3}$$

where $f$ corresponds to the previous node that the backward ant visited just before arriving at node $k$, the first node of the path from the node to the destination node. $r$ reflects the goodness of the path, based on the transmission delay from node $k$ to destination node $d$; the smaller the delay is, the larger $r$ is. (See below for a detailed definition.) Consequently, the shortest path among paths that forward ants have found has the largest amount of pheromones and attracts the most forward ants. In AntNet, as the cycle of this positive feedback mechanism becomes shorter by decreasing the interval of forward ant emissions $\Delta t_f$, the speed of convergence becomes faster but the control overhead of forward and backward ants increases, which greatly hinders data transmission. Moreover, further control overhead occurs with increased network size, because the time required for forward ants to wander from sources to destinations rises exponentially, resulting in longer convergence time. We therefore propose in the following section a predictive mechanism that accelerates the convergence speed of established paths while retaining low control overhead.

The parameter $r$, which determines the amount of pheromone increase, is calculated from the ant's trip time from node $k$ to destination node $d$, which is defined as $T_{k\to d}$, and the local statistical model $\mathcal{M}^k$, which is given by $\{\mathcal{M}_d^k\} = \{W_k^d, \mu_d^k, \sigma_d^k\}$, where $W_k^d$ is the best travelling time from node $k$ to destination $d$ over the last observation window of size $w$, and $\mu_d^k$ and $\sigma_d^k$ are respectively the average and dispersion of travelling times over the last observation window $w$. Specifically, $r$ is given by

$$r = c_1\left(\frac{W_k^d}{T_{k\to d}}\right) + c_2\left(\frac{I_{sup}-I_{inf}}{(I_{sup}-I_{inf})+(T_{k\to d}-I_{inf})}\right), \tag{2.4}$$

where $I_{sup}$ and $I_{inf}$ are estimates of the limit of an approximate confidence interval for $\mu$, given by

$$I_{inf} = W_k^d, \tag{2.5}$$

$$I_{sup} = \mu_d^k + z(\sigma_d^k/\sqrt{w}), \tag{2.6}$$

where $c_1$, $c_2$, and $z$ are coefficients, and $(c_1, c_2, z)$ is set to $(0.7, 0.3, 1.7)$ following [1]. Moreover, $r$ is squashed by means of a function $s(x)$:

$$s(x) = \left(1 + \text{exp}\left(\frac{a}{x|N_k|}\right)\right), \tag{2.7}$$

$$r \leftarrow \frac{s(r)}{s(1)}, \tag{2.8}$$

where $a$ is a coefficient.

In this study, we use $m \times r$ $(m \in (0, 1])$ as a substitute for $r$ in Eq. (2.3). For fast convergence of pheromones against environmental change, $m$ may be set to 1.0 as in the original AntNet. However, we found that doing so leads to large path fluctuations. We therefore set $m$ to a rather moderate value of 0.5, which we expect to result in fast convergence without path fluctuations in the proposed predictive mechanism.

A data message is forwarded to a next hop node based on pheromone values, where the selection probability $R_{nd}^k$ that neighbour node $n$ is chosen as the next hop node toward destination node $d$ is given as $\frac{(\tau_{nd}^k)^\epsilon}{\sum_{j \in N_k}(\tau_{jd}^k)^\epsilon}$ $(\epsilon \geq 0)$. Therefore, data messages follow the shortest path among paths that forward ants found.

## 2.3 Predictive Mechanism for AntNet

In this section, we propose a predictive mechanism for AntNet. We consider prediction only from pheromone changes (Figure 2.1) and updates of pheromones independent of the internal control in AntNet.

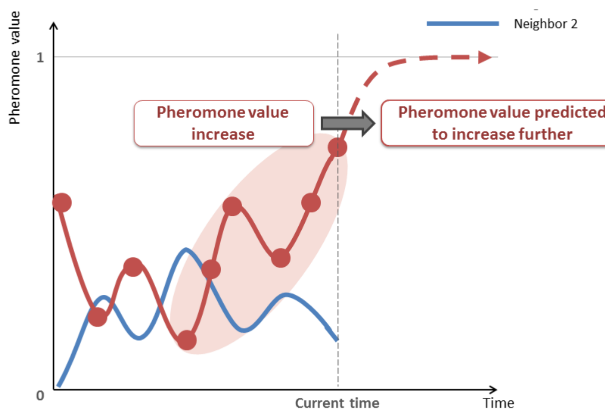


Figure 2.1: Prediction with increased pheromones rate. A path whose pheromones are increasing obtains a large amount of pheromones.

### 2.3.1 Overview

In the original AntNet, it is difficult for the system to quickly adapt to changing network conditions, because each component uses only current local information. Our approach introduces a predictive mechanism in which components observe their past behaviour, predict the future state of the system, and then control their behaviour according to the predicted future state.

In the proposed method, AntNet is modified as follows. We introduce *predictive ants* alongside the forward and backward ants, and use*increased rates of pheromone values* as an indicator for predictive control. Each node launches predictive ants at regular intervals $\Delta t_p$ $(< \Delta t_f)$. A predictive ant arriving at a neighbour node remembers the increased rate of pheromones in the neighbour node, and returns to its originating node. On its return, the predictive ant boosts pheromone accumulation for the neighbour node for faster path convergence if its increase rates are high. The predictive ants increase control overhead, but in turn the overhead caused by forward and backward ants is reduced because prediction shortens the recovery time, and predictive ants can furthermore simultaneously collect the increase rate of different destination nodes. Our predictive mechanism therefore accelerates the convergence speed of path establishment with low control overhead.

Each node uses a pheromone table $\mathcal{T}^k$ as routing information, as in the original AntNet. $\mathcal{T}^k = \{\mathcal{T}_d^k\}$, where $\mathcal{T}_d^k$ is a list of pheromone values $\tau_{nd}^k \in [0,1]$ for all neighbour node $n \in N_k$ regarding destination node $d$,$\mathcal{T}_d^k = \{\tau_{nd}^k\}$. $N_k$ is a set of neighbour nodes of node $k$. $\tau_{nd}^k$ is initialised to $\frac{1}{|N_k|}$. In the proposed method, forward and backward ants behave as in AntNet. Namely, a forward ant stochastically selects a next hop node to visit according to pheromone values by Eq. (2.1) and Eq. (2.2), and pheromone values are updated by backward ants by Eq. (2.3). The pheromone value is used for next-hop selection by ants and data messages.

In the proposed method, each node has an increase rate table $\mathcal{E}^k$ for prediction in addition to the pheromone table $\mathcal{T}^k$. $\mathcal{E}^k = \{\mathcal{E}_d^k\}$, where $\mathcal{E}_d^k$ is a list of increase rates of the pheromone values $e_{nd}^k \in [0,1]$ for all neighbour node $n \in N_k$ regarding destination node $d$. $e_{nd}^k$ is initialised to zero.

Node $k$, which receives a backward ant from node $f \in N_k$, updates the increase rate $e_{nd}^k \in [0,1]$ of all its neighbour nodes $n \in N_k$ regarding destination node $d$ by Eq. (2.9):

$$e_{nd}^k \leftarrow \begin{cases} (1-\beta)e_{nd}^k + \beta, & \text{if } n = f \\ (1-\beta)e_{nd}^k, & \text{otherwise} \end{cases}, \tag{2.9}$$

where $\beta \in [0,1]$ is a parameter that determines the weight of individual pheromone increments. Eq. (2.9) slowly changes the increase rate $e_{nd}^k$, because rapid changes lead to system instability.

### 2.3.2 Pheromone Update by Predictive Ants and Data Transmission

In the proposed method, each node $k$ predicts better paths that will possibly obtain a large amount of pheromones by sending predictive ants to its all neighbour node at regular intervals $\Delta t_p$. A predictive ant that arrives at neighbour node $f \in N_k$ remembers node $f$'s increase rate table, $\mathcal{E}^f$, and returns to its originating node $k$. When the predictive ant returns, the pheromone table of node $k$ is updated. The pheromone value $\tau_{nd}^k$ for neighbour node $n \in N_k$ at node $k$ is updated by Eq. (2.10) if the maximum value in the increase rate table of node $f$ regarding destination node $d$, max $e_{n'd}^f$

($n' \in N_f$), exceeds $\theta_e$.

$$\tau^k_{nd} \to \begin{cases} \tau^k_{nd} + p(1 - \tau^k_{nd}), & \text{if } n = f \\ \tau^k_{nd} + p\tau^k_{nd}, & \text{otherwise} \end{cases} . \tag{2.10}$$

Here, $p$ is a parameter that determines the increasing amount of pheromones. $\theta_e$ is a threshold of the value of an pheromone increase rate. A neighbour node whose increase rate exceeds $\theta_e$ is considered to collect further pheromones. A lower $\theta_e$ allows predictive ants to update pheromone values on the path with a low pheromone increase rate. In consequence, the pheromone values on longer paths may be increased by prediction. On the contrary, it is difficult to adapt to environmental change with higher $\theta_e$. Therefore, $\theta_e$ is set to 0.5 in numerical examples. Even if the max value of $e^f_{n'd}$ exceeds $\theta_e$, the pheromone values are not updated when $\mathcal{E}^f_d$ has not been updated, because node $f$ received a predictive ant from node $k$ the last time. Similar to the pheromone increase rate, the pheromone value $\tau^k_{nd}$ changes slowly as shown in Eq. (2.10) to avoid system instability. Moreover, forward and backward ants rarely visit nodes that have a heavy load or are distant from any source or destination node, because the positive feedback through pheromones causes almost all ants to follow a short path (see Section 2.2). Our predictive mechanism therefore prevents nodes from sending predictive ants when they do not receive backward ants for a fixed period, the interval of forward ants $\Delta t_f \times 10$, to reduce the overhead of predictive ants.

A data message selects a next hop node based on pheromone values as in AntNet: the selection probability $R^k_{nd}$ that neighbour node $n$ is chosen as a next hop node for destination node $d$ is given as $\frac{(\tau^k_{nd})^\epsilon}{\sum_{j \in N_k} (\tau^k_{jd})^\epsilon}$ ($\epsilon \geq 0$). Therefore, data messages follow the shortest paths among those that forward ants found.

## 2.4 Performance Evaluation

This section presents a detailed description of the adaptability of the proposed method by introducing predictive control. For that purpose, we will mainly focus on two metrics: the path recovery time

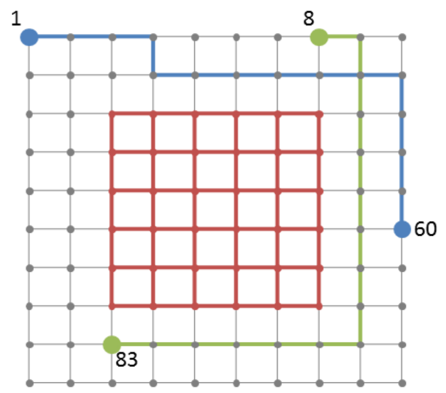


Figure 2.2: An example grid network with 100 nodes. Blue and green paths are target sessions. Background traffic takes place around the network centre with red nodes and links.

from various environmental changes, and control overhead caused by our predictive mechanism as compared to the original AntNet. We use two network models (grid and random networks) in an event-driven simulation written in Visual C++. Using those network models, in Subsection 2.4.2 we focus on performance metrics of one designated session, called the *target session* below. For the target session, packets are passed from a source node to a destination node according to our routing method (Eqs. (2.1) and (2.2)). We assume that other sessions generate packets at each node according a Poisson distribution. Then, in Subsections 2.4.2 and 2.4.3, we will consider the case where traffic from other sessions changes. We also test the case where we explicitly have multiple target sessions. See Figure 2.2 for the grid network in the two target session cases; one is from node 1 to node 60 and the other from node 8 to node 83. In Subsection 2.4.3, we also focus on reduction of the control overhead (in the number of control packets) using a multiple-sessions scenario. In Subsection 2.4.4, we consider the case where several nodes fail simultaneously, to show that the proposed method has high adaptability to environmental change other than traffic changes, such as network failures.

### 2.4.1 Simulation Settings

The simulation focuses on one or more *target session(s)*. In the target session, data packets are generated and sent from source nodes to destination nodes with a Poisson process with an intensity of 1 packet per second. Data packets on the target session follow a path established by forward, backward, and predictive ants. We use the following network configuration in experiments for grid and random networks, respectively.

- Grid network:

  100 nodes are distributed on a $10 \times 10$ grid with 30 m separation.

- Random network:

  338 nodes are deployed uniform-randomly in a $300 \times 300$ m area. The number of nodes is determined so as to compose a connected graph.

In both cases, the communication range of each node is set to 30 m. The bandwidth for each link is 1 Mbps, and the propagation delay is 0.001 s per packet, regardless of the propagation distance. Following [1], the size of all types of ant packets is set to $24+8\times|V|$ bytes, where $|V|$ is the number of nodes visited by the ant packet. The ant packet size increases with the number of visited nodes, because ants remember their visited nodes and arrival times at each node to update pheromone values according to the path delay. The data packet size is set to 1,000 bytes.

In each of the above simulation settings, we confirmed that each path converges to a stable state where forward ants of each target session repeatedly select the same path within 1,000 s from the simulation start. After 1,000 s have passed from the simulation start, we change the environment, increasing traffic in Subsections 2.4.2 and 2.4.3, and simulating a node failure in Subsection 2.4.4. Simulations end at 5,000 s from the simulation start. In the case of traffic changes in Subsections 2.4.2 and 2.4.3, an increase of traffic from *background traffic sessions* occurs around the network centre at 1,000 s from the simulation start. The background traffic sessions are intended to create a hot spot in the network centre to investigate how the routing protocol detours around heavily loaded areas. Specifically, we set the background traffic session to be a session between two adjacent nodes within the network centre. The packet generation rate is 100 packets per second in each direction. See Figure 2.2 for the case of two target sessions and a background traffic session in the 100-node grid network. Because the background traffic sessions are one-hop sessions, their paths are deterministic irrespective of pheromone values, which are thus not affected. In the simulation, the background traffic sessions start packet transmission at 1,000 s. The hot spot is $6 \times 6$ nodes in the centre of the grid network, and $180 \times 180$ m in the centre of the random network. We evaluate the time and the control overhead for paths to recover after hot spot generation, with and without our predictive mechanism. When evaluating node failures (Subsection 2.4.4), failures occur after 1,000 s from the simulation start. In this experiment, 20 randomly chosen nodes fail in a $6 \times 6$ area in the centre of the grid network at 1,000 s.

Regarding performance metrics, we consider the recovery time to be the time from the occurrence of the environmental change until path recovery. However, it is difficult to rigorously define path recovery because our system is constantly changing. We therefore define path recovery in these experiments as 10 consecutive selections of the same path for more than 90% of all target sessions, with the average delay along the paths being less than 150% the average delay of last 10 paths selected by forward ants immediately before the environmental change. This path recovery check is carried out each time a backward ant reaches a source node. The control overhead is the total number of travelled hops of control messages (forward and backward ants in both mechanisms, and additionally predictive ants in the proposed method) of all target sessions from the occurrence of environmental change until path recovery. The recovery time and the control overhead shown in this study are averaged values over 300 simulation runs for each parameter setting, except for cases

Table 2.1: Parameter settings in simulation evaluation

| Parameter | Value |
|---|---|
| $\Delta t_f$ | 100ms $\sim$ 1s |
| $\Delta t_p$ | 100ms |
| $m$ | 0.5 |
| $p$ | 0.005 $\sim$ 0.1 |
| $\beta$ | 0.2 |
| $\theta_e$ | 0.5 |

Table 2.2: Parameter settings of AntNet in [1]

| Parameter | Value |
|---|---|
| $\eta$ | 0.005 |
| $c$ | 0.3 |
| $c_1$ | 0.7 |
| $c_2$ | 0.3 |
| $z$ | 1.7 |
| $w$ | $5\times(c/\eta)$ |
| $a$ | 5 |
| $\epsilon$ | 1.4 |

where convergence was not achieved by the end of the simulation run. We explicitly note such cases in the results.

In the experiments, the interval of predictive ant emissions $\Delta t_p$ is set to 100 ms, and we change the interval of forward ant emissions $\Delta t_f$ from 100 ms to 1 s. The parameter $\beta$, which determines the weight of individual pheromone increments in Eq. (2.9), is set to 0.2. The parameter $p$, which increases the amount of pheromones in Eq. (2.10), is changed from 0.005 to 0.1. Table 2.1 shows the experimental parameter settings. We set the coefficient $\alpha$ in Eq. (2.2) to a comparatively small value to obtain path stability. $\alpha$ is set to 0.004 in the grid network and to 0.0004 in the random network. $\alpha$ is an important coefficient that affects the convergence speed of path establishment, but discussion of the relation of $\alpha$ and convergence speed is beyond the scope of this study; refer to [69] for a detailed discussion of this relation. In [1], $\alpha$ is set to 0.2–0.5 to find shorter paths,

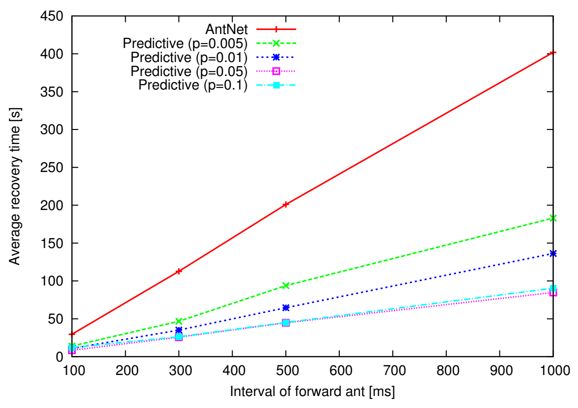


Figure 2.3: Path recovery time (grid network, 1 session)

which leads to many path fluctuations. However, we need to set $a$ to a small value (0.0004–0.004) for fair comparison, because we focus on path convergence in this study. Moreover, it is difficult to compare the original AntNet and the proposed method with the default $a$ setting [1], because the value of $a$ depends on the network size. Other parameters of AntNet are set according to their default settings [1], shown in Table 2.2.

### 2.4.2 Evaluation with Single Target Session

Here we present the simulation results with one target session to show that our predictive mechanism improves adaptability to environmental change. In each simulation, after traffic changes occur at 1,000 s, a short path that avoids the network centre is re-established by both AntNet and the proposed method.

Figures 2.3 and 2.4 show the simulation results with the grid network. We set the node at the top-left corner as the source node, and the one at the bottom-right corner as the destination node for the target session. These figures depict the recovery time and the control overhead for the interval of forward ant emissions. As shown in Figure 2.3, the recovery time of the proposed method is shorter than that of AntNet. Furthermore, the proposed method is superior to AntNet for any value of $p$.

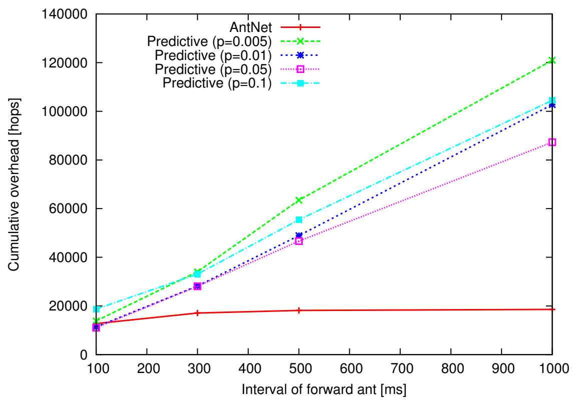


Figure 2.4: Cumulative overhead (grid network, 1 session)

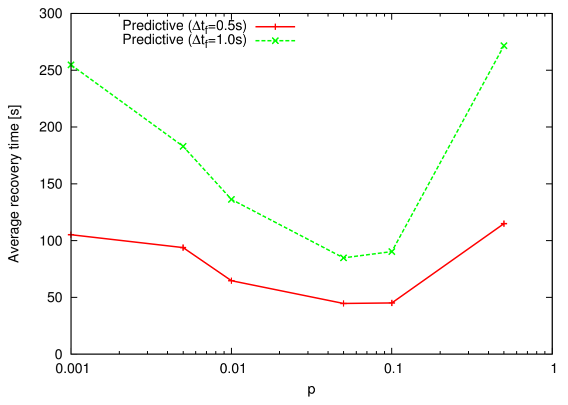


Figure 2.5: Path recovery time for parameter $p$ (grid network, 1 session)

In the original AntNet, most forward ants go through a path that has more pheromones than others, even if there is a better path, because pheromone updates depend on current pheromone values. It therefore takes a long time to re-establish a shorter path when the quality of an existing path falls due to traffic or other environmental changes. In the proposed method, however, predictive ants update pheromone values at each node while taking into account changes of pheromone values on its neighbour nodes. The proposed method boosts pheromone accumulation on a shorter path whose pheromone values are still low but increasing, so path re-establishment after environmental change is accelerated.

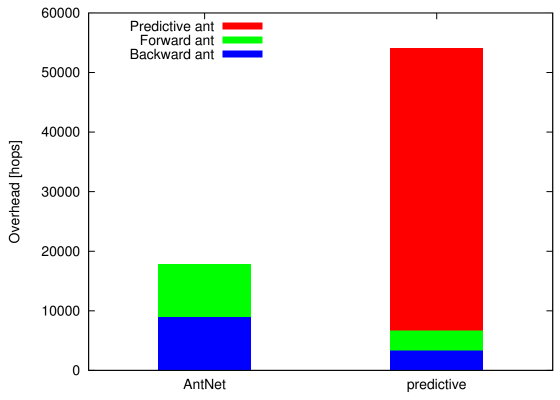


Figure 2.6: Cumulative overhead of ants (grid network, 1 session)

The recovery time is shorter In the proposed method, especially when the parameter $p$ is 0.05 (Figure 2.3). To reveal the relation between recovery speed and $p$, we show the recovery time for $p = (0.001, 0.005, 0.01, 0.05, 0.1, 0.5)$ when the interval of forward ant emissions ($= \Delta t_f$) is 0.5 s and 1.0 s in Figure 2.5. As shown in this figure, the recovery time is worse when $p$ is too high or too low. If $p$ is too low, the prediction effect is too small to accelerate path reestablishment, making the behaviour of the proposed method similar to the original AntNet with low $p$. A higher value of $p$ contrastingly leads to heavy pheromone concentration at one neighbour node. This decreases the stochastic nature, thus increasing the deterministic behaviour, of path exploration, despite stochastic features playing an important role for the discovery of shorter paths in an ant-based routing mechanism. In consequence, a loose control with appropriate $p$ leads to a better recovery time.

As shown in Figure 2.4, the proposed method has much higher control overhead than does AntNet, due to the predictive ants. Each node in the proposed method which receives a backward ant regularly sends predictive ants to each of its neighbour nodes every 0.1 s to obtain neighbour node information. Figure 2.6 shows a breakdown of the control overhead when the interval of forward ants is 0.3 s and the parameter $p$ is 0.005. Data packets of each background traffic session,

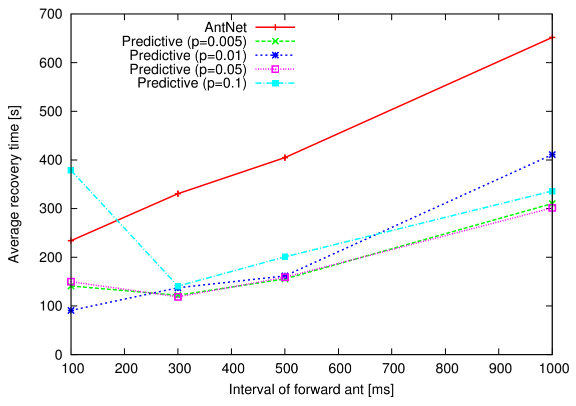


Figure 2.7: Path recovery time (random network, 1 session)

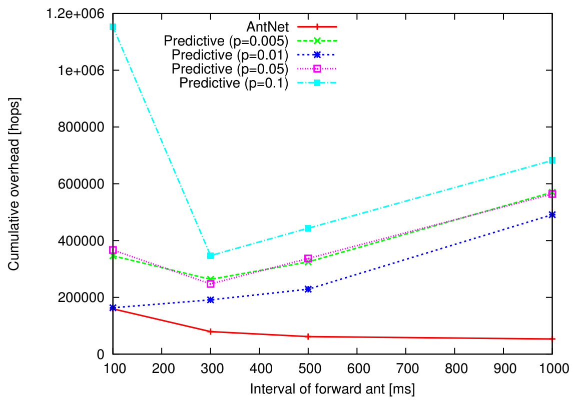


Figure 2.8: Cumulative overhead (random network, 1 session)

whose overhead is not included in this figure, follow a fixed path regardless of pheromone values and thus are not involved in pheromone updates. In other words, pheromone values are updated only by the forward, backward, and predictive ants of target sessions. As shown in Figure 2.6, the control overhead of forward and backward ants in the proposed method is smaller than that of AntNet, but the total overhead of the proposed method is high due to the predictive ants. This seems to be a serious drawback, but the overhead of predictive ants becomes negligible as the number of sessions increases. We describe the overhead for predictive ants in the multiple session scenario in the following subsection.

Figures 2.7 and 2.8 show the simulation results with the random network. A target session is generated between the two nodes furthest from each other over the diagonal line from the top-left corner to the bottom-right corner in the network. These figures depict the recovery time and the control overhead for the interval of forward ant emissions. As shown in Figure 2.7, the recovery time of the proposed method is shorter than that of AntNet, but the recovery time is worse when $\Delta t_p = 100$ ms. In the proposed method, increased rates of pheromones, which are indicators for predictive control, are updated simultaneously as backward ants update pheromone values. Therefore, if the interval of predictive ant emissions is too long against that of pheromone updates by backward ants, it is difficult to correctly predict a path that will collect more pheromones, and moreover the control with a predictive mechanism competes against the pheromone updates by backward ants, which results in path fluctuations. These are the reasons why the recovery time of the proposed method is longer than AntNet when $\Delta t_p$ is small. In consequence, it is important to properly set the interval of forward ant emissions and that of predictive ant emissions to correctly predict paths that will collect more pheromones.

In this experiment, there are cases where path recovery could not be achieved by the end of the simulation, due to path oscillations. Such oscillations are caused by the arrival of many forward ants before a backward ant returns. In AntNet, an intermediate node distant from a destination has to wait a long time for a backward ant. While it is waiting for a backward ant, new forward ants visit it and stochastically choose a next hop node. This results in establishment of multiple paths, and sometimes paths do not converge until the end of the simulation run. The ratio of path recovery within a given simulation time over 300 simulation runs is strongly correlated with the recovery time shown in Figure 2.7. In other words, the effect of path oscillations is lowered by enhancing the convergence speed.

As shown in Figure 2.8, the control overhead of the proposed method is much larger than that of the original AntNet. In particular, there are larger differences in control overhead between the original AntNet and the proposed method, as compared to the results in a grid network (Figure 2.4). These differences are caused by the average degree of nodes, about 9.6 in the random network and 3.6 in the grid network. Therefore, over 2.5 times more predictive ants are transmitted by each node in the random network than in the grid network.

Through simulation evaluation with one target session, we have shown the effect of prediction in AntNet. Our predictive mechanism improves the adaptability against traffic changes not only in the regular network but also the random network. However, the overhead of predictive ants in the proposed method is much larger than that of AntNet in the case of only one target session. This problem is further discussed in the next subsection by considering the case of multiple target sessions.

### 2.4.3 Evaluation with Multiple Target Sessions

In the previous subsection, we presented evaluations for fundamental performance of our predictive mechanism with one target session. In this subsection, we provide further simulation results to show that the overhead for predictive ants becomes negligible in a multiple session scenario. In experiments with multiple target sessions, path recovery is sometimes not achieved due to path oscillations, but the effect of path oscillations is lowered by enhancing the convergence speed (see Subsection 2.4.2).

Figures 2.9 and 2.10 show the simulation results for the grid network. In this evaluation, 100 target sessions start at the beginning of simulation and source nodes and destination nodes are selected at random except for $6 \times 6$ nodes in the network centre. These figures depict the recovery time and the control overhead for the interval of forward ant emissions. The recovery time of the proposed method is shorter than that of AntNet (Figure 2.9). This result is similar to the one session results shown in Figure 2.3. Figure 2.10 shows that the control overhead of the proposed method is

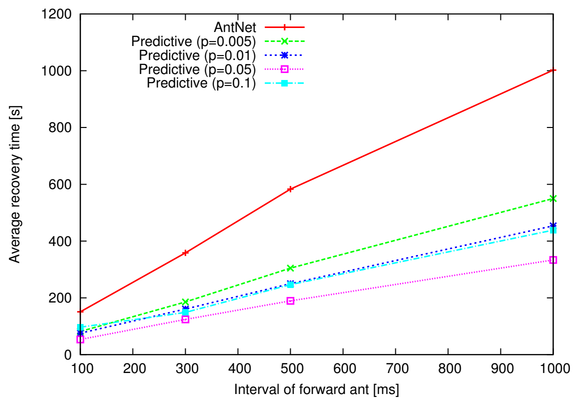


Figure 2.9: Path recovery time (grid network, 100 session)

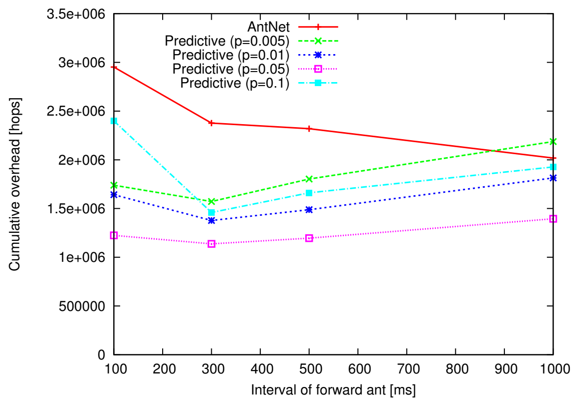


Figure 2.10: Cumulative overhead (grid network, 100 session)

smaller than or similar to that of AntNet. Figure 2.11 shows a breakdown of the control overhead when the interval of forward ants is 0.3 s and the parameter $p$ in the proposed method is 0.005. The overhead of predictive ants is large in the proposed method as compared to AntNet, since the latter has no predictive ants. The overhead caused by forward and backward ants in the proposed method is comparatively reduced, however, because the recovery time is shortened with prediction. Moreover, predictive ants can simultaneously collect increase rates for different destination nodes. In consequence, the overhead of predictive ants becomes trivial as the number of sessions increases.

Figures 2.12 and 2.13 show the simulation results with the random network. In this evaluation,

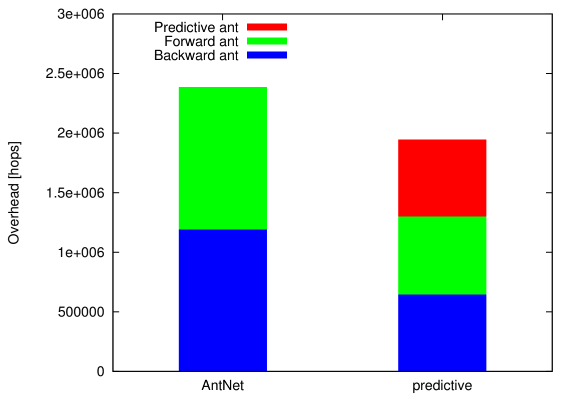


Figure 2.11: Cumulative overhead of ants (grid network, 100 session)

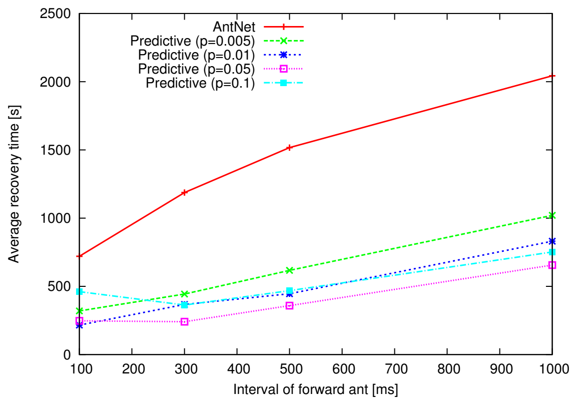


Figure 2.12: Path recovery time (random network, 100 session)

100 target sessions start at the beginning of simulation, and a source node and a destination node are selected at random (excluding a 180 $\times$ 180 m area in the network centre). These figures depict the recovery time and the control overhead for the interval of forward ant emissions. As shown in Figures 2.12 and 2.13, the recovery time of the proposed method is shorter and the control overhead of the proposed method is lower than that of AntNet. Similar to Figure 2.8, there are larger differences of the control overhead between the original AntNet and the proposed method, as compared to evaluation with the grid network (Figure 2.10). However, the control overhead will be further reduced by adding a mechanism where forward and backward ants or data packets collecting

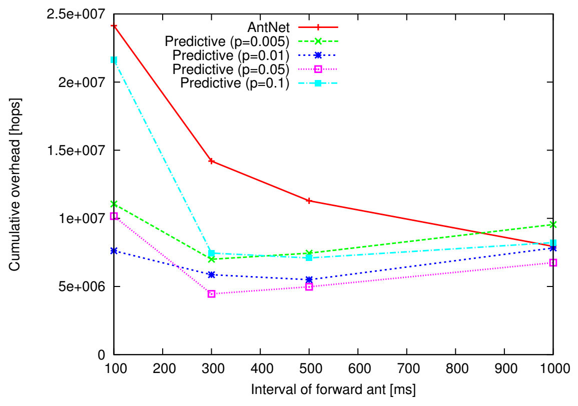


Figure 2.13: Cumulative overhead (random network, 100 session)

past pheromone information substitutes for predictive ants.

In conclusion, adaptability against traffic changes is improved by introducing a predictive mechanism, while control overhead is reduced in the multiple session scenario. We next show another scenario of environmental change, node failure.

### 2.4.4 Adaptability to Node Failures

Lastly, we provide simulation results in the case where 20 of 100 nodes in the grid network fail, to show that the proposed method has high adaptability to environmental change other than traffic changes. Figures 2.14 and 2.15 show simulation results with the grid network. In this evaluation, 100 target sessions start at the beginning of simulation and source nodes and destination nodes are selected at random (excluding $6 \times 6$ nodes in the network centre). These figures depict the recovery time and control overhead for the interval of forward ant emissions.

As shown in Figure 2.14, the proposed method is superior to AntNet in most cases over a broad range of $p$. Similar to cases where traffic changes occur (Subsections 2.4.2 and 2.4.3), pheromone updates in accordance not only with current pheromone values but with past pheromone changes play an important role in faster recovery from node failures. The recovery time is reduced at most

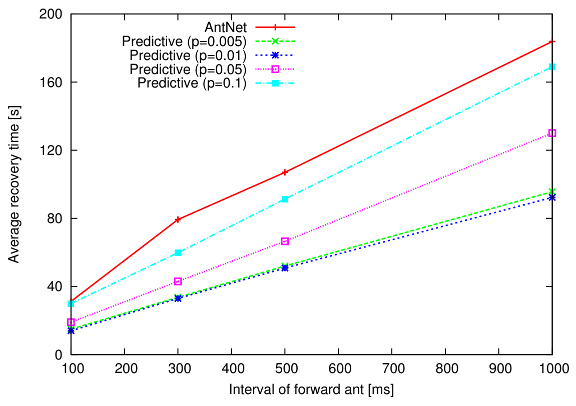


Figure 2.14: Path recovery time (node failure)

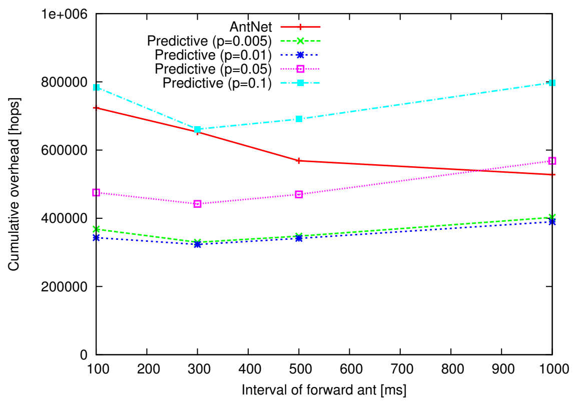


Figure 2.15: Cumulative overhead (node failure)

by about 40% in this evaluation, despite reductions of 60% in the cases where traffic changes occur (Figs 2.3, 2.7, 2.9 and 2.12). In this evaluation, 20% of all nodes fail at 1,000 s from the simulation start. In other words, the network size gets smaller, which reduces the amount of positive feedback needed for path establishment. Therefore, our predictive mechanism for accelerating positive feedbacks has a smaller effect in cases where node failures occur. The proposed method nonetheless accelerates path recovery, even under such extreme environmental change.

As shown in Figure 2.15, the control overhead is smaller than or similar to that of AntNet. Similar to the evaluation in Subsections 2.4.2 and 2.4.3, the control overhead of forward and backward ants is reduced by introducing our predictive mechanism, and predictive ants can simultaneously collect past pheromone information for several destination nodes. The control overhead of predictive ants is therefore also negligible in cases where node failures occur.

In conclusion, the proposed predictive mechanism enhances adaptability for both node failures and traffic changes, while retaining low control overhead.

## 2.5 Summary

In a self-organising system, each component behaves according to only current local information, which leads to slow adaptation to environmental change. To rapidly adapt to changing conditions, it is therefore necessary that systems be controlled considering the future state of systems as predicted by observing system behaviour. We propose and evaluate a predictive mechanism for AntNet, which is a simple and basic example of ACO-based routing. Simulation results show that the proposed method can facilitate path reestablishment when the network environment changes. Moreover, simulation evaluations showed that the control overhead needed for prediction becomes smaller in multiple session scenarios. Even in a more realistic environment where forward, backward, and predictive ants are lost in a network, ants can quickly re-establish other paths because they explore the network not deterministically but rather stochastically, and the positive feedback through pheromones leads to ants following shorter paths.

In future work, we will consider more general designs of a predictive mechanism for self-organising systems to realise future networks. There are various information network controls besides AntNet based on self-organisation, such as potential-based routing for wireless sensor networks [70] and clock synchronisation inspired by firefly behaviour [71]. Such self-organising systems inherently have the same problems as AntNet, so we will adopt predictive mechanisms to such self-organising systems to enhance their adaptability.

# Chapter 3

# Website Vulnerability Scanning Detection Inspired by Biological Adaptation Toward Diversifying Communication Services

## 3.1 Introduction

Web-based attacks on web servers that provide web services are increasing rapidly as the Internet becomes an increasingly important infrastructure and web services become more widespread. Moreover, cyber-terrorism targeting governments, corporations, and other large organizations is increasing, which is becoming a serious problem. However, it is difficult to detect all vulnerabilities in web servers, which can be targets of web-based attacks, due to the rapid growth in the diversity of web services. In other words, detecting attacks using known vulnerabilities is insufficient for preventing all web-based attacks. Therefore, we must collect and analyze information on web-based attacks in order to detect unknown attacks.

To collect web-based attack information, systems called *web honeypots*, which collect and monitor web attacks targeting web servers, are deployed [2, 72]. There are two types of web honeypots: *low interaction* and *high interaction* [73]. Low interaction honeypots emulate vulnerable OSs and applications, whereas high interaction honeypots accommodate actual OS applications. High interaction web honeypots can actually be under attacks, and are therefore used to collect and analyze a variety of web-based attacks [2, 72]. In this study, the honeypots referred to are high interaction web honeypots.

Honeypots monitor not only malicious accesses but also normal accesses such as crawler accesses by search engines. Therefore, we need to identify the malicious accesses from a large number of collected accesses. Researchers and engineers usually identify malicious accesses manually in most cases, but this is becoming difficult due to the rapid increase in traffic. Therefore, a method to identify malicious accesses automatically is necessary. A related study proposed a method of detecting web attacks that involves first identifying accesses by crawlers and then assuming the others to be malicious accesses [2]. In this scheme of crawler classification, the authors first identify accesses by well-known crawlers such as Google and then identify similar accesses as accesses by other crawlers. However, as web services become increasingly diverse, accesses by web crawlers that browse web pages for web service indexing are also becoming diverse. This leads to more difficulty in detecting crawlers using only known information. Consequently, adapting to such diverse accesses is a challenging task for crawler classification.

In this study, we adopted a bio-inspired clustering scheme for the crawler classification. Natural organisms behave individually and autonomously using only local information, and as a result, a global pattern or behavior emerges at a macroscopic level. Therefore, such mechanisms are advantageous for classifying a lot of data and for detecting unknown malicious threats. Because of these advantages, bio-inspired clustering schemes have been studied by many researchers [74]. In this study, we use AntTree [75, 76], an ant-based clustering scheme, for crawler classification because ants are typical social insects, and application of models based on their behavior is a major research theme. AntTree was inspired by the behavior exhibited by ants in which they form chains with each other to construct a tree structure. In our application, a tree structure enables us to easily interpret and analyze individual clusters of nodes (data). Moreover, AntTree, of course, retains high scalability and adaptability, which are inherent characteristics of bio-inspired mechanisms. We show that AntTree can identify crawlers more precisely than a conventional scheme [2] can through an evaluation done using communication logs collected in a real network.

The remainder of this chapter is as follows. We introduce related work in Section 3.2 and explain AntTree in Section 3.3. Then, we propose a crawler classification scheme using AntTree in Section 3.4, and in Section 3.5, we explain our evaluation of our proposed scheme using communication logs collected in a real network. Finally, we conclude the paper and briefly discuss future work in Section 3.6.

## 3.2 Related Work

In this section, we introduce previous studies about web-based attacks and bio-inspired clustering.

### 3.2.1 Web Attack Detection

There are two kinds of methods to detect web attacks: *signature detection*, which uses signatures from known malicious threats, and *anomaly detection*, which detects unknown malicious threats by collecting and analyzing web attacks. The rapid expansion in the number of web applications makes it difficult to detect all vulnerabilities in the applications. Therefore, it is necessary to collect and analyze accesses in order to detect unknown attacks. One method achieves that purpose by capturing traffic on the PCs and servers of users [77,78], and another method collects web attacks by utilizing honeypots [79]. The former method enables us to detect malicious accesses precisely, but there are some problems in implementing this kind of method. For example, traffic captures invade the user's privacy and reduce the service quality. Therefore, the latter one is more appropriate, and we focus on it in this study.

### 3.2.2 Bio-inspired Clustering

Swarms of certain animals such as ants, birds, and fish seem to behave intelligently at a macroscopic level, although each component behaves individually and autonomously based only on local information and simple rules. Because of their high scalability, adaptability, robustness, and flexibility, bio-inspired mechanisms are applied in various fields such as industry, chemistry, and economics. Such mechanisms are also applied to clustering schemes. Typical examples include clustering schemes based on particle swarm optimization (PSO) [80] and ant colony optimization (ACO) [81]. Such bio-inspired clustering schemes are suitable for web attack detection because of the following three reasons:

1. for classifying large volumes of data with a low computation cost,
2. for detecting unknown malicious threats,
3. for adapting to dynamically changing traffic (including normal and malicious traffic).

## 3.3 Self-organized Clustering Method: AntTree

In this section, we explain AntTree [75,76], which we adopt to detect accesses by crawlers. AntTree is an ant-based clustering scheme that was inspired by the coordinated behavior exhibited by ants in which they form chains with each other to construct a tree structure. This coordinated behavior is shown in swarms of *Linepithema humile* (Argentine ants) and *Oecophylla longinoda* (weaver ants). *Oecophylla longinoda* ants construct the tree structure in order to make a bridge across an open space, e.g., between leaves or branches, and to build their nest with leaves [82].

### 3.3.1 Overview

AntTree is a clustering scheme in which data that imitate ants chain with other data according to their similarity to construct a tree structure.

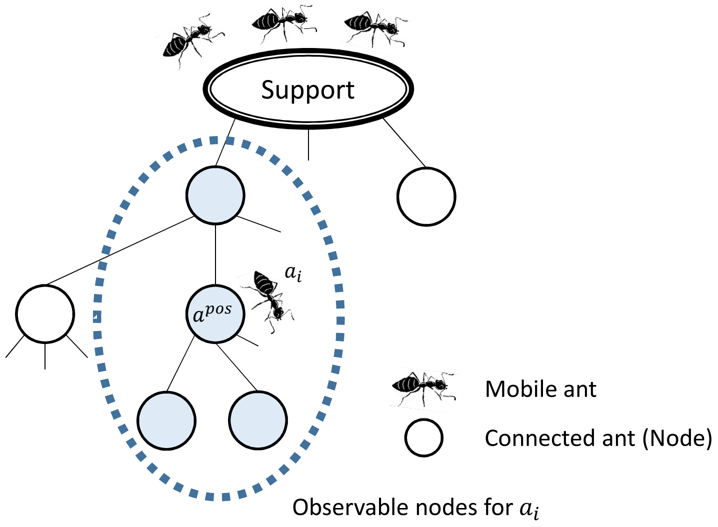


Figure 3.1: AntTree

In AntTree, one piece of data corresponds to a mobile agent called an *ant*. These ants chain together to construct a tree structure as shown in Fig. 3.1. At first, all ants are mobile and exist in the root of the tree, which is called the *support*. Then, the ants start to move away from the support one by one. When an ant starts to move away from the support, it moves among ants that are already connected to the tree (nodes), comparing itself with its neighbor nodes. If the current node is the most similar to the ant within its neighbor nodes, the ant stops moving and becomes a descendant node of the current node. When the ant is connected to the tree, the following ant starts to move away from the support. Note that in this study, *neighbors* refers to the nodes where the ant currently exists as well as its parent/descendant nodes.

In this scheme, the similarity between ants $a_i$ and $a_j$ $(i, j \in [1, \cdots, N]$, where $N$ is the number of ants) is shown by $Sim(a_i, a_j)$ $(\in [0, 1])$. Ant $a_i$ has similarity threshold $T_{Sim}(a_i)$ and dissimilarity threshold $T_{Dissim}(a_i)$. When comparing itself with ant $a_j$, ant $a_i$ assumes that it is similar to $a_j$ if $Sim(a_i, a_j) \geq T_{Sim}(a_i)$ is satisfied. On the other hand, ant $a_i$ assumes that it is not similar to $a_j$

if $Sim(a_i, a_j) < T_{Dissim}(a_i)$ is satisfied. Thresholds $T_{Sim}(a_i)$ and $T_{Dissim}(a_i)$ are updated while ant $a_i$ moves around the tree, and they finally converge to the values that are proper for exploring similar nodes.

### 3.3.2 Algorithm

In this subsection, we explain the behavior of ants in detail.

At first, the tree only consists of the support, and all ants exist on it. Then, the first ant starts to move and becomes a descendant node of the support. While the first ant is connected to the support, the next ant starts to move away from the support.

The following ant $a_i$ first compares itself with the descendant nodes of the support. If there are similar nodes to ant $a_i$, i.e., there are nodes $a_j$ that satisfy $Sim(a_i, a_j) \geq T_{Sim}(a_i)$, within the descendant nodes of the support, $a_i$ moves to the descendant node that is most similar to $a_i$. In contrast, if all descendant nodes are not similar to ant $a_i$, i.e., $Sim(a_i, a_j) < T_{Dissim}(a_i)$ is satisfied for all descendant nodes $a_j$, $a_i$ becomes a new descendant node of the support and stops moving. When ant $a_i$ is connected to the support, the next ant starts to move. Note that the maximum number of descendant nodes of each node (including the support) is limited to $l$. Therefore, if the support already has $l$ descendant nodes, ant $a_i$ decreases its similarity threshold $T_{Sim}(a_i)$ by using (3.1), which is explained later, and moves to the descendant node whose similarity to $a_i$ is the highest. Otherwise, ant $a_i$ updates its similarity threshold $T_{Sim}(a_i)$ and dissimilarity threshold $T_{Dissim}(a_i)$ by using (3.1) and (3.2) and then moves to the descendant node whose similarity to $a_i$ is highest.

$$T_{Sim}(a_i) \leftarrow T_{Sim}(a_i) \times \alpha_1, \tag{3.1}$$

$$T_{Dissim}(a_i) \leftarrow T_{Dissim}(a_i) + \alpha_2. \tag{3.2}$$

In the case where ant $a_i$ compares itself with ant $a_j$, $a_i$ is likely to be classified in other clusters if similarity threshold $T_{Sim}(a_i)$ is high and dissimilarity threshold $T_{Dissim}(a_i)$ is high. On the other hand, ants are likely to be classified in the same cluster if both $T_{Sim}(a_i)$ and $T_{Dissim}(a_i)$ are low. At first, $T_{Sim}(a_i)$ is initialized to $1$ and $T_{Dissim}(a_i)$ to $0$. Ants update these thresholds while moving around the tree, and the ants finally obtain the proper values. $\alpha_1$ and $\alpha_2$ are parameters that determine the decrease amount of $T_{Sim}(a_i)$ and the increase amount of $T_{Dissim}(a_i)$, respectively, on their updates. With higher $\alpha_1$ and $\alpha_2$, ants can find similar or dissimilar nodes faster, which increases the clustering speed but reduces the accuracy of detection.

When ant $a_i$ arrives at nodes $a^{pos}$ except for the support, $a_i$ first compares itself with $a^{pos}$. If ant $a_i$ is similar to node $a^{pos}$, i.e., $Sim(a_i, a^{pos}) \geq T_{Sim}(a_i)$ is satisfied, $a_i$ then compares itself with neighbor nodes of node $a^{pos}$. If none of the neighbor nodes are similar to ant $a_i$, i.e., $Sim(a_i, a_j) < T_{Dissim}(a_i)$ is satisfied for all neighbor nodes $a_j$, $a_i$ becomes a new descendant node of node $a^{pos}$ and stops moving. If node $a^{pos}$ already has $l$ descendant nodes, ant $a_i$ randomly moves to a neighbor node. In contrast, if there are nodes $a_j$ that do not satisfy $Sim(a_i, a_j) < T_{Dissim}(a_i)$, ant $a_i$ updates $T_{Sim}(a_i)$ and $T_{Dissim}(a_i)$ by (3.1), (3.2) and then randomly moves to a neighbor node.

If ant $a_i$ arrives at nodes $a^{pos}$ and $Sim(a_i, a^{pos}) \geq T_{Sim}(a_i)$ is not satisfied, $a_i$ randomly moves to a neighbor node.

## 3.4 Crawler Classification using AntTree

In this section, we explain how to classify crawlers using AntTree.

### 3.4.1 Similarity

In AntTree, the similarity $Sim(a_i, a_j)$ between ants $a_i$ and $a_j$ is a metric used by moving ants for exploring and constructing a tree structure as explained in Section 3.3. As in [75], we define $Sim(a_i, a_j)$ by (3.3) using the Euclidean distance $d(a_i, a_j)$ between ants $a_i$ and $a_j$.

$$Sim(a_i, a_j) = 1 - d(a_i, a_j). \tag{3.3}$$

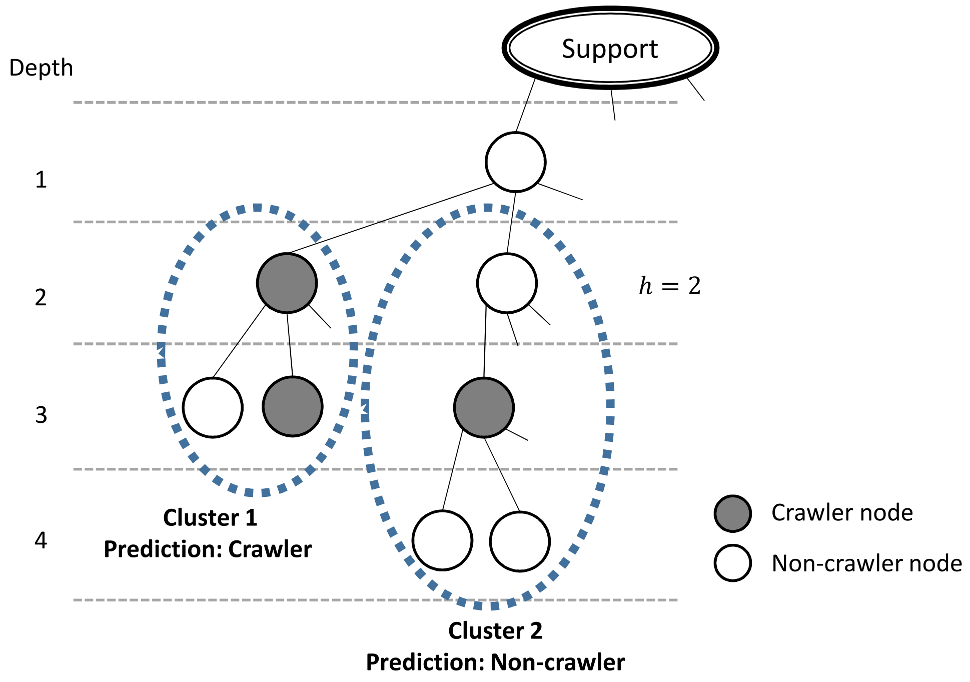


Figure 3.2: Cluster interpretation for AntTree ($h = 2$)

The higher similarity $Sim(a_i, a_j)$ is, the more similar ants $a_i$ and $a_j$ are. When an ant (a datum) has $M$ features $\{v_{i_1}, \cdots, v_{i_M}\}$, $d(a_i, a_j)$ is given by

$$d(a_i, a_j) = \sqrt{\frac{1}{M}\sum_{k=1}^{M}(v_{i_k} - v_{j_k})^2}. \tag{3.4}$$

The feature vector we designed for classifying crawlers is explained in Subsection 3.5.3.

### 3.4.2 Cluster interpretation

We assume that a cluster corresponds to a subtree whose root is an $h$ depth node of the tree constructed by AntTree. Figure 3.2 shows an example of cluster interpretation of AntTree with $h = 2$.

Each cluster is classified according to which type of data is a majority in the cluster. For example, if the number of crawler nodes is larger than other nodes within a cluster, the cluster is classified to the Crawler cluster, as shown in Fig. 3.2. If the number of crawler nodes are equal to that of non-crawler nodes in a cluster, the cluster is classified to the type of the root node of the cluster.

## 3.5 Evaluation and Discussion

### 3.5.1 Overview

Here, we discuss the evaluation of crawler classification using AntTree using communication logs collected in a real network. Communication logs of accesses by Google are easy to identify because Google opens the information of its crawlers to the public. Therefore, we first identify the Google crawlers and then classify communication logs of accesses by other crawlers. We explain the data set and the feature vector in Subsections 3.5.2 and 3.5.3, respectively.

In this evaluation, we compared the crawler classification using AntTree with the conventional scheme proposed in [2]. In the conventional scheme, crawlers are classified in accordance with the features of well-known crawlers. In this evaluation, we used Google as well-known crawlers because Google crawlers are easy to identify using public information. Then, we classified communication logs with other crawlers using the features of Google crawlers. We used RandomForest [83] as the classification algorithm to evaluate the conventional scheme.

Note that we used a program written in C++ for the evaluation of AntTree, and RapidMiner 5 for the evaluation of the conventional scheme.

### 3.5.2 Data Set

We used HTTP communication logs collected in a real network for our evaluation. These logs were collected by 37 honeypots [84] from August 29, 2013 to January 14, 2014. Each log included information of request packets that honeypots receive and the responses of honeypots to these request packets. We attached the following labels to about 220.3 million logs collected by honeypots.

- *Google* (about 8.2 million): Communication logs of accesses by Google crawlers are labeled

Table 3.1: Connection logs collected by honeypots

| Label | Number |
|---|---|
| Google | 8,276,246 |
| Crawler | 1,502,254 |
| Non-crawler | 11,547,739 |
| Other | 710,708 |
| Total | 22,036,947 |

Google. Google logs are classified in accordance with source IP addresses and UserAgents, which Google opens to the public.

- *Crawler* (about 1.5 million): Communication logs with crawlers other than Google are labeled Crawler. Crawler logs are classified in accordance with source IP addresses and UserAgents that researchers and engineers detect by manually analyzing communication logs. Some examples of the detected crawlers include Baidu, Bing, and Microsoft.

- *Non-crawler* (about 11 million): Communication logs with others are labeled Non-crawler. Non-crawler logs include malicious logs.

- *Other* (about 0.71 millions): Other logs correspond to communication logs that are not categorized to either of the aforementioned labels. Most Other logs lack information for their analysis. Therefore, we did not use Other logs in the evaluation.

We used 3,004,508 communication logs including 1,502,254 Crawler logs and 1,502,254 Non-crawler logs as the test data set for this evaluation. Note that we sampled and used only 1,502,254 Non-crawler logs out of all of the Non-crawler logs. In order to evaluate the performance regardless of the access distribution over time, we sampled Non-crawler logs randomly.

In the crawler detection scheme [2], the authors firstly identify well-known crawlers in accordance with source IP addresses and UserAgent, and then identify other accesses having similar

Table 3.2: Number of features

| Feature | Number |
|---|---|
| Request | 89 |
| Response | 37 |
| Total | 126 |

features to well-known crawlers as other crawlers. To quantitatively compare the crawler classification using AntTree with that of the conventional scheme [2], we classified crawlers as follows in the evaluation of the conventional scheme.

1. We used Google as the well-known crawlers and randomly sampled 1,502,254 Google logs. Then, we used 3,004,508 logs including 1,502,254 Google logs and 1,502,254 Non-crawler logs as the learning data set.
2. We produced the classification model using the learning data set with RandomForest. In the evaluation of the conventional scheme, we used the same feature vector as that of AntTree explained in Subsection 3.5.3.
3. We classified the test data set in accordance with the classification model. That is, we classified Crawler logs in accordance with the features of Google logs.

### 3.5.3 Feature Vector

We designed a feature vector and used it for crawler classification. There are two types of features: information on request packets and responses to those request packets.

- Request packets:

  Information on HTTP request packets that the honeypots received. Specifically, we use the following information for the crawler classification.

  - Request information: the request URL, the communication method (GET, POST, etc.)

  - Packet header: the UserAgent, the referer, the source/destination port number, the communication protocol (HTTP or HTTPS)
  - Packet body: the body length

- Responses to request packets:

  Information on responses of honeypots to request packets. In detail, we use the following information for the classification.

  - Response type: StatusCodes (200, 404, etc.) defined in RFC 2616
  - Response information: text types (HTML, CSS, etc.) and character encodings (UTF-8, ISO-8859-1, etc.) of content included in response packets if the honeypots send them

All features are normalized and described by real values in $[0, 1]$ for equalizing the weights of features. Feature $v_{i_j}$ of data $i$ ($i \in [1, \cdots, N]$, $j \in [1, \cdots, M]$) is normalized by

$$\frac{v_{i_j} - \min_{n\in[1,\cdots,N]} v_{n_j}}{\max_{n\in[1,\cdots,N]} v_{n_j} - \min_{n\in[1,\cdots,N]} v_{n_j}}.$$

As an example of features, we explain how to use request URLs included in request packets as features.

1. We use the number of characters of the request URL as a feature.
2. We separate the request URL into path and query parts. The path part gives the location (path) of the requested resource, and the query part specifies parameters. Then, we use the following values as features.

   - The number of levels of the path requested in the path part and the average number of characters of each level of the path.

Table 3.3: Parameter settings of AntTree

| Parameter | Value |
|---|---|
| $l$ | 5 |
| $h$ | 3 |
| $\alpha_1$ | 0.95 |
| $\alpha_2$ | 0.20 |

Table 3.4: AntTree

| | | Prediction | | Recall |
|---|---|---|---|---|
| | | Crawler | Non-crawler | |
| Label | Crawler | 1,259,976 | 242,278 | 83.87% |
| | Non-crawler | 76,417 | 1,425,837 | 94.91% |
| Precision | | 94.28% | 85.48% | |

Table 3.5: Conventional scheme [2]

| | | Prediction | | Recall |
|---|---|---|---|---|
| | | Crawler | Non-crawler | |
| Label | Crawler | 1,241,437 | 260,817 | 82.64% |
| | Non-crawler | 105,952 | 1,396,302 | 92.95% |
| Precision | | 92.14% | 84.26% | |

- The number of parameters specified in the query part, and the average number of characters of each parameter.

3. We describe the request URL in the form of the regular expression [85] for converting text information to numerical information. We use the types of character strings (string, integer, hex, etc.) and the ratios of strings of each type in the regular expression as features.

The number of features used in this evaluation is indicated in Table 3.2.

### 3.5.4 Results and Discussion

Here, we discuss our evaluation in which we compared the crawler classification performance of AntTree and the conventional scheme. In this evaluation, we used recall and precision as evaluation metrics. The definitions of these metrics are as follows.

- Recall: the fraction of data that are correctly classified within data to which the same label is attached. Given set $\mathcal{L}_A$ of data with label $A$ and set $\mathcal{C}_A$ of data classified to $A$, recall of $A$ is calculated by

$$\text{Recall} = \frac{|\mathcal{L}_A \bigcap \mathcal{C}_A|}{|\mathcal{L}_A|}.$$

- Precision: the fraction of data that are correctly classified within data classified to the same category. Given $\mathcal{L}_A$ and $\mathcal{C}_A$, the precision of $A$ is calculated by

$$\text{Precision} = \frac{|\mathcal{L}_A \bigcap \mathcal{C}_A|}{|\mathcal{C}_A|}.$$

The results of the crawler classification using AntTree and the conventional scheme are given in Tables 3.4 and 3.5, respectively. In these tables, *label* corresponds to the label attached to each log, as explained in Subsection 3.5.2, and *prediction* corresponds to the result of classification by AntTree or the conventional scheme.

As shown in the Tables, the precision and recall of Crawler and Non-crawler are higher with AntTree than with the conventional scheme. This indicates that Crawler logs can be classified more precisely by using AntTree than the conventional scheme. This is because it is difficult to identify all crawlers according to the features of well-known crawlers (in this evaluation, Google crawlers) due to the diversity of crawler accesses. On the contrary, AntTree is effective for classifying various accesses because similar logs are classified to the same cluster in AntTree whether their features are known or unknown.

Moreover, we found through this evaluation that AntTree can classify data whose features are minor in the entire data set although minorities of features often make us to overlook them. Most of existing clustering schemes including hierarchical and partitioning-optimization schemes are sensitive to outliers and noises and are likely to merge small clusters with large clusters [86], which complicates the classification of small clusters. Therefore, these existing schemes have difficulty in adapting to diverse accesses. On the contrary, AntNet can classify clusters accurately regardless of these cluster size because each datum explores similar kinds of data using only local information while moving over the tree. Therefore, AntTree can detect new types of accesses from just a few communication logs, which makes AntTree able to adapt to dynamical changes of features of accesses. We will investigate this point thoroughly in the future.

In conclusion, AntTree, a bio-inspired clustering scheme, can classify crawlers more accurately than the conventional scheme can and is suitable for classifying various accesses.

## 3.6 Summary

Due to the rapid growth in the diversity of web services, it is becoming increasingly difficult to classify a large number of communication logs using only known features that are detected manually. In this study, we adopted AntTree, an ant-based clustering scheme, to achieve crawler classification that can adapt to a diverse range of web services. Our evaluation results indicated that AntTree can classify crawlers more accurately than the conventional scheme.

As a future task, we will evaluate AntTree by considering the changes in communication features. Meanwhile, statistical information of communication logs, e.g., the intervals and the distribution of packet arrivals, would be an important cue for detecting crawlers and web-based attacks. Therefore, we will use such statistical features for the classification of communication logs.

# Chapter 4

# Controlling Large-scale Self-organized Networks with Lightweight Cost for Fast Adaptation to Changing Environment

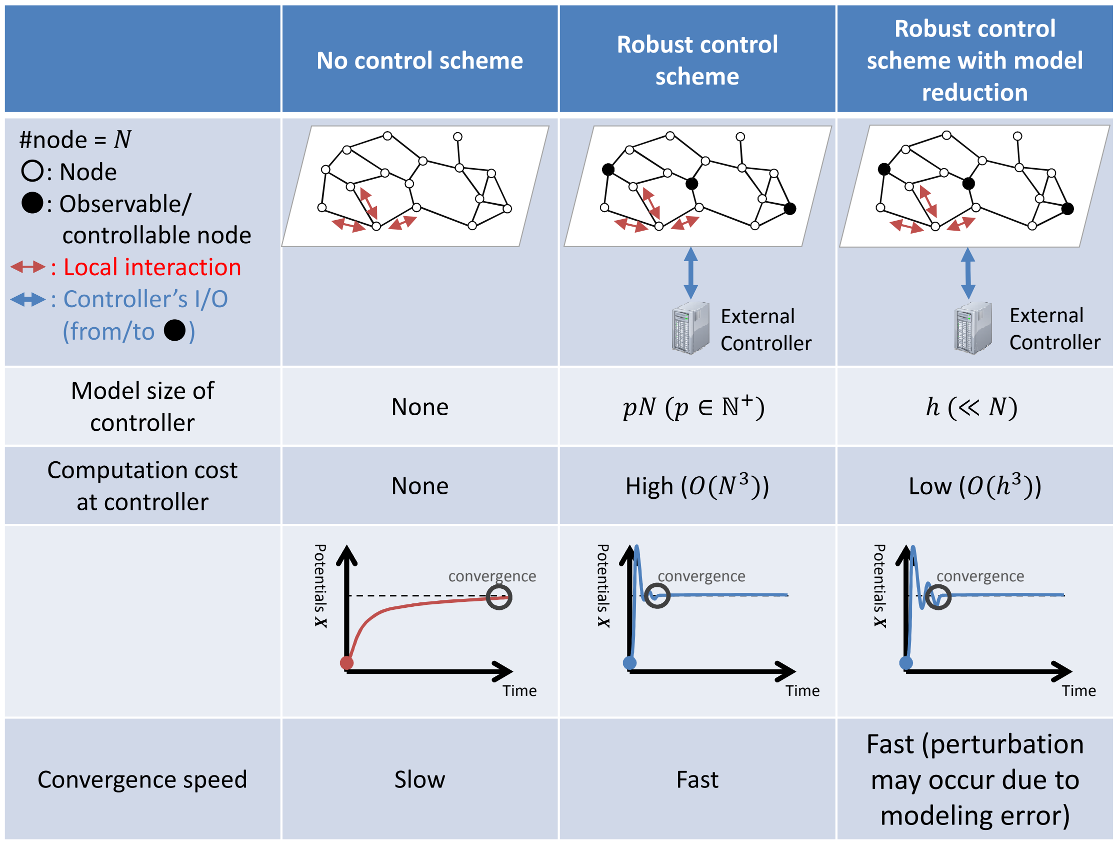
## 4.1 Introduction

Self-organization, where components behave individually and autonomously, is a natural phenomenon in natural distributed systems [17, 18]. In a self-organizing system, each component follows simple rules using locally available information. Through direct or indirect interactions among components, a global behavior or pattern emerges on a macroscopic level without a central control entity. In a self-organizing system, up-to-date information regarding the entire system or many other components is unnecessary, which considerably reduces computational cost and communication overhead for collecting global information. This localized control leads to a capability of handling local failures and small environmental changes by interaction of local components. Thus, self-organizing systems are expected to automatically recover from failures and adapt to environmental changes, without involving centralized control. These are reasons why a variety of self-organization-based models have been applied to information networking such as routing, synchronization, and task assignment [11, 12]. In future large-scale, complex networks we can expect features such as scalability, adaptability, and robustness to be improved to an extent not possible by conventional network control methods [8].

Although self-organization control without global knowledge of the current network state has various benefits, such control has critical limitations that complicate practical use in industrial and business systems [13]. It may take a long time for global patterns to emerge in large-scale systems, because they appear as a consequence of interactions between autonomous components. This property also leads to slow adaptation to large environmental changes, which is difficult to solve solely by local interaction in self-organizing systems. Also, self-organizing systems that use only local information sometimes fall into local optima. On the other hand, in conventional centralized systems global information can reach an optimal solution, though the required computational cost to do so often leads to unrealistic solutions.

Such limitations raised from real applications brought about the idea of guided self-organization, where the self-organizing system is controlled through some constraints [14–16]. For example, the authors of [31,70] use the concept of guided self-organization, where an external observer/controller guides self-organizing optical network [31] and sensor network [70] systems through a feedback mechanism that leads them to a desired state. The self-organizing system can then be controlled through fully observed information. However, enhancing the convergence speed to reach an optimal or semi-optimal solution remains as an outstanding task.

We previously introduced an external controller having an optimal feedback mechanism to self-organizing systems in [45]. The external controller collects information regarding the network such as node states and network topology via a partial set of nodes directly monitored by the controller, and estimates system dynamics using a mathematical model that describes the network dynamics, then determines optimal control inputs based on robust control theory [47] for facilitating the convergence. Optimal feedback mechanisms for controlling dynamical systems have been researched



| | No control scheme | Robust control scheme | Robust control scheme with model reduction |
|---|---|---|---|
| #node = $N$<br>○: Node<br>●: Observable/ controllable node<br>↔ : Local interaction<br>↔ : Controller's I/O (from/to ●) | | External Controller | External Controller |
| Model size of controller | None | $pN$ ($p \in \mathbb{N}^+$) | $h$ ($\ll N$) |
| Computation cost at controller | None | High ($O(N^3)$) | Low ($O(h^3)$) |
| | Potentials $X$<br>convergence<br>Time | Potentials $X$<br>convergence<br>Time | Potentials $X$<br>convergence<br>Time |
| Convergence speed | Slow | Fast | Fast (perturbation may occur due to modeling error) |

Figure 4.1: Advantages of optimal feedback with model reduction ($\mathbb{N}^+$ is the set of all non-zero natural numbers)

for long years in the field of control theory [48]. A controller monitors a system and provides an optimal control feedback that minimizes the cost function [48] based on a mathematical model of the system. Note that there are various errors such as modeling errors and unexpected disturbances so that the optimal control feedback is not necessarily optimal at any time in real systems. Simulation results in [45] showed that the mechanism improves the convergence speed of self-organization, however, in large-scale networks especially, it is generally difficult for the controller to collect detailed network information and to estimate the network dynamics [87], because doing so requires considerable costs and, even worse, loses the advantage of scalability of self-organizing systems.

The contribution of this study is to converge self-organizing network systems with lightweight cost while retaining a high performance (such as high convergence and adaptability in our proposal) of robust control. To that end, we first regulate the area in which the external controller collects node states to reduce the cost for collecting information (Figure 4.1). If the controller collects all node states, the communication overhead is extremely large and traffic congestion would occur. Therefore, we limit the area from which the controller can collect information to nodes that can be reached within several hops from the monitored nodes. The controller can estimate the states of all nodes from only a part of them given the network topology, because the state of each node is determined in accordance with local interactions based on prescribed rules [47]. That is, the controller acquires network information with lower communication overhead, then calculates optimal feedback inputs. In our scheme, a controller has an internal model estimating the actual network system. The order $(N_{dim})$ of that model is proportional to that of the actual network model, namely the number of nodes. Note that we need $O({N_{dim}}^2)$ for controller state updates at each time instance, and also $O({N_{dim}}^3)$ computation for controller re-design when the network topology changes. The computational cost is thus extremely large in large-scale networks when we use the original model, whose order is proportional to the number of nodes. We therefore reduce the order of the network model (consequently that of $N_{dim}$) to decrease the computational cost via a model reduction technique [49]. As shown in Figure 4.1, model reduction refers to reducing the number of state variables in dynamical systems, while retaining suitable input/output characteristics. Note that there is a trade-off between the optimization of control input and the number of state variables, i.e., between the convergence speed and the computational cost. Intuitively, a reduced-order model leads to the loss of the original model while decreasing the computational cost. This trade-off relation requires us to further examine the number $h$ of state variables of the reduced-order model.

The effectiveness of our proposal is evaluated through computer simulation studies where we consider potential-based routing—a self-organizing routing mechanism for wireless sensor networks—with optimal feedback, and we evaluate the convergence speed after environmental changes. Wireless sensor networks represent one suitable application of our proposal. This is because our proposal requires the expensive computation of control input for an external powerful controller and not for each component of a self-organizing system. Therefore, the performance of the whole system can be improved even though each component in the systems does not need much computational power and energy consumption. Optimality of our feedback mechanism is analytically guaranteed in synchronous systems, but not in asynchronous systems. Then, we first assume a wireless sensor network where nodes behave asynchronously and the controller can observe the network state via the monitored nodes experiencing communication delay. To evaluate the robustness of our proposal against information loss in the controller, we next simulate an environment where several nodes fail and the controller cannot immediately detect node failures. Through these evaluations, we will show that optimal feedback using a reduced-order model can enhance the convergence speed of self-organizing systems at fairly low cost, even if the network is large. Moreover, optimal feedback does not have a negative impact on the robustness that is a notable property of self-organizing systems. Note that potential-based routing for wireless sensor networks is just an example of application for our proposal.

The remainder of this chapter is organized as follows. Firstly, we briefly explain potential-based routing in Section 4.2. We propose and explain potential-based routing with optimal feedback in Section 4.3, and explain a reduced-order model with which the controller estimates the network dynamics in Section 4.4. We then show fast adaptation to an environmental change of the proposed method through simulation and give a discussion of our proposal in Section 4.5. Finally, in Section 4.6 we present our conclusions and suggest areas for future work.

## 4.2 Potential-based Routing

Potential-based routing is a self-organizing routing mechanism being active in the fields of wireless sensor networks, mobile ad-hoc networks, and information centric networks [70, 88–93]. Here we assume that potential-based routing is used in wireless sensor networks. In potential-based routing, each node has a scalar value called its *potential*, and data packets are forwarded to a neighbor whose potential is smaller than the forwarder's. In wireless sensor networks, data packets are generally sent to a sink node, and the fewer hops from the sink node a node is, the lower the potential value assigned to the node. The simple forwarding rule "forward data to a neighboring node with lower potential" can therefore result in data packet collection toward sink nodes, as illustrated in Figure 4.2. Potential-based routing has high scalability because each node uses only local information for calculating potentials and a local rule for forwarding data. Furthermore, it can achieve load balancing and consequently network lifetime improvement by calculating potentials using information such as flow rates, queue lengths, or remaining energy [90]. In Subsections 4.2.1 and 4.2.2, we describe a potential field construction method and show how to select a next hop node using the gradient of the field.

### 4.2.1 Potential Field Construction

Sheikhattar and Kalantari [91] focused on the convergence of potential-based routing and achieved enhancement of the potential convergence speed. They proposed a potential calculation method based not only on current potentials but also on last potentials to accelerate potential convergence. Node $n$'s potential at time $t$, $\theta_n(t)$, is calculated by Equation (4.1).

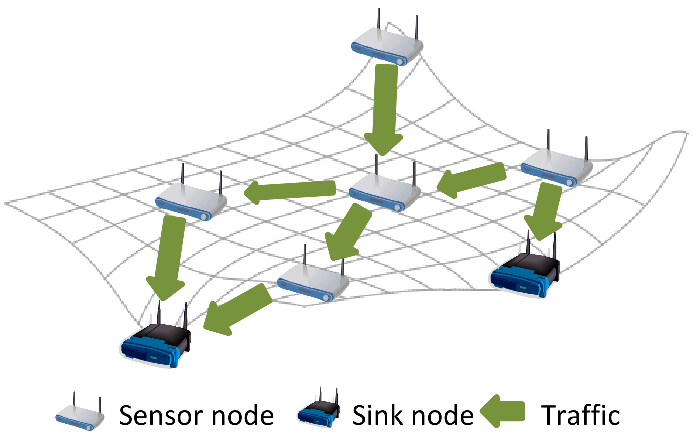


Figure 4.2: Potential-based routing

$$\begin{aligned}\theta_n(t+1) =& (\alpha+1)\theta_n(t) - \alpha\theta_n(t-1) \\ & + \beta\sigma_n \left( \sum_{k\in\mathcal{N}_b(n)} \{\theta_k(t) - \theta_n(t)\} + f_n(t) \right)\end{aligned} \tag{4.1}$$

where $\mathcal{N}_b(n)$ is the set of neighbors of node $n$, and $\alpha$ is a parameter that determines the weights of the current and the last potential values when calculating the next potential. Larger $\alpha$ means that the weight of the last potential value is larger and therefore the system becomes less subject to current noise, though the convergence speed is slower. Parameter $\beta$ determines the influence amount of neighbor node potentials. $\sigma_n$ is defined as $\sigma_0/|\mathcal{N}_b(n)|$ ($\sigma_0$ is a parameter), and $f_n(t)$ corresponds to the flow rate of node $n$ at time $t$. If $f_n(t)$ is a positive value it means the data generation rate of node $n$, whereas if $f_n(t)$ is negative it means the rate of data packets delivered to node $n$. For sink node $n$, $f_n(t)$ corresponds to targeted flow rates which are given by the network manager. If the flow conservation constraint is upheld, that is, $\sum_{n\in\{1,\cdots,N\}} f_n(t) = 0$, a potential field is constructed so that actual rates of data packets delivered to nodes satisfy given flow rates, i.e., all gradients, which are potential differences between next hop nodes, correspond to the appropriate flow rates between next hop nodes.

The convergence speed based on Equation (4.1) is faster than simple Jacobi iterations (such as our previous work [70]), but still takes a long time to converge due to its calculation being based only on local information (see Section 4.5 for an example convergence). In both cases, potentials of nodes converge as a result of the iterative behavior (i.e., potential updates of nodes), and therefore, the convergence speed is faster with a shorter interval of potential updates. Instead of relying on only local interactions among nodes, we introduce into potential-based routing an external controller that observes network states (potential values), estimates its future state, and regulates potentials of a partial set of nodes for faster convergence.

### 4.2.2 Routing

If a node has a data packet, it forwards it according to the potential values of itself and its neighbors. In our potential-based routing, when a sensor node generates or receives a data packet, it probabilistically selects a next node that is assigned a lower potential value than itself, and the packet eventually arrives at a sink node. Specifically, a next-hop node is selected proportionally with potential values, that is, the probability $p_{i\to j}(t)$ that sensor node $i$ selects a neighbor node $j$ as the next-hop node for a data packet at time $t$ is given by

$$p_{i\to j}(t) = \begin{cases} \frac{\theta_i(t)-\theta_j(t)}{\sum_{k\in\mathcal{N}_l(i)}\{\theta_i(t)-\theta_k(t)\}}, & \text{if } j \in \mathcal{N}_l(i) \\ 0, & \text{otherwise} \end{cases},$$

where $\mathcal{N}_l(i)$ is the set of node $i$'s neighbor nodes that are assigned lower potential values than node $i$. If node $i$ has no neighbor node with lower potential, that is, $|\mathcal{N}_l(i)| = 0$, the data packet is not sent to any node and is dropped, but such cases generally happen only in transient cases, such as node failures or changes of potential values at the sink node.

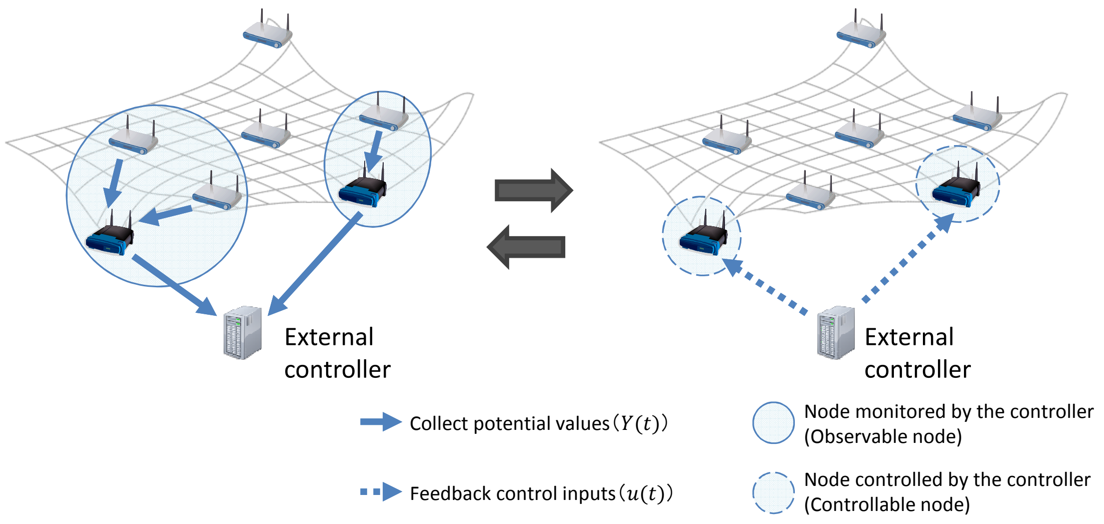

External controller
External controller
Collect potential values $(Y(t))$
Feedback control inputs $(u(t))$
Node monitored by the controller (Observable node)
Node controlled by the controller (Controllable node)

Figure 4.3: Potential-based routing with a contoller's feedback, where an external controller collects potential values from observable nodes and introduces control inputs to controllable nodes periodically.

## 4.3 Potential-based Routing with Optimal Feedback

### 4.3.1 Overview

In this section, we describe a model of network dynamics and explain our optimal control scheme. A *controller* monitors network information, in particular the potential values of a partial set of nodes, which we call *observable nodes*. The controller then returns suitable control inputs to a partial set of nodes, which we call *controllable nodes*), for accelerating convergence of the potential distribution toward the target potential distribution. In this study, target potentials are estimates of converged potential values derived from current information, specified in Subsection 4.3.2. We assume that the controller and sink nodes are power-supplied so that these sink nodes can have direct reliable connections to sensor nodes and the controller at all times. Therefore, in our proposal, the controller monitors network information and provides control inputs via them, as illustrated in Figure 4.3.

In our proposal, the area over which the controller monitors potential values is limited to several hops from sink nodes for reducing communication overhead. Of course, the controller cannot directly get node potentials outside the area, but it can estimate them by utilizing the potential dynamics model, which describes potential changes based on local node interactions. When receiving control inputs from the controller, sink nodes diffuse the information, changing potential amounts over the entire network through local interactions of sensor nodes (Equation (4.2) in Subsection 4.3.2). This estimation of the controller requires high computational cost and the reduced-order model can reduce this computational cost for estimating potential values of non-observable nodes, as will be explained in Section 4.4. Note that information of the network topology is needed for designing a controller, as is flow rates of nodes for calculating target potential values. Such information is difficult to estimate, but is reported to the controller only when it changes, because we assume that changing intervals of the network topology and flow rates are lower than the convergence time of potentials. This assumption is feasible because the potential convergence is generally achieved as a result of the iterative behavior (nodes' potential updates and the controller's feedback) in potential-based routing with an optimal feedback so that frequencies of potential updates and controls need to be much higher than those of changes in the network topology and flow rates.

### 4.3.2 Network Dynamics

Let the dynamics of potentials be given by a deterministic discrete-time model. In our proposal we represent $f_n(t) = \bar{f}_n + d_n(t)$, where $\bar{f}_n$ is the stationary flow rate and $d_n$ is non-stationary fluctuation (white Gaussian noise) at node $n$. Then, as in [91], each potential is updated by

$$\begin{aligned}\theta_n(t+1) &= (\alpha+1)\theta_n(t) - \alpha\theta_n(t-1) \\ &+ \beta\sigma_n \left( \sum_{k\in\mathcal{N}_b(n)} \{\theta_k(t) - \theta_n(t)\} + \bar{f}_n + d_n(t) \right) + \eta_n(t),\end{aligned} \tag{4.2}$$

where $\eta_n$ represents feedback input received from the controller. If node $n$ does not receive

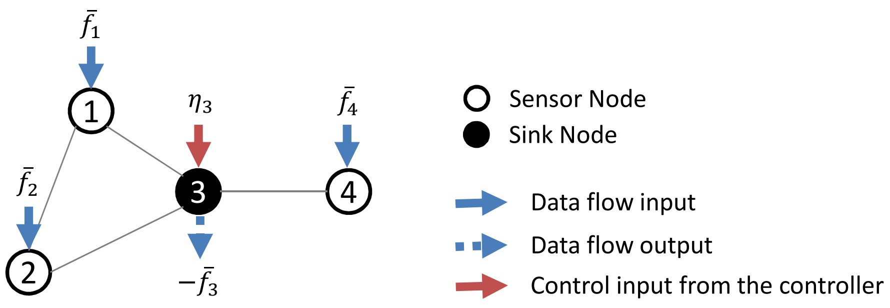


Figure 4.4: An example of a network with 3 sensor nodes and a sink node. $\bar{f}_1$, $\bar{f}_2$, $\bar{f}_3$ and $\bar{f}_4$ are stationary flow rates for each node. $\eta_3$ is the control input to node 3 from the controller (the other nodes are not directly controlled by the controller, and $\eta_1$, $\eta_2$ and $\eta_4$ is 0 at all times). $d_1$, $d_2$, $d_3$ and $d_4$ are non-stationary fluctuations at each node although they are not described in this figure.

any feedback input directly from the controller, then $\eta_n(t) = 0$. In our proposal, the controller collects and estimates the node potentials, and provides to the network feedback inputs $\boldsymbol{u}(t) = [\eta_1(t)\ \eta_2(t)\ \cdots\ \eta_{N_{ctrl}}(t)]^T$ (where $N_{ctrl}$ denotes the number of nodes that receive feedback from the controller), as described later. Because of feedback inputs, we consider that node potentials can converge faster than in the non-control scheme (Equation (4.1)) where each node updates its potential based only on local interactions with neighbors. In [91], $\sigma_n$ is set to $\sigma_0/|\mathcal{N}_b(n)|$, but this value may lead to oscillation of potentials in some situations since Equation (4.9) (after-mentioned) has no solution. In this study we therefore set $\sigma_n$ to the constant value $\sigma$ for all $n$ ($n \in \{1, 2, \cdots, N\}$). Now we define $\boldsymbol{\theta}_i(t)$ as $[\theta_i(t)\ \theta_i(t+1)]$, and Equation (4.2) is rewritten by

$$\boldsymbol{\theta}_n(t+1) = \boldsymbol{A}_1\boldsymbol{\theta}_n(t) + \boldsymbol{A}_0\left(\sum_{k\in\mathcal{N}_b(n)}\{\boldsymbol{\theta}_k(t) - \boldsymbol{\theta}_n(t)\} + \boldsymbol{M}\left(\bar{f}_n + d_n(t)\right)\right) + \boldsymbol{M}\eta_n(t), \quad (4.3)$$

where

$$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 1 \\ -\alpha & \alpha+1 \end{bmatrix}, \boldsymbol{A}_0 = \begin{bmatrix} 0 & 0 \\ 0 & \beta\sigma \end{bmatrix}, \boldsymbol{M} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

In our proposal, the controller estimates potentials of unobservable nodes and future potentials, and determines control inputs with $H^\infty$ optimization (described below), allowing us to describe the system dynamics in the form of the state space model. We first formulate the potential dynamics of all nodes. For example, we consider the case of a network of 4 nodes as shown in Figure 4.4. From Equation (4.3), the potential dynamics of node 1 is given by

$$\boldsymbol{\theta}_1(t+1) = \boldsymbol{A}_1\boldsymbol{\theta}_1(t) + \boldsymbol{A}_0\left(\sum_{k\in\{2,3\}}\{\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_1(t)\} + \boldsymbol{M}\left(\bar{f}_1 + d_1(t)\right)\right). \tag{4.4}$$

Similarly to Equation (4.4), the potential dynamics of node 2, 3 and 4 are given, respectively, by:

$$\begin{aligned}
\boldsymbol{\theta}_2(t+1) &= \boldsymbol{A}_1\boldsymbol{\theta}_2(t) + \boldsymbol{A}_0\left(\sum_{k\in\{1,3\}}\{\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_2(t)\} + \boldsymbol{M}\left(\bar{f}_2 + d_2(t)\right)\right), && (4.5)\\
\boldsymbol{\theta}_3(t+1) &= \boldsymbol{A}_1\boldsymbol{\theta}_3(t) + \boldsymbol{A}_0\left(\sum_{k\in\{1,2,4\}}\{\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_3(t)\} + \boldsymbol{M}\left(\bar{f}_3 + d_3(t)\right)\right) + \boldsymbol{M}\eta_3(t), && (4.6)\\
\boldsymbol{\theta}_4(t+1) &= \boldsymbol{A}_1\boldsymbol{\theta}_4(t) + \boldsymbol{A}_0\left(\sum_{k\in\{3\}}\{\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_4(t)\} + \boldsymbol{M}\left(\bar{f}_4 + d_4(t)\right)\right). && (4.7)
\end{aligned}$$

From Equations (4.4)–(4.7), the potential dynamics of the network shown in Figure 4.4 is given by

$$\begin{aligned}
\begin{bmatrix}\boldsymbol{\theta}_1(t+1)\\ \boldsymbol{\theta}_2(t+1)\\ \boldsymbol{\theta}_3(t+1)\\ \boldsymbol{\theta}_4(t+1)\end{bmatrix} &= \left(\boldsymbol{I}_{4\times4}\otimes\boldsymbol{A}_1 - \begin{bmatrix}2&-1&-1&0\\-1&2&-1&0\\-1&-1&3&-1\\0&0&-1&1\end{bmatrix}\otimes\boldsymbol{A}_0\right)\begin{bmatrix}\boldsymbol{\theta}_1(t)\\ \boldsymbol{\theta}_2(t)\\ \boldsymbol{\theta}_3(t)\\ \boldsymbol{\theta}_4(t)\end{bmatrix}\\
&+ \left(\beta\sigma\left(\begin{bmatrix}f_1\\f_2\\f_3\\f_4\end{bmatrix} + \begin{bmatrix}d_1\\d_2\\d_3\\d_4\end{bmatrix}\right) + \begin{bmatrix}0\\0\\\eta_3\\0\end{bmatrix}\right)\otimes\boldsymbol{M}, && (4.8)
\end{aligned}$$

where $\boldsymbol{I}_{N\times N}$ is the identity matrix of $N\times N$. The matrix $\begin{bmatrix}2&-1&-1&0\\-1&2&-1&0\\-1&-1&3&-1\\0&0&-1&1\end{bmatrix}$ corresponds to the graph Laplacian, which indicates the network topology. We refer to the graph Laplacian as $\boldsymbol{\Gamma}$ in this study. The element $l_{ij}$ of graph Laplacian $\boldsymbol{\Gamma}$ is given by

$$l_{ij} = \begin{cases} \deg(i), & \text{if } i = j \\ -1, & \text{if } i \neq j \wedge \text{node } i \text{ is adjacent to node } j \\ 0, & \text{otherwise} \end{cases},$$

where $\deg(i)$ corresponds to the degree of node $i$. The operation $\otimes$ that is called the Kronecker product is an operation on two matrices. Given an $m \times n$ matrix $\boldsymbol{J}$ and a $p \times q$ matrix $\boldsymbol{K}$, the Kronecker product $\boldsymbol{J} \otimes \boldsymbol{K}$ is given by

$$\boldsymbol{J} \otimes \boldsymbol{K} = \begin{bmatrix} j_{11}\boldsymbol{K} & \cdots & j_{1n}\boldsymbol{K} \\ \vdots & & \vdots \\ j_{m1}\boldsymbol{K} & \cdots & j_{mn}\boldsymbol{K} \end{bmatrix}.$$

Then, we define a vector $\boldsymbol{\Theta}(t)$ that shows potential values for all nodes as

$$\boldsymbol{\Theta}(t) := [\boldsymbol{\theta}_1(t)\ \boldsymbol{\theta}_2(t)\ \cdots\ \boldsymbol{\theta}_N(t)]^T.$$

As in Equation (4.8), the potential dynamics of all nodes is described with $\boldsymbol{\Theta}(t)$ as

$$\boldsymbol{\Theta}(t+1) = \boldsymbol{A}\boldsymbol{\Theta}(t) + (\beta\sigma(\boldsymbol{F} + \boldsymbol{D}(t)) + \boldsymbol{E}\boldsymbol{u}(t)) \otimes \boldsymbol{M},$$

where

$$\boldsymbol{A} = \boldsymbol{I}_{N\times N} \otimes \boldsymbol{A}_1 - \boldsymbol{\Gamma} \otimes \boldsymbol{A}_0.$$

The flow rate vector $\boldsymbol{F}$ is defined as $\left[\bar{f}_1\ \cdots\ \bar{f}_N\right]^T$, and the fluctuation rate vector $\boldsymbol{D}(t)$ is defined as $[d_1(t)\ \cdots\ d_N(t)]^T$. Note that the $(N \times N_{ctrl})$-matrix $\boldsymbol{E}$ specifies the controllable node, that is, the element $e_{ij} \in \{0, 1\}$ of $\boldsymbol{E}$ is 1 if and only if node $i$ receives the $j$-th element of $\boldsymbol{u}(t)$ as control input $\eta_i(t)$ (in the case with the network shown in Figure 4.4, $\boldsymbol{E}$ is set to $[0\ 0\ 1\ 0]^T$). With the larger number of controllable nodes, i.e., with denser $\boldsymbol{E}$, the influence of the optimal control is larger so that the convergence speed is faster. On the contrary, significant properties originating from self-organization such as scalability and adaptability could be lost with too many controllable nodes.

Under these dynamics, the target potential distribution is given by a solution of

$$(\boldsymbol{I}_{2N\times 2N} - \boldsymbol{A})\bar{\boldsymbol{\Theta}} = \beta\sigma \boldsymbol{F} \otimes \boldsymbol{M}. \tag{4.9}$$

Then, we define $\boldsymbol{X}(t)$ as the regulation error $\bar{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(t)$, which shows the differences between the convergence values and the current values of potentials for each node. The potential dynamics can be rewritten using $\boldsymbol{X}(t)$ in the form of the state space representation by Equation (4.10), with which the controller can estimate potentials of unobservable nodes and future potentials.

$$\boldsymbol{X}(t+1) = \boldsymbol{A}\boldsymbol{X}(t) + \boldsymbol{B}_1\boldsymbol{d}(t) + \boldsymbol{B}_2\boldsymbol{u}(t). \tag{4.10}$$

Here, the $N_{ctrl}$-dim vector $\boldsymbol{u}$ (resp. $N$-dim vector $\boldsymbol{d}$) concatenates $\eta_n(t)$ for controllable nodes (resp. $d_n$). $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ characterizes the effect of noises and control inputs, respectively, given by

$$\boldsymbol{B}_1 = \boldsymbol{B}_1' \otimes \begin{bmatrix} 0 \\ \beta\sigma \end{bmatrix}, \boldsymbol{B}_2 = \boldsymbol{E} \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

$\boldsymbol{B}_1'$ characterizes the variance/co-variance of the noise distribution injected at each node. For simplicity, we choose $\boldsymbol{B}_1' = \boldsymbol{I}_{N\times N}$ in this study, which implies each node has independent noise with unit variance. Note that the dynamics (Equation (4.10)) depends on the graph Laplacian $\boldsymbol{\Gamma}$, but not on the flow rate $\bar{f}$.

### 4.3.3 Optimal Controller Design

We next explain the controller dynamics. We consider the case where the controller can observe

$$\begin{aligned} \boldsymbol{Y}(t) &= (\boldsymbol{H}^T \otimes \boldsymbol{I}_{2\times 2})\boldsymbol{X}(t) \\ &= (\boldsymbol{H}^T \otimes \boldsymbol{I}_{2\times 2})(\bar{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(t)), \end{aligned} \tag{4.11}$$

where $\boldsymbol{Y}(t)$ is a $2N_{obs}$-dim vector, with $N_{obs}$ being the number of observable nodes, and an $(N \times N_{obs})$ matrix $\boldsymbol{H}$ determines observable nodes in the same manner as $\boldsymbol{E}$ (if the controller can only observe node 3 in the network shown in Figure 4.4, $\boldsymbol{H}$ is set to $[0\ 0\ 1\ 0]^T$). Note that the controller can estimate the potential values of all nodes based on the information of only a partial set of nodes, and therefore, not all elements of $\boldsymbol{H}$ need to be set to 1. If more nodes are monitored by the controller, it can estimate potential values of all nodes more precisely while the communication overhead for collecting information of observable nodes becomes much larger. The future node potentials can be estimated by Equation (4.10) using observable information expressed with Equation (4.11). Considering the future potentials, the controller calculates a $\boldsymbol{u}(t)$ that accelerates the convergence speed of potentials.

Then, the control input is calculated according to

$$\tilde{\boldsymbol{X}}(t+1) = \boldsymbol{A}_c\tilde{\boldsymbol{X}}(t) + \boldsymbol{B}_c\boldsymbol{Y}(t), \tag{4.12}$$

$$\boldsymbol{u}(t) = \boldsymbol{C}_c\tilde{\boldsymbol{X}}(t) + \boldsymbol{D}_c\boldsymbol{Y}(t), \tag{4.13}$$

where $2N$-dim vector $\tilde{\boldsymbol{X}}$ is an internal model for the controller and the controller estimates potentials of all nodes with the model. $\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $\boldsymbol{C}_c$, and $\boldsymbol{D}_c$ are design parameters.

Concerning the performance criteria, let us define

$$\phi(k) = \boldsymbol{X}(k)^T\boldsymbol{X}(k) + r\boldsymbol{u}(k)^T\boldsymbol{u}(k)$$

as the stage cost where $r$ specifies the trade-off between convergence speed and input energy. With a larger $r$, control inputs become smaller and the stability of the system is enhanced. Namely, potentials change more gently, whereas the convergence speed of potentials becomes slower. Our design objective is then to minimize the worst-case error

$$\sup_{\boldsymbol{d}} \frac{\sum_{k=0}^{\infty}\phi(k)}{\sum_{k=0}^{\infty}\boldsymbol{d}(k)^T\boldsymbol{d}(k)}.$$

This min-max type problem is called $H^\infty$ optimization [47]. Given $\boldsymbol{A}$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$ and $\boldsymbol{H}$, it is known that the optimal $\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $\boldsymbol{C}_c$, $\boldsymbol{D}_c$ can be obtained based on semi-definite programming (see also Section 4.5). In general, the degree of the optimal controller, that is the size of $\tilde{\boldsymbol{X}}$, is the same as that of the system to be controlled, which is $2N$ in this case. Note that we need to solve this optimization problem only when the connection topology changes. In other words, we only have to update $\tilde{\boldsymbol{X}}$ in Equation (4.12) and compute Equation (4.13) so long as the connection topology does not change.

In [45], we previously proposed potential-based routing with an optimal control that is explained in this section, but we only evaluate the situation where the controller observes all nodes, i.e., $N_{obs} = N$ and $\boldsymbol{H} = \boldsymbol{I}_{N\times N}$. The upper bound of the order of the communication cost for observation is $O(NP_{avg})$ where $P_{avg}$ corresponds to the average path length of the network. Note that the average path length of the network generally depends on the number $N$ of nodes. For example, the average path length of Erdös-Rényi network (a random network) [94] and Barabási-Albert model (a scale-free network) [95] is $O(\log N)$, and that of Watts-Strogatz network (a small-world network) [96] is $O(N)$ if $N < N^*$ and $O(\log N)$ if $N > N^*$[1]. Therefore, in this study, we show that the controller can estimate potentials of non-observable nodes based on only a part of nodes' potentials with simulation evaluations in Section 4.5. Note that, if the average path length is quite small against the network size as in the case with WWW, the external controller can observe a large number of nodes with small communication cost, which allows the external controller to estimate the potential dynamics more correctly.

Moreover, as the number of nodes increases, that is, the size of vector $\boldsymbol{X}(t)$ becomes large, it becomes unfeasible to control the system using the model expressed in Equation (4.10) and Equation (4.11), owing to the considerable computational cost. The design parameters ($\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $\boldsymbol{C}_c$, and $\boldsymbol{D}_c$) are calculated with $O(N^3)$ because the computational cost is proportion to the cube of the size of $\boldsymbol{X}$ [97]. Control inputs ($\boldsymbol{u}(t)$) are calculated with $O(N^2)$ because they are calculated by

[1] $N^*$ is a length which depends on the randomness $p$ of a small-world network. See [95] for details.

Equations (4.12) and (4.13). We therefore approximate the model with a reduced-order model for reducing the computational cost.

## 4.4 Model Reduction

In our proposal, the controller uses a reduced-order model for estimating network dynamics to reduce the computational cost. In this section, we explain the model reduction of the network dynamics model described in Section 4.3.

In the reduced-order model, the dynamics of the system is expressed as an $(h \times 1)$-vector $\boldsymbol{X}_r(t)$ whose elements are linear transformations of the original model $\boldsymbol{X}(t)$, and the reduced-order model is given by Equations (4.14) and (4.15).

$$\boldsymbol{X}_r(t+1) = \boldsymbol{A}_r\boldsymbol{X}_r(t) + \boldsymbol{B}_{r1}\boldsymbol{d}(t) + \boldsymbol{B}_{r2}\boldsymbol{u}_r(t), \tag{4.14}$$

$$\boldsymbol{Y}_r(t) = \boldsymbol{C}_r\boldsymbol{X}_r(t) + \boldsymbol{D}_r\boldsymbol{d}(t). \tag{4.15}$$

Here, $\boldsymbol{X}_r$ is an $h$-dim vector with $h$ $(< 2N)$ being the order of the approximate model, and $u_r(t)$ corresponds to control inputs provided by the controller with the reduced-order model. We need to choose matrices $\boldsymbol{A}_r$, $\boldsymbol{B}_{r1}$, $\boldsymbol{B}_{r2}$, $\boldsymbol{C}_r$, and $\boldsymbol{D}_r$ of compatible dimensions such that $\boldsymbol{Y}_r(t) \approx \boldsymbol{Y}(t)$ for all input $\boldsymbol{u}_r(t)$, $\boldsymbol{d}_r(t)$. Many researchers have studied a variety of methods to approximate a model with a reduced-order model to control large, complex systems [49]. In our proposal, we approximate the original model based on a '*balanced realization*' that is highly compatible with the model expressed in the state space representation [47, 49]. In model reduction, a reduced-order model needs to have the same response characteristics as the original model for the accurate estimation of potentials. Here, response characteristics mean the effectiveness of inputs to the system.

The control input is then calculated in the same manner as $\boldsymbol{u}_r(t)$ according to

$$\tilde{\boldsymbol{X}}_r(t+1) = \boldsymbol{A}_{cr}\tilde{\boldsymbol{X}}_r(t) + \boldsymbol{B}_{cr}\boldsymbol{Y}_r(t), \tag{4.16}$$

$$\boldsymbol{u}_r(t) = \boldsymbol{C}_{cr}\tilde{\boldsymbol{X}}_r(t) + \boldsymbol{D}_{cr}\boldsymbol{Y}_r(t), \tag{4.17}$$

where $\boldsymbol{A}_{cr}$, $\boldsymbol{B}_{cr}$, $\boldsymbol{C}_{cr}$, and $\boldsymbol{D}_{cr}$ are design parameters. Now, we can use $\boldsymbol{Y}(t)$ for $\boldsymbol{Y}_r(t)$ in Equation (4.17) under the well-designed $\boldsymbol{C}_r$, and $\boldsymbol{D}_r$ because of the relation of $\boldsymbol{Y}_r(t) \approx \boldsymbol{Y}(t)$. We similarly obtain the $H^\infty$ optimal controller of order $h$ for the reduced system. These parameters determine how potential values observed by the controller affect feedback inputs.

A reduced-order model can describe the system dynamics with a constant number $h$ of state variables and the computational cost can be reduced using it. With a reduced-order model, the design parameters ($\boldsymbol{A}_{cr}$, $\boldsymbol{B}_{cr}$, $\boldsymbol{C}_{cr}$, and $\boldsymbol{D}_{cr}$) are calculated with $O(h^3)$, and control inputs ($\boldsymbol{u}_r(t)$) are calculated with $O(h^2)$. In general, a model that has more state variables lets the controller estimate more correctly, but the computational cost is larger. In contrast, the computational cost is smaller but the estimation error can be larger in a model that has fewer state variables. Therefore, $h$ needs to be properly determined in accordance with the requirements or system properties (see also Section 4.5).

In Section 4.5, we conduct a network simulation to reveal the packet-level dynamics and to show that Equations (4.2)–(4.17) assuming synchrony of nodes can be adopted to asynchronous network systems.

## 4.5 Performance Evaluation

In this section, we evaluate our proposal to show that the convergence speed is enhanced by introducing optimal feedback using the reduced-order model. We conduct computer simulation and evaluate the convergence speed comparing our proposal (potential-based routing with optimal feedback (PBR-opt) and one using the model reduction (PBR-opt-mr)) with the non-control scheme proposed in [91]. First, in Subsection 4.5.2, we evaluate the potential convergence speed after traffic changes to show that our proposed methods (PBR-opt and PBR-opt-mr) enhance the convergence speed of potentials. Moreover, we show that PBR-opt-mr can be adapted to large-scale networks in Subsection 4.5.3. Next, in Subsection 4.5.4 for demonstrating the robustness of our proposal, we consider a changing environment where several nodes fail but the controller does not collect network topology information, and thus cannot detect these failures.

In conducting simulation experiments, for the network simulator we use an event-driven packet-level simulator written in Visual C++ that calls MATLAB functions *dhinflmi*[2] to design the optimal controller and *balred*[3] to obtain a reduced-order model on a 64 bit operating system with Intel(R) Xeon(R) CPU with 2.70 GHz, and 64.0 GB memory. The simulator is of our own making and, in the MAC layer, each node sends information about its own potential to its neighbors for their potential updates using intermittent receiver-driven data transmission (IRDT) [70], an asynchronous receive-drive data transmission protocol, in the MAC layer. Note that this setting does not mean that our proposal depends on specific MAC layer protocols such as IRDT. In the physical layer, we use a simple disk model for wireless communication, where the packet reception between two nodes is successful if two nodes are within the pre-defined communication range. Also we use a simple packet collision model where a receiver node always drops a packet if other packets arrives during the reception of the packet. In the simulator, the asynchrony of components, i.e., the controller

[2]*dhinflmi* is a function for designing the $H^\infty$ optimal controller [97] of a discrete system. Given parameter matrices $\boldsymbol{A}$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\boldsymbol{C}$, $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ for the system dynamics $\boldsymbol{X}(t+1) = \boldsymbol{A}\boldsymbol{X}(t) + \boldsymbol{B}_1\boldsymbol{D}(t) + \boldsymbol{B}_2\boldsymbol{u}(t)$ and the information $\boldsymbol{Y}(t) = \boldsymbol{C}\boldsymbol{X}(t) + \boldsymbol{D}_1\boldsymbol{D}(t) + \boldsymbol{D}_2\boldsymbol{u}(t)$ that is observed by the controller, $dhinflmi$ computes design parameters $\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $\boldsymbol{C}_c$ and $\boldsymbol{D}_c$ that are referred in Equations (4.12), (4.13.) The matrix $\boldsymbol{D}(t)$ shows non-stability fluctuations, $\boldsymbol{u}(t)$ are control inputs, $\boldsymbol{A}, \boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{C}, \boldsymbol{D}_1, \boldsymbol{D}_2$

[3]*balred* is a function for reducing the model order [98]. Given design parameter matrices $\boldsymbol{A}$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\boldsymbol{C}$, $\boldsymbol{D}_1$ for the original model and the degree $h$ of a reduced-order model, *balred* computes a $h$th-order approximation of the original model.

and all nodes, is implemented as follows. The controller and all nodes do not match their timing to provide feedback and to update their potentials. The asynchronous behavior of all nodes is implemented on a single thread with an event queue in our simulator. Note that we set the interval of the control feedback by the controller and the intervals of potential updates in each node to be equal in accordance with Equations (4.10) and (4.14) so that the controller can estimate potentials of nodes with small errors.

### 4.5.1 Simulation Settings

We evaluate changes of potentials in potential-based routing and the number of data packets delivered to each sink node after traffic changes or node failures. Network models used in evaluations consist of an external controller and sensor/sink nodes. Only sink nodes are directly connected to the controller, so the controller monitors network states via sink nodes and sends suitable control inputs to sink nodes. The controller provides feedback to each sink node at interval $T_f$, while at interval $T_p$, each node updates its next potential value. Typically, $T_f = T_p$ for matching with the potential dynamics described by Equation (4.2). In real situations, it is difficult to monitor up-to-date potential values of all nodes in the network because of the overhead for collecting potential values, especially when the number of nodes increases. Therefore, here we consider that the controller can directly monitor only nodes within $p$ hops from the sink nodes. That is, nodes within $p$ hops from the sink nodes send a control message to the sink nodes at interval $T_i$ to notify the controller of their potential values. Otherwise, the controller does not monitor node potentials outside $p$ hops from sink nodes, and only estimates them using the model (Equation (4.14)), which describes potential dynamics through local interactions among nodes. The interval of control message emission $T_i$ is set to 50 s, $p$ is set to 2, and the $i$-th element $h_i$ of $\boldsymbol{H}$ is set to

$$h_i = \begin{cases} 1, & \text{if node } i \text{ is within } p \text{ hops from sink nodes} \\ 0, & \text{otherwise} \end{cases}.$$

At the beginning of the simulation, potential values of all nodes, including sensor nodes and sink nodes, are initialized to $0$. During the first $1,000$ s, each node exchanges its potential values with neighbor nodes and updates its potential value at interval $T_p$ according to Equation (4.2) so that the potential values are stabilized. For this, we do not generate data packets during that time duration. At $1,000$ s data packets begin to be generated at sensor nodes according to the Poisson process with their flow rate.

We evaluate the convergence speed of potentials and data packets delivered to each sink node after traffic changes. To measure the convergence speed of potentials, we define the degree of the potential convergence $\epsilon_n(t)$ for each node that is given by

$$\epsilon_n(t) = \frac{|\bar{\theta}_n - \theta_n(t)|}{|\bar{\theta}_n|},$$

where $\bar{\theta}_n$ corresponds to the target potential value of node $n$. We consider convergence to be achieved when $\epsilon_n(t)$ for all nodes becomes sufficiently small. Convergence time is defined as the minimum time taken by all sensor and sink nodes to satisfy the condition

$$\epsilon_n(t) < c, \tag{4.18}$$

where $c\,(\geq 0)$ is a constant value. In an ideal situation where all nodes are completely synchronized and there is no noise, all sensor and sink nodes satisfy Equation (4.18) in the end even if $c = 0$, but in an actual situation, not all nodes can satisfy Equation (4.18). The larger the value of $c$ is, the shorter the time needed for the achievement of the potential convergence is. Therefore, $c$ needs to be set carefully according to the purpose of the evaluation. In this study, we set $c$ to $0.2$ in order to

Table 4.1: Parameter settings

| parameter | value |
|---|---|
| $\alpha$ | 0.4 |
| $\beta$ | 0.2 |
| $\sigma$ | 0.1 |
| $r$ | 10 |
| $T_f$ | 50 s |
| $T_p$ | 50 s |

evaluate the convergence speed strictly.

Energy efficiency is a significant challenging task in wireless sensor networks. With respect to our proposal, the computational cost of each sensor node is fairly low because each sensor node behaves based only on local information and simple rules. This is an inherent characteristic of self-organizing systems. Moreover, our proposal can achieve load balancing by setting flow matrix $\boldsymbol{F}$ so that each sink node receives data packets equally, which we explain in Subsection 4.5.2. Finally, our proposal does not depend on specific MAC layer protocols. Therefore, we can reduce energy for data transmission between adjacent nodes by introducing energy-efficient MAC layer protocols such as IRDT that we use in this evaluation.

Parameters are summarized in Table 4.1. Optimal parameters $(\alpha, \beta\sigma)$ for PBR-no-ctrl are given in [91] [4], but potentials of nodes diverge to infinity and do not converge. Therefore, we use original settings shown in Table 4.1 even for PBR-no-ctrl. All results presented below are averaged over 40 simulation runs for each parameter setting.

### 4.5.2 Performance evaluation of reduced order controller

Here, we evaluate the convergence speed considering constraints in wireless sensor networks.

[4]Optimal $\alpha$ and $\beta\sigma$ are respectively given by $\frac{1-\xi}{1+\xi}$ and $\frac{\alpha+1}{2}$ for the fastest convergence rate of potentials, where $\xi = \sqrt{1-\nu_1^2}$. $\nu_1$ is the spectrum radius of the matrix $\boldsymbol{S} = \boldsymbol{I}_{N\times N} - \boldsymbol{G}^{-1}\boldsymbol{\Gamma}$ where $\boldsymbol{G}$ is the degree matrix and $\boldsymbol{\Gamma}$ is the graph Laplacian of the network. The spectrum radius of the matrix $\boldsymbol{S}$ is defined as the max value among absolute values of eigenvalues of $\boldsymbol{S}$. See [91] for more details.

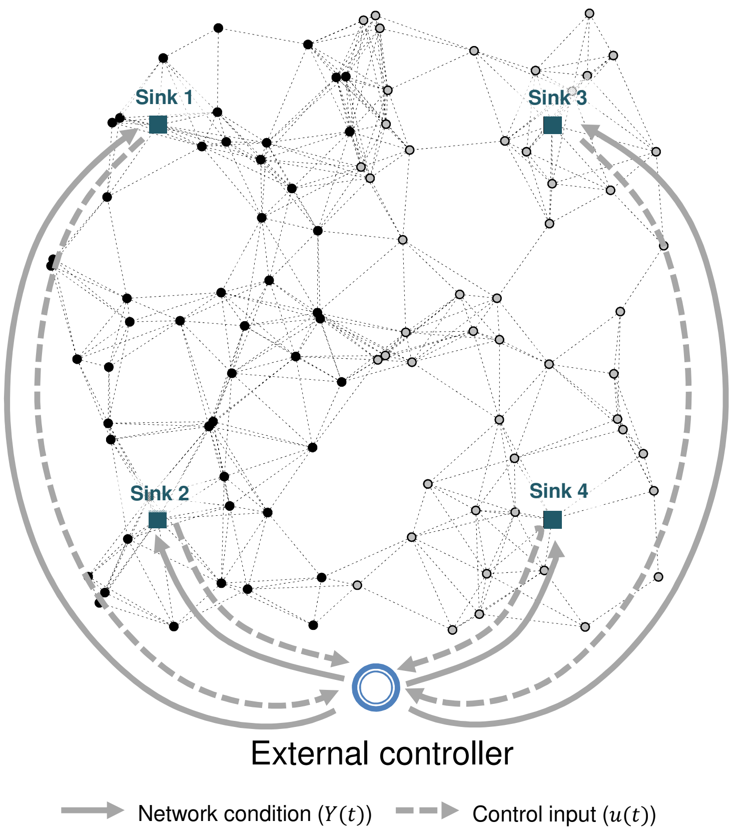
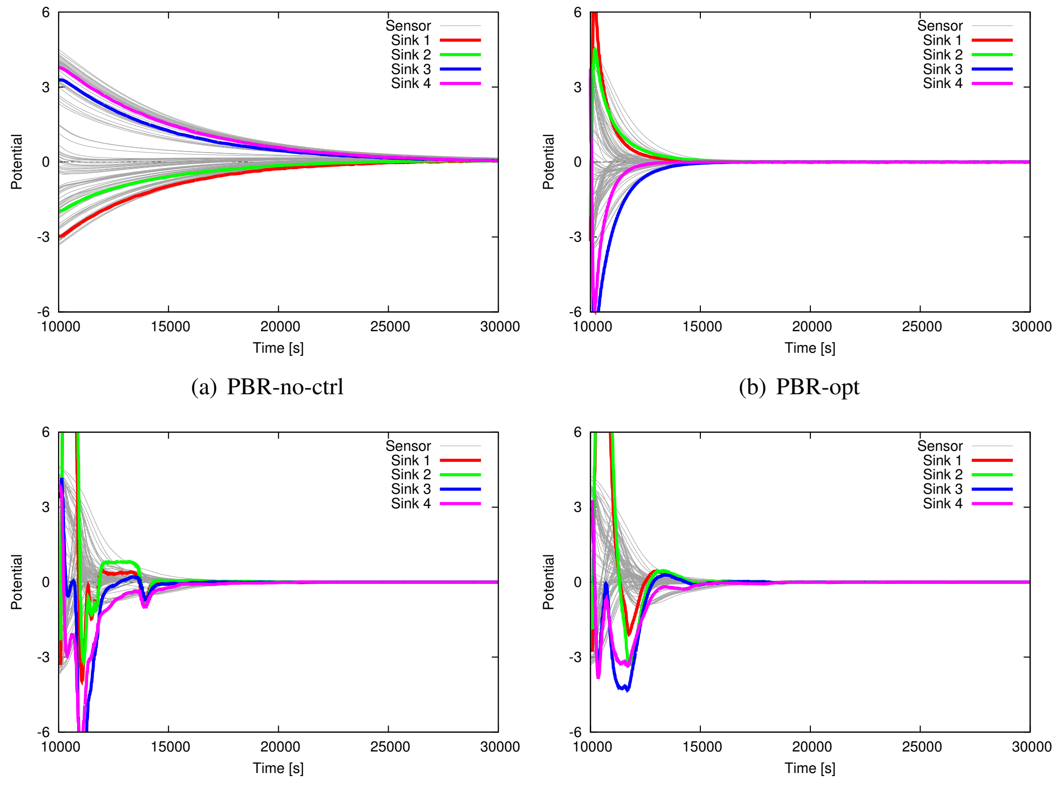


Figure 4.5: Network topology ($N = 104$)

Figure 4.5 shows the network model with 104 nodes (including four sink nodes) used for this evaluation. Sink nodes are illustrated with squares and sensor nodes with dots in that figure. In this network model described in Figure 4.5, every sensor node is within 3 hops from a sink node. The number of sensor nodes within 2 hops from a sink node is 86, that is, the external controller observes potentials of 86 sensor nodes in this network. In this evaluation, at $10,000$ s from the beginning of the simulation, data packet generation rates of sensor nodes are changed to examine the convergence speed of our method. Data packet generation rates are initially set to be $0.0005$ packets/s for sensor nodes in the left half of Figure 4.5 (illustrated with black dots), and $0.0015$ packets/s for sensor nodes in the right half (illustrated with gray dots). After traffic changes at $10,000$ s, data packet



(a) PBR-no-ctrl (b) PBR-opt

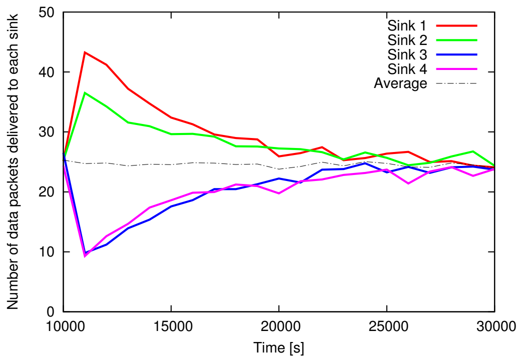
(c) PBR-opt-mr ($h = 3$) (d) PBR-opt-mr ($h = 100$)

Figure 4.6: Potential convergence in case of traffic changes

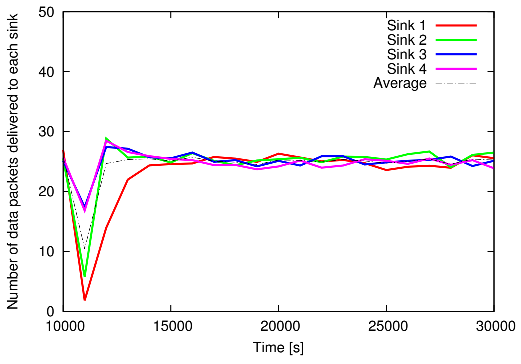
generation rates are increased to $0.0015$ packets/s for the left half sensor nodes and decreased to $0.0005$ packets/s for the right half nodes. The average data generation rate of a node of $0.001$ packets/s corresponds to $\bar{f}_n = 1$, so before traffic changes the flow rate vector $\boldsymbol{F} = [\bar{f}_1 \cdots \bar{f}_N]^T$ is given by

$$\bar{f}_i = \begin{cases} 0.5, & \text{if } i \in \mathcal{N}_{sen_l} \\ 1.5, & \text{if } i \in \mathcal{N}_{sen_r} \\ -\frac{|\mathcal{N}_{sen}|}{|\mathcal{N}_{sin}|}, & \text{if } i \in \mathcal{N}_{sin} \end{cases}, \tag{4.19}$$

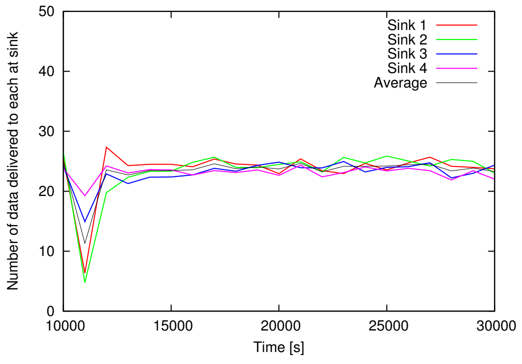
where $\mathcal{N}_{sen}$ corresponds to the set of sensor nodes, and $\mathcal{N}_{sin}$ corresponds to that of sink nodes. $\mathcal{N}_{sen_l}$ is the set of sensor nodes in the left half of Figure 4.5 and $\mathcal{N}_{sen_r}$ is that of sensor nodes in



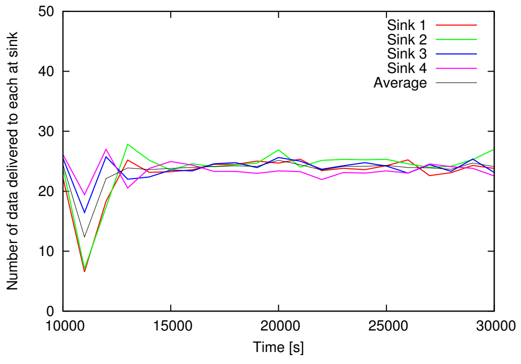
(a) PBR-no-ctrl



(b) PBR-opt



(c) PBR-opt-mr ($h = 3$)



(d) PBR-opt-mr ($h = 100$)

Figure 4.7: Data packets delivered to each sink node in case of traffic changes

the right half. Note that we construct the potential fields such that all sink nodes can receive data packets equally because load balancing is a challenging task in wireless sensor networks. Thus, the flow rate at each sink node is ideally given by $|\mathcal{N}_{sen}|/|\mathcal{N}_{sin}|$ $(= 25)$. Similarly, after the traffic changes, the $i$-th element of $\boldsymbol{F}$ is given by

$$\bar{f}_i = \begin{cases} 1.5, & \text{if } i \in \mathcal{N}_{sen_l} \\ 0.5, & \text{if } i \in \mathcal{N}_{sen_r} \\ -\frac{|\mathcal{N}_{sen}|}{|\mathcal{N}_{sin}|}, & \text{if } i \in \mathcal{N}_{sin} \end{cases}, \tag{4.20}$$

In this evaluation, we conduct simulation in the case where flows of data packets massively change the network as mentioned above in order to show that our proposal can adapt to massive environmental changes. A system that can adapt to massive environmental changes is considered to be able to adapt to minor fluctuations. Moreover, we assume that the data generation rate does not change until $10,000$ s from the beginning of the simulation in order to compare our proposal and the non-control scheme under the same conditions.

In the evaluation of PBR-opt-mr, degree $h$ of the reduced order model is set to 3 and 100. As explained in Section 4.4, with lower $h$, the computational cost is lower but the estimation error is larger, which results in the slow convergence speed of potentials. Therefore, we evaluate not only in the case with low $h$ $(= 3)$ but also in the case with high $h$ $(= 100)$ for clarifying how the degree of the reduced order model effects potential convergence.

In the evaluation of PBR-opt, the design parameters ($\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $\boldsymbol{C}_c$, $\boldsymbol{D}_c$) cannot be calculated with the *dhinflmi* function when $N = 104$, i.e., the size of the model is 208, due to the considerable computational cost. This motivates us to reduce the computational burden upon controller parameter design and controller state updates. For comparison, we utilize an optimal static controller designed by the *dlqr* function[5], where the controller is assumed able to monitor the latest conditions of all nodes with no communication delay or control overhead (i.e., $h_i = 1$ for all nodes). It is known that static controllers can achieve satisfactory performance with such state feedback settings [47].

Figure 4.6 shows changes of potential values of the non-control scheme, PBR-opt, and PBR-opt-mr with $h = 3$, and 100. More exactly, Figure 4.6 plots $\boldsymbol{X}(t) = \bar{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(t)$ against time $t$, and the potential convergence is achieved when each element of $\boldsymbol{X}(t)$ is sufficiently close to 0, that is, Equation (4.18) is satisfied. In the figure, thick lines correspond to potential changes of the two

[5]The function *dlqr* is linear-quandratic (LQ) state-feedback regulator for discrete-time state space system. Given parameter matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $q$ and $r$ for the system dynamics $\boldsymbol{X}(t+1) = \boldsymbol{X}(t) + \boldsymbol{B}\boldsymbol{u}(t)$, *dlqr* computes $\boldsymbol{K}$, $\boldsymbol{S}$ and $e$. Then, control inputs $\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{X}(t)$ using the optimal gain matrix $\boldsymbol{K}$ minimizes the quandratic cost function $J(u) = \sum_{n=1}^{\infty} (q||\boldsymbol{X}(n)|| + r||\boldsymbol{u}(n)||)$ for the system. Note that the controller needs to observe all nodes for calculating control inputs by $\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{X}(t)$. computational cost for calculating $\boldsymbol{K}$ with *dlqr* function is much lower than that for calculating $\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $\boldsymbol{C}_c$ and $\boldsymbol{D}_c$ with *dhinflmi* function.

Table 4.2: Potential convergence time

| Scheme | Convergence time [s] |
|---|---|
| PBR-no-ctrl | 70,417 |
| PBR-opt | 9,174 |
| PBR-opt-mr ($h = 3$) | 11,765 |
| PBR-opt-mr ($h = 100$) | 9,313 |

sink nodes, and thin lines correspond to those of other sensor nodes. Sink node potentials change more than those of sensor nodes, because sink nodes receive feedback inputs $\boldsymbol{u}$ directly from the controller. Feedback inputs provided to each sink node are different, so the potential change of each sink node varies. On the other hand, sensor nodes are indirectly affected by feedback inputs via sink nodes through potential updates by Equation (4.2). As shown in the figure, if the potential value of a sink node increases (resp. decreases) due to feedback inputs from the controller, potential values of sensor nodes near the sink node also increase (resp. decrease).

As shown in Figure 4.6, our proposal can enhance the convergence speed of potentials, as compared with the case of no control scheme, and Table 4.2 shows the time needed from traffic changes until the potential convergence is achieved when $c = 0.2$. Figure 4.6 shows potential changes only within $10,000$–$30,000$ s but it takes $70,417$ s for the potential convergence, i.e., satisfying Equation (4.18), with PBR-no-ctrl as shown in Table 4.2. As a result, potential convergence is accelerated by about $5.99$ times and $7.56$ times due to PBR-opt-mr with $h = 3$ and $100$, respectively. Compared with PBR-opt, potential convergence speed is $0.780$ times and $0.985$ times using PBR-opt-mr with $h = 3$ and $100$, respectively. These results indicate that if $h$ is properly chosen, the convergence speed of our proposal can be approximately equal to that of PBR-opt (about $7.68$ times as fast as the original). The convergence speed of PBR-opt-mr with $h = 3$ is a little bit slower than that of PBR-opt-mr with $h = 100$, because the smaller $h$ is the larger the approximation error is. Specifically, if $h$ is smaller than 3, potentials diverge and cannot converge to targeted potential values although we do not show the result in this study. However, from the point of view of computational cost, when $h$ is 3, PBR-opt-mr can enhance the convergence speed with much lower computational cost than PBR-opt, because the cost of the design parameter calculation is reduced from $O(N^3)$ to $O(h^3)$ and the cost of the control input calculation is reduced from $O(N^2)$ to $O(h^2)$ by the model reduction. Note that the computational cost for designing the controller of PBR-opt using *dhinflmi* is too large to compute in a practical time as explained above. Therefore, PBR-opt cannot be adapt to large-scale networks. It is worth mentioning that the controller of PBR-opt-mr observes only node potentials within two hops of sink nodes (i.e., $p = 2$). The controller estimates potentials of non-observable and future potentials using Equation (4.14), then determines the optimal feedback inputs. This indicates that the controller does not need to collect information of the entire network to enhance the convergence speed of potentials. In general, it becomes more difficult for the controller to estimate potentials correctly as the number of nodes observed by the controller becomes smaller. We will investigate the trade-off between the number of observable nodes and the accuracy of the potential estimation as future work.

At about $14,000$ s in Figure 4.6(c), potentials of nodes rapidly fluctuate. This is because the controller cannot receive control messages, which collect potential information and inform the controller about them, due to packet drops so that the controller temporarily provides irrelevant control inputs. However, such fluctuations are temporary and potentials immediately converge again. Therefore, these fluctuations have little effect on transmission of data packets as shown in Figure 4.7(c) which is mentioned below. Moreover, in the case where the controller collects potential information via sensor nodes (i.e., not sink nodes), traffic congestion around sink nodes is reduced, and therefore, these fluctuations caused by packet drops will be reduced.

Estimation errors of potentials cannot be completely avoided in real networks, because nodes are not synchronized unless the potential dynamics described by Equation (4.2), and the potential values the controller collects are not always correct due to communication delays, dropped data, or interference. In this evaluation, traffic congestion occurs around sink nodes because the controller collects the network information via sink nodes, and $1.32\%$, $1.33\%$ of the control messages for collecting potential values are dropped when using PBR-opt-mr with $h = 3$, $100$ respectively, leading to estimation errors of potentials. Moreover, the asynchrony of the controller and nodes is also a cause of estimation errors because PBR-opt-mr (and also PBR-opt) inherently assumes synchronous systems. Nevertheless, our proposal can achieve fast convergence of potentials despite such errors, which clearly shows that our proposal works well in an asynchronous environment where noise or disturbance exists.

Figure 4.7 shows the average number of data packets delivered to each sink node every $1,000$ s. In each case, the number of data packets delivered to each sink node becomes disproportionate after the traffic changes at $10,000$ s. Then sink nodes gradually become able to receive data packets equally, because potentials are updated to adapt to the current packet rate. We can observe that the traffic convergence is also accelerated by optimal feedback. This is because the potential convergence speed is enhanced by the optimal feedback mechanism.

One problem we can find is that our proposal reduces the average number of data packets delivered to each sink node immediately after traffic changes. This is because some sink nodes temporarily have the largest potential values within their communication ranges according to the control inputs, so data packets cannot arrive at sink nodes. Therefore, a partial set of data packets would drop when the controller make large changes to the potentials, which contributes to the faster convergence speed of potentials. However, the data packet drops are immediately reduced and the traffic finally converges faster than the non-control scheme because of the faster potential convergence. Note that in an actual situation data packets may be retransmitted instantly. Here, we evaluate only the case where data packets are never retransmitted, because the main purpose of this study is to reveal the upper limit of convergence speed of self-organizing systems. Moreover, Figures 4.7(c) and 4.7(d) show worst case scenarios for temporary packet drops, because the controller changes sink node potentials (in other words, data packet destinations), so many data packets are dropped when a sink node temporarily gets the highest potential within its communication range due to control inputs. If the controller provides an optimal feedback to several sensor nodes where only some data packets arrive, the number of data packet drops will be smaller. With a lower $r$ sink nodes are more likely to be assigned higher potential values, since the controller can make large changes to the potentials, whereas the recovery speed of data packets delivered to each sink node becomes faster. This indicates that there is a trade-off between the convergence speed of potentials and potential fluctuations.

It is also worth mentioning that the traffic convergence when using PBR-opt-mr with $h = 3$ is as fast as that with $h = 100$ as shown in Figures 4.7(c) and 4.7(d), although potentials oscillate more frequently until the potential convergence is achieved with $h = 3$ compared to the case of $h = 100$ (Figures 4.6(c) and 4.6(d)). This indicates that temporary oscillations of potential values because of approximation errors have little effect on data packet transmissions.

In this subsection, we have shown that an optimal control by the external controller is effective in wireless sensor networks where the capacity and the energy of each node are limited. Moreover, PBR-opt-mr enhanced the convergence speed of potentials with much lower computational cost than PBR-opt. That indicates that a reduced-order model reflects the dominant characteristics of the original model. It is also worth mentioning that even when some amount of approximation error exists, fast convergence of potentials can be achieved. In this evaluation, however, we have assumed that the controller has up-to-date topology information. Therefore, in Subsection 4.5.4 we conduct a simulation experiment where several nodes fail and the controller cannot detect the

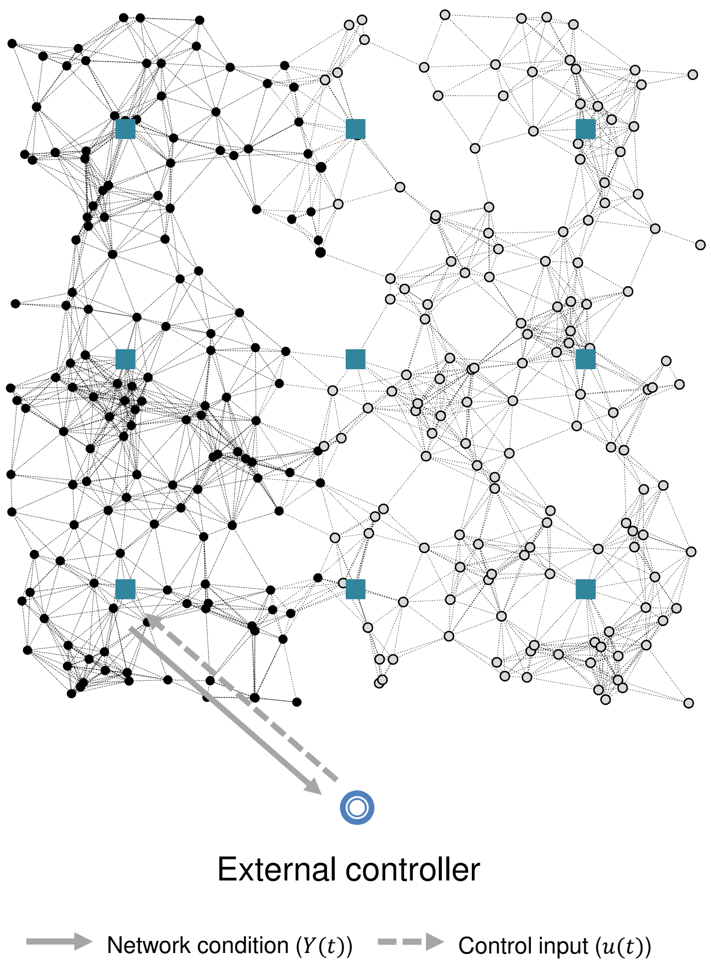


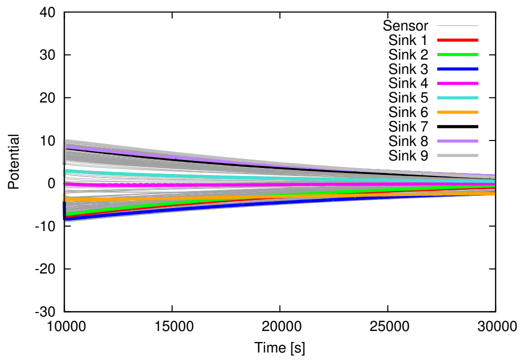
Figure 4.8: Network topology ($N = 309$). Sink nodes are illustrated with squares and sensor nodes with dots in this figure. In this figure, only one sink node seems to be connected to the external controller, but all sink nodes actually are wired with the external controller.

failures (i.e., the controller provides feedback inputs which lead potentials to converge to values that are not adequate to the network).

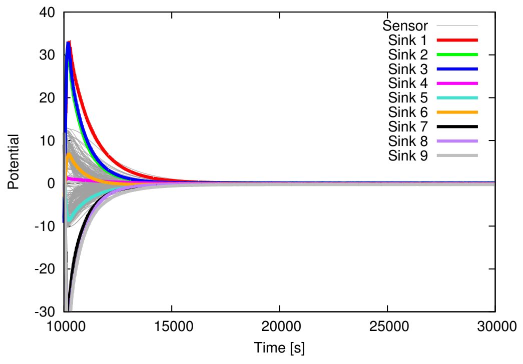
### 4.5.3 Scalability of reduced order controller

Next, we conduct a simulation on a network with 309 nodes including 9 sink nodes in order to reveal the scalability of our proposal in large-scale networks. Specifically, we prove that our proposal can facilitate convergence of a self-organizing system with light-weight cost even if the network size is large. Figure 4.8 shows the network model that is used in this evaluation. In this network model described in Figure 4.8, every sensor node is within 3 hops from a sink node. The number of sensor nodes within 2 hops from a sink node is 271, that is, the external controller observes potentials of 271 sensor nodes in this network. In this evaluation, at $10,000$ s from the beginning of the simulation, data packet generation rates of sensor nodes are changed to examine the convergence speed of our method similarly to the scenario of the evaluation in Subsection 4.5.2. Data packet generation rates are initially set to be $0.0005$ packets/s for sensor nodes in the left half of Figure 4.8 (illustrated with black dots), and $0.0015$ packets/s for sensor nodes in the right half (illustrated with gray dots). After traffic changes at $10,000$ s, data packet generation rates are increased to $0.0015$ packets/s for the left half sensor nodes and decreased to $0.0005$ packets/s for the right half nodes. The average data generation rate of a node of $0.001$ packets/s corresponds to $\bar{f}_n = 1$, so before traffic changes the flow rate vector $\boldsymbol{F} = \begin{bmatrix}\bar{f}_1 & \cdots & \bar{f}_N\end{bmatrix}^T$ is given by Equation (4.19) and after the traffic changes, $\boldsymbol{F}$ is given by Equation (4.20).

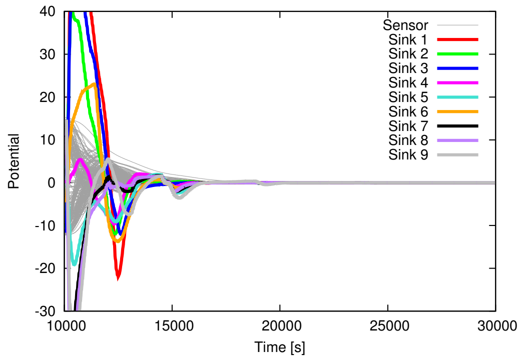
Figure 4.9 shows changes of potential values of the non-control scheme, PBR-opt, and PBR-opt-mr with $h = 20$, and $100$, and Table 4.3 shows the time needed from traffic changes until the potential convergence is achieved when $c = 0.2$. Figure 4.6 shows potential changes only within $10,000$–$30,000$ s but it takes more than $20,000$ s for the potential convergence with PBR-no-ctrl and PBR-opt as shown in Table 4.3. As a result, the potential convergence speed is accelerated by about $7.03$ times, $22.4$ times and $22.6$ times due to PBR-opt, PBR-opt-mr with $h = 20$ and $100$, respectively, compared to the case with PBR-no-ctrl. Table 4.3 indicates that potentials converge faster with the larger number of state variables. This characteristic is valid because the controller is generally able to estimate potentials of nodes more correctly with the larger number of state variables. Table 4.3 also shows that the convergence speed of potentials with PBR-opt-mr is faster than



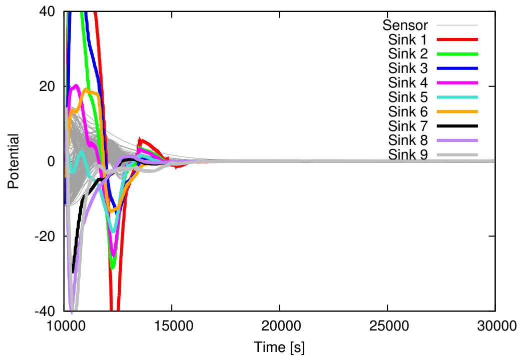
(a) PBR-no-ctrl



(b) PBR-opt



(c) PBR-opt-mr ($h = 20$)

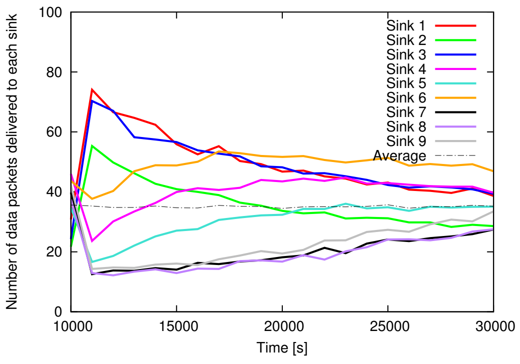


(d) PBR-opt-mr ($h = 100$)

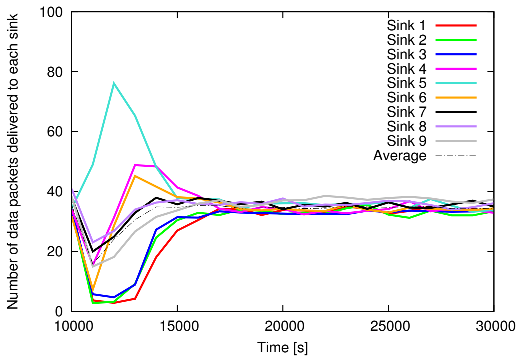
Figure 4.9: Potential convergence in the large-scale network

that with PBR-opt. Feedback inputs provided by the external controller with PBR-opt are optimal in an ideal situation where the external controller and all nodes are synchronized and the external controller can obtain potential information without any communication delays. However, in a real situation, the optimality is not guaranteed because there are estimation errors due to the asynchrony of the system, data packet drops and communication delays. Therefore, the convergence speed with PBR-opt is not always faster than that with PBR-opt-mr. On the contrary, the convergence speed of potentials with PBR-opt seems not to be so different from that with PBR-opt-mr in Figure 4.9. This indicates that PBR-opt works as well as PBR-opt-mr in the real setting of wireless sensor networks.

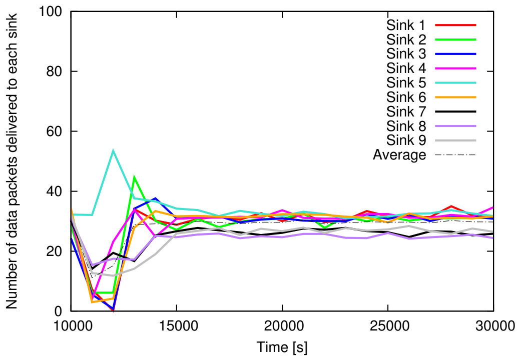
Compared with the evaluation of the smaller scale network in Subsection 4.5.2, degree $h$ of



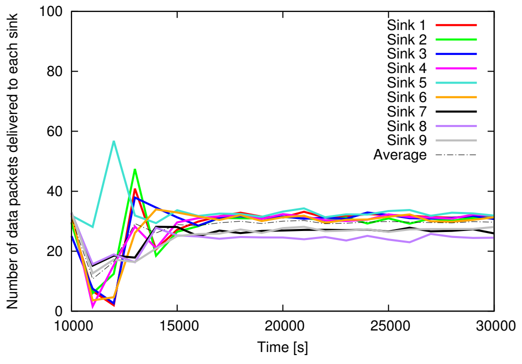
(a) PBR-no-ctrl



(b) PBR-opt



(c) PBR-opt-mr ($h = 20$)



(d) PBR-opt-mr ($h = 100$)

Figure 4.10: Data packets delivered to each sink node in the large-scale network

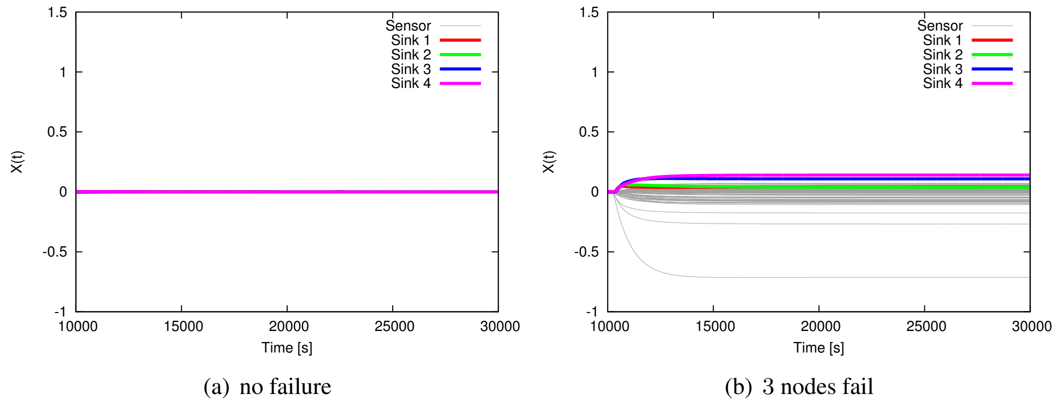
a reduced-order model needs to be larger. This is because a reduced-order model needs a larger number of state variables in order to keep the dominant characteristics of the original model with a larger number of nodes. When $h$ is set to 3 as similar to the evaluation in Subsection 4.5.2, the controller cannot estimate correctly potentials of nodes and the potential convergence speed is quite slow although we do not show the result in Figure 4.9. In this study, we only show the result of evaluations using PBR-opt-mr with $h = 20, 100$, but our proposal can achieve the potential convergence even if $h = 1$, that is, the lower bound of degree $h$ for the network shown in Figure 4.8 is 1, which is smaller than the lower bound of $h$ for the network of 104 nodes shown in Figure 4.5 ($=$ 3). The lower bound of $h$ deeply depends on the network topology, and so we will investigate the

Table 4.3: Potential convergence time in the large-scale network

| Scheme | Convergence time [s] |
|---|---:|
| PBR-no-ctrl | 363,675 |
| PBR-opt | 51,712 |
| PBR-opt-mr ($h = 20$) | 16,234 |
| PBR-opt-mr ($h = 100$) | 16,064 |

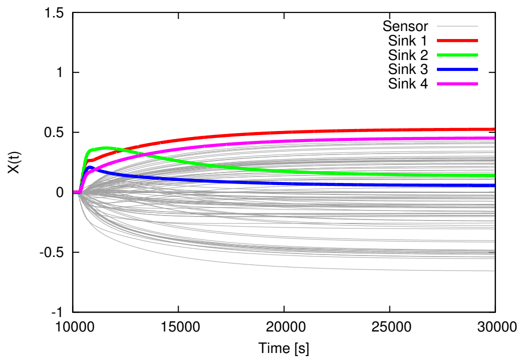
scalability of the reduced order model in the future. In PBR-opt-mr, the computational cost for designing the controller is $O(h^3)$ and that for calculating control inputs is $O(h^2)$. Therefore, the computational cost of PBR-opt-mr with $h = 100$ in Subsection 4.5.2 and that in Subsection 4.5.2 is approximately the same, although the number of nodes is 104 and 309, respectively. Moreover, the number of controllable and observable nodes increases compared with the evaluation of the smaller network. That is because the communication overhead for collecting the information of nodes far from observable nodes is large, and these nodes also take a long time to receive the effect of the optimal feedback that is provided to sink nodes.

Figure 4.10 shows the number of data packets delivered to each sink node every $1,000$ s. As shown in this figure, the traffic convergence speed is accelerated because the potential convergence speed is improved by an optimal feedback. As shown in Figures 4.9(c) and 4.9(d), many sink nodes have potential values higher than their target potential values according to control inputs provided by the controller. Therefore, the decrease rate of data packet arrivals in sink nodes is higher than that in evaluation of 104 nodes. However, this is not always true because control inputs deeply depend on the network topology and flow rates. From Figure 4.10, the average number of data packets delivered to each sink node with PBR-opt-mr is a bit smaller than that with PBR-no-ctrl and PBR-opt. This is because a partial set of data packets and control messages drops near sink nodes because a lot of data packets and control messages are delivered to sink nodes. The number of data packet and control messages dropped will be reduced if the controller collects potential



(a) no failure

(b) 3 nodes fail



(c) 10 nodes fail

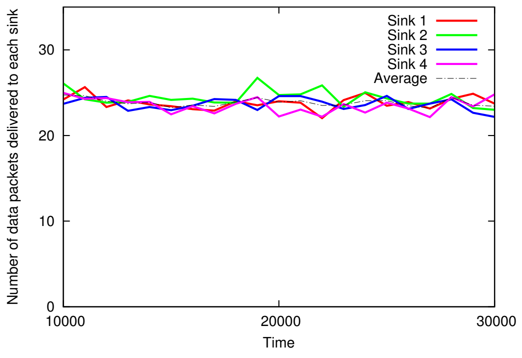
Figure 4.11: Potential convergence in case of node failures (PBR-opt-mr with $h = 3$)

information via several sensor nodes.

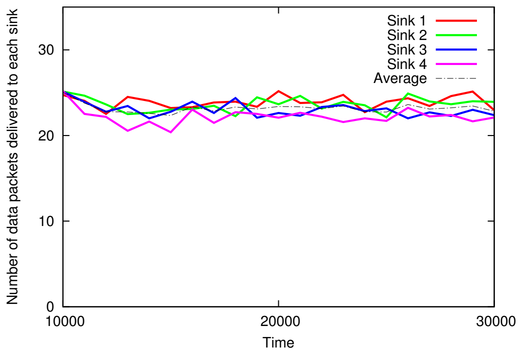
In conclusion, our proposal can accelerate the convergence speed even in large-scale networks with the lower computational cost. However, the lower bound degree of a reduced-order model deeply depends on the network topology.

### 4.5.4 Robustness against Node Failures

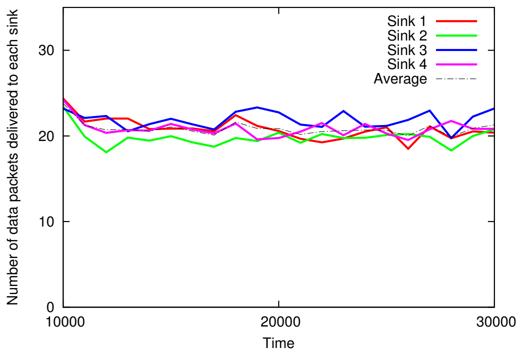
We finally investigate the robustness of PBR-opt-mr. The network model of $104$ nodes is used in this evaluation. In this evaluation, several nodes fail at $10,000$ s from the start of the simulation, and we examine changes of potentials and data packets delivered to each sink node after node failures. At $10,000$ s from the beginning of the simulation, $q$ nodes fail. Failing nodes are randomly



(a) no failure



(b) 3 nodes fail



(c) 10 nodes fail

Figure 4.12: Data packets delivered to each sink node in case of node failures (PBR-opt-mr with $h = 3$)

selected from nodes more than $p$ hops from the sink nodes, so the controller cannot detect them. Instead, each sensor node constantly detect their current neighbors in a self-organizing manner. To avoid the influence of traffic changes, data packet generation rates of all sensor nodes are fixed at 0.001 packets/s, so the $i$-th element $f_i$ of $\boldsymbol{F}$ is given by

$$\bar{f}_i = \begin{cases} 1, & \text{if } i \in \mathcal{N}_{sen} \\ -\frac{|\mathcal{N}_{sen}|}{|\mathcal{N}_{sin}|}, & \text{if } i \in \mathcal{N}_{sin} \end{cases} .$$

The controller monitors only nodes within $p$ $(= 2)$ hops from the sink nodes, as in Subsection 4.5.2. That is, nodes within two hops from the sink nodes send a control message to the sink nodes at

Table 4.4: Data arrival rate

| $q$ | Data arrival rate [%] |
|---|---|
| 0 | 99.12 |
| 3 | 99.03 |
| 10 | 99.18 |

intervals of $T_i$ (50 s), and transfer their potentials to the controller. The degree $h$ of a reduced-order model is set to 3. Other simulation settings are the same as in Subsection 4.5.2.

Figure 4.11 shows the potential changes with $q = 0$, 3, and 10. With $q = 0$, no nodes fail. The figure plots $\boldsymbol{X}(t) = \bar{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(t)$ against time $t$. Note that $\bar{\boldsymbol{\Theta}}$ of each case is the same as $\bar{\boldsymbol{\Theta}}$ in the case where no node fails, because the controller cannot detect the node failures. The reason why the convergence speed is faster than Figure 4.6 in Subsection 4.5.2 is that data generation rates of all sensor nodes are fixed in this evaluation. As the figure shows, potentials converge immediately after the node failures, but potential values at the time when they converge are different from $\bar{\boldsymbol{\Theta}}$. This is because $\bar{\boldsymbol{\Theta}}$ changes due to the topology change, but the controller cannot detect the change. However, the potential convergence is achieved soon after node failures and these failures only slightly affect the data packets delivered to each sink node, as explained below.

Figure 4.12 shows the number of data packets delivered to each sink node every $1,000$ s, and the average number of packets. As the figure shows, each sink node receives data packets, equally even in the case where node failures occur and the controller cannot detect them. With $q = 10$, the number of data packets delivered to sink nodes varies, but these differences are not very large. Compared with the case where node failures do not occur, the average number of data packets delivered to each node is smaller when $q = 3$, 10. This is because the number of data packets generated in the entire network is decreased due to the node failures. The data arrival rate is shown in Table 4.4. The data arrival rate corresponds to the ratio of data packets that arrive at any sink nodes to all generated data packets. This result indicates that the data arrival rate in the case where node failures occur is approximately equal to that in the case where no node fails. In conclusion, optimal feedback by the external controller can enhance convergence while maintaining the high robustness against inaccurate information due to node failures that is an essential characteristic of self-organizing systems.

## 4.6 Summary

In self-organizing systems, each component behaves according to only local information, which leads to slow convergence. We propose and evaluate potential-based routing with optimal feedback using a reduced-order model, where a controller monitors and estimates system states, and provides optimal feedback for the fastest convergence. Simulation results have shown that optimal feedback using a reduced-order model can facilitate the convergence of potentials while reducing costs for collecting system information and estimating system dynamics. Moreover, our proposal has high scalability and robustness against information loss from node failures.

On the contrary, our proposal remains some challenging tasks. First, the optimal feedback mechanism assumes that the controller has the information of the network topology and the flow rates, which reduces the practicality of our proposal. Second, the potential convergence is achieved as a result of the iterative behavior, i.e., the controller's optimal feedback and nodes' potential updates, so that potential cannot converge if environmental changes occur more frequently than the iterative behavior. Third, the controllability and the stability of networks depends on the network topology. Finally, the optimal control improves the convergence speed of potentials but also causes potential fluctuations as shown by simulation results. These fluctuations lead to data packet drops because sink nodes temporarily have the highest potentials among their neighbors. There is a trade-off between the improvement of the potential convergence speed and potential fluctuations.

We are now studying a distributed version where several controllers independently monitor, estimate and control the system dynamics. For this purpose, it would be a promising direction to design hierarchical controllers based on *the clustered model reduction* [99]. With optimal control by a controller, considerable control overhead is needed for collecting network states and estimating network dynamics information when the network is large. By introducing a distributed control mechanism, control overhead can be expected to be reduced because it is not necessary to collect network-wide information.

# Chapter 5

# Hierarchical Optimal Control Method for Controlling Large-scale Self-organizing Networks

## 5.1 Introduction

Self-organization, in which components behave individually and autonomously, is a natural phenomenon in distributed systems [17, 18]. In a self-organizing system, each component follows simple rules that rely on locally available information only. Through direct or indirect interactions among components, a global behavior or pattern emerges at a macroscopic level without a central control entity. In a self-organizing system, up-to-date information regarding the entire system or many other components is unnecessary; this considerably reduces the need to incur the computational cost and communication overhead that collecting global information would entail. This localized behavior affords the ability to handle local failures and small environmental changes via the interaction of local components. Thus, self-organizing systems are expected to automatically recover from failures and adapt to environmental changes. Therefore, various models based on self-organization have been applied to information networking such as routing, synchronization, and task assignment [11, 12]. In future large-scale, complex networks, we can expect that features such as scalability, adaptability, and robustness will be improved to an extent not possible by conventional network control methods [8].

Although self-organization control that does not use global knowledge about the current network state has various benefits, such control has critical limitations that complicate practical use in industrial and business systems [13]. One drawback is that it may take a long time for global patterns to emerge in large-scale systems because they appear as a consequence of interactions between autonomous components. This property also leads to slow adaptation to large environmental changes, which is difficult to solve by solely local interactions in self-organizing systems. Furthermore, self-organizing systems that use local information only sometimes become trapped in local optima. In contrast, in conventional centralized systems, global information can lead to an optimal solution, though the required computational cost may be extremely high.

Such limitations of self-organization noted in real applications led to the idea of guided self-organization, in which the self-organizing system is controlled through some constraints [16, 100]. For example, the authors of [31, 70] used the concept of guided self-organization, in which an external observer/controller guides self-organizing optical networks [31] and sensor network [70] systems through a feedback mechanism that leads them to a desired state. These studies showed that self-organizing systems can be guided to the desired state by introducing an external observer/controller. Along another line, we proposed an optimal feedback mechanism for self-organizing systems [45] to enhance the speed of convergence to an optimal or semi-optimal solution. In this mechanism, an external controller collects information about the network and provides optimal feedback inputs to cause faster convergence. For calculating optimal feedback inputs, the external controller estimates the states of nodes in the entire network by using a mathematical model that describes the network dynamics. The simulation results in [45] showed that optimal control by the external controller can facilitate the convergence of self-organizing networks.

We showed that the introduction of an external controller into self-organizing networks can accelerate their convergence; however, this mechanism still has problems with scalability. The mechanism requires topology information about the whole network, and the computational cost for designing the estimation model is roughly proportional to the cube of the number of nodes, which makes it increasingly difficult for the external controller to collect topology information and to obtain the estimation model of the whole network as the network size becomes larger. These problems become critical when topological changes occur, because the estimation model needs to be redesigned to include the latest topology information so that the external controller can guide the network to converge to a targeted state.

In this study, we propose a hierarchical optimal feedback mechanism to control a self-organizing network while maintaining the high scalability, adaptability, and robustness that are originally inherent in self-organizing systems. The basic idea of the mechanism is shown in [50], in which a network has been partitioned into several sub-networks that are controlled in a hierarchical manner by two types of controllers, i.e., a *central controller* and *sub-controllers*. Each sub-controller monitors a different sub-network, that is, only a part of the entire network, and provides optimal feedback inputs to the sub-network so that fast convergence can be achieved. Specifically, a sub-controller collects information regarding the corresponding sub-network, such as node states and network topology, via a partial set of nodes monitored by the sub-controller. Then, the sub-controller estimates the dynamics of the sub-network by using a mathematical model that describes the sub-network dynamics to calculate optimal feedback control inputs that facilitate the convergence speed of self-organizing systems. This is based on robust control theory [47]. In contrast, the role of a central controller is to guide sub-networks to achieve the identical global optimality. If each sub-controller determines the optimal inputs selfishly, the interactions among the sub-networks may cause network instability. This is because each individual sub-controller can monitor the node states of only its own sub-network, even though sub-networks actually interact with each other. A

Table 5.1: Computational costs in each scheme. $N$ is the total number of nodes; $N_{sub}$ is the number of nodes in a sub-network; $h_{ex}$ is the number of state variables in the estimation model used by the external controller in PBR-opt-mr; $h$ and $h_{sub}$ are the number of state variables in the estimation model used by the central controller and each sub-controllers, respectively, in PBR-h-opt-mr.

| Scheme | Controller type | $N_{dim}$ | Computational cost | |
|---|---|---|---|---|
| | | | Controller design | Calculation of feedback inputs |
| PBR-no-ctrl | - | - | 0 | 0 |
| PBR-opt-mr | External | $h_{ex}(< 2N)$ | $O(N_{dim}^3)$ | $O(N_{dim}^2)$ |
| PBR-h-opt | Sub | $2N_{sub}(< 2N)$ | $O(N_{dim}^3)$ | $O(N_{dim}^2)$ |
| | Central | $2N$ | $O(N_{dim}^2)$ | $O(N_{dim}^2)$ |
| PBR-h-opt-mr | Sub | $h_{sub}(< 2N_{sub})$ | $O(N_{dim}^3)$ | $O(N_{dim}^2)$ |
| | Central | $h(< 2N)$ | $O(N_{dim}^2)$ | $O(N_{dim}^2)$ |

central controller obtains information about the network from sub-controllers, estimates the degrees of interactions among sub-networks, and then returns feedback inputs to the sub-controllers. To reduce the computational cost of the central controller and sub-controllers, we reduce the number of state variables of the estimation models for both the central controller and the sub-controllers by using a model reduction technique. In contrast, [50] used the original estimation model, which has a larger number of state variables than the reduced model. Model reduction refers to reducing the number of state variables in dynamical systems while retaining suitable input/output characteristics.

This study realizes a low-cost self-organizing network system that shows high performance (such as high convergence, scalability, and adaptability) of optimal control and reveals the effectiveness of a hierarchical optimal feedback mechanism through computer simulation. The number of state variables of the estimation model is proportional to the size of the actual network model, namely, the number of nodes. Therefore, the computational cost incurred for the dynamics estimation and the control input calculation of an external controller or a sub-controller increase rapidly as the number of nodes becomes large. The external controller or sub-controller incurs a computational cost on the order of $O({N_{dim}}^2)$ for the dynamics estimation at each time instance and of

$O({N_{dim}}^3)$ for the redesign of the estimation model when the network topology changes, where $N_{dim}$ is the number of state variables of the estimation model. Therefore, with hierarchical control, the sum of the computational cost and communication cost of each sub-controller is much smaller than the computational cost of the external controller proposed in [45] (Table 5.1). Moreover, we reduce the number of state variables of the estimation model ($N_{dim}$) via a model reduction technique [49] to reduce the computational cost. Note that there is a trade-off between the accuracy of the estimation model and the computational cost (i.e., between the accuracy of the estimation model and the number of state variables). Intuitively, the smaller size of a reduced model leads to the loss of the original model while decreasing the computational cost. This trade-off relation requires us to further examine the number $N_{dim}$ of state variables of the model. Furthermore, local environmental changes can be handled by the corresponding sub-network because each sub-controller monitors/controls the corresponding sub-network in an individual and automatic manner. This contributes to high scalability and adaptability of the system. In contrast, with the non-hierarchical method, the estimation model needs to be redesigned entirely.

The effectiveness of our proposal is evaluated through computer simulation studies in which we consider potential-based routing—a self-organizing routing mechanism for wireless sensor networks—with optimal feedback and evaluate the convergence speed after environmental changes. Our proposal is suitable for wireless sensor networks because the simple behavior of each component in self-organizing systems needs neither much capability nor much energy, whereas optimal control by the external controller improves the performance of these systems.

The optimality of our feedback mechanism is analytically guaranteed in synchronous systems but not in asynchronous systems. We first assume a wireless sensor network where nodes behave asynchronously and the sub-controller can observe the network state via nodes that have a connection to the sub-controller, although one with a communication delay. To evaluate the scalability of our proposal, we next conduct a simulation with a larger network. Finally, we evaluate our method in the case where a new sub-network is added to the network and show that local environmental changes can be handled by the corresponding sub-controller with our method. Through these evaluations, we show that hierarchical optimal feedback using a reduced-order model can enhance the convergence speed of self-organizing systems at fairly low cost, even if the network is large. Note that potential-based routing for wireless sensor networks is an example of an application of our proposal, and our proposal can be adapted to other types of mechanisms, networks, and situations.

The remainder of this chapter is organized as follows. First, we propose and explain potential-based routing with optimal feedback in Section 5.2. Note that we already explained potential-based routing in Chapter 4. We then show, through simulation, the proposed method's ability to facilitate fast adaptation to environmental changes and discuss our proposal in Section 5.3. Finally, in Section 5.4, we present our conclusions and suggest areas for future work.

## 5.2 Potential-based Routing with Hierarchical Optimal Feedback

In this section, we describe a model of the network dynamics and explain our hierarchical optimal control scheme.

### 5.2.1 Overview

The proposed hierarchical optimal feedback mechanism is shown in Figure 5.1. A network is partitioned into several sub-networks. Each sub-network is connected to a sub-controller via a part of the nodes that belong to the sub-network, whereas a central controller is connected to all sub-controllers.

We describe the network as $G = \{\mathcal{V}, \mathcal{E}\}$. The set $\mathcal{V}$ corresponds to a set of nodes in network $G$. The set $\mathcal{E}$ is given by $\{e_{n,m}; \forall n, m \in \mathcal{V}\}$, where $e_{n,m}$ corresponds to a link between node $n$ and $m$. Network $G$ is partitioned into sub-networks $\{G^i = \{\mathcal{V}^i, \mathcal{E}^i\}; \forall i \in \mathcal{S}\}$, where $\mathcal{S}$ corresponds to a set

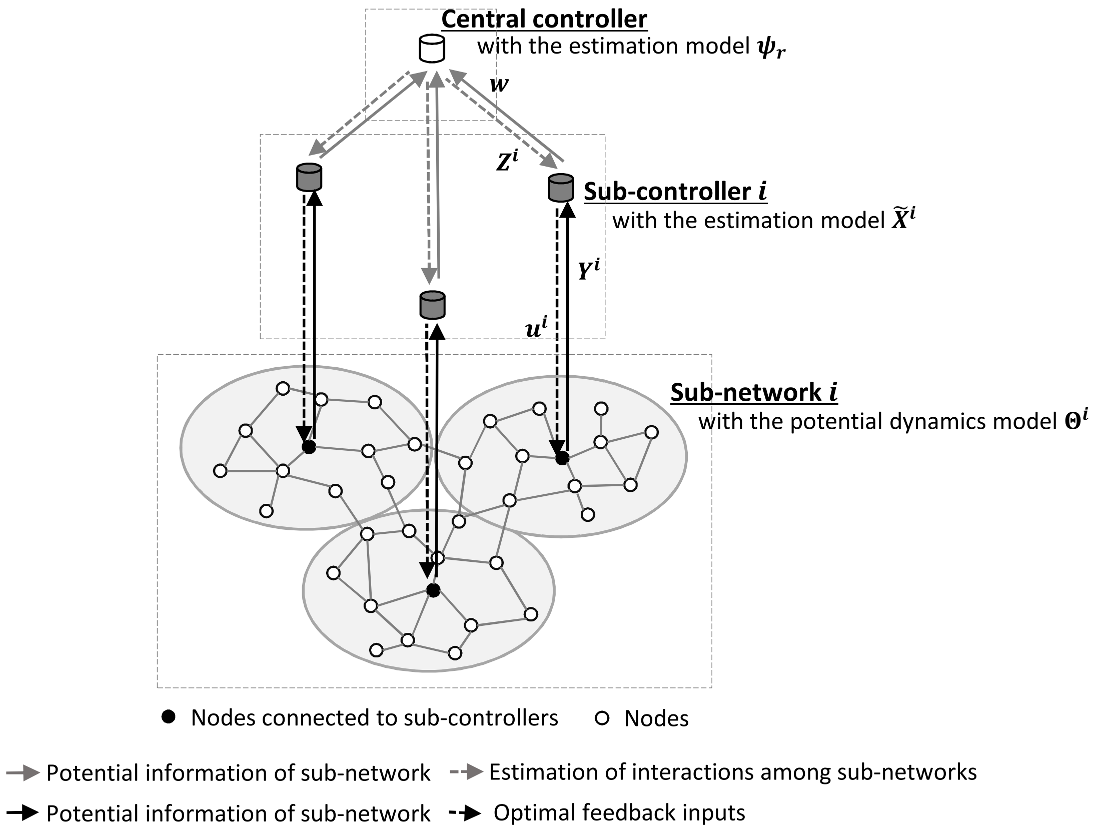

Central controller
with the estimation model $\psi_r$
$w$
$Z^i$
Sub-controller $i$
with the estimation model $\tilde{X}^i$
$Y^i$
$u^i$
Sub-network $i$
with the potential dynamics model $\Theta^i$
● Nodes connected to sub-controllers
○ Nodes
Potential information of sub-network
Estimation of interactions among sub-networks
Potential information of sub-network
Optimal feedback inputs

Figure 5.1: Potential-based routing with a hierarchical optimal control scheme, where sub-controllers monitor/control their own area, and the central controller collects potential values from sub-controllers and provides feedback inputs to them periodically.

of sub-networks. $\mathcal{V}^i$ is the set of nodes belonging to sub-network $i$ and $\mathcal{E}^i = \{e_{n,m}; \forall n, m \in \mathcal{V}^i\}$. $G^i$ is a connected graph for each $i$ ($i \in \mathcal{S}$) and $\mathcal{V}^i \cap \mathcal{V}^j = \varnothing$ for all $i, j$ ($i, j \in \mathcal{S}, i \neq j$). If node $n$ ($\in \mathcal{V}^i$) and $m$ ($\in \mathcal{V}^j$) are directly connected by a link $e_{n,m}$, then sub-networks $i$ and $j$ interact with each other via the link.

A *sub-controller* monitors information about its corresponding sub-network, and in particular the potential values of a partial set of nodes, which we call *observable nodes*. The sub-controller then returns suitable control inputs to another partial set of nodes, which we call *controllable nodes*, for accelerating the convergence of the potential distribution of the sub-network toward the target potential distribution, which is described in Section 5.2.2. The control input from the sub-controller affects the potential values of the controllable nodes, whose effect propagates to the entire network through the local interaction (Equation (5.1) of Section 5.2.2). The information about the sub-network topology is needed for designing the sub-controller. Furthermore, we need the flow rates of nodes in the sub-network for calculating target potential values. Although this information cannot be estimated with our proposal, we assume that changing the frequencies of the network topology and flow rates occurs much less frequently than changes to potentials and that changes can be reported to the sub-controllers only when they occur. Such data collection is feasible with the proper choice of control interval.

The role of the *central controller* is to estimate interactions among sub-networks and to provide feedback inputs to sub-controllers so that the whole network reaches the targeted state. For this purpose, the central controller obtains the network information from sub-controllers and then returns suitable feedback inputs to sub-controllers. Without the central controller, the ignored interactions among sub-networks lead to an undesirable target potential distribution and can even cause network instability. Because the estimation by the central controller incurs a high computational cost, we introduce the reduced-order model to reduce this computational cost for estimating the potential values of non-observable nodes, as explained in Section 5.2.2.

We assume that the sub-controllers and sink nodes are supplied power so that these sink nodes can have a reliable direct connection with sensor nodes and sub-controllers at all times. Therefore, in our proposal, sub-controllers monitor sub-network information and provide control inputs via connected sink nodes. We consider controllable nodes as sink nodes and observable nodes as nodes within $p$ hops from sink nodes. For estimating the interactions among sub-networks, the central controller obtains the potential information about all observable nodes from the sub-controllers.

### 5.2.2 Dynamics of Sub-networks

Let the dynamics of the potentials be given by a deterministic discrete-time model. In our proposal, the update rule of each potential is the same as that in [101], except for controllable nodes that additionally receive feedback inputs sent by each sub-controller. Node $n$ updates its potential at time $t$ by Equation (5.1).

$$\begin{aligned}\theta_n(t+1) &= (\alpha+1)\theta_n(t) - \alpha\theta_n(t-1) \\ &+ \beta\sigma_n\left(\sum_{k\in\mathcal{N}(n)}\{\theta_k(t)-\theta_n(t)\} + f_n\right) + \eta_n(t).\end{aligned} \tag{5.1}$$

Here, $\eta_n$ represents the feedback input sent by the controller. If node $n$ is not controllable, then $\eta_n(t) = 0$. Sub-controllers collect and estimate the node potentials of each sub-network and provide feedback inputs $\boldsymbol{u}(t) = [\eta_1(t)\ \eta_2(t)\ \cdots\ \eta_{N_{ctrl}}(t)]^T$ (where $N_{ctrl}$ denotes the number of controllable nodes). Node potentials can converge faster than in the non-control scheme described by Equation (**??**), where each node updates its potential based on only local interactions with neighbors. We set $\sigma_n$ to a constant value $\sigma$ $(0 < \sigma < 1)$ for all $n$ $(\in \{1, 2, \cdots, N\})$ because the original value of $\sigma_n$ $(= 1/|\mathcal{N}(n)|)$ proposed in [101] leads to oscillation of potentials in some situations.

Next, we describe the potential dynamics of sub-networks. We define $\mathcal{S}$ as a set of sub-networks, where a sub-network $i$ $(\in \mathcal{S})$ includes $N^i$ nodes. The potential values of nodes in sub-network $i$ are described as a vector $\boldsymbol{\Theta}^i(t) = \left[\boldsymbol{\theta}_1^i(t)^T\ \boldsymbol{\theta}_2^i(t)^T\ \cdots\ \boldsymbol{\theta}_{N^i}^i(t)^T\right]^T$ using $\boldsymbol{\theta}_n(t) = [\theta_n(t)\ \theta_n(t+1)]^T$. The potential dynamics of sub-network $i$ is given by Equation (5.2) using flow matrix $\boldsymbol{F}^i$ and control inputs $\boldsymbol{u}^i$.

$$\begin{aligned}\boldsymbol{\Theta}^i(t+1) &= \boldsymbol{A}^i\boldsymbol{\Theta}^i(t) + \sum_{j\in\mathcal{S}-\{i\}}\boldsymbol{A}^{i,j}\boldsymbol{\Theta}^j(t) \\ &+ \left(\beta\sigma\boldsymbol{F}^i + \boldsymbol{u}^i(t)\right)\otimes\boldsymbol{M}_0.\end{aligned} \tag{5.2}$$

Here, $\boldsymbol{A}^i$ and $\boldsymbol{A}^{i,j}$ are matrices that describe interactions, respectively, within sub-network $i$ and between sub-network $i$ and $j$[1].

Under these dynamics, the potentials $\boldsymbol{\Theta}^i$ converge to $\bar{\boldsymbol{\Theta}}^i$ and we consider them as the target potentials, as given by the solution of

$$(\boldsymbol{A}^i - \boldsymbol{I}_{2N^i \times 2N^i})\bar{\boldsymbol{\Theta}}^i + \sum_{j \in \mathcal{S}-\{i\}} \boldsymbol{A}^{i,j}\bar{\boldsymbol{\Theta}}^j = -\beta\sigma \boldsymbol{F}^i \otimes \boldsymbol{M}_0. \tag{5.3}$$

In the evaluation of Section 5.3, the target potential distribution is denoted by

$$\bar{\boldsymbol{\Theta}} = \begin{bmatrix} \bar{\boldsymbol{\Theta}}^{1^T} & \cdots & \bar{\boldsymbol{\Theta}}^{|\mathcal{S}|^T} \end{bmatrix}^T.$$

Each sub-controller calculates the target potential distribution of the corresponding sub-network in advance, using Equation (5.3). Note that information about the target potentials is not needed to design the estimation model of the sub-controller.

### 5.2.3 Optimal Control of Sub-networks by Sub-controllers

A sub-controller collects potential information from observable nodes in the corresponding sub-network. With the collected information, the sub-controller estimates the dynamics of the sub-network to provide optimal feedback inputs to the sub-network.

[1] $\boldsymbol{A}^i$ and $\boldsymbol{A}^{i,j}$ are given by

$$\begin{aligned} \boldsymbol{A}^i &= \boldsymbol{I}_{N^i \times N^i} \otimes \boldsymbol{A}_1 + (\boldsymbol{L}^i_{intra} - \boldsymbol{G}^i) \otimes \boldsymbol{A}_0, \\ \boldsymbol{A}^{i,j} &= \boldsymbol{L}^{i,j}_{inter} \otimes \boldsymbol{A}_0, \end{aligned}$$

where the $(N^i \times N^i)$-matrix $\boldsymbol{L}^i_{intra}$ corresponds to the adjacency matrix of sub-network $i$. $\boldsymbol{L}^{i,j}_{inter}$ is an $(N^i \times N^j)$-matrix that describes links connecting between sub-networks $i$ and $j$. The element $l^{i,j}_{inter}(n,m) \in \{0,1\}$ of $\boldsymbol{L}^{i,j}_{inter}$ is 1 if and only if there is a link between node $n$ of sub-network $i$ and node $m$ of sub-network $j$. The $(N^i \times N^i)$-matrix $\boldsymbol{G}^i$ describes the degrees of nodes that belong to sub-network $i$. The element $g^i(n,n)$ of $\boldsymbol{G}^i$ corresponds to the number of links, one of whose edges is node $n$, and $g^i(n,m) = 0$ if $n \neq m$. $\boldsymbol{I}_{N \times N}$ is the $N \times N$ identity matrix. $\boldsymbol{A}_1$, $\boldsymbol{A}_0$, and $\boldsymbol{M}_0$ are given by

$$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 1 \\ -\alpha & \alpha+1 \end{bmatrix}, \boldsymbol{A}_0 = \begin{bmatrix} 0 & 0 \\ 0 & \beta\sigma \end{bmatrix}, \boldsymbol{M}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

**Design of Sub-controllers**

Next, we explain the design of the sub-controllers. For all $i$ ($\in \mathcal{S}$), sub-network $i$ is connected to sub-controller $i$, and sub-controller $i$ monitors potentials $\boldsymbol{Y}^i$ of observable nodes in sub-network $i$ via nodes that have a connection to the sub-controller. $\boldsymbol{Y}^i(t)$ is a $2N^i_{obs}$-dimension vector that is given by $\boldsymbol{Y}^i(t) = \boldsymbol{C}^i\boldsymbol{X}^i(t)$[2] using $\boldsymbol{X}^i(t) = \bar{\boldsymbol{\Theta}} - \boldsymbol{\Theta}^i(t)$, where $N^i_{obs}$ is the number of observable nodes in sub-network $i$. Note that $\boldsymbol{Y}^i$ shows the gap between the current and target potential values at the observable nodes. Then, sub-controller $i$ estimates $\boldsymbol{X}^i(t)$ from observable information $\boldsymbol{Y}^i$. In this, $\tilde{\boldsymbol{X}}^i(t)$, which is the estimation of $\boldsymbol{X}^i(t)$ by sub-controller $i$, is a $2N^i$-dimension vector and is given by Equations (5.4) and (5.5)[3].

$$\tilde{\boldsymbol{X}}^i(t+1) = \boldsymbol{A}^i\tilde{\boldsymbol{X}}^i(t) + \boldsymbol{B}^i\boldsymbol{u}^i(t) + \boldsymbol{Q}^i\left(\boldsymbol{Y}^i(t) - \tilde{\boldsymbol{Y}}^i(t)\right) + \boldsymbol{Z}^i(t), \tag{5.4}$$

$$\tilde{\boldsymbol{Y}}^i(t) = \boldsymbol{C}^i\tilde{\boldsymbol{X}}^i(t), \tag{5.5}$$

where $\boldsymbol{Z}^i$ indicates interactions among sub-networks. Sub-controller $i$ receives $\boldsymbol{Z}^i$ from the central controller and calculates a feedback input that accelerates the convergence speed of potentials, which we explain in Section 5.2.4. If $\tilde{\boldsymbol{X}}^i(t)$ is close to 0, then the potentials are estimated to be

[2]$(2N^i_{obs} \times 2N^i)$-matrix $\boldsymbol{C}^i$ determines the observable nodes. The element $c^{2i}(n,m)$, $c^{2i+1}(n,m) \in \{0,1\}$ of $\boldsymbol{C}^i$ is 1 if and only if sub-controller $i$ monitors the potential value of node $m$ as the $n$th element of $\boldsymbol{Y}^i$. If more nodes are monitored by sub-controller $i$, i.e., when $N^i_{obs}$ is larger, it can estimate potential values of nodes in sub-network $i$ more precisely, although the communication overhead for collecting information about the observable nodes becomes much larger.

[3]We need to select $\boldsymbol{Q}^i$ such that it satisfies "$\boldsymbol{A}^i - \boldsymbol{Q}^i\boldsymbol{C}^i$ is stable," which allows the potentials to converge. Equation (5.4) is rewritten as

$$\tilde{\boldsymbol{X}}^i(t+1) = (\boldsymbol{A}^i - \boldsymbol{Q}^i\boldsymbol{C}^i)\tilde{\boldsymbol{X}}^i(t) + \boldsymbol{B}^i\boldsymbol{u}^i(t) + \boldsymbol{Q}^i\boldsymbol{Y}^i(t) + \boldsymbol{Z}^i(t).$$

If $\boldsymbol{A}^i - \boldsymbol{Q}^i\boldsymbol{C}^i$ is not stable, then $\tilde{\boldsymbol{X}}^i$ cannot converge, and the elements of $\tilde{\boldsymbol{X}}^i$ will diverge to infinite values for all inputs $\boldsymbol{u}^i$, $\boldsymbol{Y}^i$, $\boldsymbol{Z}^i$. This means that sub-controller $i$ cannot estimate potentials properly, which leads the sub-controller to provide improper feedback inputs $\boldsymbol{u}^i$ to the network. The matrix $\boldsymbol{B}^i$ is given by

$$\boldsymbol{B}^i = \boldsymbol{S}^i \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

The $(N^i \times N^i_{ctrl})$-matrix $\boldsymbol{S}^i$ specifies the controllable node of sub-network $i$, that is, the element $s^i(n,m) \in \{0,1\}$ of $\boldsymbol{S}^i$ is 1 if and only if node $n$ receives the $m$th element of $\boldsymbol{u}^i(t)$ as control input $\eta_n(t)$. With a larger number of controllable nodes, i.e., with a denser $\boldsymbol{S}^i$, the influence of the optimal control is larger so that the convergence speed should be faster, although significant properties originating from self-organization, such as scalability and adaptability, could be lost with too many controllable nodes.

close to their target values. Next, the feedback input $\boldsymbol{u}^i(t)$ is calculated according to Equation (5.6)[4].

$$\boldsymbol{u}^i(t) = -\boldsymbol{V}^i \tilde{\boldsymbol{X}}^i(t). \tag{5.6}$$

With the estimation model described by Equations (5.4), (5.5), and (5.6), which has $2N^i$ state variables, the optimal feedback $\boldsymbol{u}^i(t)$ is calculated with computational cost $O({N^i}^2)$. In other words, the computational cost for calculating the optimal feedback increases rapidly as the size of the corresponding sub-network becomes large. Therefore, sub-controllers use reduced-order models that have fewer state variables for estimating the interactions among sub-networks with smaller computational cost. Many researchers have studied various methods to approximate a model with a reduced-order model to control large, complex systems [49]. In our proposal, we approximate the original model based on a '*balanced realization*' that is highly compatible with the model expressed in the state space representation [47, 49]. In model reduction, a reduced-order model needs to have approximately the same response characteristics as the original model for the accurate estimation of potentials. Here, response characteristics mean the effectiveness of inputs to the system.

### Model Reduction for Sub-controllers

In the reduced-order model for sub-controller $i$, the estimation model is expressed as an $(h^i \times 1)$-vector $\tilde{\boldsymbol{X}}^i_r$ whose elements are linear transformations of the original model $\tilde{\boldsymbol{X}}^i$. $h^i$ is given by the network manager.

The reduced-order model is given by

[4] $\boldsymbol{V}^i$ is the optimal gain matrix that minimizes the quadratic cost function $H^i(\boldsymbol{u}^i) = \sum_{n=1}^{\infty}\left(||\tilde{\boldsymbol{X}}^i(n)|| + r||\boldsymbol{u}^i(n)||\right)$ for the system. $r$ is a parameter that regulates the trade-off between the convergence speed and the stability of potentials. With lower values of $r$, the convergence speed of potentials is faster, but potentials change by larger amounts at one time because the sub-controller is allowed to provide larger values of $\boldsymbol{u}^i(t)$.

$$\tilde{\boldsymbol{X}}_r^i(t+1) = \boldsymbol{A}_r^i \tilde{\boldsymbol{X}}_r^i(t) + \boldsymbol{B}_r^i \boldsymbol{u}_r^i(t) + \boldsymbol{Q}_r^i \left( \boldsymbol{Y}^i(t) - \tilde{\boldsymbol{Y}}^i(t) \right) + \boldsymbol{Z}^i(t), \tag{5.7}$$

$$\tilde{\boldsymbol{Y}}^i(t) = \boldsymbol{C}_r^i \tilde{\boldsymbol{X}}_r^i(t), \tag{5.8}$$

$$\boldsymbol{u}_r^i(t) = -\boldsymbol{V}_r^i \tilde{\boldsymbol{X}}_r^i(t). \tag{5.9}$$

Here, $2N_{ctrl}^i$-dimension vector $\boldsymbol{u}_r^i(t)$ corresponds to feedback inputs provided to a sub-network by the corresponding sub-controller with the reduced-order model. Note that control inputs $\boldsymbol{u}_r^i(t)$ given by Equation (5.9) need to be approximately the same as $\boldsymbol{u}^i(t)$, which is calculated using the original model described by Equations (5.4)–(5.6). We choose matrices $\boldsymbol{A}_r^i$, $\boldsymbol{B}_r^i$, $\boldsymbol{Q}_r^i$, $\boldsymbol{C}_r^i$, and $\boldsymbol{V}_r^i$ of compatible dimensions such that $\boldsymbol{u}_r^i(t) \approx \boldsymbol{u}^i(t)$ for all input $\boldsymbol{Y}^i(t)$, $\boldsymbol{Z}^i(t)$ by using *balred* function[5].

A reduced-order model can be described with a constant number $h^i$ of state variables, and the computational cost can be reduced using it. With a reduced-order model, feedback inputs $\boldsymbol{u}_r^i(t)$ are calculated with computational cost $O({h^i}^2)$. In general, a model that has more state variables allows the controller to perform estimations more accurately; however, the computational cost is larger. In contrast, the computational cost is smaller but the estimation error can be larger in a model that has fewer state variables. Therefore, $h^i$ needs to be properly determined in accordance with the requirements or system properties.

### 5.2.4 Estimation of Interactions among Sub-networks

#### Design of Central Controller

Here, we explain how the central controller estimates interactions among sub-networks. The central controller obtains potential information $\boldsymbol{w}(t)$ from sub-controllers for estimating the interactions

[5]*balred* is a function for reducing the model order [98]. Given design parameter matrices $\boldsymbol{A}$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\boldsymbol{C}$, and $\boldsymbol{D}_1$ for the original model and the degree $h$ of a reduced-order model, *balred* computes an $h$th-order approximation of the original model.

among sub-networks. $\boldsymbol{w}(t)$ is a $2N_{obs}$-dimension vector, with $N_{obs}$ being the number of nodes whose potentials are collected by the central controller, and it is given as $\boldsymbol{w}(t) = \boldsymbol{W}\boldsymbol{X}(t)$[6]. In this study, we set $\boldsymbol{w}(t)$ to $\left[\boldsymbol{Y}^1(t)^T\ \boldsymbol{Y}^2(t)^T\ \cdots\ \boldsymbol{Y}^{|\mathcal{S}|}(t)^T\right]^T$. Note that observable nodes for the sub-controllers and nodes whose potential information is collected by the central controller are not necessarily the same, i.e., $\boldsymbol{w}(t)$ does not need to be equal to $\left[\boldsymbol{Y}^1(t)^T\ \boldsymbol{Y}^2(t)^T\ \cdots\ \boldsymbol{Y}^{|\mathcal{S}|}(t)^T\right]^T$.

Then, the central controller estimates degrees $\boldsymbol{Z}(t)$ of interactions among sub-networks using Equations (5.11) and (5.12)[7] [99].

$$\boldsymbol{\psi}(t+1) = \boldsymbol{J}\boldsymbol{\psi}(t) + \boldsymbol{K}\boldsymbol{u}(t) + \boldsymbol{O}\boldsymbol{w}(t), \tag{5.11}$$

$$\boldsymbol{Z}(t) = \boldsymbol{T}\boldsymbol{\psi}(t), \tag{5.12}$$

where $2N$-dimension vector $\boldsymbol{\psi}$ describes an estimation model for the central controller, using which the central controller estimates the interactions among sub-networks, and $\boldsymbol{Z}(t)$, which corresponds to $\left[\boldsymbol{Z}^1(t)^T\ \boldsymbol{Z}^2(t)^T\ \cdots\ \boldsymbol{Z}^{|\mathcal{S}|}(t)^T\right]^T$, is provided to each sub-controller by the central controller.

With the estimation model described by Equations (5.11) and (5.12), which has $2N$ state variables, degrees $\boldsymbol{Z}(t)$ are calculated with $O(N^2)$. In other words, the computational cost is extremely

[6]$\boldsymbol{X}(t)$ describes the potential dynamics of all nodes and $\boldsymbol{X}(t)$ is defined as $\left[\boldsymbol{X}^1(t)^T\ \boldsymbol{X}^2(t)^T\ \cdots\ \boldsymbol{X}^{|\mathcal{S}|}(t)^T\right]^T$. The dynamics of the entire network is given by

$$\boldsymbol{X}(t+1) = \boldsymbol{A}\boldsymbol{X}(t) + \boldsymbol{B}\boldsymbol{u}(t), \tag{5.10}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}^1 & \boldsymbol{A}^{1,2} & \cdots & \boldsymbol{A}^{1,|\mathcal{S}|} \\ \boldsymbol{A}^{2,1} & \boldsymbol{A}^2 & \cdots & \boldsymbol{A}^{2,|\mathcal{S}|} \\ \vdots & & & \vdots \\ \boldsymbol{A}^{|\mathcal{S}|,1} & \boldsymbol{A}^{|\mathcal{S}|,2} & \cdots & \boldsymbol{A}^{|\mathcal{S}|} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}^1 \\ \boldsymbol{B}^2 \\ \vdots \\ \boldsymbol{B}^{|\mathcal{S}|} \end{bmatrix}.$$

$\boldsymbol{W}$ is a $(2N_{obs} \times 2N)$-matrix that determines nodes whose potential are collected by the central controller in the same manner as $\boldsymbol{C}^i$.

[7]$\boldsymbol{J}$, $\boldsymbol{K}$, $\boldsymbol{O}$, and $\boldsymbol{T}$ are given by

$$\boldsymbol{J} = \boldsymbol{A} - \boldsymbol{Q}\boldsymbol{W}, \boldsymbol{K} = \boldsymbol{B}, \boldsymbol{O} = \boldsymbol{Q},$$

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}^{1,2} & \cdots & \boldsymbol{A}^{1,|\mathcal{S}|} \\ \boldsymbol{A}^{2,1} & \boldsymbol{0} & \cdots & \boldsymbol{A}^{2,|\mathcal{S}|} \\ \vdots & & & \vdots \\ \boldsymbol{A}^{|\mathcal{S}|,1} & \boldsymbol{A}^{|\mathcal{S}|,2} & \cdots & \boldsymbol{0} \end{bmatrix}.$$

$\boldsymbol{Q}$ is a $(2N \times 2N_{obs})$-matrix that satisfies "$\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{W}$ is stable." $\boldsymbol{0}$ is a zero matrix.

large in large-scale networks. In our proposal, therefore, the central controller uses a reduced-order model that has fewer state variables, as described in Section 5.2.3, to estimate interactions among sub-networks with smaller computational cost.

**Model Reduction for Central Controller**

In the reduced-order model for the central controller, the estimation model is expressed as an $(h \times 1)$-vector $\boldsymbol{\psi}_r(t)$ whose elements are linear transformations of the original model $\boldsymbol{\psi}(t)$, and the reduced-order model is given by Equations (5.13) and (5.14).

$$\boldsymbol{\psi}_r(t+1) = \boldsymbol{J}_r \boldsymbol{\psi}_r(t) + \boldsymbol{K}_r \boldsymbol{u}(t) + \boldsymbol{O}_r \boldsymbol{w}(t), \tag{5.13}$$

$$\boldsymbol{Z}_r(t) = \boldsymbol{T}_r \boldsymbol{\psi}_r(t). \tag{5.14}$$

Here, the $2N$-dimension vector $\boldsymbol{Z}_r(t)$ corresponds to feedback inputs provided to sub-controllers by the central controller with the reduced-order model. We need to choose matrices $\boldsymbol{J}_r$, $\boldsymbol{K}_r$, $\boldsymbol{O}_r$, and $\boldsymbol{T}_r$ of compatible dimensions such that $\boldsymbol{Z}_r(t) \approx \boldsymbol{Z}(t)$ for all inputs $\boldsymbol{u}(t)$, $\boldsymbol{w}(t)$.

A reduced-order model can be described with a constant number $h$ of state variables, and the computational cost can be thereby reduced. With a reduced-order model, feedback inputs $\boldsymbol{Z}_r(t)$ are calculated with computational cost $O(h^2)$. The value of $h$ needs to be properly determined because there is a trade-off between the accuracy of the potential estimation and the computational cost for the estimation.

## 5.3 Performance Evaluation

In this section, we evaluate our proposal to clarify the advantages and properties of hierarchical optimal feedback using the reduced-order model. We conduct a computer simulation and evaluate the convergence speed by comparing the results with our proposal (potential-based routing using hierarchical optimal feedback with and without model reduction [PBR-h-opt-mr and PBR-h-opt, respectively]) with potential-based routing using non-hierarchical optimal feedback including model reduction (PBR-opt-mr) as proposed in [45], and with the non-control scheme (PBR-no-ctrl) proposed in [101]. First, in Section 5.3.2, we evaluate the potential convergence speed after traffic changes to show that our proposed methods (PBR-h-opt-mr and PBR-h-opt) enhance the convergence speed of potentials. Moreover, we show that our proposal can be adapted to massive and frequent environmental changes, in Section 5.3.3. Finally, in Section 5.3.4, to demonstrate that our proposal can handle topological changes via redesigning the estimation model of the corresponding sub-controller, we evaluate the case in which a new sub-network is added to the network.

In conducting simulation experiments, for the network simulator, we use an event-driven packet-level simulator written in Visual C++ that calls MATLAB functions *dlqr*[8] to design an optimal central controller and sub-controllers with PBR-h-opt and PBR-h-opt-mr, *dhinflmi*[9] to design an optimal external controller with PBR-opt-mr, and *balred* to obtain a reduced-order model with PBR-opt-mr and PBR-h-opt-mr on a 64-bit PC with an Intel(R) Xeon(R) CPU with 2.70 GHz and 64.0 GB memory. The simulator has been developed by us. In the MAC layer, each node sends information about its own potential to its neighbors for their potential updates using intermittent receiver-driven data transmission (IRDT) [70], an asynchronous receiver-driven data transmission protocol. Note that this setting does not mean that our proposal depends on specific MAC layer

[8]The function *dlqr* is a linear-quadratic (LQ) state-feedback regulator for a discrete-time state space system. Given parameter matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $q$, and $r$ for the system dynamics $\boldsymbol{X}(t+1) = \boldsymbol{X}(t) + \boldsymbol{B}\boldsymbol{u}(t)$, *dlqr* computes $\boldsymbol{K}$, $\boldsymbol{S}$, and $e$. Then, control inputs $\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{X}(t)$ using the optimal gain matrix $\boldsymbol{K}$ minimize the quadratic cost function $J(u) = \sum_{n=1}^{\infty} (q||\boldsymbol{X}(n)|| + r||\boldsymbol{u}(n)||)$ for the system.

[9]*dhinflmi* is a function for designing the $H^{\infty}$ optimal controller [97] of a discrete system. Given parameter matrices $\boldsymbol{A}$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\boldsymbol{C}$, $\boldsymbol{D}_1$, and $\boldsymbol{D}_2$ for the system dynamics $\boldsymbol{X}(t+1) = \boldsymbol{A}\boldsymbol{X}(t) + \boldsymbol{B}_1\boldsymbol{D}(t) + \boldsymbol{B}_2\boldsymbol{u}(t)$ and the information $\boldsymbol{Y}(t) = \boldsymbol{C}\boldsymbol{X}(t) + \boldsymbol{D}_1\boldsymbol{D}(t) + \boldsymbol{D}_2\boldsymbol{u}(t)$ that is observed by the controller, *dhinflmi* computes the design parameters $\boldsymbol{A}_c$, $\boldsymbol{B}_{c1}$, $\boldsymbol{B}_{c2}$, $\boldsymbol{C}_c$, $\boldsymbol{D}_{c1}$, and $\boldsymbol{D}_{c2}$. The $n$th element of the matrix $\boldsymbol{D}(t)$ corresponds to an unexpected fluctuation in node $n$, such as a disturbance. $\boldsymbol{u}(t)$ are control inputs and are given by

$$\begin{aligned}\tilde{\boldsymbol{X}}(t+1) &= \boldsymbol{A}_c\tilde{\boldsymbol{X}}(t) + \boldsymbol{B}_{c1}\boldsymbol{D}(t) + \boldsymbol{B}_{c2}\boldsymbol{Y}(t),\\ \boldsymbol{u}(t) &= \boldsymbol{C}_c\tilde{\boldsymbol{X}}(t) + \boldsymbol{D}_{c1}\boldsymbol{D}(t) + \boldsymbol{D}_{c2}\boldsymbol{Y}(t).\end{aligned}$$

protocols. We use a disk model as a physical layer model where data packets drop with chance $100\%$ if they collide with each other. As the capacity of each sensor node is limited in wireless sensor networks, we set the queue size of each sensor node to $1$. In the simulator, nodes are not synchronized; in contrast, controllers, including the central controller and sub-controllers, are synchronized. Nodes do not match their timing to receive feedback from the corresponding sub-controller or the external controller and update their potentials. The asynchronous behavior of all nodes is implemented on a single thread with an event queue in our simulator. We set the interval of the control feedback from the central controller to sub-controllers, that of the control feedback from sub-controllers to corresponding controllable nodes, and that of potential updates in each node to be equal so that the central controller and sub-controllers can estimate the dynamics of the network with small errors.

### 5.3.1 Simulation Settings

We evaluate changes in potentials in potential-based routing and the number of data packets delivered to each sink node after environmental changes, such as traffic changes or the addition of a new sub-network. The network models used in the evaluations consist of sensor and sink nodes. For controlling these networks, we introduce a central controller and several sub-controllers. A network model is partitioned into several sub-networks including both sink and sensor nodes. Sink nodes in the same sub-network are directly connected to a sub-controller, and all sub-controllers are connected to the central controller. A sub-controller monitors the state of the corresponding sub-network via corresponding sink nodes, estimates the dynamics of the sub-network, and then sends suitable control inputs to these sink nodes to cause faster convergence of potentials. In contrast, the central controller obtains information about the network from sub-controllers, estimates interaction among sub-networks, and then provides feedback inputs to these sub-controllers. The central controller and sub-controllers provide feedback inputs at intervals $T_c$ and $T_s$, respectively, and each node updates its next potential value at an interval $T_p$. Typically, $T_c = T_s = T_p$ for matching with the potential dynamics described by Equations (5.2) and (5.4). In our evaluations, we consider a control cycle ($T_c = T_s = T_p$) as a unit time step.

In real situations, it is difficult for sub-controllers to monitor up-to-date potential values of all nodes in the corresponding sub-networks because of the overhead for collecting potential values, especially when the number of nodes of each sub-network increases. Therefore, here we consider that sub-controllers can monitor only nodes within $p$ hops from the corresponding sink nodes. Nodes within $p$ hops from sink nodes send a control message to sink nodes at interval $T_s$ to notify sub-controllers of their potential values. The $n$th element $c_n^i$ of $\boldsymbol{C}^i$ ($i \in \mathcal{S}$), which determines a set of monitorable nodes in Equation (5.5) or (5.8), is set to

$$c_n^i = \begin{cases} 1, & \text{if node } n \text{ is within } p \text{ hops from a sink node in sub-network } i \\ 0, & \text{otherwise} \end{cases} .$$

Sub-controllers do not directly monitor node potentials beyond $p$ hops from their connected sink nodes; instead, they only estimate them, using the estimation model (Equation (5.4) with (5.5) or Equation (5.7) with (5.8)), which describes the potential dynamics through local interactions among nodes. Each sub-controller calculates an optimal control using the potential information obtained through the observation and estimation. Similarly, the central controller obtains the potential information about observable nodes from all sub-controllers at interval $T_c$ for estimating interactions among sub-networks. The $n$th element $w_n$ of $\boldsymbol{W}$, which determines a set $\boldsymbol{w}$ of nodes whose potential information is sent from sub-controllers to the central controller in Section 5.2.4, is set to

$$w_n = \begin{cases} 1, & \text{if node } n \text{ is within } p \text{ hops from a sink node} \\ 0, & \text{otherwise} \end{cases} .$$

At the beginning of the simulation, the potential values of all nodes, including sensor nodes and sink nodes, are initialized to 0. During the first 20 steps, each node exchanges its potential value with neighbor nodes and updates its potential value at interval $T_p$ according to Equation (5.1) so that the potential values are stabilized. For removing the influence of differences among schemes, we do not generate data packets during that time. At 20 steps, data packets begin to be generated at sensor nodes according to the Poisson process with their flow rates.

We evaluate the convergence speed of potentials and data packets delivered to each sink node after traffic changes. To measure the convergence speed of potentials, we define the degree $\epsilon_n(t)$ of the potential convergence for each node that is given by

$$\epsilon_n(t) = |\bar{\theta}_n - \theta_n(t)|,$$

where $\bar{\theta}_n$ corresponds to the target potential value of node $n$. We consider convergence to be achieved when $\epsilon_n(t)$ is sufficiently small for all nodes. The convergence time of potentials is defined as the minimum time taken by all sensor and sink nodes to satisfy the condition

$$\epsilon_n(t) < \delta, \tag{5.15}$$

where $\delta$ is a constant value. In an ideal situation, where all nodes are completely synchronized and there is no noise, all sensor and sink nodes satisfy Equation (5.15) finally, even if $\delta = 0$; however, in an actual situation, not all nodes can satisfy Equation (5.15). Therefore, we set $\delta$ to $0.01$ for the convergence evaluation.

Energy efficiency is a significant and challenging task for wireless sensor networks. With respect to our proposal, each sensor node acts on local information and simple rules so that the computational cost for each sensor node is very low. This is an inherent characteristic of self-organizing systems. Moreover, our proposal can achieve load balancing by setting the flow matrix $\boldsymbol{F}$ such that each sink node receives data packets at the same rate as other sink nodes, which we explain in Section 5.3.2. Finally, our proposal does not depend on specific MAC-layer protocols. Therefore, we can reduce the energy required for data transfer between adjacent nodes by introducing energy-efficient MAC-layer protocols, such as IRDT (which we use in this evaluation).

Table 5.2: Parameter settings

| Parameter | Value |
|---|---|
| $\alpha$ | 0.4 |
| $\beta$ | 0.2 |
| $\sigma$ | 0.1 |
| $r$ | 10 |

The parameter values are summarized in Table 5.2. The optimal parameters $(\alpha, \beta, \sigma)$ for PBR-no-ctrl are given in [101] [10], but potentials of nodes diverge to infinity and do not converge. Therefore, we use the same settings shown in Table 5.2 even for PBR-no-ctrl. All results presented below are averaged over 30 simulation runs for each parameter setting.

### 5.3.2 Performance Evaluation of Hierarchical Control Method with Reduced-order Model

First, we evaluate the convergence speed of potentials after traffic changes considering the constraints in wireless sensor networks. To reveal the performance and properties of our proposal, we consider the case where the data generation rates of a partial set of sensor nodes change once, simultaneously.

Figure 5.2 shows the network model with 309 nodes (including 9 sink nodes) that is used in this evaluation. In this figure, sink nodes (resp., sensor nodes) are illustrated with squares (resp., dots). As shown in the figure, each sub-network is connected to a sub-controller via the sink node, whereas the central controller has direct connections with all sub-controllers. Each sub-controller collects potential information about nodes within 2 hops from its connected sink node, that is, $p$ is set to 2. Details of the number of nodes observed by each sub-controller are shown in Table 5.3.

In this evaluation, the data generation rates are initially set to be $0.050$ packets/step for all sensor

[10] Optimal $\alpha$ and $\beta \cdot \sigma$ are respectively given by $\frac{1-\xi}{1+\xi}$ and $\frac{\alpha+1}{2}$ for the fastest convergence rate of potentials, where $\xi = \sqrt{1-\nu_1^2}$. $\nu_1$ is the spectrum radius of the matrix $\boldsymbol{S} = \boldsymbol{I}_{N\times N} - \boldsymbol{G}^{-1}\boldsymbol{\Gamma}$, where $\boldsymbol{G}$ is the degree matrix and $\boldsymbol{\Gamma}$ is the graph Laplacian of the network. The spectrum radius of the matrix $\boldsymbol{S}$ is defined as the maximum among the absolute values of the eigenvalues of $\boldsymbol{S}$. See [101] for more details.

Table 5.3: Observable nodes in each sub-network with PBR-h-opt and PBR-h-opt-mr

| Sub-network No. | No. of nodes | No. of observable nodes |
|---|---|---|
| 1 | 43 | 10 |
| 2 | 39 | 17 |
| 3 | 38 | 15 |
| 4 | 51 | 16 |
| 5 | 12 | 3 |
| 6 | 25 | 9 |
| 7 | 56 | 22 |
| 8 | 16 | 10 |
| 9 | 29 | 14 |
| Total | 309 | 116 |

nodes of the network shown in Figure 5.2. At 200 steps from the beginning of the simulation, the data packet generation rates of sensor nodes are changed to examine the convergence speed of our method. After traffic changes at 200 steps, the data packet generation rates are increased to 0.075 packets/step for a partial set $\mathcal{N}_{inc}$ ($\in \mathcal{N}_{sen}^{1}$) of sensor nodes included in sub-network 1. The number of nodes whose data generation rates increase is set to 20 (i.e., $|\mathcal{N}_{inc}| = 20$). Here, $\mathcal{N}_{sen}$ corresponds to the set of all sensor nodes, $\mathcal{N}_{sen}^{i}$ corresponds to the set of sensor nodes in sub-network $i$, and $\mathcal{N}_{sin}$ corresponds to the set of all sink nodes in the network. The initial data generation rate of sensor nodes (0.050 packets/step) corresponds to $\bar{f}_n = -1.0$; therefore, before traffic changes, the flow rate vector $\boldsymbol{F} = \begin{bmatrix} \bar{f}_1 & \cdots & \bar{f}_N \end{bmatrix}^T$ is given by

$$\bar{f}_n = \begin{cases} -1.0, & \text{if } n \in \mathcal{N}_{sen} \\ \frac{1.0 \times |\mathcal{N}_{sen}|}{|\mathcal{N}_{sin}|}, & \text{if } n \in \mathcal{N}_{sin} \end{cases} . \tag{5.16}$$

Note that we construct the potential fields such that all sink nodes can receive data packets equally because load balancing is known to be a challenging task for wireless sensor networks. Thus, the flow rate at each sink node is ideally given by $\frac{1.0 \times |\mathcal{N}_{sen}|}{|\mathcal{N}_{sin}|}$ $(= 25)$. Similarly, after the traffic changes, $\boldsymbol{F}$ is given by

$$\bar{f}_n = \begin{cases} -1.0, & \text{if } n \in \mathcal{N}_{sen} - \mathcal{N}_{inc} \\ -1.5, & \text{if } n \in \mathcal{N}_{inc} \\ \frac{1.0\times(|\mathcal{N}_{sen}|-|\mathcal{N}_{inc}|)+1.5\times|\mathcal{N}_{inc}|}{|\mathcal{N}_{sin}|}, & \text{if } n \in \mathcal{N}_{sin} \end{cases} . \tag{5.17}$$

Because the data generation rates of sensors included in $\mathcal{N}_{inc}$ increase, the flow rate at each sink node also increases to $\frac{1.0\times(|\mathcal{N}_{sen}|-|\mathcal{N}_{inc}|)+1.5\times|\mathcal{N}_{inc}|}{|\mathcal{N}_{sin}|}$. We calculate the identical target values of potentials using these flow rates (Equations (5.16) and (5.17)) by Equation (5.3). In the evaluation of PBR-h-opt-mr, we set $h^i$ to the same value ($h_{sub}$) for all $i \in \mathcal{S}$. The degrees $(h, h_{sub})$ of the reduced order models for controllers are set to $(18, 2)$ and $(27, 3)$. Although it is difficult to quantitatively express the relation between $h$ and $h_{sub}$ (we note that $h = |\mathcal{S}| \cdot h_{sub}$ is not always satisfied), $h$ is typically larger than $h_{sub}$ because the central controller manages a larger amount of information than each sub-controller. The degree $h_{ex}$ of the external controller in PBR-opt-mr is set to 18.

Figure 5.3 shows the changes in the potential values of PBR-no-ctrl, PBR-opt-mr, PBR-h-opt, and PBR-h-opt-mr. More precisely, this figure shows a plot of $\boldsymbol{X}(s) = \bar{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(s)$ against step $s$, and potential convergence is achieved when each element of $\boldsymbol{X}(s)$ is sufficiently close to 0, that is, when Equation (5.15) is satisfied. In this figure, thick colored lines correspond to potential changes of the 9 sink nodes and thin gray lines correspond to potential changes of sensor nodes. Sink node potentials change more than those of sensor nodes because we set sink nodes as controllable nodes and they receive feedback inputs $\boldsymbol{u}$ from the corresponding sub-controllers, whereas sensor nodes are indirectly affected by them via sink nodes. Feedback inputs provided to each sink node are different, and so the potential change of each sink node varies.

As shown in Figure 5.3, the convergence speed of potentials is enhanced by introducing optimal feedback mechanisms, including non-hierarchical and hierarchical mechanisms. This proves the effectiveness of optimal feedback in a hierarchical manner. The central controller can estimate interactions among sub-networks correctly while sub-controllers can estimate the potential dynamics of

Table 5.4: Potential convergence time in the case with a large-scale network

| Scheme | Convergence time [step] |
|---|---|
| PBR-no-ctrl | 1484.0 |
| PBR-opt-mr ($h_{ex} = 18$) | 137.61 |
| PBR-h-opt | 140.47 |
| PBR-h-opt-mr ($h = 18,\ h^i = 2$) | 186.18 |
| PBR-h-opt-mr ($h = 27,\ h^i = 3$) | 166.18 |

their corresponding sub-networks in our proposal. It is worth mentioning that each sub-controller of PBR-h-opt and PBR-h-opt-mr observes node potentials within at most 2 hops of the sink nodes in its corresponding sub-network. Sub-controller $i$ estimates the potentials of non-observable and future potentials of nodes in its corresponding sub-network using feedback inputs $\boldsymbol{Z}^i$, which are provided from the central controller to each sub-network (Equation (5.4) or (5.7)), and then determines the optimal feedback inputs $\boldsymbol{u}^i$. Our result indicates that sub-controllers can correctly estimate node potentials in their corresponding sub-network even though they do not directly observe nodes outside their corresponding sub-network. Table 5.4 shows the time needed from traffic changes until the potential convergence is achieved (with $\delta = 0.01$). Figure 5.3 shows potential changes only within 180–600 steps; however, it takes 1484.0 steps for the potential to converge with PBR-no-ctrl, as shown in Table 5.4. From the result, compared with PBR-no-ctrl, potential convergence is accelerated around 10.8 fold with PBR-opt-mr, 10.6 fold with PBR-h-opt, and 7.97 and 8.93 fold with PBR-h-opt-mr with $(h, h_{sub}) = (18, 2)$ and $(h, h_{sub}) = (27, 3)$, respectively. The convergence speed of potentials with our proposed methods (i.e., PBR-h-opt and PBR-h-opt-mr) is a bit slower than that with PBR-opt-mr. This is because each sub-controller can directly collect and estimate the potential values of only the nodes in the corresponding sub-network, and they can receive the potential information about other sub-networks only via the central controller.

From the viewpoint of computational cost, the computational cost $O({h^i}^2)$ of each sub-controller

Table 5.5: Computational time for each scheme

| Scheme | Controller Type | Computational time [s] | | |
|---|---|---|---|---|
| | | Controller design (*dlqr* or *dhinflmi*) | Model reduction (*balred*) | Calculation of feedback inputs |
| PBR-no-ctrl | – | 0 | 0 | 0 |
| PBR-opt-mr ($h_{ex} = 18$) | External | 6398 | 4.084 | $6.313 \times 10^{-5}$ |
| PBR-h-opt | Sub | 0.04003 | 0 | $9.360 \times 10^{-5}$ |
| | Central | 30.75 | 0 | $1.798 \times 10^{-3}$ |
| PBR-h-opt-mr ($h = 18, h_{sub} = 2$) | Sub | 0.08880 | 0.04311 | $3.853 \times 10^{-6}$ |
| | Central | 35.38 | 2.196 | $3.815 \times 10^{-5}$ |
| PBR-h-opt-mr ($h = 27, h_{sub} = 3$) | Sub | 0.09006 | 0.04396 | $4.125 \times 10^{-6}$ |
| | Central | 34.35 | 2.428 | $4.810 \times 10^{-5}$ |

for estimating the potential dynamics and calculating control inputs is much smaller than the computational cost $O(h^2)$ of the external controller proposed in [45], where only one controller monitors the potential dynamics of all nodes and then calculates control inputs for the entire network. This is because the number of state variables that suffices to describe the network/sub-network is generally larger if the number of nodes in the network/sub-network is larger. Moreover, given the information about the network topology and flow rates, the central controller is designed with the computational cost $O(h^2)$ as described in Section 5.2.4, which is much smaller than the computational cost $O(h^3)$ needed for designing the external controller proposed in [45]. Table 5.5 shows the computational time required for the controller design, model reduction, and calculation of feedback inputs. In Table 5.5, the computational time for the controller design and the model reduction results from one simulation run for each scheme are shown, as measured by using the *timeit* function in MATLAB. The computational time for the calculation of feedback inputs is averaged over 30 calculations of $\boldsymbol{u}$ or $\boldsymbol{u}^i$ for each scheme and measured by using the *QueryPerformanceCounter* function in C++. In this table, the computational time for the sub-controller corresponds to the average amount of time required for each sub-controller. As shown in the table, the computational time for the controller design with our proposal (PBR-h-opt and PBR-h-opt-mr) is much smaller than that with PBR-opt-mr. Moreover, the time required for the calculation of feedback inputs with our proposal is a little smaller than that with PBR-opt-mr. It is worth mentioning that because feedback inputs are calculated at every step, the difference in the computational time for each step is small but makes a good contribution to the reduction in the computational time in the long run. It is true that the non-hierarchical scheme is superior to the hierarchical one in the convergence speed of potentials, but the non-hierarchical scheme cannot adapt to the rapid growth of information networks in scale [5–7] due to its high computation cost. On the contrary, our proposal can enhance the convergence speed of potentials with low computational cost, even in large-scale networks. Consequently, the hierarchical optimal feedback mechanism is more scalable than the non-hierarchical one.

In general, as the degree of the reduced-order model becomes smaller, the approximation error becomes larger while the computational cost (described by $O({h^i}^2)$) becomes much smaller. Actually, as shown in Figures 5.3(c)–5.3(e) and Table 5.4, the convergence speed of PBR-h-opt-mr with $h = 18$ and $h_{sub} = 2$ is a bit slower than that of PBR-h-opt and PBR-h-opt-mr for higher values of $h$ and $h_{sub}$. Consequently, there is a trade-off between the computational cost and the performance of our proposal.

The estimation errors of the potentials cannot be avoided completely because of disturbances, modeling errors of the system dynamics, and so on. Specifically, in our evaluations, nodes are not synchronized against the potential dynamics described by Equation (5.1) and the potential values the controller collects are not always correct owing to communication delays, dropped data, or interference. Our evaluations prove that our proposal can work even when there are such estimation errors. In this evaluation, a partial set of control messages drop because traffic congestion occurs around sink nodes. This occurs because the external controller/sub-controllers collect the network information via sink nodes in potential-based routing with optimal feedback. Control messages are sent by sensor nodes within $p$ hops from sink nodes for sending potential information to sub-controllers. If control messages drop, sub-controllers cannot receive potential information, which leads to errors in the estimation of potentials. Moreover, the asynchrony of the controller and nodes is also a cause of estimation errors because PBR-h-opt and PBR-h-opt-mr, as well as PBR-opt-mr, inherently assume synchronous systems. Nevertheless, our proposal can achieve fast convergence of potentials despite such errors, which clearly shows that our proposal works well in an asynchronous environment where noise or disturbances exist.

In Sections 5.3.2, we have shown that a hierarchical optimal control by the central controller and sub-controllers is effective in wireless sensor networks where the capacity and energy of each node are limited. Moreover, PBR-h-opt-mr enhanced the convergence speed of potentials with much lower computational cost than PBR-opt-mr. This indicates that a reduced-order model reflects the dominant characteristics of the original model. It is also worth mentioning that even when some amount of approximation error exists, fast convergence of potentials can be achieved.

### 5.3.3 Adaptability to Massive and Frequent Environmental Changes

Next, we evaluate the changes in potential values and traffics in cases where the data generation rates of all sensor nodes change several times, using this to demonstrate that our proposal can work even if there are frequent and massive environmental changes. Specifically, we use the network model with 104 nodes (including 4 sink nodes) described in Figure 5.4 for this evaluation, and the data packet generation rates of sensor nodes in sub-network 1 (resp., 3) and that in sub-network 2 (resp., 4) are exchanged every 200 steps. Each sub-controller collects potential information about nodes within 1 hop from its connected sink node; that is, $p$ is set to 1.

The data packet generation rates are initially set to be 0.025 packets/step for sensor nodes in sub-networks 1 and 3 of Figure 5.4, and 0.075 packets/step for sensor nodes in sub-networks 2 and

4. After traffic changes at 200 steps, the data packet generation rates are increased to $0.075$ packets/step for the left half sensor nodes and decreased to $0.025$ packets/step for the right half nodes. Subsequently, the data packet generation rates of sensor nodes in sub-network 1, 3 and that in sub-network 2, 4 are exchanged at interval 200 step. The average data generation rate of a node of $0.050$ packets/step corresponds to $\bar{f}_n = -1.0$, and therefore, the flow rate vector $\boldsymbol{F}$ during the first 200 steps is given by

$$\bar{f}_n = \begin{cases} -0.5, & \text{if } n \in \mathcal{N}_{sen}^1 \cup \mathcal{N}_{sen}^3 \\ -1.5, & \text{if } n \in \mathcal{N}_{sen}^2 \cup \mathcal{N}_{sen}^4 \\ \frac{0.5\times(|\mathcal{N}_{sen}^1|+|\mathcal{N}_{sen}^3|)+1.5\times(|\mathcal{N}_{sen}^2|+|\mathcal{N}_{sen}^4|)}{|\mathcal{N}_{sin}|}, & \text{if } n \in \mathcal{N}_{sin} \end{cases} . \tag{5.18}$$

Similarly, $\boldsymbol{F}$ during the next 200 steps is given by

$$\bar{f}_n = \begin{cases} -1.5, & \text{if } n \in \mathcal{N}_{sen}^1 \cup \mathcal{N}_{sen}^3 \\ -0.5, & \text{if } n \in \mathcal{N}_{sen}^2 \cup \mathcal{N}_{sen}^4 \\ \frac{1.5\times(|\mathcal{N}_{sen}^1|+|\mathcal{N}_{sen}^3|)+0.5\times(|\mathcal{N}_{sen}^2|+|\mathcal{N}_{sen}^4|)}{|\mathcal{N}_{sin}|}, & \text{if } n \in \mathcal{N}_{sin} \end{cases} . \tag{5.19}$$

Figure 5.5 shows the changes in the potential values of PBR-no-ctrl, PBR-opt-mr with $h_{ex} = 8$, and PBR-h-opt-mr with $h = 8$ and $h^i = 2$. Specifically, Figure 5.5 shows a plot of $\boldsymbol{X}(s) = \bar{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(s)$ against step $s$. As shown in Figure 5.5, the hierarchical control feedback mechanism can enhance the convergence speed of potentials, as compared with the case of the non-control scheme.

Figure 5.6 shows changes in the number of data packets delivered to each sink node every 20 steps. In each case, the number of data packets delivered to each sink node becomes disproportionate after the traffic changes at 200 steps. Then sink nodes gradually become able to receive data packets equally, because potentials are updated to adapt to the current packet rate. We can observe that the traffic convergence speed is also accelerated by optimal feedback. This is because the potential convergence speed is enhanced by the optimal feedback mechanism. As shown in Figures 5.5(a) and 5.6(a), as the convergence speed of potentials is too low, traffic cannot adapt to frequent and massive changes with PBR-no-ctrl. In contrast, by introducing optimal control mechanisms, the adaptability to adapt to such changes is improved (Figures 5.5(b)–5.5(d) and 5.6(b)–5.6(d)).

One problem we note is that our proposal reduces the average number of data packets delivered to each sink node immediately after traffic changes. This is because some sink nodes temporarily have the largest potential values within their communication ranges according to the control inputs, so data packets cannot arrive at sink nodes. Therefore, data packets will drop when the controller makes large changes to the potentials, which contributes to the faster convergence speed of potentials. However, the data packet drops are immediately reduced and the traffic finally converges faster than the non-control scheme because of the faster potential convergence. Note that in an actual situation, data packets may be retransmitted instantly. Here, we evaluate only the case in which data packets are never retransmitted because the main purpose of this study is to reveal the upper limit of convergence speed of self-organizing systems. Moreover, Figures 5.6(b)–5.6(d) show worst-case scenarios for temporal packet drops, because the controller changes sink node potentials (in other words, data packet destinations), and therefore, many data packets are dropped when a sink node temporarily gets the highest potential within its communication range owing to control inputs. If the sub-controller provides an optimal feedback to several sensor nodes where only some data packets arrive, the number of data packet drops will be smaller. With a lower $r$, sink nodes are more likely to be assigned higher potential values as the sub-controller can make large changes to the potentials, whereas the recovery speed of data packets delivered to each sink node increases. This indicates that there is trade-off between the convergence speed of potentials and potential fluctuations.

In Section 5.3.3, we have shown that our proposal can adapt to frequent and massive environmental changes.

### 5.3.4 Integration of Two Different Networks

We finally evaluate how our proposal works when a new sub-network is added to the network. The network model shown in Figure 5.4 is used in this evaluation. First, the network only consists of nodes of sub-networks 1, 2, and 3. Then, at 200 steps from the beginning of the simulation, sub-network 4 is added to the network. To evaluate the influence of the addition of a new sub-network, and not that of traffic changes, the data packet generation rates of all sensor nodes are fixed at $0.050$ packets/step, and therefore, $\boldsymbol{F}$ before the addition of a new sub-network is given by

$$\bar{f}_n = \begin{cases} -1.0, & \text{if } n \in \mathcal{N}^1_{sen} \cup \mathcal{N}^2_{sen} \cup \mathcal{N}^3_{sen} \\ \frac{1.0 \times |\mathcal{N}^1_{sen}| + |\mathcal{N}^2_{sen}| + |\mathcal{N}^3_{sen}|}{|\mathcal{N}^1_{sin}| + |\mathcal{N}^2_{sin}| + |\mathcal{N}^3_{sin}|}, & \text{if } n \in \mathcal{N}^1_{sin} \cup \mathcal{N}^2_{sin} \cup \mathcal{N}^3_{sin} \end{cases}. \tag{5.20}$$

Then, $\boldsymbol{F}$ after the addition of a new sub-network is given by

$$\bar{f}_n = \begin{cases} -1.0, & \text{if } n \in \mathcal{N}_{sen} \\ \frac{1.0 \times |\mathcal{N}_{sen}|}{|\mathcal{N}_{sin}|}, & \text{if } n \in \mathcal{N}_{sin} \end{cases}. \tag{5.21}$$

In this evaluation, we set degrees $(h, h_{sub})$ to $(6, 2)$ before the addition of a new sub-network at $200$ steps, whereas we set $(h, h_{sub})$ to $(8, 2)$ after $200$ steps with PBR-h-opt-mr. After the addition of a sub-network, we increase the degree $h$ of the estimation model for the central controller because the number of nodes in the entire network increases. With PBR-opt-mr, we set degree $h_{ex}$ to 6, 8 before and after the addition of a new sub-network, respectively. Other simulation settings are the same as in Section 5.3.3.

Figure 5.7 shows the potential changes with PBR-no-ctrl, PBR-opt-mr, PBR-h-opt, and PBR-h-opt-mr, and Table 5.6 shows the convergence time of potentials. The figure plots $\boldsymbol{X}(t) = \bar{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(t)$ against step $s$. As the figure and table show, the convergence speed of potentials is improved by $1.87$ times with PBR-opt-mr, $2.02$ times with PBR-h-opt, and $2.75$ times with PBR-h-opt-mr. This indicates that both the hierarchical optimal feedback mechanism and the non-hierarchical one can enhance the convergence speed in cases where a new sub-network is added to the network.

Table 5.6: Potential convergence time in the case of adding a new sub-network

| Scheme | Convergence time [step] |
|---|---|
| PBR-no-ctrl | 234.71 |
| PBR-opt-mr ($h_{ex} = 8$) | 125.33 |
| PBR-h-opt | 116.28 |
| PBR-h-opt-mr ($h = 8,\ h^i = 2$) | 85.500 |

The dynamics of potentials depends on the network topology, and therefore, the estimation model of the dynamics also depends on the network topology. In other words, unlike cases of traffic changes (Subsections 5.3.2 and 5.3.3), the estimation model needs to be redesigned when topological changes occur. With PBR-opt-mr, the estimation model needs to be redesigned entirely and the computational cost for redesigning is given by $O(h^3)$. On the contrary, with PBR-h-opt and PBR-h-opt-mr, the estimation model is designed for each sub-network. Therefore, only the estimation model of the sub-network whose topology changes needs to be redesigned and the computational cost is given by $O({N^i}^3)$ and $O({h^i}^3)$ with PBR-h-opt and PBR-h-opt-mr, respectively. The degree of the approximation model for a sub-network is generally smaller than that for the entire network. Consequently, the computational cost for redesigning the estimation model due to topological changes is smaller with the hierarchical optimal feedback mechanism (PBR-h-opt-mr) than that with the non-hierarchical one (PBR-opt-mr).

In conclusion, hierarchical optimal feedback by several controllers can work with low computational cost even in the case of topological changes in the network. Specifically, owing to the high adaptability to the integration of several networks as shown in this evaluation, our proposal can be adapted to much larger-scale networks.

## 5.4 Summary

In self-organizing systems, each component behaves according to only local information, which leads to slow convergence. We propose and evaluate potential-based routing with hierarchical optimal feedback using a reduced-order model, in which several controllers monitor and estimates system states, and provide optimal feedback in a hierarchical manner. Simulation results have shown that hierarchical optimal feedback using a reduced-order model can facilitate the convergence of potentials while reducing costs for collecting system information and estimating system dynamics. Moreover, our proposal has high scalability because the computational cost is much smaller than in the non-hierarchical scheme.

However, some challenges still need to be overcome. First, the optimal feedback mechanism assumes that the controller has information about the network topology and flow rates, which reduces the practicality of our proposal. Second, the potential convergence is achieved as a result of the iterative behavior, i.e., the controller's optimal feedback and nodes' potential updates, and therefore, the potential cannot converge if environmental changes occur more frequently than the iterative behavior. Third, the controllability and stability of the networks depends on the network topology. Finally, the optimal control improves the convergence speed of potentials but also causes potential fluctuations, as shown by the simulation results. These fluctuations lead to data packet drops because sink nodes temporally have the highest potentials between their neighbors. There is a trade-off between the improvement of the potential convergence speed and the potential fluctuations.

We are now studying a predictive control method that can adapt to the dynamically changing network. For this purpose, it would be a promising direction to design a controller based on Bayesian inference.

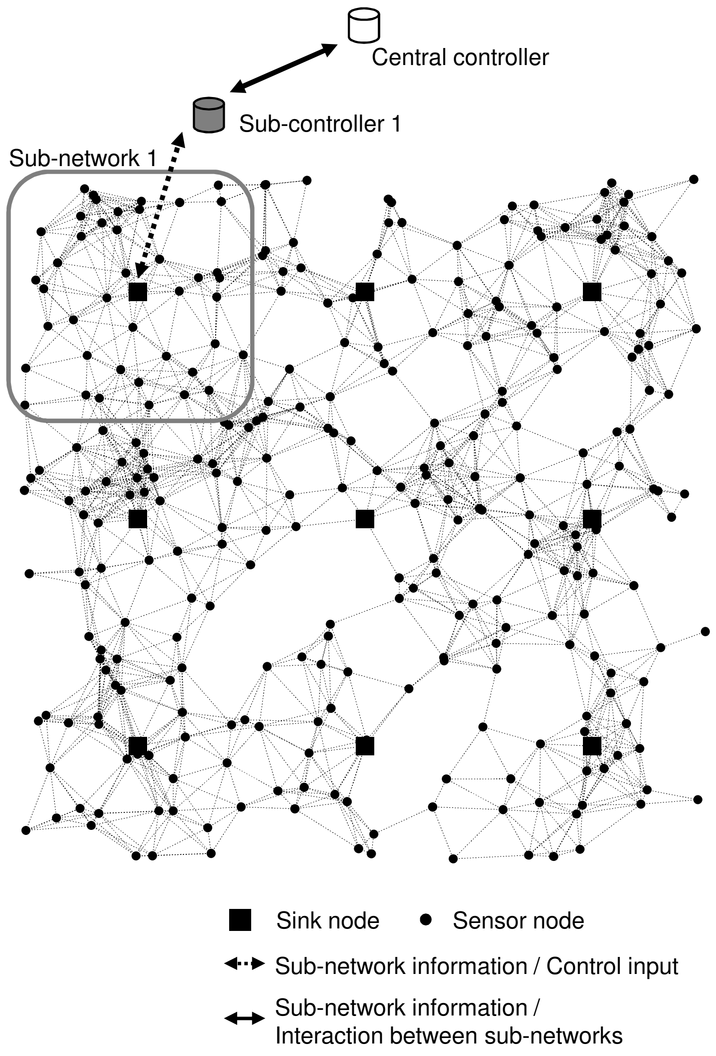
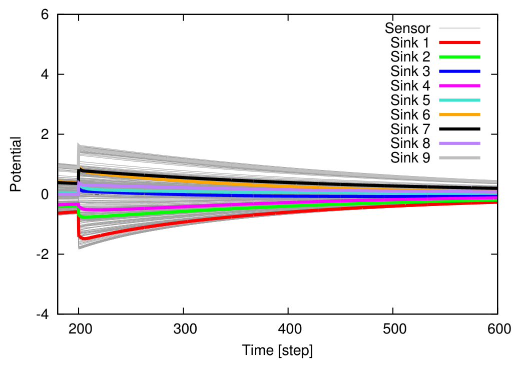
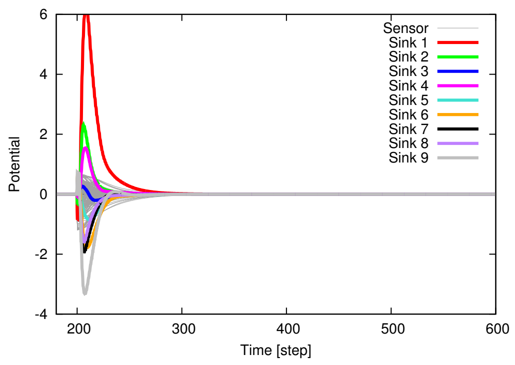

Central controller
Sub-controller 1
Sub-network 1
■ Sink node
● Sensor node
Sub-network information / Control input
Sub-network information /
Interaction between sub-networks

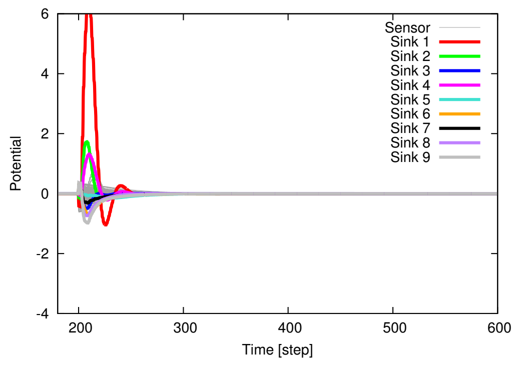
Figure 5.2: Network topology ($N = 309$). Sink nodes are illustrated with squares and sensor nodes with dots. Only one node is shown to be connected to a sub-controller, and the central controller shows a connection with only one sub-controller. However, each sink node is actually wired with a sub-controller, and the central controller has connections with 9 sub-controllers.



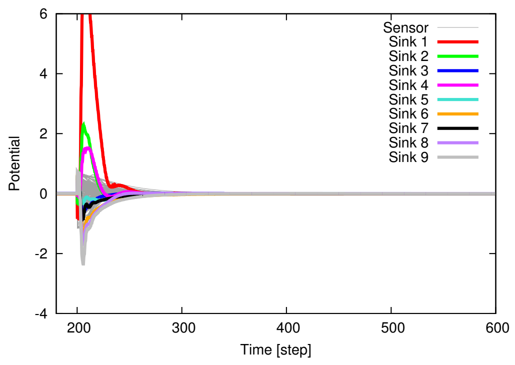
(a) PBR-no-ctrl



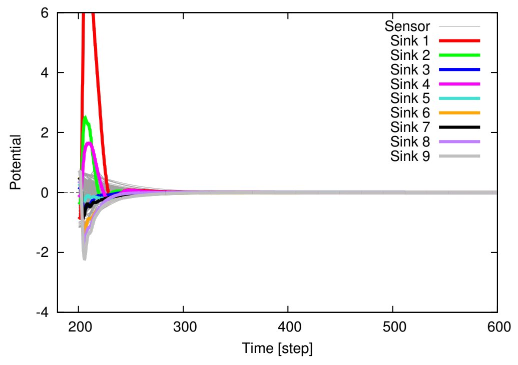
(b) PBR-opt-mr ($h_{ex} = 18$)



(c) PBR-h-opt



(d) PBR-h-opt-mr ($h = 18,\ h^i = 2$)



(e) PBR-h-opt-mr ($h = 27,\ h^i = 3$)

Figure 5.3: Potential convergence in the case with a large-scale network

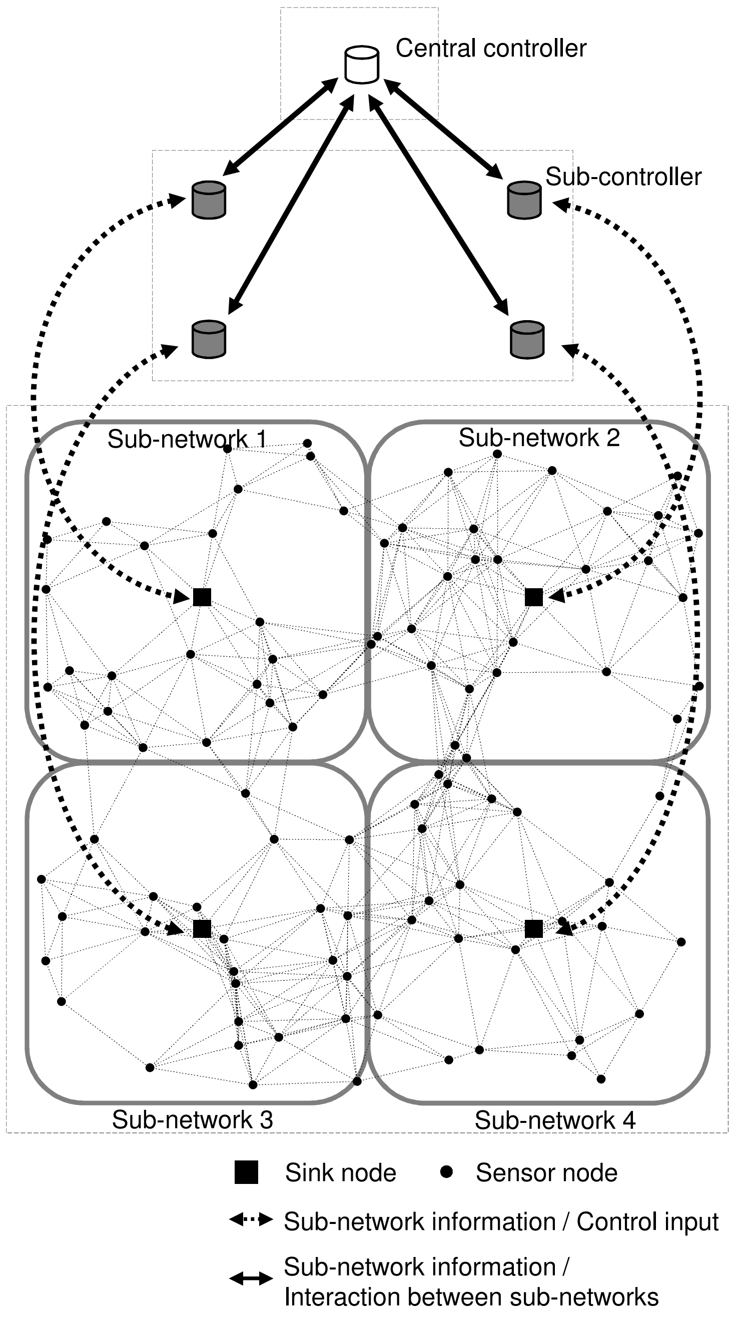
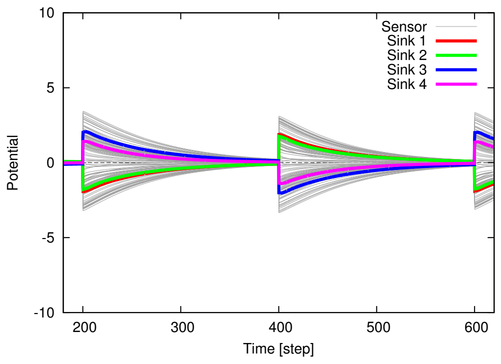
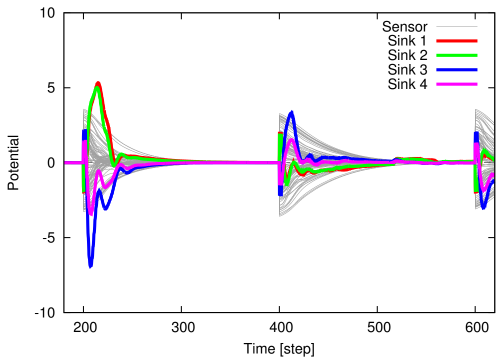


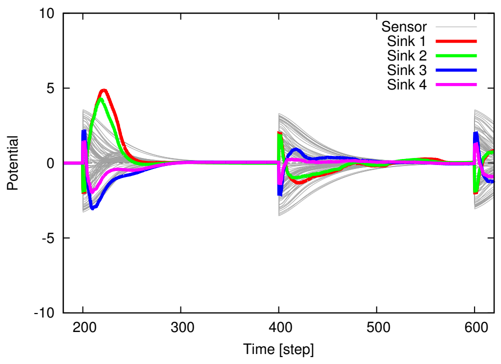
Figure 5.4: Network topology ($N = 104$)



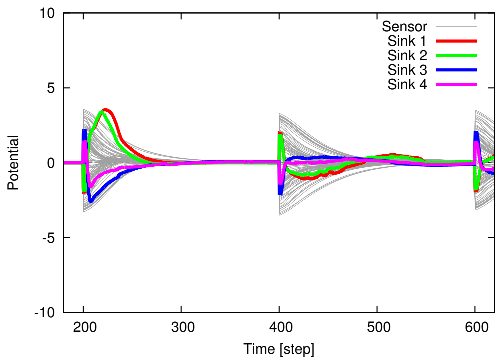
(a) PBR-no-ctrl



(b) PBR-opt-mr ($h_{ex} = 3$)

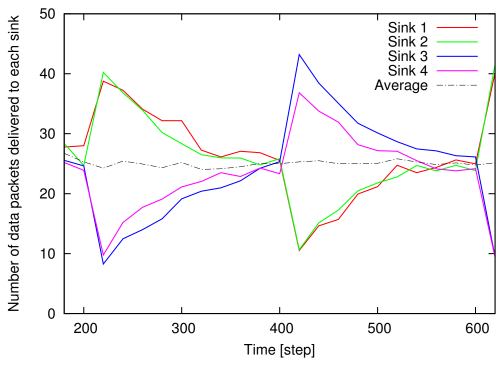


(c) PBR-h-opt

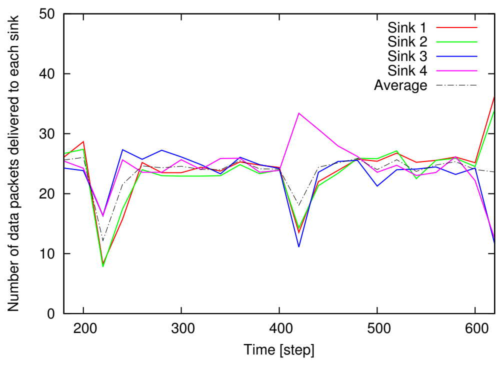


(d) PBR-h-opt-mr ($h = 8,\ h^i = 2$)

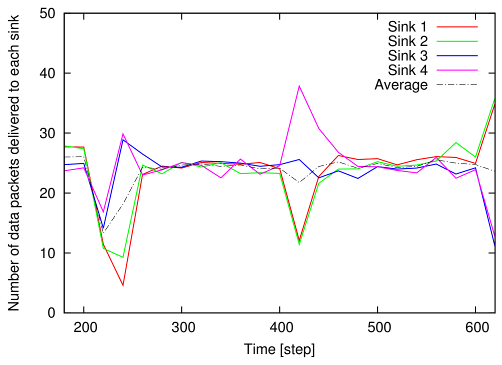
Figure 5.5: Potential convergence in case of traffic changes



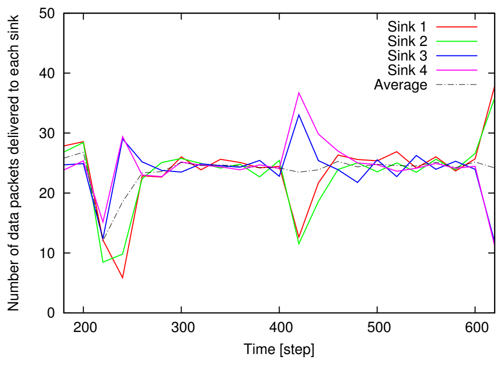
(a) PBR-no-ctrl



(b) PBR-opt-mr ($h_{ex} = 8$)

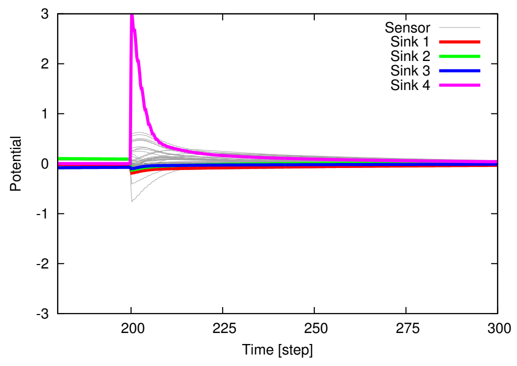


(c) PBR-h-opt

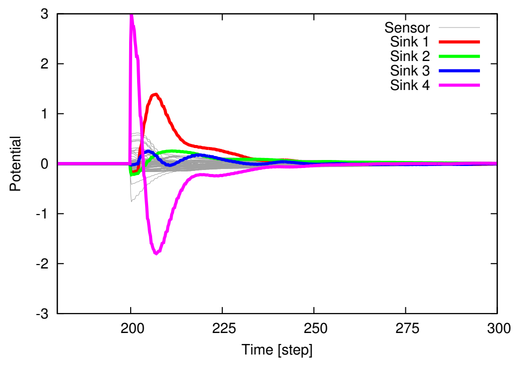


(d) PBR-h-opt-mr ($h = 8,\ h^i = 2$)

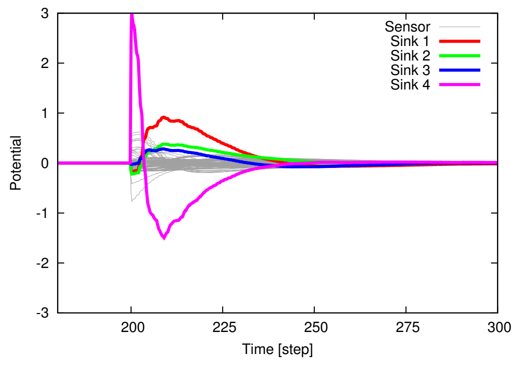
Figure 5.6: Data packets delivered to each sink node in case of traffic changes



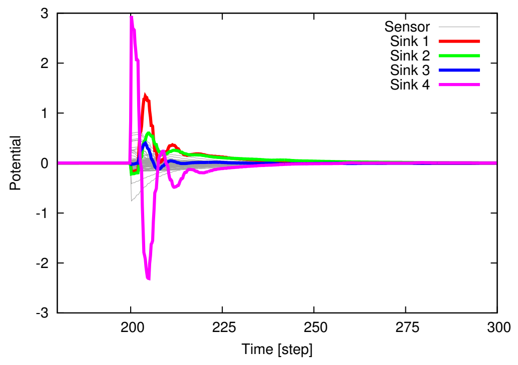
(a) PBR-no-ctrl



(b) PBR-opt-mr ($h_{ex} = 8$)



(c) PBR-h-opt



(d) PBR-h-opt-mr ($h = 8,\ h^i = 2$)

Figure 5.7: Potential convergence in case of addition of new sub-network

# Chapter 6

# Conclusion

For future growth of information networks, it is essential to develop a new network architecture which has high scalability, adaptability, and robustness. Self-organizing control is a promising idea to realize it. Especially, biological systems are inherent self-organizing. However, such systems have some disadvantages that complicates implementation of industrial and business systems. In this thesis, therefore we study managed self-organizing network control inspired by adaptive biological systems.

First, in Chapter 2, we introduce a predict mechanism to self-organizing systems and examine the property and the limit of self-organizing systems where global behavior or pattern emerges based on only local interactions among components. To rapidly adapt to changing conditions, it is therefore necessary that systems be controlled considering the future state of systems as predicted by observing system behavior. We propose and evaluate a predictive mechanism for AntNet, which is a simple and basic example of ACO-based routing. Simulation results show that the proposed method can facilitate path reestablishment when the network environment changes. Moreover, simulation evaluations showed that the control overhead needed for prediction becomes smaller in multiple session scenarios. Even in a more realistic environment where forward, backward, and predictive ants are lost in a network, ants can quickly re-establish other paths because they explore the network not deterministically but rather stochastically, and the positive feedback through pheromones leads to ants following shorter paths.

In Chapter 3, we next introduce a bio-inspired clustering scheme to crawler classification and evaluate it using data collected in a real network. Due to the rapid growth in the diversity of web services, it is becoming increasingly difficult to classify a large number of communication logs using only known features that are detected manually. In this study, we adopted AntTree, an ant-based clustering scheme, to achieve crawler classification that can adapt to a diverse range of web services. Our evaluation results indicated that AntTree can classify crawlers more accurately than the conventional scheme.

Through accomplishments of studies mentioned above, we show that the upper limitation of the convergence speed of self-organizing systems without any centralized control. In Chapter 4, therefore, we propose an external controller using an optimal control method as a mean of managing self-organizing for fast adaptation. We take potential-based routing, a self-organizing routing mechanism, and introduce an optimal control method to it. With the control method, a controller monitors and estimates system states, and provides optimal feedback for the fastest convergence. Simulation results have shown that optimal feedback using a reduced-order model can facilitate the convergence of potentials while reducing costs for collecting system information and estimating system dynamics. Moreover, our proposal has high scalability and robustness against information loss from node failures.

Finally, in Chapter 5, we propose and evaluate potential-based routing with hierarchical optimal feedback. Through previous work, we show the fastest convergence speed of the linear system, but, especially in large-scale networks, it is difficult for only one external controller to collect and manage information of the whole network. On the other hand, in this study, we divide a network to several controllers and divide external controller's roles to a central controller and several sub-controllers for reducing the computation and communication cost for each controller. Simulation results have shown that hierarchical optimal feedback using a reduced-order model can facilitate the convergence of potentials while reducing costs for collecting system information and estimating system dynamics. Moreover, our proposal has high scalability because the computational cost is much smaller than in the non-hierarchical scheme.

In the future, information networks will grow in scale and complexity, and requirements to network-based services will becomes more and more diverse. Therefore, it is essential to study managed self-organizing control methods more deeply for implementation of them in industrial and business systems. For future work, we will further research the managed control framework for self-organizing systems. For this purpose, it would be a promising direction to design a controller based on Bayesian inference.

# Bibliography

[1] G. Di Caro and M. Dorigo, “AntNet: Distributed stigmergetic control for communications networks,” *Journal of Artificial Intelligence Research*, vol. 9, pp. 317–365, Dec. 1998.

[2] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi, “Heat-seeking honeypots: design and experience,” in *Proceedings of the 20th International Conference on World Wide Web*, Mar. 2011, pp. 207–216.

[3] F. L. Lewis, “Wireless sensor networks,” *Smart environments: technologies, protocols, and applications*, pp. 11–46, 2004.

[4] I. F. Akyildiz and I. H. Kasimoglu, “Wireless sensor and actor networks: research challenges,” *Ad hoc networks*, vol. 2, no. 4, pp. 351–367, Oct. 2004.

[5] V. C. Güngör, D. Sahin, T. Kocak, S. Ergüt, C. Buccella, C. Cecati, and G. P. Hancke, “Smart grid technologies: communication technologies and standards,” *IEEE transactions on Industrial informatics*, vol. 7, no. 4, pp. 529–539, Nov. 2011.

[6] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, “M2m: From mobile to embedded internet,” *Communications Magazine, IEEE*, vol. 49, no. 4, pp. 36–43, Apr. 2011.

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[8] S. Balasubramaniam, K. Leibnitz, P. Lio, D. Botvich, and M. Murata, “Biological principles for future internet architecture design,” *IEEE Communications Magazine*, vol. 49, no. 7, pp. 44–52, Jul. 2011.

[9] Y. U. Cao, A. S. Fukunaga, and A. Kahng, “Cooperative mobile robotics: Antecedents and directions,” *Autonomous robots*, vol. 4, no. 1, pp. 7–27, Mar. 1997.

[10] F. Dressler and O. B. Akan, “A survey on bio-inspired networking,” *Computer Networks*, vol. 54, no. 6, pp. 881–900, Apr. 2010.

[11] C. Zheng and D. C. Sicker, “A survey on biologically inspired algorithms for computer networking,” *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1160–1191, Jan. 2013.

[12] Z. Zhang, K. Long, J. Wang, and F. Dressler, “On swarm intelligence inspired self-organized networking: its bionic mechanisms, designing principles and optimization approaches,” *Communications Surveys & Tutorials*, vol. 16, pp. 513–537, Jul. 2013.

[13] F. Dressler, *Self-organization in sensor and actor networks*. Wiley, Jan. 2008.

[14] J. Branke, M. Mnif, C. Muller-Schloer, and H. Prothmann, “Organic computing - addressing complexity by controlled self-organization,” in *Proceedings of IEEE ISoLA Workshop on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, Nov. 2006, pp. 185–191.

[15] H. Schmeck, C. Müller-Schloer, E. Çakar, M. Mnif, and U. Richter, “Adaptivity and self-organization in organic computing systems,” *ACM Transactions on Autonomous and Adaptive Systems*, vol. 5, no. 3, pp. 10:1–10:32, Sep. 2010. [Online]. Available: http://doi.acm.org/10.1145/1837909.1837911

[16] M. Prokopenko, *Guided self-organization: inception*. Springer, Jan. 2014.

[17] X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, *Swarm intelligence and bio-inspired computation: theory and applications*. Elsevier, May 2013.

[18] C.-M. Pintea, *Advances in bio-inspired computing for combinatorial optimization problems*. Springer, Jun. 2014.

[19] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics: Cybernetics*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[20] T. Stützle and H. H. Hoos, "MAX–MIN ant system," *Future generation computer systems*, vol. 16, no. 8, pp. 889–914, Jun. 2000.

[21] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, "Ant-based load balancing in telecommunications networks," *Adaptive behavior*, vol. 5, no. 2, pp. 169–207, Jan. 1997.

[22] M. Heissenbüttel and T. Braun, "Ants-based routing in large scale mobile ad-hoc networks," in *Proceedings of Kommunikation in Verteilten Systemen (KiVS 2003)*, Mar. 2003, pp. 91–99.

[23] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load-balancing: survey and new directions," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, vol. 33, no. 5, pp. 560–572, Sep. 2003.

[24] G. Di Caro, F. Ducatelle, and L. M. Gambardella, "AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks," *European Transactions on Telecommunications*, vol. 16, no. 5, pp. 443–455, Sep. 2005.

[25] A. Kashiwagi, I. Urabe, K. Kaneko, and T. Yomo, “Adaptive response of a gene network to environmental changes by fitness-induced attractor selection,” *PloS one*, vol. 1, no. 1, p. e49, Dec. 2006.

[26] K. Leibnitz, N. Wakamiya, and M. Murata, “Resilient multi-path routing based on a biological attractor selection scheme,” in *Biologically Inspired Approaches to Advanced Information Technology*. Springer, Jan. 2006, pp. 48–63.

[27] ——, “WSN17-3: Self-adaptive ad-hoc/sensor network routing with attractor-selection,” in *2006 IEEE Global Telecommunications Conference (GLOBECOM’06)*. IEEE, Nov. 2006, pp. 1–5.

[28] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, “Adaptive virtual network topology control based on attractor selection,” *Journal of Lightwave Technology*, vol. 28, no. 11, pp. 1720–1731, Jun. 2010.

[29] R. E. Mirollo and S. H. Strogatz, “Synchronization of pulse-coupled biological oscillators,” *SIAM Journal on Applied Mathematics*, vol. 50, no. 6, pp. 1645–1662, Dec. 1990.

[30] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, “Firefly-inspired sensor network synchronicity with realistic radio effects,” in *Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM, Nov. 2005, pp. 142–153.

[31] S. Arakawa, Y. Minami, Y. Koizumi, T. Miyamura, K. Shiomoto, and M. Murata, “A managed self-organization of virtual network topology controls in WDM-based optical networks,” *Journal of Optical Communications*, vol. 32, no. 4, pp. 233–242, Dec. 2011.

[32] D. Kominami, M. Sugano, M. Murata, and T. Hatauchi, “Energy-efficient receiver-driven wireless mesh sensor networks,” *Sensors*, vol. 11, no. 1, pp. 111–137, Dec. 2010.

[33] N. Kuze, D. Kominami, and M. Murata, "A predictive mechanism for enhancing adaptability of self-organised routing," *International Journal of Bio-Inspired Computation*, vol. 6, no. 6, pp. 384–396, Jan. 2015.

[34] N. Kuze, N. Wakamiya, and M. Murata, "Proposal and evaluation of ant-based routing with prediction," in *Proceedings of the Fifth International Workshop on Guided Self-Organization (GSO-2012)*, Sep. 2012.

[35] ——, "Proposal and evaluation of ant-based routing with autonomous zoning for convergence improvement," in *Proceedings of The 15th International Conference on Network-Based Information Systems (NBiS-2012)*, Sep. 2012, pp. 290–297.

[36] N. Kuze, N. Wakamiya, D. Kominami, and M. Murata, "Proposal and evaluation of a predictive mechanism for ant-based routing," in *Proceedings of International Conference on Emerging Network Intelligence (EMERGING 2013)*, Sep. 2013, pp. 7–12.

[37] N. Kuze, N. Wakamiya, and M. Murata, "Proposal and evaluation of ant-based routing with autonomous zoning," *Technical Report of IEICE (IN2011-196)*, vol. 111, no. 469, pp. 353–358, Mar. 2012.

[38] N. Kuze, D. Kominami, K. Kashima, T. Hashimoto, and M. Murata, "Robustness of potential-based routing with model predictive control," *Technical Report of IEICE (IN2013-195)*, vol. 113, no. 473, pp. 305–310, Mar. 2014.

[39] C. Summerfield, T. Egner, M. Greene, E. Koechlin, J. Mangels, and J. Hirsch, "Predictive codes for forthcoming perception in the frontal cortex," *Science*, vol. 314, no. 5803, pp. 1311–1314, Nov. 2006.

[40] A. K. Seth, “The cybernetic bayesian brain: From interoceptive inference to sensorimotor contingencies.” Open MIND, Jan. 2015, pp. 1–24.

[41] H.-T. Zhang, M. Z. Chen, G. B. Stan, T. Zhou, and J. M. Maciejowski, “Collective behavior coordination with predictive mechanisms,” *IEEE Circuits and Systems Magazine*, vol. 8, no. 3, pp. 67–85, Aug. 2008.

[42] N. Kuze, S. Ishikura, T. Yagi, D. Chiba, and M. Murata, “Crawler classification using ant-based clustering scheme,” in *to be presented at International Conference for Internet Technology and Secured Transactions (ICITST-2015)*, Dec. 2015.

[43] ——, “Website vulnerability scanning detection inspired by biological adaptation toward diversifying communication services,” *Computer Security Symposium 2015 (CSS 2015)*, Oct. 2015.

[44] N. Kuze, D. Kominami, K. Kashima, T. Hashimoto, and M. Murata, “Controlling large-scale self-organized networks with lightweight cost for fast adaptation to changing environments,” *to appear in ACM Transactions on Autonomous and Adaptive Systems*, 2015.

[45] ——, “Enhancing convergence with optimal feedback for controlled self-organizing network,” in *Proceedings of The 2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, Sep. 2014, pp. 1–7.

[46] ——, “Potential-based routing with optimal feedback using reduced order model for controlled self-organizing networks,” *Technical Report of IEICE (IN2014-32)*, vol. 114, no. 139, pp. 13–18, Jul. 2014.

[47] K. Zhou, J. C. Doyle, K. Glover *et al.*, *Robust and optimal control.* New Jersey: Prentice Hall, Aug. 1995.

[48] D. E. Kirk, *Optimal control theory: an introduction*. Massachusetts: Courier Corporation, Apr. 2012.

[49] A. Antoulas, D. Sorensen, and S. Gugercin, "A survey of model reduction methods for large-scale systems," *Contemporary mathematics*, vol. 280, pp. 193–219, Oct. 2006.

[50] N. Kuze, D. Kominami, K. Kashima, T. Hashimoto, and M. Murata, "Hierarchical optimal control method for controlling self-organized networks with light-weight cost," in *to be presented at IEEE Global Communications Conference (GLOBECOM 2015)*, Dec. 2015.

[51] National Science Foundation (NSF), *Global Environment for Network Innovations (GENI)*, http://www.geni.net [retrieved: August, 2013], 2007.

[52] ——, *NSF future internet architecture project (FIA)*, http://www.nets-fia.net/ [retrieved: August, 2013], 2010.

[53] European Commission, *Seventh Framework Programme (FP7)*, http://cordis.europa.eu/fp7/home_en.html [retrieved: August, 2013], 2006.

[54] National Institute of Information and Communications Technology (NICT), *New-Generation Network (NWGN) Project*, http://www.nict.go.jp/en/nrh/nwgn/ [retrieved: August, 2013], 2008.

[55] M. Meisel, V. Pappas, and L. Zhang, "A taxonomy of biologically inspired research in computer networking," *Computer Networks*, vol. 54, no. 6, pp. 901–916, Apr. 2010.

[56] X. Feng, Y. Liang, and L. Jiao, "Bio–inspired optimisation approach for data association in target tracking," *International Journal of Wireless and Mobile Computing*, vol. 6, no. 3, pp. 299–304, Aug. 2013.

[57] G.-Y. Zhang, J.-C. Zeng, and S.-D. Xue, "Research on task allocation of multi–target search with swarm robots," *International Journal of Wireless and Mobile Computing*, vol. 7, no. 3, pp. 297–304, May 2014.

[58] P. R. Montague, P. Dayan, C. Person, and T. J. Sejnowski, "Bee foraging in uncertain environments using predictive hebbian learning," *Nature*, vol. 377, no. 6551, pp. 725–728, Oct. 1995.

[59] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[60] A. Abdelaziz, R. Osama, and S. Elkhodary, "Application of ant colony optimization and harmony search algorithms to reconfiguration of radial distribution networks with distributed generations," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 86–94, Jun. 2012.

[61] H. Ahangarikiasari, M. Saraji, and M. Torabi, "Investigation of code complexity of an innovative algorithm based on ACO in weighted graph traversing and compare it to traditional ACO and bellman-ford," *Journal of Bioinformatics and Intelligent Control*, vol. 2, no. 1, pp. 73–78, Mar. 2013.

[62] M. Saleem, G. A. Di Caro, and M. Farooq, "Swarm intelligence based routing protocol for wireless sensor networks: survey and future directions," *Information Sciences*, vol. 181, no. 20, pp. 4597–4624, Oct. 2011.

[63] K. Saleem and N. Fisal, "Enhanced ant colony algorithm for self-optimized data assured routing in wireless sensor networks," in *Proceedings of 2012 18th IEEE International Conference on Networks (ICON)*. IEEE, Dec. 2012, pp. 422–427.

[64] S. Dhillon and P. Van Mieghem, "Performance analysis of the AntNet algorithm," *Computer Networks*, vol. 51, no. 8, pp. 2104–2125, Jun. 2007.

[65] L. Carvelli and G. Sebastiani, "Some issues of ACO algorithm convergence," in *Ant Colony Optimization - Method and Application*, A. Ostfeld, Ed. InTech, Feb. 2011, ch. 4.

[66] Y. Zhang, L. D. Kuhn, and M. P. J. Fromherz, "Improvements on ant routing for sensor networks," in *Ant Colony Optimization and Swarm Intelligence*. Springer, Sep. 2004, vol. 3172, pp. 154–165.

[67] A. A. Moghanjoughi, S. Khatun, B. M. Ali, and R. S. A. R. Abdullah, "QoS based fair load-balancing: Paradigm to IANRA routing algorithm for wireless networks (WNs)," in *Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008)*. IEEE, Dec. 2008, pp. 104–109.

[68] A. A. A. Radwan, T. M. Mahmoud, and E. H. Hussein, "AntNet-RSLR: A proposed ant routing protocol for MANETs," in *Proceedings of 2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)*. IEEE, Apr. 2011, pp. 1–6.

[69] K. Park and Y. Kim, "Analysis of AntNet routing scheme by using queueing model," *Computer Communications*, vol. 31, no. 13, pp. 2951–2958, Aug. 2008.

[70] D. Kominami, M. Sugano, M. Murata, and T. Hatauchi, "Controlled and self-organized routing for large-scale wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 1, pp. 13:1–13:27, Nov. 2013.

[71] A. Tyrrell and G. Auer, "Imposing a reference timing onto firefly synchronization in wireless networks," in *Proceedings of 65th IEEE Vehicular Technology Conference (VTC2007-Spring)*. IEEE, Apr. 2007, pp. 222–226.

[72] D. Canali and D. Balzarotti, “Behind the scenes of online attacks: an analysis of exploitation behaviors on the web,” in *Proceedings of 20th Annual Network & Distributed System Security Symposium (NDSS 2013)*, Feb. 2013.

[73] N. Provos and T. Holz, *Virtual honeypots: from botnet tracking to intrusion detection*. Addison-Wesley Professional, Jul. 2007.

[74] C. Grosan, A. Abraham, and M. Chis, *Swarm intelligence in data mining*. Springer Berlin Heidelberg, 2006, vol. 34.

[75] H. Azzag, N. Monmarche, M. Slimane, and G. Venturini, “AntTree: a new model for clustering with artificial ants,” in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2003)*, vol. 4, Dec. 2003, pp. 2642–2647.

[76] H. Azzag, G. Venturini, A. Oliver, and C. Guinot, “A hierarchical ant based clustering algorithm and its use in three real-world applications,” *European Journal of Operational Research*, vol. 179, no. 3, pp. 906–922, Jun. 2007.

[77] C. Kruegel, G. Vigna, and W. Robertson, “A multi-model approach to the detection of web-based attacks,” *Computer Networks*, vol. 48, no. 5, pp. 717–738, Feb. 2005.

[78] T. H. Kim, K. Kim, J. Kim, and S. J. Hong, “Profile-based web application security system with positive model selection,” in *Proceedings of the 2nd Joint Workshop on Information Security*, Aug. 2007.

[79] T. Yagi, N. Tanimoto, and T. Hariu, “Design of provider-provisioned website protection scheme against malware distribution,” *IEICE transactions on communications*, vol. 93, no. 5, pp. 1122–1130, May 2010.

[80] S. Alam, G. Dobbie, Y. S. Koh, P. Riddle, and S. U. Rehman, “Research on particle swarm optimization based clustering: A systematic review of literature and techniques,” *Swarm and Evolutionary Computation*, vol. 17, pp. 1–13, Aug. 2014.

[81] O. M. Jafar and R. Sivakumar, “Ant-based clustering algorithms: a brief survey,” *International journal of computer theory and engineering*, vol. 2, no. 5, pp. 1793–8201, Oct. 2010.

[82] A. Lioni, C. Sauwens, G. Theraulaz, and J.-L. Deneubourg, “Chain formation in oecophylla ionginoda,” *Journal of Insect Behavior*, vol. 14, no. 5, pp. 679–696, Sep. 2001.

[83] A. Liaw and M. Wiener, “Classification and regression by randomforest,” *R news*, vol. 2, no. 3, pp. 18–22, Dec. 2002.

[84] T. Yagi, N. Tanimoto, and T. Hariu, “Intelligent high-interaction web honeypots based on URL conversion scheme,” *IEICE transactions on communications*, vol. 94, no. 5, pp. 1339–1347, May 2011.

[85] T. Nelms, R. Perdisci, and M. Ahamad, “ExecScent: mining for new C&C domains in live networks with adaptive control protocol templates,” in *Proceedings of 22nd USENIX Security Symposium*, Aug. 2013, pp. 589–604.

[86] R. Xu and D. Wunsch, “Survey of clustering algorithms,” *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005.

[87] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios, “Robust optimization of large-scale systems,” *Operations research*, vol. 43, no. 2, pp. 264–281, Apr. 1995.

[88] A. Basu, A. Lin, and S. Ramanathan, “Routing using potentials: a dynamic traffic-aware routing algorithm,” in *Proceedings of the 2003 conference on Applications, technologies,*

*architectures, and protocols for computer communications.* USA: ACM, Aug. 2003, pp. 37–48.

[89] S. Jung, M. Kserawi, D. Lee, and J.-K. Rhee, "Distributed potential field based routing and autonomous load balancing for wireless mesh networks," *IEEE Communications Letters*, vol. 13, no. 6, pp. 429–431, Jun. 2009.

[90] C. Wu, R. Yuan, and H. Zhou, "A novel load balanced and lifetime maximization routing protocol in wireless sensor networks," in *Proceedings of the 67th IEEE Vehicular Technology Conference*. Singapore: IEEE, May 2008, pp. 113–117.

[91] A. Sheikhattar and M. Kalantari, "Fast convergence scheme for potential-based routing in wireless sensor networks," in *Proceedings of Wireless Communications and Networking Conference (WCNC)*. Shanghai: IEEE, Apr. 2013, pp. 1980–1985.

[92] S. Eum, Y. Shoji, M. Murata, and N. Nishinaga, "Design and implementation of icn-enabled ieee 802.11 access points as nano data centers," *Journal of Network and Computer Applications*, Aug. 2014.

[93] M. Lee, J. Song, K. Cho, S. Pack, J. Kangasharju, Y. Choi, and T. Taekyoung Kwon, "Scan: Content discovery for information-centric networking," *Computer Networks*, Oct. 2014.

[94] P. Erdős and A. Rényi, "On the strength of connectedness of a random graph," *Acta Mathematica Hungarica*, vol. 12, no. 1, pp. 261–267, Dec. 1961.

[95] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 47–97, Jan. 2002.

[96] D. J. Watts and S. H. Strogatz, "Collective dynamics of ‘ small-world ’networks," *Nature*, vol. 393, pp. 440–442, 1998.

[97] P. Gahinet and P. Apkarian, "A linear matrix inequality approach to h $\infty$ control," *International Journal of Robust and Nonlinear Control*, vol. 4, no. 4, pp. 421–448, Mar. 1994.

[98] A. Varga, "Balancing free square-root algorithm for computing singular perturbation approximations," in *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, vol. 2. Brighton: IEEE, Dec. 1991, pp. 1062–1065.

[99] T. Ishizaki, K. Kashima, J.-i. Imura, and K. Aihara, "Model reduction and clusterization of large-scale bidirectional networks," *Automatic Control, IEEE Transaction on*, vol. 59, no. 1, pp. 48–63, Jan. 2014.

[100] C. Müller-Schloer, H. Schmeck, and T. Ungerer, *Organic computing-a paradigm shift for complex systems*. Berlin: Birkhaeuser, May 2011.

[101] A. Sheikhattar and M. Kalantari, "Distributed load balancing using alternating direction method of multipliers," in *Proceedings of 2014 IEEE Global Communications Conference (GLOBECOM 2014)*. USA: IEEE, Dec. 2014, pp. 392–398.