SELECTIVELY DECENTRALIZED REINFORCEMENT LEARNING

by

Thanh Minh Nguyen

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Indianapolis, Indiana

August 2018

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Snehasis Mukhopadhyay, Chair
Department of Computer and Information Science
Dr. Mohammad Al Hasan
Department of Computer and Information Science
Dr. Shiaofen Fang
Department of Computer and Information Science
Dr. George Mohler
Department of Computer and Information Science

Approved by:
Dr. Mihran Tuceryan
Head of the Graduate Program

Grandpa, it is a proof showing that I can do more than what you expect!

ACKNOWLEDGMENTS

This thesis marks the end-point of my 9-year study at Indiana University Purdue University Indianapolis (IUPUI). Coincidently, the length of the period is the same to the length of the First Vietnam War, which was the turning point building up the military tradition in my family. After a long campaign, I have finally become a fine veteran in the Field of Science.

First, I would like to honor Dr. Snehasis Mukhopadhay, who have been mentoring me for nearly 8 years. He has been the chief architect on my success, similar to Helmuth von Moltke the Elder on the birth of modern Germany.

Second, I appreciate Dr. Fang Shiaofen, Dr. Mohammad Al Hasan and Dr. George Mohler. I definitely could not claim this victory without the excellent and dedicated advices from all of them.

Third, I name Dr. Jake Chen from University of Alabama at Birmingham and Dr. Syed Aun Muhammad from Bahauddin Zakariya University Multan Pakistan and Miss Sara Ibrahim from School of Medicine – IUPUI as trusted allies. All of them taught me Biology and allowed me to be able to transform from a pure Computer Scientist to a Translational Scientist. I also thank other junior allies from China, especially Sisi Zeng, Ni Cao, Lin Ma, Jinlei Guo and Tongbin Zhang, who gained significant achievement in career thank to my teaching. They gave me excellent ground to train my leadership skills, and some of them made some contribution in this thesis.

Finally, in every victory, I turn my heart toward my family, my home-based, where I build the soldier-like personality, and the most important allies. I am speechless whenever I think of Son Dang, my dear wife, who had been enduring the most difficult hardship with me during this 9-year campaign. If I usually joke myself as the divine strategist Zhuge Liang, then she must be my Huang Yueying. I cannot find any word to thank Dad, who never likes me saying ‘thank you’ to him, but always gave me the critical support in my toughest spots, because he are a true veteran. And, at the end of every victory, I recall Grandpa, one of the heroes in the legendary Dien Bien Phu and the First Vietnam War, because it is him who first taught me how to be an elite veteran.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Author: Nguyen, Thanh, Minh. PhD
Institution: Purdue University
Degree Received: August 2018
Title: Selectively Decentralized Reinforcement Learning
Major Professor: Snehasis Mukhopadhyay

The main contributions in this thesis include the selectively decentralized method in solving multi-agent reinforcement learning problems and the discretized Markov-decision-process (MDP) algorithm to compute the sub-optimal learning policy in completely unknown learning and control problems. These contributions tackle several challenges in multi-agent reinforcement learning: the unknown and dynamic nature of the learning environment, the difficulty in computing the closed-form solution of the learning problem, the slow learning performance in large-scale systems, and the questions of how/when/to whom the learning agents should communicate among themselves. Through this thesis, the selectively decentralized method, which evaluates all of the possible communicative strategies, not only increases the learning speed, achieves better learning goals but also could learn the communicative policy for each learning agent. Compared to the other state-of-the-art approaches, this thesis's contributions offer two advantages. First, the selectively decentralized method could incorporate a wide range of well-known algorithms, including the discretized MDP, in single-agent reinforcement learning; meanwhile, the state-of-the-art approaches usually could be applied for one class of algorithms. Second, the discretized MDP algorithm could compute the sub-optimal learning policy when the environment is described in general nonlinear format; meanwhile, the other state-of-the-art approaches often assume that the environment is in limited format, particularly in feedback-linearization form. This thesis also discusses several alternative approaches for multi-agent learning, including Multidisciplinary Optimization. In addition, this thesis shows how the selectively decentralized method could successfully solve several real-worlds problems, particularly in mechanical and biological systems.

1. INTRODUCTION

This thesis is devoted to reinforcement learning, which is one of the most attractive areas in Computer Science, and probably in Philosophy. In this thesis, I propose, explore, and discuss selective decentralization as a new approach in decentralized reinforcement learning, which is also cited in the literature as distributed learning or multi-agent learning. Briefly, this thesis answers the questions: how the computational agent learn how to act optimally when it does not know the learning environment, and how multiple agents collaborate to learn faster in large-scale learning problems. Here, the notion of 'large' is problem-specific instead of just number of dimensions. For example, in aircraft smart control, a system of more than ten state parameter would be considered as large, while ten is still a small number in big data area.

Let us begin with an infamous Japanese cartoon named 'Doraemon', a science-fiction gadget cat who has been motivating many Asian children to love and pursue robotic and artificial intelligence career [1]. In the series of the same name, Doraemon is a robot coming from the 22nd century to take care of Nobita, a weak, under-performed and somewhat lazy child. Doraemon is extremely fond of Japanese rice cake; therefore, the child Nobita usually offer Doraemon cakes to reward his service or to ask for help. In section titled 'Scary rice cake' [2], again, the lazy Nobita shows Doraemon many rice cakes; however, the condition is that Doraemon must complete Nobita's 60 homework on the last night of the due date. Given such a large amount of work in short time and unfamiliarity with primary school homework, Doraemon knows that he needs extra help. Then, he uses his special ability to travel across the time to ask his copies living in several hours later in the future to come back to his time and help. The multiple Doraemons divide the work among themselves, try to learn and solve primary school homework, and exchange opinions on right or

wrong solutions. Will the Doreamons succeed in this heavy task to enjoy the rice cake or will they fail and fighting each other? At the end, both of these outcomes occur: the gadget cats do not agree many times during the work, sometimes fighting each other, but they manage to finish the work and enjoy the cakes.

The short story above demonstrate many points in this thesis. First, the gadget cats have a mission in which he does not know the domain knowledge; therefore, they must learn and act concurrently. Second, his task is to provide homework solutions but there is no precise feedback telling him whether or not his solutions are right or wrong. These two points are essential in reinforcement learning. Third, the task is too large to complete by one gadget; therefore, the task needs the distributed execution. Forth, the outcome of the task depends on the communication and collaboration among the Doreamon copies, which is also the essential element in decentralized learning.

From these points above, the remaining of the introduction will be divided into three sections. The first section reviews the concepts and state-of-the-art techniques in reinforcement learning. The second section reviews decentralized learning and multi-agent learning systems. The third section briefly reviews Hamilton-Jacobi-Bellman equation, which is the universal mathematical description in many reinforcement learning problems.

1.1 Reinforcement Learning: overview

Reinforcement learning, briefly, is how the agents learn what to do when there is no instruction telling what the agent should do [3]. The lack of instruction could be in many scenarios. First, when the problem is naturally exploratory, there is usually neither guidance nor domain knowledge

for the agent to make the decision. This usually happens when the problem is relatively new for human. A typical example in this scenario was fighting Ozma boss in Final Fantasy IX game; this super boss took hundreds or thousands of players to play multiple rounds to discover the winning strategy [4] in 2000. In another scenario, the learning outome could only be seen after so many phases that the 'good' action in an early moment may fail later, or a 'bad' action in an early moment could bring favorable outcome. This scenario often occurs in chess game or searching missions. An example of this scenario is The Game of the Century chess, where Bobby Fischer made a queen scarification move, which appeared to be a suicide at turn 17, but at the end claimed the victory [5] at turn 38. In addition, when the environment is known to be uncertain, there is no guarantee of optimality in every action. We see this scenario when Markov decision process [6] is applied.

In general, reinforcement learning is about trial-and-error interactions with a dynamic environment [7]. The reinforcement learning problem includes the following elements

- The agent who can perceive and perform action $\mathbf{u} \in \boldsymbol{A}$, where $\boldsymbol{A}$ is the set of all actions.

- The environment where the agent operates in. The agent can perceive information from the environment, which is also called state **x**. In this thesis, we assume that the environment is fully accessible for the agent, which means the agent can obtain complete, accurate and up-to-date information from the environment [3].

- The dynamic transition of the environment: the mapping $\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t))$, in which $\mathbf{x}(t+1)$ is the new state, $\mathbf{x}(t)$ is the exiting state and $\mathbf{u}(t)$ is the agent's action at time t. In reinforcement learning, the agent initially does not know the transition.

- The reinforcement feedback $R(\mathbf{x}(t), \mathbf{u}(t))$, which the agent knows in order to determine how well it performs at time t. This is the agent's learning goal.

The reinforcement learning agent, also called the intelligent agent, aims to maximize the cumulative feedback $R(\mathbf{x}(t), \mathbf{u}(t))$ for the long time. For this objective, the agent tries to find the policy function $\mathbf{u} = v(\mathbf{x})$, which tells the agent which action it should perform in a given state.

In general, the reinforcement learning techniques need to solve two major challenges to optimize the reinforcement feedback: the unknown nature of the environment and computing the action **u** as a function of state **x**. For the first challenges, system identification has been widely used to approximate the dynamic environment [8-11]. For the second challenge, as it is claimed that reinforcement learning is the direct adaptive optimal control problem [12], in theory, solving a reinforcement learning problem is solving the equivalent Hamilton-Jacobi-Bellman (HJB) equation, which is the fundamental of optimal control [13]. However, the closed-form solution of the HJB equation is very difficult to find in general. In the special case of linear system, the HJB equation becomes the well-known Riccati equation with the complete closed-form solution [14]. In nonlinear systems, researchers have been focusing on approximation methods to tackle nonlinear HJB equation problem such as [15-18]. However, these methods are only limited for the systems in feedback-linearizable forms.

Reinforcement learning techniques could be categorized based on how the techniques compute the action **u**. In one category, the reinforcement learning techniques, inspired by the theory of dynamic programming and HJB equation [19-21], try to estimate the expected long-time reward at every state **x**, also called state utility, given the most updated policy the agent has. After executing each action and seeing new reward, the agent updates the estimation of state utility, and updates its policy to maximize the chance of getting higher utility based on the new estimation. In this

category, the agent usually needs to employ the dynamic transition of the environment for its state utility estimation. Therefore, this category is called model-based methods [22-24]. In another category, the learning agent keeps a history of reward for all state-action pairs and increments the history of reward after executing each action in a given state. In this type of approach, the agent does not need the transition of the environment since the update is only done incrementally on a short term. Therefore, this category is called model-free methods, in which Q-learning is a typical example [25-27]. Another category of reinforcement learning approaches is called reflex design, in which the agent directly represent policy function in some form, especially in differentiable form. After seeing the new reward, the agent directly adjusts the parameters of the policy function, usually by the gradient methods [28, 29]. Overall, only the model-based category could directly estimate the solution for the HJB equation; while the other two categories do not.

From another perspective, reinforcement learning techniques need to balance the trade-off between exploration and exploitation [3]. Exploration refers to the amount of learning samples that the agent should experience to find the optimal policy. Exploitation refers to the maximization of reward at any instance. Due to unknown nature of reinforcement learning, maximization of the short-term reward with insufficient knowledge may not lead to the optimal policy in long term. A typical example of this trade-off could be found in the multi-bandit-arm problem [30]. Similar to the multi-bandit-arm problem, most of the reinforcement learning algorithm has a learning rate parameters to control how fast the learning is. In addition, some reinforcement learning techniques apply the Boltzmann distribution, which directs how likely the agent chooses alternative options rather than the most updated policy as it knows [31-33].

1.2 Decentralized reinforcement learning

As reinforcement learning has been applied in more complex real-world problems, the interest in multi-agent, also called distributed or decentralized, learning has been increasing for the recent years. There are many scenarios in which single-agent, or centralized learning, is not suitable. First, when the learning problem is large, especially when the environment is represented by high-dimensional vectors, the single agent may not have sufficient computational power to solve such a problem. In addition, even if the single/centralized agent have sufficient computational power, it still has to tackle the curse of dimensionality [34, 35], which may obstruct the agent learning optimal policy, or the slow exploration [36], which makes the learning impractical. In these scenarios, decentralized learning may be a more practical approach. Second, when the learning problem is naturally distributed, such as coordinating multiple players in the infamous Civilization game series [37] where each artificial agent may control one nation collaborating-competing with the others in the world-map, it may be more suitable to apply multi-agent learning. Third, even when the problem is neither high-dimension nor naturally distributed but composed from multiple disciplinary, it is more feasible to apply distributed learning, in which each learning agent is an expert in single disciplinary. One typical example of this problem in aircraft control, where the aero dynamic agent applies fluid dynamics law to manage the air-pressure on the aircraft wing and the structure agent applies the material law to manage the deflection and shape of the wing [38].

It has been shown that decentralized reinforcement learning shows several advantages, compared to centralized learning. First, as expected, decentralized learning allows the agents to operate on less dimension, which could avoid the curse of dimensionality an improve the learning speed [39]. Second, decentralized learning offers more robustness and improves fault-tolerance: if one agent fails in learning, the other agents could compensate for it in the overall learning problem resulting

in only graceful degradation of performance. Third, decentralized learning is less susceptible to uncertain system parameters [40].

In order to avoid instability and show these advantages, the decentralized reinforcement learning has to overcome four major challenges. First, when the agents should communicate with the others? Second, to whom should a single agent communicate? Third, which types of information should the single agent share with the others? Forth, how does a single agent use the information to improve its own learning performance and contribute to the overall performance? Adding to the complexity of decentralized learning, the answers for these questions must consider the learning environment: collaborative (when all of the agents work together to achieve a common goal), competitive (when the agents compete with the others to decide a winner), or mixed (including both collaboration and competition). Thus, the 'communication' questions are still opened. More details about these questions could be found in [41-43].

To the extent of our knowledge, the state-of-the-art decentralized learning could be divided into two categories: partial communication and multi-model switching. In partial communication, each agent is responsible to for its own communication: when, to whom, which to share and how to use information, depending on the agent's state variables and communication costs [44]. Some of the recent state-of-the-art techniques in partial communication demonstrate how each agent decide the communication in Q-learning problems [45-47], partially ordered subsystems [48], fuzzy logic systems [49, 50] and probabilistic control sharing systems [51]. In multi-model-switching, the entire system has K policies to allow the agents to communicate, and the entire system has a central communicator who is responsible to switch the communication policy depending on the resulting

performance [52-55]. Also, criteria to decide policy switch may depends on the domain-specific optimization of the problem, such as power efficiency function in energy system [56, 57] and aerodynamic performance in hypersonic vehicle systems [58]. Our thesis focus on the multi-model-switching approach, whose complexity is, at most, as of Bell's number [59]. It is known that the Bell's number grows more than exponentially.

1.3 Principles of Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman (HJB) equation, proposed by Richard Bellman in 1950s [60], is the general theory for optimal control and reinforcement learning. The equation includes two fundamental elements: a cost function $C(\mathbf{x}, \mathbf{u})$ (the reinforcement feedback), which forms the long-term optimization objective, and the system function $F(\mathbf{x}, \mathbf{u})$, which forms the thresholds of the optimization. The continuous HJB equation [60] is as follow: find $\mathbf{u}(t)$ as a function of $(\mathbf{x}(t))$ to minimize

$$J = \int_0^T C(\mathbf{x}(t), \mathbf{u}(t) + D(\mathbf{x}(T)) \quad (1.1)$$

In which T is the period which the optimization applies and $D(\mathbf{x}(T))$ is the specific cost function at the end of the period. When $T \to \infty$, the optimization criteria becomes

$$J = \int_0^\infty C(\mathbf{x}(t), \mathbf{u}(t) \quad (1.2)$$

The optimization constrains is represented as

$$\mathbf{x}'(t) = F(\mathbf{x}(t), \mathbf{u}(t)) \quad (1.3)$$

in which $\mathbf{x}'(t)$ stands for the first derivative of $\mathbf{x}$. The HJB equation in discrete-time form is: minimize

$$J = \sum_{t=0}^{T} C\big(\mathbf{x}(t), \mathbf{u}(t)\big) + D\big(\mathbf{x}(T)\big) \quad (1.4)$$

or

$$J = \sum_{t=0}^{\infty} C\big(\mathbf{x}(t), \mathbf{u}(t)\big) \quad (1.5)$$

when $T \to \infty$. The constrains is

$$\mathbf{x}(t+1) = F(\mathbf{x}(t), \mathbf{u}(t)) \quad (1.6)$$

In the special case of linear system, the HJB equation becomes the well-known Riccati equation [61]: minimizing

$$J = \sum_{0}^{\infty} \mathbf{x}(t)^{\boldsymbol{T}}\mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^{T}\mathbf{R}\mathbf{u}(t) \quad (1.7)$$

subject to

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad (1.8)$$

where **Q** and **R** are semi-positive-definite square matrixes.

The most important principle of the HJB equation is the dynamic optimization, which includes two characteristics. First, the solution $\mathbf{u}(t)$ for equation starting at time t ($t = 0$) should be also the solution at time $t+1$ ($t = 1$) [13]. Therefore, in discrete-time system with finite T , the HJB equation could be solved by inverse dynamic programming [62]: use the solution at time t to find the solution at t-1. Second, due to the inter-dependence of $\mathbf{u}(t)$ and the optimization objective J, these two factor could be iteratively estimated [19].

Theoretically, the solution of the HJB equation follows calculus of variation [63]. For the linear system, the HJB equation becomes the well-known Riccati equation with complete solution [14]. However, in most of the real-world cases, the system is nonlinear where the closed-form solution for HJB equation is very difficult to find. Therefore, researchers have been focusing on approximation methods to tackle nonlinear HJB equation problem such as [15-18]. Generally,

these efforts focus on the nonlinear feedback-linearization system, in which the closed-form solution for the approximation of HJB equation has been found [64]. In the other hand , a simple idea is to discretize the nonlinear system to convert it into a Markov-Decision-Process (MDP) and solve it by the policy iteration algorithm [65]. Such discretization of continuous-state nonlinear control systems has been examined in [66-68]. However, according to the best of our knowledge, the sufficient and the closeness of the MDP approximation to the real solution of the HJB equation has not been well-established.

Overall, the introduction highlights two major areas which will be answered by this thesis:

- How to approximate the solution of the HJB equation by the MDP approach? What are the necessary condition for the MDP's approximated solution to approach the HJB equation's solution? How closed these two solutions are?

- The communication among the agents: when? To whom? What to share? And how to use the sharing information?

The remaining chapters of this thesis are organized as follow. In chapter 2, I show the grid-based discretization method to convert the HJB equation into a MDP problem. In addition, I prove some necessary condition for the MDP's solution toward the HJB's solution. In chapter 3, I propose the selective decentralization approach for the communication among the agent. In chapter 4, I show that the selective decentralization approach outperforms the centralized approach in Q-learning, which is a typical model-free reinforcement learning technique. In chapter 5, I show the performance of selective decentralization in system identification, which is the first sub-problem in model-based reinforcement learning. In chapter 6, I show that the selective decentralization also improves the learning performance, in combination with the MDP approximation for HJB

equation. In chapter 7, I explore the capability of multidisciplinary optimization (MDO), which is a popular technique in aircraft control to coordinate multiple subsystems, in reinforcement learning. In chapter 8, I add the system noise into the learning problems and see how the learning techniques above would perform in the noisy cases. Chapter 9 demonstrates how the proposed selective decentralization improves the learning performance in real-world problems of mechanics and system biology. Chapter 10 concludes the thesis on how well it tackles the major challenges in general and decentralized reinforcement learning.

2. MARKOV DECISION PROCESS

As mentioned in the introduction, the HJB equation's solution has the dynamic programming principles, which is also shared by the Markov Decision Process's (MDP) solution [69]. Therefore, a simple idea is to discretize the nonlinear system to convert it into a Markov-Decision-Process (MDP) and solved it by well-known MDP algorithms [70]. Such discretization of continuous-state nonlinear control systems have been studied in [66-68]. In addition, convergence results for decentralized learning in Markov systems have been derived in [71, 72]. However, from our knowledge, the theoretical proof about the existence and closeness of the MDP's solution in the general form HJB equation has not been widely explored.

The main challenge in approximating the HJB equation's solution by MDP is the transformation from the HJB's continuity to MDP's discreteness. Therefore, the proof of MDP's existence and closeness toward the HJB depends on how to discretize the HJB equation. From this argument, this chapter is organized as followed. First, I briefly review the MDP problem. Second, I describe the problem statement of the HJB equation in adaptive control, which has be claimed to be equivalent to reinforcement learning [12]. Third, I demonstrate the discretization to transform the continuous HJB equation toward the discrete MDP problem, in which I define the 'discretized resolution' concept (further details could be found in publications [78, 85]). Forth, I prove some sufficient conditions for the discrete MDP's states to approach the discrete the continuous HJB's states in the long-term, which guarantee the closeness of these two problems toward each other [78]. Fifth, I prove the existence of the MDP's solution which can be used as a 'stabilizer' for the HJB – adaptive control equation. Finally, I show some simulation experiments in which the MDP's solution could stabilize the system as the HJB equation aims for [78].

2.1 Overview of the MDP problem

Formally, an MDP problem, which is discrete-time, has the following elements [73]

- A set of discrete states $\boldsymbol{S}$, which is assumed to be fully accessible for the learning agent.

- A set of actions $\boldsymbol{A}$, which the agent could do at every time step.

- The state transition distributions $\boldsymbol{P}$, also written $\boldsymbol{P}(\mathbf{x}_2 \mid \mathbf{x}_1, \mathbf{u})$ in which $\mathbf{x}_1, \mathbf{x}_2 \in \boldsymbol{S}$ and $\mathbf{u} \in \boldsymbol{A}$, tell how likely to reach a new state next time when executing a specific action at a specific state.

- A reward function R: $\boldsymbol{S} \to \Re$ or $\boldsymbol{S}\times\boldsymbol{A} \to \Re$ to tell how good a specific state (or state-action) is instantly. This is the reinforcement feedback for the agent.

- A discount factor $\gamma < 1$ to define the long-term learning objective of the agent. The closer γ to 1, the more long-term the agent needs to aim for.

The learning objective is

$$\sum_{t=0}^{\infty} \gamma^t R(\mathbf{x}(t)) \quad (2.1)$$

and the agent needs to find a policy function $\mathbf{u} = v(\mathbf{x})$ to optimize (2.1).

Compared to the HJB equation, we can see that most of the elements in the MDP problem are the same to the corresponding ones in the discrete-time HJB equation, except the state transition. In the HJB equation, the state transition is deterministic

$$\mathbf{x}_2 = f(\mathbf{x}_1, \mathbf{u}) \quad (2.2)$$

Meaning that at state $\mathbf{x}_1$, executing action $\mathbf{u}$ only lead to one specific state $\mathbf{x}_2$. In the other hands, the MDP is non-deterministic, which means executing the same action $\mathbf{u}$ at state $\mathbf{x}_1$ may lead to other states rather than $\mathbf{x}_2$. In addition, it is easy to see that if the MDP state-transition is less diverge: $\boldsymbol{P}(\mathbf{x}_2 \mid \mathbf{x}_1, \mathbf{u}) \to 1$, then the discrete-state and non-deterministic MDP behaves closer to the continuous and deterministic HJB equation.

From the dynamic programming perspective, the MDP problem could be solved by policy iteration algorithm [65]. The main idea is to maintain an estimation of the learning objective, called utility function, and iteratively update the policy / utility function. To be more specific, given an arbitrary policy $\mathbf{u} = v'(\mathbf{x})$, we can always compute the utility function $V_{v'}(\mathbf{x})$ for every initial state $\mathbf{x}$

$$V_{v'}(x) = \sum_{t=0}^{\infty} \gamma^t R\big(\mathbf{x}(t)\big) \ , \mathbf{x}(0) = \mathbf{x}$$

$$\text{or } V_{v'}(x) = \sum_{t=0}^{\infty} \gamma^t R\big(\mathbf{x}(t), \mathbf{u}(t)\big) \ , \mathbf{x}(0) = \mathbf{x} \quad (2.3)$$

depending on how the function R is defined. Then, at time t, the agent may review all of the possible actions and choose the action maximizing the expected next utility

$$\underset{\mathbf{u}}{\text{argmax}} \left(R(\mathbf{x}(t)) + \sum_{\forall \mathbf{x}'} \boldsymbol{P}(\mathbf{x}'|\mathbf{x}(t), \mathbf{u})\, V_{v'}(\mathbf{x}') \right) \quad (2.4)$$

Equation (2.4) shows that the policy is updated. Then, the new policy in (2.4) is used to re-compute (2.3). Theoretically, the iterative processes (2.3-2.4) has been proved to converge [71, 72], where both the policy no longer change in (2.4) and the state utility reaches the maximum.

2.2 Problem statement: HJB equation to optimally stabilize the system

In this thesis, we focus on discrete time, continuous-state, time-invariant system in the general format [78, 85]

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (2.5)$$

Where $\mathbf{x} \in \Re^N$ stands for the N dimensional bounded state vector, $\mathbf{u} \in \Re^M$ stands for the M dimensional bounded control unit, t stands for the iteration number, $\mathbf{x}(0)$ is given and $f: \Re^N \times \Re^M \to \Re^N$ is an continuously differentiable unknown function. Here, the symmetric boundaries $[-\chi, \chi]$ and $[-\mu, \mu]$ for all components of $\mathbf{x}$ and $\mathbf{u}$ are known. Let $p: \Re^N \to \Re$ and $q: \Re^M \to \Re$ be the two

continuously semi-definite negative and differentiable reward functions with the following properties

$$p(\mathbf{x}_1) \leq p(\mathbf{x}_2) \Leftrightarrow \|\mathbf{x}_1\| \geq \|\mathbf{x}_2\| \text{ and } p(\mathbf{0}) = 0 \quad (2.6)$$

$$q(\mathbf{u}_1) \leq q(\mathbf{u}_2) \Leftrightarrow \|\mathbf{u}_1\| \geq \|\mathbf{u}_2\| \text{ and } q(\mathbf{0}) = 0 \quad (2.7)$$

where $\|\mathbf{x}\|$ denotes the second norm of $\mathbf{x}$. The main objective is to learn the control unit $\mathbf{u}$ such that

$$\mathbf{x}(t) \to 0, \mathbf{u}(t) \to 0 \text{ as } t \to \infty \quad (2.8)$$

To formulate an optimal control or learning problem, we convert the objective in (2.8) into a more formal control problem with discount factor $0 < \gamma \to 1$

$$J(\mathbf{x}_0) = \sum_{t=0}^{\infty} \gamma^t \left(p\big(\mathbf{x}(t)\big) + q\big(\mathbf{u}(t)\big)\right) \quad (2.9)$$

It is easy to see that $p(\mathbf{x})$ and $q(\mathbf{u})$ is designed such that the learning objective is optimal only when the system is stabilized.

2.3 Discretization to setup the MDP problem

2.3.1 Discretizing the state and control vector space

Let G be the number of intervals in each dimension of $\mathbf{x}$ and $\mathbf{u}$ for which we uniformly divide the dimension into small grids. Therefore, the entire state space is divided into G^N small hyper cubes and the control space is divided into G^M small hyper cubes. All points inside a hyper cube are discretely represented by the center of the hyper cube. Points on the borders between two hyper cubes are represented by the center of the 'left' hypercube. Mathematically, the discretization process is described by the following formulas

$$\mathbf{x}[i] \to \theta_x + \chi/G \quad \forall i \in [1, N] \;\; and \; \mathbf{x}[i] \in [\theta_x, \theta_x + 2\,\chi/G) \quad (2.10)$$

$$\mathbf{u}[i] \to \theta_u + \mu/G \quad \forall i \in [1, M] \;\; and \; \mathbf{u}[i] \in [\theta_u, \theta_u + 2\,\mu/G) \quad (2.11)$$

where $\theta_x \in \{-\chi, -\chi + 2\chi/G, -\chi + 4\chi/G, ..., \chi - 2\chi/G\}$ and $\theta_u \in \{-\mu, -\mu + 2\mu/G, -\mu + 4\mu/G, ..., \mu - 2\mu/G\}$, which are the 'left' boundaries in the hyper cubes.

Let $\delta = \max(2\chi/G, 2\mu/G)$. It is easy to see that inside each small hyper cube, the largest distance between any two points, or the 'main diagonal', is bounded by

$$\sqrt{\delta^2 + \delta^2 + \cdots + \delta^2} = \sqrt{N\delta^2} = \sqrt{N}\delta \quad (2.12)$$

in the state space and by $\sqrt{M}\delta$ in the control space. The left side of (2.12) has N terms for **x** dimension or M terms for **u** dimension. Trivially, $G \to \infty \Leftrightarrow \delta \to 0$, or the discretization is more precise.

From this point, for any state vector **x**, we denote $\mathbf{x}_{dis}$ as **x**'s discretized form; for any control vector **u**, we denote $\mathbf{u}_{dis}$ as **u**'s discretized form. We also denote ($\mathbf{x}_{dis}$) and ($\mathbf{u}_{dis}$) as the hypercube where every **x**'s and **u**'s discretization is $\mathbf{x}_{dis}$ and $\mathbf{u}_{dis}$, correspondingly. Formally, from (2.10) and (2.11), we have

$$(\mathbf{x}_{\text{dis}}) = [\mathbf{x}_{\text{dis}}(i) - \chi/G\,, \mathbf{x}_{\text{dis}}(i) + \chi/G)\,]\ \ \forall i \in \{1,2,3 \ldots N\} \quad (2.13)$$

and

$$(\mathbf{u}_{\text{dis}}) = [\mathbf{u}_{\text{dis}}(i) - \mu/G\,, \mathbf{u}_{\text{dis}}(i) + \mu/G)\,]\ \ \forall i \in \{1,2,3 \ldots M\} \quad (2.14)$$

2.3.2 Setting up the state transition matrix for the MDP problem

The state transition matrix for the MDP problem, which contains all conditional probability $\boldsymbol{P}(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$, has the dimension of $G^N \times G^M \times G^N$. It is easy to observe that for each triple ($\mathbf{x'}_{dis}$, $\mathbf{x}_{dis}$, $\mathbf{u}_{dis}$) the conditional probability $\boldsymbol{P}(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$ is

$$\boldsymbol{P}(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis}) = \frac{\iiint_{(\mathbf{x}_{\text{dis}})\times(\mathbf{u}_{\text{dis}})\times(\mathbf{x'}_{\text{dis}})} d\mathbf{x}d\mathbf{u}d\mathbf{x'}}{\iiint_{(\mathbf{x}_{\text{dis}})\times(\mathbf{u}_{\text{dis}})\times \mathrm{C}} d\mathbf{x}d\mathbf{u}d\mathbf{x'}} \quad (2.15)$$

where C is the subspace containing all possible value of $f(\mathbf{x}, \mathbf{u})$ $\forall \mathbf{x},\mathbf{u} \in (\mathbf{x}_{dis}) \times (\mathbf{u}_{dis})$. In our problem statement, since f is unknown, we replace f by $\hat{f}$, which is approximated by the neural network. Figure 2.1 illustrates a simple case of this conditional probability when $N = 1$. Although the integral could be approximated by the Monte Carlo method [74], the simpler method to approximate $\boldsymbol{P}(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$ is as follow.

- Generate a large number of S points $(\mathbf{x}, \mathbf{u})$ following the uniform distribution in $(\mathbf{x}_{dis}) \times (\mathbf{u}_{dis})$. Here, we emphasize that the computation of $\boldsymbol{P}(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$ does not use any sample $(\mathbf{x}(t), \mathbf{u}(t))$. These S points are randomly generated without any prior knowledge of f to avoid bias.

- Count the number of points T such that $\hat{f}(\mathbf{x}, \mathbf{u}) \in (\mathbf{x'}_{dis})$.

- Then $T/S \rightarrow \boldsymbol{P}(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$ when $S \rightarrow \infty$.

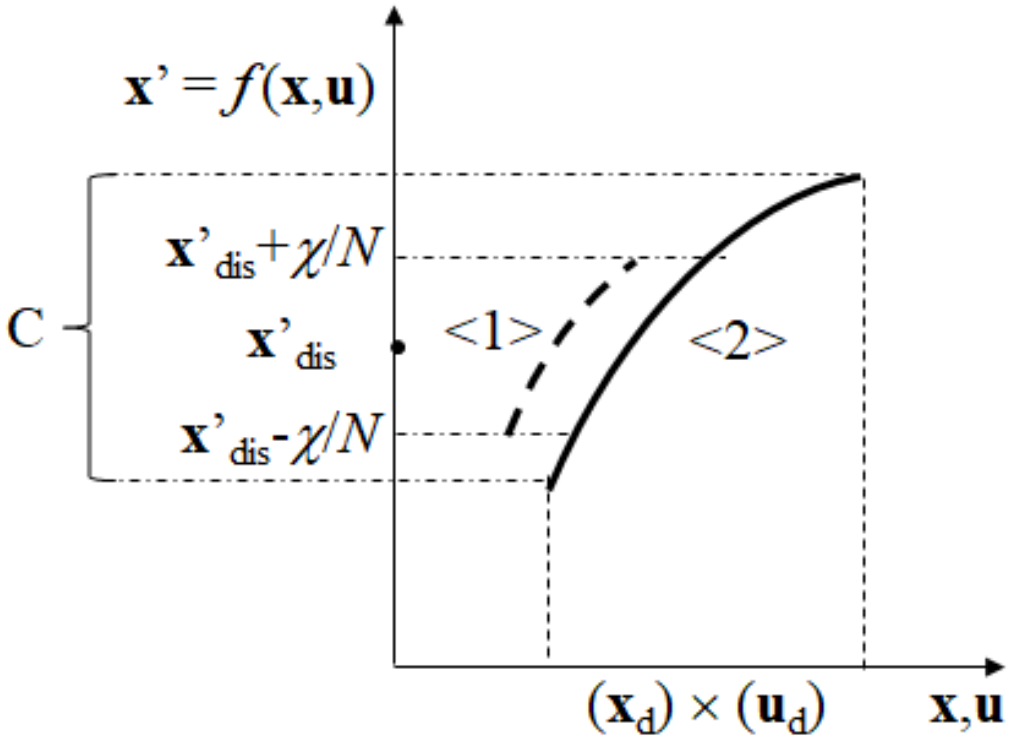

Figure 2.1. An example of (2.15) in one-dimension state space. <1>, the dash surface, is the numerator in (2.15). <2>, the bold surface, is the denominator of (2.15).

2.3.3 State value function in MDP problem

In (2.9), from Bellman's principle of optimality [75], for the solution **u**(t) of the HJB equation (2.5)-(2.9), we have

$$J(\mathbf{x}(t)) = p(\mathbf{x}(t)) + q(\mathbf{u}(t)) + \sum_{\tau=1}^{\infty} \gamma^t \left(p(\mathbf{x}(\tau)) + q(\mathbf{u}(\tau)) \right)$$

$$= p(\mathbf{x}(t)) + q(\mathbf{u}(t)) + J(\mathbf{x}(t+1)) \quad (2.16)$$

Because f is stable at the origin, from (2.2) and (2.3), $J(\mathbf{0}) = 0$. Since the state value function in the HJB equation (2.5)-(2.9) contains a discount factor, we define the corresponding value function in the MDP as

$$R(\mathbf{x}_{\text{dis}}(t)) = p(\mathbf{x}_{\text{dis}}(t)) + q(\mathbf{u}_{\text{dis}}(t))$$

$$+ \sum_{\tau=1}^{\infty} \gamma^t \sum_{\forall \mathbf{x}_{\text{dis}}'} P(\mathbf{x}_{\text{dis}}'(\tau) | \mathbf{x}_{\text{dis}}(\tau-1), \mathbf{u}_{\text{dis}}(\tau-1)) \left(p(\mathbf{x}_{\text{dis}}'(\tau)) + q(\mathbf{u}_{\text{dis}}'(\tau)) \right)$$

$$= p(\mathbf{x}_{\text{dis}}(t)) + q(\mathbf{u}_{\text{dis}}(t)) + \gamma \sum_{\forall \mathbf{x}_{\text{dis}}'} P(\mathbf{x}_{\text{dis}}'(t+1) | \mathbf{x}_{\text{dis}}(t), \mathbf{u}_{\text{dis}}(t)) R(\mathbf{x}'_{\text{dis}}(t+1)) \quad (2.17)$$

And $R(\mathbf{x}_{\text{dis}}) = 0$ if ($\mathbf{x}_{\text{dis}}$) contains **0** or has **0** on the boundary.

2.4 The closeness of the MDP's states toward the HJB's states

In this section, we examine several conditions for $\mathbf{x}_{\text{dis}}(t)$ and $\mathbf{u}_{\text{dis}}(t)$, acquired by the discretized MDP method, converge to $\mathbf{x}(t)$ and $\mathbf{u}(t)$ when $t \to \infty$. More specifically, we answer the following questions. First, suppose that we know an admissible control $\mathbf{u}(t) = g(\mathbf{x}(t))$ and discretize this admissible control (without the MDP policy iteration algorithm), what is the boundary of $|\mathbf{x}(t) - \mathbf{x}_{\text{dis}}(t)|$? Second, without any knowledge of the admissible control, in which condition the MDP solution could near-optimally stabilize the system? To simplify the analysis, in this section, we assume that f is known. Although this assumption is not applicable for reinforcement learning,

this assumption is logical given that the neural network, as the functional approximator $\hat{f}$, could approximate any arbitrary function given sufficient training sample [15, 76, 77]. The main content of the theoretical analysis is taken from my paper [78].

2.4.1 The autonomous system

When we linearize an autonomous system using Taylor series expansion

$$\mathbf{x}(t+1) = f(\mathbf{x}(t)) \quad (2.18)$$

at point **p** in the domain of *f*, we have

$$f(\mathbf{x}) \approx f(\mathbf{p}) + \mathbf{M}(\mathbf{x}-\mathbf{p}) \quad (2.19)$$

where **M** is the matrix of partial derivative of *f* on **x** at **p**

$$\mathbf{M} = \begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{\mathbf{x}=\mathbf{p}} & \left.\frac{\partial f_1}{\partial x_2}\right|_{\mathbf{x}=\mathbf{p}} & \cdots & \left.\frac{\partial f_1}{\partial x_d}\right|_{\mathbf{x}=\mathbf{p}} \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{\mathbf{x}=\mathbf{p}} & \left.\frac{\partial f_2}{\partial x_2}\right|_{\mathbf{x}=\mathbf{p}} & \cdots & \left.\frac{\partial f_2}{\partial x_d}\right|_{\mathbf{x}=\mathbf{p}} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial f_d}{\partial x_1}\right|_{\mathbf{x}=\mathbf{p}} & \left.\frac{\partial f_d}{\partial x_2}\right|_{\mathbf{x}=\mathbf{p}} & \cdots & \left.\frac{\partial f_d}{\partial x_d}\right|_{\mathbf{x}=\mathbf{p}} \end{bmatrix} \quad (2.20)$$

For the close region ($\mathbf{x}_{dis}$) (2.13) which includes the boundary and contains **x**, let C_η be the set of all $\mathbf{x}(t+\eta)$ computed by tracking all points in ($\mathbf{x}_{dis}$) on *f* after η time points. Obviously C_η has to be a close region because it is spanned from a close region by a continuous function. Therefore, there exists two points $\mathbf{x}_1(t+\eta)$ and $\mathbf{x}_2(t+\eta)$ such that $|\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta)|$ is the maximum for all pairs of points in C_η. There must exist two chains: { $\mathbf{x}_1(t)$, $\mathbf{x}_1(t+1)$, **...,** $\mathbf{x}_1(t+\eta-1)$ } and { $\mathbf{x}_2(t)$, $\mathbf{x}_2(t+1)$, **...,** $\mathbf{x}_2(t+\eta-1)$ } such that $\mathbf{x}_1(t+\eta) = f(\mathbf{x}_1(t+\eta-1)) = f(f(\mathbf{x}_1(t+\eta-2))) = ... = f^{\eta}(\mathbf{x}_1(t))$ and $\mathbf{x}_2(t+\eta) = f(\mathbf{x}_2(t+\eta-1)) = f(f(\mathbf{x}_2(t+\eta-2))) = ... = f^{\eta}(\mathbf{x}_2(t))$. Applying the Taylor series expansion, we have

$$\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta) = f^{\eta}(\mathbf{x}_1(t)) - f^{\eta}(\mathbf{x}_2(t)) = \frac{\partial f^{\eta}}{\partial(\mathbf{x}_2(t))}(\mathbf{x}_1(t) - \mathbf{x}_2(t)) + O(\delta^2) \quad (2.21)$$

Apply the derivative chain rule for $\frac{\partial f^{\eta}}{\partial(\mathbf{x}_2(t))}$, we have

$$\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta) = \frac{\partial f}{\partial(\mathbf{x}_2(t+\eta-1))} \times \frac{\partial f}{\partial(\mathbf{x}_2(t+\eta-2))} \times \ldots \frac{\partial f}{\partial(\mathbf{x}_2(t))}(\mathbf{x}_1(t) - \mathbf{x}_2(t)) + O(\delta^2)$$

$$= \mathbf{M}_{\mathbf{x}_2(t+\eta-1)} \times \mathbf{M}_{\mathbf{x}_2(t+\eta-2)} \times \ldots \times \mathbf{M}_{\mathbf{x}_2(t)}(\mathbf{x}_1(t) - \mathbf{x}_2(t)) + O(\delta^2) \quad (2.22)$$

Therefore,

$$\| \mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta) \| \leq \| \mathbf{M}_{\mathbf{x}_2(t+\eta-1)} \times \mathbf{M}_{\mathbf{x}_2(t+\eta-2)} \times \ldots \times \mathbf{M}_{\mathbf{x}_2(t)}(\mathbf{x}_1(t) - \mathbf{x}_2(t)) \| \quad (2.23)$$

where each matrix **M** is setup according to (2.20). From (2.20) and (2.23), we have the following necessary conditions for the $\mathbf{x}_{dis}(t+\eta)$ approaches to $\mathbf{x}(t+\eta)$.

2.4.1.1 Theorem 2.1

If all matrices **M** *generated by* (2.20) *have no eigenvalue outside the unit circle on the complex plane, then* $\mathbf{x}_{dis}(t+\eta)$ *approaches to* $\mathbf{x}(t+\eta)$ *as* $G \to \infty$.

The proof is as follow. Let λ be the most prominent eigenvalue of all matrices **M** with the largest magnitude. Then from (2.23)

$$| \mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta) | \leq | \mathbf{M}_{\mathbf{x}_2(t+\eta-1)} \times \mathbf{M}_{\mathbf{x}_2(t+\eta-2)} \times \ldots \times \mathbf{M}_{\mathbf{x}_2(t)}(\mathbf{x}_1(t) - \mathbf{x}_2(t)) |$$

$$\leq |\lambda|^{\eta} | \mathbf{x}_1(t) - \mathbf{x}_2(t)| \quad (2.24)$$

In (2.12), we showed that the distance between any two points in ($\mathbf{x}_{dis}$) cannot be larger than the 'main diagonal' $\sqrt{N}\delta$. Therefore,

$$| \mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta) | \leq |\lambda|^{\eta} | \mathbf{x}_1(t) - \mathbf{x}_2(t)| \leq |\lambda|^{\eta} \sqrt{N}\delta \quad (2.25)$$

Since $|\lambda| \leq 1$, $|\lambda|^{\eta}$ is finite, even with $\eta \to \infty$. Therefore, with $G \to \infty \Leftrightarrow \delta \to 0$, $|\lambda|^{\eta}\,\delta \to 0$. From the method we used in constructing the MDP, $\mathbf{x}_{dis}(t+\eta)$ also falls in $\mathbf{C}_{\eta}$. Thus $|\,\mathbf{x}(t+\eta) - \mathbf{x}_{dis}(t+\eta)\,| \leq |\,\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta)\,|$ will also approaches 0.

2.4.1.2 Theorem 2.2

If the system (2.18) *has an* ***asymptotic equilibrium point*** **p** *such that the linearized matrix* $\mathbf{M_p}$ *has all eigenvalues inside the unit circle of the complex plane, then* $\mathbf{x}_{dis}(t+\eta)$ *approaches to* $\mathbf{x}(t+\eta)$ *as* $G \to \infty$.

The proof is as follow. Since the derivative of f is continuous, there must exist a region $\mathbf{C_\varepsilon}$ with size $\boldsymbol{\varepsilon}$ around **p** such that all of the derivative matrices **M** in that region have all eigenvalues within the unit complex circle. Let λ be the eigenvalue with the largest magnitude among these matrices. In addition, since (2.1) has an *asymptotic equilibrium point*, after a finite time T, $\mathbf{x}(t)$ must be inside $\mathbf{C_\varepsilon}$. Then, from (2.23)

$$|\,\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta)\,| \leq \left|\,\mathbf{M}_{\mathbf{x}_2(t+\eta-1)} \times \mathbf{M}_{\mathbf{x}_2(t+\eta-2)} \times \ldots \times \mathbf{M}_{\mathbf{x}_2(t)}(\,\mathbf{x}_1(t) - \mathbf{x}_2(t))\,\right|$$

$$= \left|\,\mathbf{M}_{\mathbf{x}_2(t+\eta-1)} \times \mathbf{M}_{\mathbf{x}_2(t+\eta-2)} \times \ldots \times \mathbf{M}_{\mathbf{x}_2(T)} \times \right. \quad \text{(this has } \eta - T + 1 \text{ factors)}$$

$$\left.\mathbf{M}_{\mathbf{x}_2(T+1)} \times \mathbf{M}_{\mathbf{x}_2(T+2)} \times \ldots \times \mathbf{M}_{\mathbf{x}_2(t)}\;(\,\mathbf{x}_1(t) - \mathbf{x}_2(t))\,\right| \quad \text{(this has } T \text{ factors)}$$

$$\leq |\lambda|^{\eta-T+1} \times |\lambda_T| \times |\lambda_{T-1}| \times \ldots \times |\lambda_1| \times |\,\mathbf{x}_1(t) - \mathbf{x}_2(t)|$$

$$\leq |\lambda|^{\eta-T+1} \times |\lambda_T| \times |\lambda_{T-1}| \times \ldots \times |\lambda_1| \sqrt{N}\delta \quad (2.26)$$

Because λ is within the complex unit circle, $|\lambda|^{\eta-T+1}$ is finite as $\eta \to \infty$. $|\lambda_T| \times |\lambda_{T-1}| \times \ldots \times |\lambda_1|$ is also finite since T is finite. Therefore, $|\lambda|^{\eta-T+1} \times |\lambda_T| \times |\lambda_{T-1}| \times \ldots \times |\lambda_1| \sqrt{N}\delta$ approaches to 0 as $G \to \infty$ (or $\delta \to 0$), which leads to $|\,\mathbf{x}(t+\eta) - \mathbf{x}_{dis}(t+\eta)\,|$ approaching 0.

2.4.1.3 Theorem 2.3

For a special case: If the system is asymptotically stable at 0 (regardless of the linearization), then $\mathbf{x}_{dis}(t+\eta)$ *approaches to* $\mathbf{x}(t+\eta)$ *as* $G \to \infty$ (or $\delta \to 0$).

The proof for this statement is relatively simpler. For any discretization threshold δ, we can guarantee that the state $\mathbf{x}(t)$ will fall inside the region $[-\delta,\delta]$ at some finite time T, and remain in $[-\delta, \delta]$ $\forall t > T$. This fact implies that with discretization, the MDP will have an absorbing state specified by the region $[-\delta, \delta]$. In addition, regardless of the starting state $\mathbf{x}(0)$ and $\mathbf{x}_{dis}(0)$, there must be a path toward the absorbing state/region. Therefore, the MDP will eventually bring $\mathbf{x}_{dis}(t)$ to the absorbing state after some finite time L. Thus, after $\max(T, L)$, both $\mathbf{x}_{dis}(t)$ and $\mathbf{x}(t)$ will stay inside $[-\delta,\delta]$. Therefore, $|\mathbf{x}(t) - \mathbf{x}_{dis}(t)| \leq \delta$ as $t \to \infty$.

In Figure 2.2, we show some toy examples in one-dimensional system to demonstrate the first necessary condition. The left side is the result of the system

$$\boldsymbol{x}(t+1) = \sin(\boldsymbol{x}(t)) + 0.1e^{-(\boldsymbol{x}(t))^2} \quad (2.27)$$

and the right side is the result of the system

$$\boldsymbol{x}(t+1) = \sin(\boldsymbol{x}(t)) + 1.1e^{-(\boldsymbol{x}(t))^2} \quad (2.28)$$

The state space in both of these systems is [-1.5, 1.5]; the initial $\boldsymbol{x}(0)$ is 0.5 for both of them; and we discretize the entire state space into $G = 100$ regions ($\delta = 0.02$). The derivative matrices (2.20) for systems (2.27) and (2.28) are one-dimensional functions $\cos(\boldsymbol{x}) - 0.2\boldsymbol{x}e^{-x^2}$ and $\cos(\boldsymbol{x}) - 2.2\boldsymbol{x}e^{-x^2}$, correspondingly. As in Figure 2.3, where we plot the derivative of (2.27) and (2.28) in the domain [-1.5. 1.5], system (2.27) satisfies the first necessary condition; while system (2.28)

does not. We observe that $\boldsymbol{x}$ and $\boldsymbol{x}_{dis}$ approach closely to each other in system (2.27) but not in system (2.28).

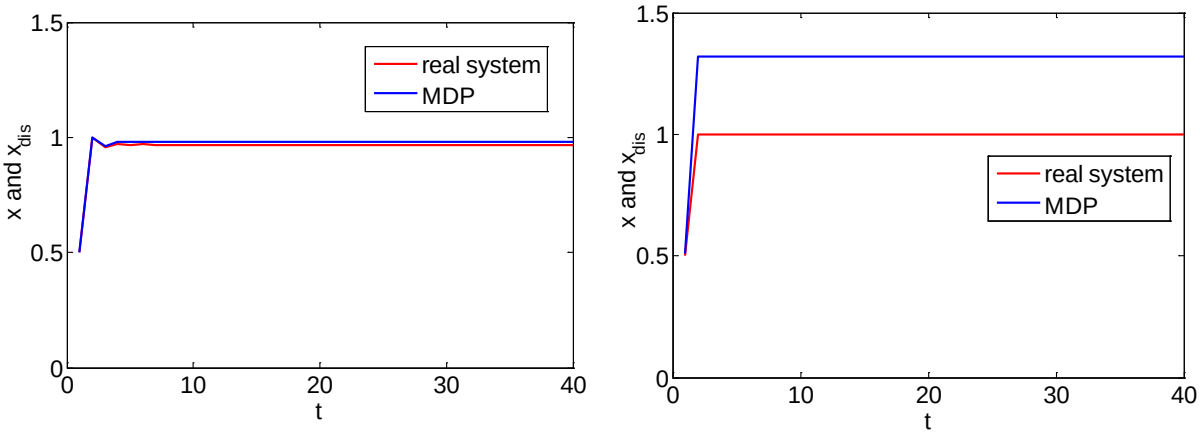

Figure 2.2. The closeness between $\boldsymbol{x}$(real system) and $\boldsymbol{x}_{dis}$ (MDP). The left figure corresponds to system (2.27). The right figure corresponds to system (2.28)

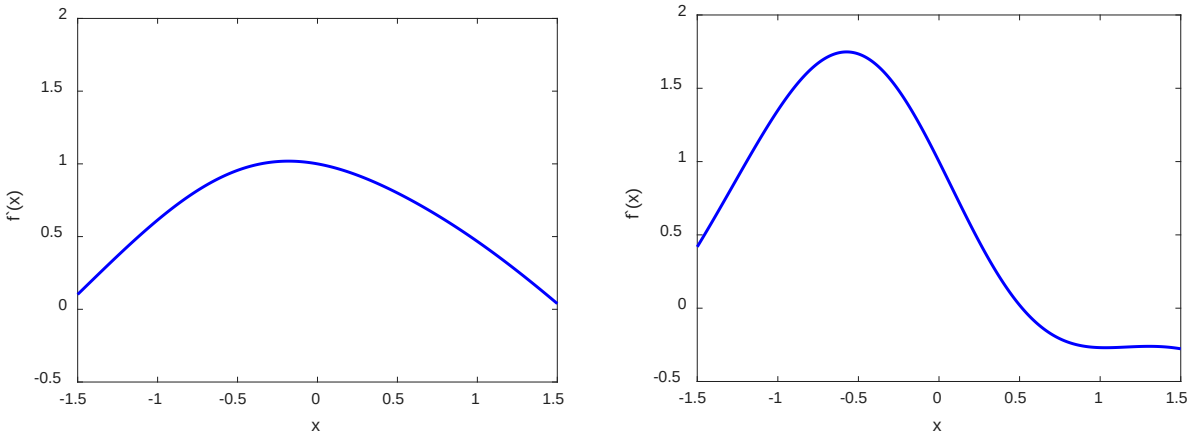

Figure 2.3. Derivative $\partial f/\partial x$ in autonomous system: (2.27) on the left and system (2.28) on the right.

2.4.2 The non-autonomous system

When we linearize the general system (2.5) using Taylor series expansion on any point $\langle \mathbf{x}, \mathbf{u} \rangle = [\mathbf{p}, \mathbf{q}]$, we have

$$f(\mathbf{x}) \approx f(\mathbf{p}, \mathbf{q}) + \mathbf{M_p}(\mathbf{x}-\mathbf{p}) + \mathbf{M_q}(\mathbf{u}-\mathbf{q}) \quad (2.29)$$

where $\mathbf{M_p}$ and $\mathbf{M_q}$ are the partial derivative of f at $[\mathbf{p}, \mathbf{q}]$

$$\mathbf{M_p} = \begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_1}{\partial x_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_1}{\partial x_d}\right|_{\mathbf{x=p,u=q}} \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_2}{\partial x_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_2}{\partial x_d}\right|_{\mathbf{x=p,u=q}} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial f_d}{\partial x_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_d}{\partial x_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_d}{\partial x_d}\right|_{\mathbf{x=p,u=q}} \end{bmatrix} \quad (2.30)$$

and

$$\mathbf{M_q} = \begin{bmatrix} \left.\frac{\partial f_1}{\partial u_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_1}{\partial u_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_1}{\partial u_g}\right|_{\mathbf{x=p,u=q}} \\ \left.\frac{\partial f_2}{\partial u_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_2}{\partial u_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_2}{\partial u_g}\right|_{\mathbf{x=p,u=q}} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial f_d}{\partial u_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_d}{\partial u_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_d}{\partial u_g}\right|_{\mathbf{x=p,u=q}} \end{bmatrix} \quad (2.31)$$

Similar to the autonomous system, for the close region ([$\mathbf{x}_{dis}$, $\mathbf{u}_{dis}$]) (2.10-2.11), including the boundary, containing [**x**, **u**], let C_η be the set of all $\mathbf{x}(t+\eta)$ computed by tracking all points in ([$\mathbf{x}_{dis}$, $\mathbf{u}_{dis}$]) on f after η time points. On the region C_η containing all possible $\mathbf{x}(t+\eta)$, there exists two points $\mathbf{x}_1(t+\eta)$ and $\mathbf{x}_2(t+\eta)$ such that $|\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta)|$ is the maximum for all pairs of points in C_η. There must exist two chains: { $[\mathbf{x}_1(t), \mathbf{u}_1(t)]$, $[\mathbf{x}_1(t+1), \mathbf{u}_1(t+1)]$, **...,** $[\mathbf{x}_1(t+\eta-1), \mathbf{u}_1(t+\eta-1$ } and { $[\mathbf{x}_2(t), \mathbf{u}_2(t)]$, $[\mathbf{x}_2(t+1), \mathbf{u}_2(t+1)]$, **...,** $[\mathbf{x}_2(t+\eta-1), \mathbf{u}_2(t+\eta-1)]$ } such that $\mathbf{x}_1(t+\eta) = f(\mathbf{x}_1(t+\eta-1), \mathbf{u}_1(t+\eta-1)) = f(f(\mathbf{x}_1(t+\eta-2), \mathbf{u}_1(t+\eta-2)))$ $= \ldots = f^{\eta}(\mathbf{x}_1(t), \mathbf{u}_1(t))$ and $\mathbf{x}_2(t+\eta) = f(\mathbf{x}_2(t+\eta-1), \mathbf{u}_2(t+\eta-1)) = f(f(\mathbf{x}_2(t+\eta-2), \mathbf{u}_2(t+\eta-2)))$ $= \ldots = f^{\eta}(\mathbf{x}_2(t), \mathbf{u}_2(t))$. Applying the Taylor series expansion, we have

$$\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta) = f^{\eta}(\mathbf{x}_1(t+\eta-1)) - f^{\eta}(\mathbf{x}_2(t+\eta-1))$$

$$= \frac{\partial f^{\eta}}{\partial(\mathbf{x}_2(t+\eta-1), \mathbf{u}_2(t+\eta-1))} ([\mathbf{x}_1(t+\eta-1), \mathbf{u}_1(t+\eta-1)] - [\mathbf{x}_2(t+\eta-1), \mathbf{u}_2(t+\eta-1)]) + O(\delta^2)$$

$$= \mathbf{M}_{\mathbf{p,x2}(t+\eta-1)}(\mathbf{x}_1(t+\eta-1) - \mathbf{x}_2(t+\eta-1)) + \mathbf{M}_{\mathbf{q,u2}(t+\eta-1)}(\mathbf{u}_1(t+\eta-1) - \mathbf{u}_2(t+\eta-1)) \quad (2.32)$$

Suppose that we have an arbitrary control law $\mathbf{u} = k(\mathbf{x})$. Taking the derivative of the control rule, we have

$$\Delta\mathbf{u} = \mathbf{M}_k \Delta\mathbf{x}$$

$$\text{Such that } \mathbf{M}_k = \begin{bmatrix} \left.\frac{\partial k_1}{\partial x_1}\right|_{\mathbf{x}=\mathbf{p}} & \left.\frac{\partial k_1}{\partial x_2}\right|_{\mathbf{x}=\mathbf{p}} & \cdots & \left.\frac{\partial k_1}{\partial x_d}\right|_{\mathbf{x}=\mathbf{p}} \\ \left.\frac{\partial k_2}{\partial x_1}\right|_{\mathbf{x}=\mathbf{p}} & \left.\frac{\partial k_2}{\partial x_2}\right|_{\mathbf{x}=\mathbf{p}} & \cdots & \left.\frac{\partial k_2}{\partial x_d}\right|_{\mathbf{x}=\mathbf{p}} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial k_g}{\partial x_1}\right|_{\mathbf{x}=\mathbf{p}} & \left.\frac{\partial k_g}{\partial x_2}\right|_{\mathbf{x}=\mathbf{p}} & \cdots & \left.\frac{\partial k_g}{\partial x_d}\right|_{\mathbf{x}=\mathbf{p}} \end{bmatrix} \quad (2.33)$$

Substitute (2.33) to (2.32), we have

$$|\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta)| = \left| \left(\mathbf{M}_{\mathbf{p},\mathbf{x2}(t+\eta\text{-}1)} \left(\mathbf{x}_1(t+\eta\text{-}1) - \mathbf{x}_2(t+\eta\text{-}1)\right) + \mathbf{M}_{\mathbf{q},\mathbf{u2}(t+\eta\text{-}1)}\right) \left(\mathbf{u}_1(t+\eta\text{-}1) - \mathbf{u}_2(t+\eta\text{-}1)\right) \right|$$

$$\leq \left| \left(\mathbf{M}_{\mathbf{p},\mathbf{x2}(t+\eta\text{-}1)} + \mathbf{M}_{\mathbf{q},\mathbf{u2}(t+\eta\text{-}1)}\ \mathbf{M}_{k\,\mathbf{x2}(t+\eta\text{-}1)}\right)\ \left(\mathbf{x}_1(t+\eta\text{-}1) - \mathbf{x}_2(t+\eta\text{-}1)\right) \right| \quad (2.34)$$

Recursively applying the derivative chain rule on $(\mathbf{x}_1(t+\eta\text{-}1) - \mathbf{x}_2(t+\eta\text{-}1))$ until $[\mathbf{x}(t), \mathbf{u}(t)]$, with the same argument from (2.32) to (2.34), we have

$$|\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta)| \leq \left| \left(\mathbf{M}_{\mathbf{p},\mathbf{x2}(t+\eta\text{-}1)} + \mathbf{M}_{\mathbf{q},\mathbf{u2}(t+\eta\text{-}1)}\ \mathbf{M}_{k\,\mathbf{x2}(t+\eta\text{-}1)}\right) \times \right.$$

$$\left(\mathbf{M}_{\mathbf{p},\mathbf{x2}(t+\eta\text{-}2)} + \mathbf{M}_{\mathbf{q},\mathbf{u2}(t+\eta\text{-}2)}\ \mathbf{M}_{k\,\mathbf{x2}(t+\eta\text{-}2)}\right) \times \ldots$$

$$\left. \left(\mathbf{M}_{\mathbf{p},\mathbf{x2}(t+1)} + \mathbf{M}_{\mathbf{q},\mathbf{u2}(t+1)}\ \mathbf{M}_{k\,\mathbf{x2}(t+1)}\right) \left(\mathbf{x}_1(t) - \mathbf{x}_2(t)\right) \right| \quad (2.35)$$

From this point, similar to the autonomous system, we have the necessary conditions for the $\mathbf{x}_{\text{dis}}(t+\eta)$ approaches to $\mathbf{x}(t+\eta)$.

2.4.2.1 Theorem 2.4

If the matrices $\mathbf{M_p} + \mathbf{M_q}\mathbf{M}_k$ *generated by* (2.30), (2.31) and (2.33) *have no eigenvalue outside the unit circle on the complex plane, then* $\mathbf{x}_{\text{dis}}(t+\eta)$ *approaches to* $\mathbf{x}(t+\eta)$ *as* $\delta \to 0$ *with any* η.

2.4.2.2 Theorem 2.5

If the system (2.5) *has an* ***asymptotic equilibrium point*** **p** *such that the linearized matrix* $\mathbf{M_p} + \mathbf{M_q}\mathbf{M}_k$ *at the equilibrium point has all eigenvalues inside the unit circle of the complex plane, then* $\mathbf{x}_{dis}(t+\eta)$ *approaches* $\mathbf{x}(t+\eta)$ *as* $\delta \to 0$ *for any* η.

We omit the proof for these two statements since the proof is almost similar to the proof we already showed in the autonomous system section

In Figure 2.4, we show some toy examples in one-dimensional system to demonstrate the first necessary condition. The left side is the result of the system

$$\boldsymbol{x}(t+1) = \sin(\boldsymbol{x}(t)) + \boldsymbol{u}(t) \text{ and control law } \boldsymbol{u}(t) = -0.5x(t) \quad (2.36)$$

and the right side is the result of the system

$$\boldsymbol{x}(t+1) = \sin(\boldsymbol{x}(t)) + \boldsymbol{u}(t) \text{ and control law } \boldsymbol{u}(t) = -2x(t) \quad (2.37)$$

The state space in both of these systems is [-1, 1]; the initial $\boldsymbol{x}(0)$ is 0.5 for both of them; and we discretize the entire state space into $G = 100$ regions ($\delta = 0.02$). In (2.36), $\mathbf{M_p} + \mathbf{M_q}\mathbf{M}_k = \cos(x(t)) - 0.5$, which is within [0.0403, 0.5]. Therefore, (2.36) meets the first necessary condition. In (37), $\mathbf{M_p} + \mathbf{M_q}\mathbf{M}_k = \cos(x(t)) - 2$, which is between [-1.5403, -1]. Therefore, (2.37) does not meet the necessary condition. As in Figure 2.4, $\mathbf{x}_{dis}(t)$ converges to $\mathbf{x}(t)$ in system (2.36), but not in system (2.37).

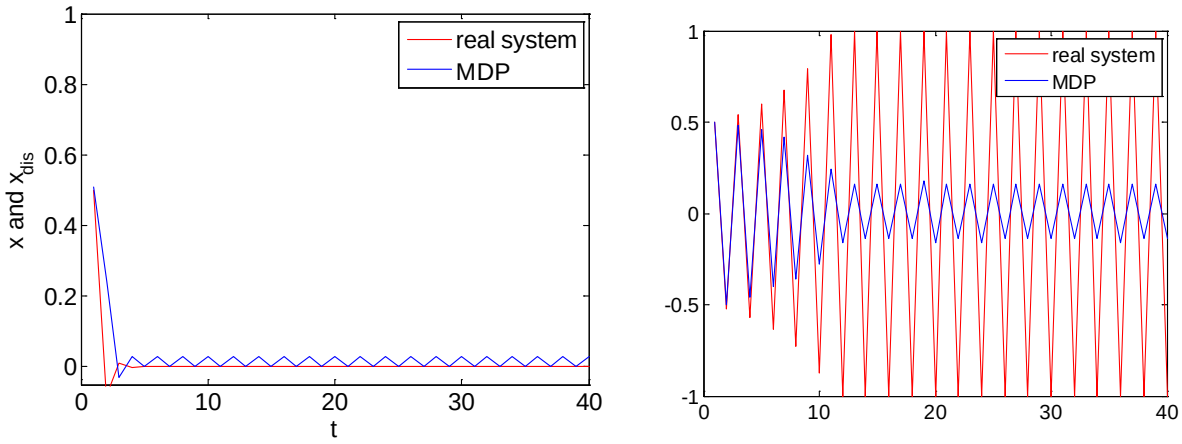

Figure 2.4. The closeness between **x**(real system) and $\mathbf{x}_{dis}$ (MDP) in non-autonomous system. The left figure corresponds to system (2.36). The right figure corresponds to system (2.37).

2.5 The existence of the MDP solution as to near-optimally stabilize the system

In this section, we show the existence of the MDP solution when the system (2.5) is stable at the equilibrium point. The stability definition is defined as follow: there exist a positive small number ε such that if $|\mathbf{x}| < \varepsilon$ then $|f(\mathbf{x}, \mathbf{0})| < \varepsilon$. With this assumption, when we choose G such that $\chi/G < \varepsilon$, the MDP will have a special state $\mathbf{x}^*_{dis} = \mathbf{0}$ with the following properties:

- The MDP's optimal policy at $\mathbf{x}^*_{dis}$ is $\mathbf{u}^*_{dis} = \mathbf{0}$.

- The later states in the MDP are also $\mathbf{x}^*_{dis}$.

The proof for these properties are relatively simple due to the properties of the state and action reward functions in (2.6) and (2.7), where the optima are at **0**. From this stability assumption of f, we prove the following statements.

2.5.1.1 Theorem 2.6

If the system (2.1) *is stable and the HJB equation* (2.5)-(2.10) *has a finite solution as* $\gamma \to 1$, *then in the MDP,* $\mathbf{x}_{dis}(t) = \mathbf{0}$ *as* $t \to \infty$.

The proof for this statement is as follow. If the HJB equation (2.5)-(2.10) has a finite solution as $\gamma \to 1$, then the control function **u**(t) has to be able to bring **x**(t) to **0** in finite time. Otherwise, the state and action rewards are always negative and will approach infinite as $\gamma \to 1$. Since **x**(t) is 0 in finite time, there must exist a path in the MDP that can reach $\mathbf{x}^*_{dis}$ with positive probability. Obviously one of these paths is the discretization of the HJB's solution **u**(t). Since the policy iteration in MDP has been proven to converge to the optimal policy [13], this policy cannot be worse that the policy induced by discretizing the HJB equation's solution. Therefore, in the MDP's optimal policy, there must exist a path from any state to $\mathbf{x}^*_{dis}$ with positive probability $\phi > 0$. With infinite number of visit t$\to \infty$, the maximum probability for **not** reaching $\mathbf{x}^*_{dis}$ is $(1-\phi)^{\infty} \to 0$.

2.5.1.2 Theorem 2.7

If all $\mathbf{M_p}$ *matrices* (2.30) *have the most prominent eigenvalues within the unit circle* $\forall \mathbf{x},\mathbf{u}$ *and* $\mathbf{x}_{dis}(t) = \mathbf{0}$ *as* $t \to \infty$ *in the MDP solution for all starting* $\mathbf{x}_{dis}(0)$, *then by applying the MDP's control unit* $\mathbf{u}_{dis}(t)$ *on* $\mathbf{x}(t)$, $|\mathbf{x}(t)| \leq \sqrt{N}\delta$.

The proof of this statement is as follow. Since we apply $\mathbf{u}_{dis}(t)$ for all $\mathbf{x}(t)$ in ($\mathbf{x}_{dis}(t)$) region, the difference of the control unit cancels. Thus, the equation (2.32) becomes

$$f(\mathbf{x}) \approx f(\mathbf{p}, \mathbf{q}) + \mathbf{M_p}(\mathbf{x}-\mathbf{p}) \quad (2.38)$$

Following the same argument from (2.30) to (2.35), we have

$$|\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta)| \leq |\ (\mathbf{M}_{\mathbf{p},\mathbf{x2}(t+\eta-1)})\ \times (\mathbf{M}_{\mathbf{p},\mathbf{x2}(t+\eta-2)})\ \times \ldots \times (\mathbf{M}_{\mathbf{p},\mathbf{x2}(t+1)})\ (\mathbf{x}_1(t) - \mathbf{x}_2(t))\ | \quad (2.39)$$

Because the most prominent eigenvalues of $\mathbf{M_p}$ are within unit circle, from (2.39), we have

$$|\mathbf{x_{dis}}(t) - \mathbf{x}(t)| \leq |\mathbf{x}_1(t+\eta) - \mathbf{x}_2(t+\eta)| \leq |\ (\mathbf{x}_1(t) - \mathbf{x}_2(t))\ | \leq \sqrt{N}\delta \quad (2.40)$$

Therefore, if $\mathbf{x}_{dis}(t) \to \mathbf{0}$, then $|\mathbf{x}(t)| \leq \sqrt{N}\delta$.

2.6 Simulation results

In this example, we choose the system

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) = sin\big(\mathbf{A}\mathbf{x}(t) + \mathbf{u}(t)\big) \quad (2.41)$$

where

$$\mathbf{A} = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

For the ease of decentralization, we choose the system such that the dimensionalities of both **x** and **u** are the same. The vector sin function is defined from each dimension as

$$\sin(\mathbf{x}) = \begin{bmatrix} \sin(\mathbf{x}_1) \\ \sin(\mathbf{x}_2) \\ \sin(\mathbf{x}_3) \end{bmatrix} \quad (2.42)$$

In addition, each dimension of **x** and **u** is between -0.35 and 0.35. The initial state **x**(0) is a vector of **0.2**. For equation (2.2) and (2.3), we choose $p(\mathbf{x}) = -\|\mathbf{x}\|^2$ and $q(\mathbf{u}) = -\|\mathbf{u}\|^2$. We set $\gamma = 0.9$. For discretization, we divide each state and control dimension into $G = 7$ grids, which means $\delta = 0.1$. It is easy to see that system (2.40) satisfy the necessary conditions for the existence of the MDP solution. Figure 2.5 shows that the MDP solution could drive the system (40) toward the stable region.

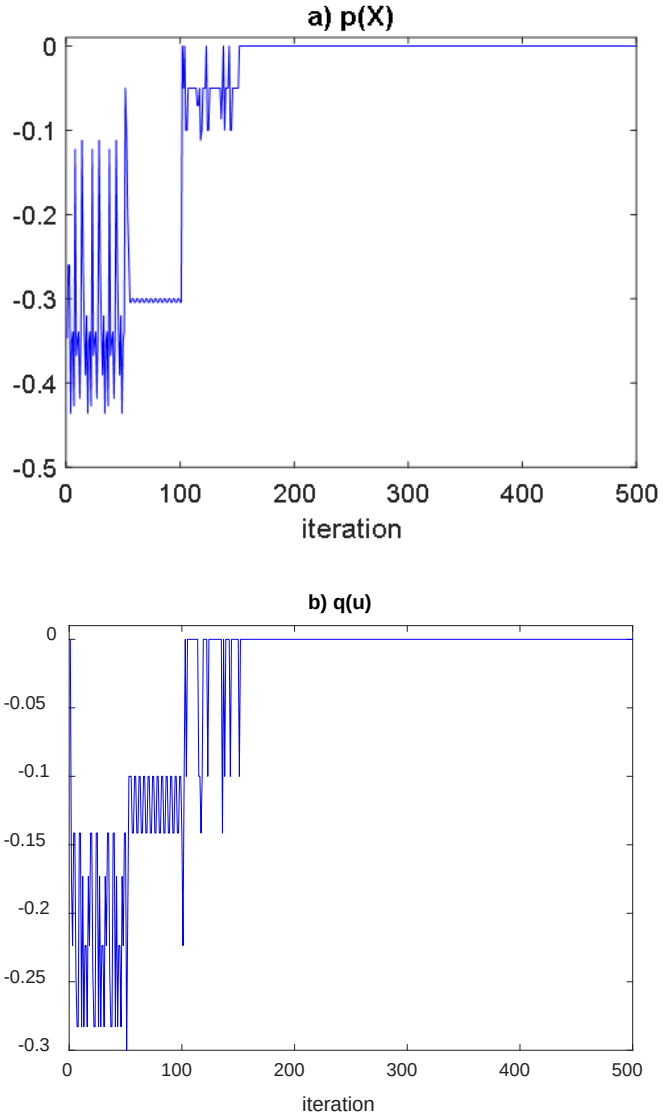

Figure 2.5. Learning performance $p(\mathbf{x})$ and $q(\mathbf{u})$ in system (2.40) with MDP solution

3. SELECTIVE DECENTRALIZATION

As mentioned in chapter 1 and chapter 2, the larger reinforcement learning problems is, the more decentralized learning techniques should be. MDP is a typical example where the number of states grows exponentially with the problem dimensionality [73], which explains why this technique may be inapplicable for even average-size problem with dimensionality around 10. In chapter 1, we already discuss the recent state-of-the-art methods in decentralized learning [44, 46, 48, 49, 51]. However, these techniques often require partial and prior knowledge about the communicative structure among the learning agents. This assumption may not always hold in reinforcement learning, given its unknown nature.

In this short chapter, we present the overview, problem statement and pseudo code for the selective decentralization technique, which is the second key contribution in this thesis. The outcome of selective decentralization would be presented in later chapters when selective decentralization is used in combination with other well-known reinforcement learning techniques. From the communication point of view, the selective decentralization belongs to multi-model switching category [52-55]. The selective decentralization could not only improve the learning performance but also learn the optimal communication scheme for the agents without any prior knowledge on communicative structure.

3.1 Problem statement of selective decentralization

Let the dynamical system (Σ) be described by the equation:

$$\Sigma: \boldsymbol{x}(t+1) = f[\boldsymbol{x}(t), \boldsymbol{u}(t), \theta] \quad (3.1)$$

where $x \in \mathbb{R}^N$ where N is a large number. It is assumed in this problem that the input $u(t)$ is known. θ is an unknown parameter vector in $\mathbb{R}^M$ where the dimension M is large. The objective is to estimate θ using measurements of the overall system.

In the problem of interest to us, the system is assumed to consist of r components of lower dimension which are interconnected. If the state vectors of the subsystems $\Sigma_1, \Sigma_2, \dots, \Sigma_r$ are respectively $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_r$, it is assumed that each subsystem can be described by the difference equation

$$\Sigma_i: \boldsymbol{x}_i(k+1) = f_i[\boldsymbol{x}_i(k), u_i(k), \theta_i] + \sigma_i g[\boldsymbol{z}_i(k)] \quad (3.2)$$

where the parameter σ_i is assumed to be small, and $[\boldsymbol{x}_i, \boldsymbol{z}_i] = \boldsymbol{x}^T$(i.e., the elements of $\boldsymbol{z}_i$ are state variables not contained in $\boldsymbol{x}_i$).

A decentralized model can be set up as:

$$\hat{\boldsymbol{x}}_i(t+1) = \hat{f}_i[\boldsymbol{x}_i(t), \boldsymbol{z}_i(t), \boldsymbol{u}(t), \theta(t)] \quad (3.3)$$

At this stage, the knowledge that each agent has about the components of z that affect it, becomes important. We distinguish between two distinct cases:

- Every agent knows all the state variables that affect its outputs (known decentralization structure).

- Every agent Σ_i knows the small set of variables in $\boldsymbol{z}_i$ that might affect its outputs, but does not know exactly which variables do affect them (unknown decentralization structure).

In the former case, Σ_i uses a single model set up with the correct decentralization structure, but in the latter case it uses multiple models corresponding to different possible decentralization

structures, and switches between them. The former is referred to as strict decentralization, and the latter as selective decentralization.

The selective decentralized control examines all of the connection schemes (also called decentralization scheme) among the agents and sets up criteria to select which scheme to help in performing the learning, depending on the specific problem. In this thesis, we use system identification error for model-based learning and cumulative Q-value [26] gained for model-free learning. Since we assume that the agents do not know anything about communicative structure, the possible of communication schemes follows the Bell's number rule, which is the number of partition in a set [59]. For example, with number of agents r = 3, we have $B(r)$ = 5, which is the third Bell's number, possible decentralization schemes: {{1, 2, 3}}, {{1, 2}, {3}}, {{1, 3}, {2}}, {{1}, {2, 3}} and {{1}, {2}, {3}}, in which each scheme has 1, 2, 2, 2 and 3 subsystem(s), correspondingly. An agent only uses its state and control variable to compute its own approximator. For example, in the linear control-learning system (see chapter 2), with scheme {{1, 2}, {3}}, we have the format $\widehat{\mathbf{A}} = \begin{bmatrix} \widehat{\mathbf{A}}_{1,2} & \\ & \widehat{\mathbf{A}}_3 \end{bmatrix}$. For identification phase, $\widehat{\mathbf{A}}_{1,2}(t)$ is computed only using $\mathbf{x}_1(t\text{-}1)$, $\mathbf{x}_2(t\text{-}1)$, $\mathbf{u}_1(t\text{-}1)$ and $\mathbf{u}_2(t\text{-}1)$, meanwhile $\widehat{\mathbf{A}}_3(t)$ is computed only using $\mathbf{x}_3(t\text{-}1)$ and $\mathbf{u}_3(t\text{-}1)$. In the decision phase, if scheme {{1, 2}, {3}} returns the lowest identification error, then from (12), we compute the next control $[\mathbf{u}_1(t), \mathbf{u}_2(t)]$ using only $\widehat{\mathbf{A}}_{1,2}(t)$ and $\mathbf{u}_3(t)$ using only $\widehat{\mathbf{A}}_3(t)$.

3.2 Pseudo code for selective decentralization

One important factor in selective decentralization is how often the central coordinator examines and decides switching the decentralization schemes. Here, let Ω be the window size, which means after how many iterations or learning instances the agent should examines and decides switching.

Let w be the window index. Then the window w covers the discrete time index from $t = (w\text{-}1)\,\Omega + 1$ to $t = w\Omega$. The central coordinator decides the switching criteria E for all communication scheme, for example, identification error. Let $E_s(w)$ be the value of E for communication scheme s at window w. Let ε and γ be two small thresholds for termination. The pseudo code for selective decentralization is as follow:

initialize b: the best decentralization scheme

E: criteria to select the best decentralization scheme,

$E_s = 0$ for all communication scheme s.

for w from 1 to the maximum window index

perform the learning technique using b

compute $E(w)$ for $B(r)$ decentralization schemes s (see $B(r)$ in the previous section)

Select the decentralization scheme with the lowest $E(w)$ as b

if w > 1 and ($E(w) < \varepsilon$ or $|E(w) - E(w\text{-}1)| / |E(w)| < \gamma$)

return

end if

end for

4. SELECTIVELY DECENTRALIZED Q-LEARNING

In this chapter, we show the impact of selective decentralization to improve Q-learning, which is one of the most well-known model-free reinforcement learning technique [27, 46, 79-81]. The key contributions of selectively decentralized Q-learning are at three points. First, selectively decentralized Q-learning would clearly show superior converging time, compared to centralized Q-learning. As the dimensionality of the learning problem increases, the converging time of selectively decentralized Q-learning is expected to be exponentially shorter than the converging time of centralized Q-learning. Second, the results in this chapter suggest that decentralized Q-learning could be practically applied as an alternative approach for unknown system control problems. In these problems, the conventional Hamilton-Jacobi-Bellman equation approach may not provide the close-form solution in general [15-18]. Third, selective decentralized Q-learning is able to learn the optimal communication scheme among the learning agents, which many recent decentralized Q-learning technique may not have.

Decentralization has been widely applied in Q-learning [26], especially when the systems are naturally distributed [82, 83]. In Q-learning, the learning agent maintains the optimal values for all state-action entries in its Q-table. In each state, the learning agent chooses the action by the highest Q-table entry for the state. After each visit, the learning agent updates the former state-action Q-value by the new state's reward and highest Q-value. Most of the decentralized Q-learning approaches adapt the partial communication idea: each subsystem manages its own communication and utilizes these communications for updating own Q-table. Although decentralized Q-learning has been well-established, there are still two open questions in this approach. First, how well decentralized Q-learning tackle the slow rate of converging weakness in

Q-learning [84]? Second, how to use apply multi-model-switching in decentralized Q-learning? There has been relatively small literature about multi-model-switching in decentralized Q-learning.

In this chapter, I address these two questions above by solving an optimal control problem by Q-learning paradigm. A brief review about Q-learning is presented in Chapter 1 with literatures [25-27]. This chapter also proves some sufficient conditions such that the Q-learning policy is guaranteed to stabilize the nonlinear system. I emphasize that in our control problems, we design the state-reward function with linearity property such that the central state-reward value is equivalent to the sum of all sub-state rewards. From this argument, I choose the cumulative gained Q-value to decide the best communication scheme. However, due to the difficulty to provide theoretical analysis for this question, I can only offer limited proofs that the Q-learning policy could guarantee to stabilize the learning-control system for a single agent. Most of our results only serve as confirmation studies, as I already published in [85]. While it is clear that the multi-agent and selective decentralization could stabilize the learning-control system, it is still unclear how the multi-agent policy could be, and how the multi-agent policy is similar to the centralized policy.

4.1 Selectively decentralized Q-learning method

4.1.1 Problem statement

In this chapter, we are interested in the systems in the general form

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (4.1)$$

where $\mathbf{x} \in \Re^N$ is the joint state vector, $\mathbf{u} \in \Re^M$ is the joint action (also called control) vector, $f : \Re^{N+M} \to \Re^N$ is a general nonlinear unknown function. We assume that f has the stable equilibrium point $f(\mathbf{0}, \mathbf{0}) = \mathbf{0}$. The main objective is to learn the sequence of action units $\mathbf{u}(t)$ to stabilize $\mathbf{x}$

$$\mathbf{x}(t) \to 0, \mathbf{u}(t) \to 0 \text{ as } t \to \infty \quad (4.2)$$

In order to apply Q-learning, we need to discretize the system (4.1). Therefore, to simplify the discretization, we assume that the system (4.1) has the following properties

- Each dimension of **x** is symmetrically bounded by $[-\chi, \chi]$, where $\chi > 0$ is a known boundary for **x**.
- Each dimension of **u** is symmetrically bounded by $[-\mu, \mu]$, where $\mu > 0$ is a known boundary for **u**.

To apply selectively decentralized Q-learning, we restate the following assumptions for system (4.1). First, the system could be decoupled in to K subsystems, where each subsystem could be assigned to an independent learning agent. Each agent knows which components of **x** and **u** belonging to it. Second, since f is unknown, each agent k does not know the relationship between the current sub-state $\mathbf{x}_k(t)$ and previous sub-state/sub-action $[\mathbf{x}_k(t\text{-}1), \mathbf{u}_k(t\text{-}1)]$. Each agent does not know the interconnection among itself and the other agents. The central coordinator unit to decide which decentralization structure could provide the best learning performance. In each scheme, there are $L \leq K$ groups such that each group contains one or more subsystems/agents communicating to execute Q-learning. In a group, inside agents do not communicate with any outside agents.

4.1.2 System discretization and reward function

Let G be the number of intervals in each dimension of **x** and **u** for which we uniformly divide the dimension into small grids. Therefore, the entire state space is divided into G^N small hyper cubes with edge $\theta_x = 2\chi/G$. The control space is divided into G^M small hyper cubes with edge $\theta_u = 2\mu/G$. All points inside a hyper cube are discretely represented by the center of the hyper cube. Points on

the border between two hyper cubes are represented by the center of the 'left' hypercube. Mathematically, the discretization process is described by the following formulas

$$\mathbf{x}[i] \to \theta_x + \chi/G \quad \forall i \in [1, N] \;\; and \; \mathbf{x}[i] \in [\theta_x, \theta_x + 2\,\chi/G) \quad (4.3)$$

$$\mathbf{u}[i] \to \theta_u + \mu/G \quad \forall i \in [1, M] \;\; and \; \mathbf{u}[i] \in [\theta_u, \theta_u + 2\,\mu/G) \quad (4.4)$$

where $\theta_x \in \{-\chi, -\chi + 2\chi/G, -\chi + 4\chi/G, ..., \chi - 2\chi/G\}$ and $\theta_u \in \{-\mu, -\mu + 2\mu/G, -\mu + 4\mu/G, ..., \mu - 2\mu/G\}$, which are the 'left' boundaries in the hyper cubes. We denote $\mathbf{x}_{\text{dis}}$ and $\mathbf{u}_{\text{dis}}$ as the discrete space and control vector of **x** and **u**, correspondingly.

With the discretization process in (4.3) and (4.4), it is important and easy to see that when *M* is odd, the zero vector **0** is one of the discrete space/control vectors. Given this condition, we define the state reward function $q(\mathbf{x})$ as

$$q(\mathbf{x}) = \sum_{i=1}^{N} q(i)\,, \text{where } q(i) = \begin{cases} -\mathbf{x}_{\text{dis}}(i)^2 \text{ if } \mathbf{x}_{\text{dis}}(i) \neq 0 \\ r \text{ if } \mathbf{x}_{\text{dis}}(i) = 0 \end{cases} \quad (4.5)$$

where $r > 0$ is a small bonus factor when the discrete $\mathbf{x}_{\text{dis}}$ is **0**, or **x** is within the hypercube containing the equilibrium point. Since our main objective is to stabilize (4.1), with reward function (4.5), the learning problem aims to maximize

$$J(\mathbf{x}) = \sum_{t=0}^{\infty} \gamma^t q\big(\mathbf{x}_{\text{dis}}(t)\big) \quad (4.6)$$

where $0 < \gamma < 1$ is the discount factor. The learning problem (4.3-4.6) is similar to a classical exploration problem in [3], where there is only one terminated state with positive reward and all of the other states show negative reward.

4.1.3 Selectively decentralized Q-learning formulation

First, we rewrite (4.5) and (4.6) for subsystem as follow. Let N_1, N_2, …N_K be the dimensionality of the K subsystems. Certainly, $N_1 + N_2 + \ldots + N_K = N$. Let $\{i_1\}$, [86], … $\{i_K\}$ be the set of indexes of $\mathbf{x}$ and $\mathbf{u}$ belonging to these subsystems. In subsystem k, we denote $\mathbf{x}\{i_k\}$ and $\mathbf{u}\{i_k\}$ as the sub-state and sub-action vectors. Thus, (4.5) becomes

$$q\big(\mathbf{x}(\{i_k\})\big) = \sum_{\forall i \in \{i_k\}} q(i)\,, \text{where } q(i) = \begin{cases} -\mathbf{x}_{\text{dis}}(i)^2 \text{ if } \mathbf{x}_{\text{dis}}(i) \neq 0 \\ r \text{ if } \mathbf{x}_{\text{dis}}(i) = 0 \end{cases} \quad (4.7)$$

In each subsystem, at each iteration, the Q-learning is executed according to [3]

$$Q[\mathbf{x}_{\text{dis}}\{i_k\}(t-1), \mathbf{u}_{\text{dis}}\{i_k\}(t-1)] = (1-\alpha) Q[\mathbf{x}_{\text{dis}}\{i_k\}(t-1), \mathbf{u}_{\text{dis}}\{i_k\}(t-1)] +$$

$$\alpha\left(q\big(\mathbf{x}_{\text{dis}}\{i_k\}(t-1)\big) + \gamma \max_{\mathbf{u'}_{\text{dis}}\{i_k\}} Q[\mathbf{x}_{\text{dis}}\{i_k\}(t), \mathbf{u'}_{\text{dis}}\{i_k\}]\right) \quad (4.8)$$

Where $Q[\mathbf{x}_{\text{dis}}\{i_k\}, \mathbf{u}_{\text{dis}}\{i_k\}]$ denotes the Q table in subsystem k and $0 < \alpha < 1$ is the learning rate.

Suppose that the decentralization scheme b partitions the entire system into L disjoint components $c_1, c_2, \ldots, c_L$ with dimensionality $N_1, N_2, \ldots, N_L$. Each component contains one or more subsystems. For any component c_l, let $\{I_l\} = \cup \{i_k\}$ be the union of indexes from all subsystems k belonging to c_l. In this component, the Q-learning is executed according to (4.7) and (4.8) with index set $\{I_l\}$. Since the number of possible decentralization schemes in a K-subsystem is B_K [59], the main question in selective decentralization is which scheme b is the 'best'. In this work, we select the scheme b returning the highest cumulative gained Q value, which is

$$\Omega(b) = \sum_{l=1}^{L} \alpha\left(\begin{array}{c} q\big(\mathbf{x}_{\text{dis}}\{I_l\}(t-1)\big) + \\ \gamma \max_{\mathbf{u'}_{\text{dis}}\{I_l\}} Q[\mathbf{x}_{\text{dis}}\{I_l\}(t), \mathbf{u'}_{\text{dis}}\{I_l\}] - Q[\mathbf{x}_{\text{dis}}\{I_l\}(t-1), \mathbf{u}_{\text{dis}}\{I_l\}(t-1)] \end{array}\right) \quad (4.9)$$

Let *w* be the window index covering the time from $t = (w\text{-}1)\,\Omega + 1$ to $t = w\Omega$. In this window, we choose the same decentralization scheme to decide the optimal action $\mathbf{u'}_{\text{dis}}$ for $\max_{\mathbf{u'}_{\text{dis}}} Q[\mathbf{x}_{\text{dis}}(t), \mathbf{u'}_{\text{dis}}]$. Larger window size implies less scheme switching. Pseudo code of procedure ***QLearning_Window*** shows more details on how we execute selectively decentralized Q-learning in each window.

Procedure *QLearning_window* (*w*)

Persistent input: Q tables in all B_K decentralization schemes

S: array to store the cumulative gained Q-value

b: best decentralization scheme

if $w = 1$

Initialize all Q tables as 0 in all decentralization schemes (4.10)

Choose a random decentralization scheme as *b*

Reset *S* to 0

for *t* from $(w\text{-}1)\,\Omega + 1$ to $w\Omega$

// use *b* to compute the action

for all components *l* in *b*

Compute $\mathbf{u}_{\text{dis}}\{I_l\}(t)$ as $\max_{\mathbf{u'}_{\text{dis}}\{I_l\}} Q[\mathbf{x}_{\text{dis}}\{I_l\}(t), \mathbf{u'}_{\text{dis}}\{I_l\}]$ (4.11)

Assembly $\mathbf{u}_{\text{dis}}(t)$from all $\mathbf{u}_{\text{dis}}\{I_l\}(t)$

// update the cumulative Q-value gained

for all decentralization schemes β

$S[\beta] = S[\beta] + \Omega(\beta)$ // formula (4.9)

Update Q tables in all components according to (8), with $\mathbf{u}_{\text{dis}}(t)$

Choose b as $\underset{\beta}{\operatorname{argmax}}\, S[\beta]$

In (4.11), if there are multiple $\mathbf{u}'_{\text{dis}}\{I_l\}$ returning the same optimal Q value (especially in (4.10) when we initialize the Q-table as 0) for $\mathbf{x}_{\text{dis}}\{I_l\}(t)$, we randomly select one instance. We choose the cumulative gained Q-value as the choice of decentralization scheme. First, it is important to note that the state-reward function (4.5), (4.7) satisfy the first linearity assumption: *For any decentralization b separating system (1) into L components such that these component are completely disjoint, the sum of components' state-rewards is equal to the centralized state-reward.*

$$q(\mathbf{x}) = \sum_{l=1}^{L} q\big(\mathbf{x}(\{I_l\})\big) \quad (4.12)$$

4.2 Sufficient conditions for the Q-learning policy to stabilize the system

The discretization in (4.3) and (4.4) allows partitioning the state space into different layers as follow:

- The 0^{th} layer contains only the hypercube with the **0**-equilibrium point.

- Recursively, the g^{th} layer ($g > 0$) contains all hyper-cubes outer-neighboring the $(g\text{-}1)^{\text{th}}$ layer.

Figure 4.1 demonstrates the layer defined above in two-dimensional state space.

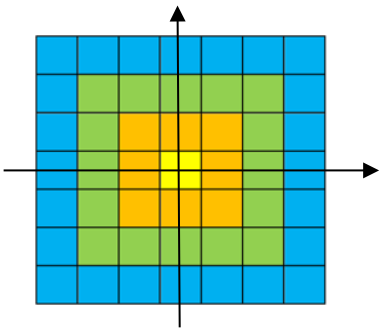

Figure 4.1. Demonstration of state-layers for discretization in two dimensions. : 0^{th} layer, : 1^{st} layer, : 2^{nd} layer, : 3^{rd} layer

In addition, by the continuity assumption and the choice of G to discretize the action state, for every state of layer $g > 0$, there must exist discrete action transiting the next state toward the inner layers.

From the discretization in (4.5) and (4.6) and the layer definition above, we setup the penalty function as follow

$$q(\mathbf{x}) = -\beta_g \|\mathbf{x}\| \quad (4.13)$$

in which g denotes the layer where $\mathbf{x}$ belongs to. β_g are positive constants chosen as follow

$$\begin{cases} \beta_0 = 0 \\ \beta_1 = 1 \\ \beta_g = \dfrac{\beta_{g-1}\sqrt{N}}{1-\gamma} \quad \forall g > 1 \end{cases} \quad (4.14)$$

In this section, I demonstrate the theoretical result on sufficient conditions for the Q-learning solution to stabilize the system (4.1). The analysis begins with well-known statements that with the Q-learning executed as in [26]: *It is guarantee that Q-learning converges the Q-table to the optimal values*. In addition, [87] shows that: *The optimal Q-value is equivalent to the optimal learning goal J(***x***).* From this point, we have the following theorems.

4.2.1.1 Theorem 4.1

For all states x belonging to the 0^{th} layer, the optimal Q-table for these states is $Q(\mathbf{x}, \mathbf{0}) = 0$, which implies that there is no need to take further action.

Proof: this theorem is trivial. Because system (4.1) is stable at the 0-equilibrium point such that

$$\|\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{0})\| \leq \delta \ \forall\ t > 0 \text{ if } \|\mathbf{x}(0)\| \leq \delta \quad (4.15)$$

and we choose $\delta = \max(2\chi/G, 2\mu/G)$ to discretize the state space as in (4.3-4.4), by taking no action $\mathbf{u} = \mathbf{0}$, all subsequent states will stay within the 0^{th} layer if $\mathbf{x}(0)$ is inside the 0^{th} layer. Therefore, the Q-table should be

$$J(\mathbf{x}) = \sum_{t=0}^{\infty} \gamma^t q\left(\mathbf{x}_{\text{dis}}(t)\right) = \sum_{t=0}^{\infty} \gamma^t \alpha_0 ||\mathbf{x}_{\text{dis}}(t)|| = 0 \quad (4.16)$$

The other actions $\mathbf{u} \neq \mathbf{0}$ may bring the state outside the 0-equilibrium point and receive penalty. In this case, the Q-table would be negative and certainly not optimal.

4.2.1.2 Theorem 4.2

With the partial derivative of system (1) respected to $\mathbf{x}$ *at* $\mathbf{x} = \mathbf{p}$ *and* $\mathbf{u} = \mathbf{q}$ *as* $\mathbf{M_{p,q}}$

$$\mathbf{M_{p,q}} = \begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_1}{\partial x_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_1}{\partial x_d}\right|_{\mathbf{x=p,u=q}} \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_2}{\partial x_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_2}{\partial x_d}\right|_{\mathbf{x=p,u=q}} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial f_d}{\partial x_1}\right|_{\mathbf{x=p,u=q}} & \left.\frac{\partial f_d}{\partial x_2}\right|_{\mathbf{x=p,u=q}} & \cdots & \left.\frac{\partial f_d}{\partial x_d}\right|_{\mathbf{x=p,u=q}} \end{bmatrix} \quad (4.17)$$

if the largest (magnitude) eigenvalue of $\mathbf{M_{p,q}}$ $\forall \mathbf{p, q}$ *is within the unit circle, then for every state not inside the 0th layer, the optimal action must be able to transit the state toward the inner layer*.

Proof: the proof starts by linearizing (1) using Taylor series expansion

$$f(\mathbf{x, q}) = f(\mathbf{p, q}) + \mathbf{M_{p,q}}(\mathbf{x-p}) \quad (4.18)$$

When $\mathbf{x}$ and $\mathbf{p}$ are in the same hypercube, $||\mathbf{x-p}|| < \delta$. Therefore, from (4.18)

$$|| f(\mathbf{x, q}) - f(\mathbf{p, q}) || = || \mathbf{M_{p,q}}(\mathbf{x-p}) || < \lambda\delta \quad (4.19)$$

where λ stands for the largest eigenvalue of $\mathbf{M_{p,q}}$. Therefore, if λ is within the unit circle,

$$|| f(\mathbf{x, q}) - f(\mathbf{p, q}) || < \delta \quad (4.20)$$

which imply that from every state in the same hypercube and executing the same action, the next state will not span on more than two layers. Therefore, for any hypercube [$\mathbf{x}_{dis}$), if any action **q** could transit a state $\mathbf{x} \in [\mathbf{x}_{dis})$ toward the inner layers, then there exist no state in [$\mathbf{x}_{dis}$) such that executing **q** could transit toward the outer layers.

In addition, it is easy to see that at the g^{th} layer, $||\mathbf{x}_{dis}||$ is bounded by

$$g\delta\left|\left|\mathbf{x}_{dis}\right|\right| = \sqrt{\sum_{i=1}^{N}(\mathbf{x}_{dis})^2} < g\delta\sqrt{N} \quad (4.21)$$

From the definition of $q(\mathbf{x})$ in (4.13) and (4.14), for every state at g^{th} layer ($g > 0$), we have

$$\sum_{t=0}^{\infty}\gamma^t\beta_g||\mathbf{x}_{\text{dis}}(t)|| < \sum_{t=0}^{\infty}\gamma^t\beta_g g\delta\sqrt{N} = \frac{\beta_g g\delta\sqrt{N}}{1-\gamma} = \beta_{g+1}g\delta \quad (4.22)$$

In (4.22), the left side is the upper bound of the Q-value ($J(\mathbf{x})$) by executing action such that the state does not transit toward the outer layer. Here, the upper bound is calculated in the worst-case scenario when the state simply stay in the g^{th} layer. In addition, from (4.16) we can see that the right side is the lower bound of Q-value by executing action such that the state transit toward the outer layers. Therefore, for any state, the optimal policy should avoid actions, which could transit toward the outer layers. Furthermore, *from any state* **x**, *if there exist an action* **u** *transiting the next state toward the inner layers, then the optimal policy should allow transiting the state toward the inner layers*.

4.2.1.3 Theorem 4.3

If the largest (magnitude) eigenvalue of $\mathbf{M_{p,q}}$ **∀p, q in** (4.17) *is within the unit circle, then the optimal policy learned by Q-learning will bring the system state toward the* 0^{th} *layer, thus stabilize the system toward the 0-equilibrium point with* δ *error.*

Proof: By the continuation assumption of system (1), we know that there exist actions allowing transition from the outer-layer states toward the inner-layer states. Here, there may be some states of the g^{th} layer such that in order to transit toward the inner-layer, the transition must pass through several other states of the same g^{th} layer. The result from theorem 2 shows that the series of discrete $\mathbf{x}_{dis}(t)$ never lead to 'outer' direction, and must lead toward 'inner' direction if there exist a path toward the inner layers. Therefore, from any discrete state, the optimal Q-learning policy must be able to transit the state toward the most-inner layer, or the 0^{th} layer. In other words, $\mathbf{x}_{dis}(T) = 0$, so $||\mathbf{x}(T)|| < \delta$ after some time T. By the stability assumption of (1), we know that $||\mathbf{x}(t)|| < \delta$.

4.3 Toy example results

4.3.1 Converging speed of selectively decentralized Q-learning

I perform experiments on several toy examples from the same class of system to show the superior converging speed of selectively decentralized Q-learning, compared to centralized Q-learning. In these examples, we examine the convergence from two points of view: the closeness of $\mathbf{x}(t)$ toward 0 and the magnitude of cumulative Q value increase. The difference in converging speed between the selectively decentralized Q-learning and the centralized Q-learning may grow exponentially with the number of subsystems.

The systems used in these examples are in the format

$$\mathbf{x}(t) = \sin(\mathbf{A}\mathbf{x}(t\text{-}1)) + \mathbf{u}(t) \quad (4.23)$$

where $\mathbf{A}$ are $N \times N$ random Markov matrices such that all diagonal entries share the same value. The vector sin function is defined from each dimension as

$$\sin(\mathbf{x}) = \begin{bmatrix} \sin(\mathbf{x}_1) \\ \sin(\mathbf{x}_2) \\ \vdots \\ \sin(\mathbf{x}_n) \end{bmatrix} \quad (4.24)$$

The coupling parameter σ on the non-diagonal entries is defined as

$$\sigma = \frac{\sum_{i \neq j} \mathbf{A}_{ij}}{\sum \mathbf{A}_{ij}} \quad \forall i, j \in [1, N] \quad (4.25)$$

In other words, σ is the ratio between the sum of non-diagonal entries in **A** and the sum of all entries in **A**. With $\sigma = 0$, **A** becomes the identity matrix or the systems are completely decouple. The systems are more couple when σ increases. For state and action variables, all state **x** and action **u** components have the range between -0.5 and 0.5 and initial state vectors **x**(0) are uniformly random numbers. For discretization (4.3-4.4), we choose $G = 5$. Therefore, $\theta_x = \theta_u = 0.2$. For Q-learning parameters (4.5-4.8), we choose $r = 0.01$, $\alpha = 0.1$ and $\gamma = 0.9$. I test system (4.24) with number of subsystems $N = 3$, 4, 5 and 6 and window size $w = 50$. For each choice of n, we repeat the experiments 100 times and report the average value due to the randomness of **A** and **x**(0).

Figures 4.2-4.5 highlight two significant advantages of selectively decentralized Q-learning, compared to centralized Q-learning. First, selectively decentralized Q-learning converges faster in both completely decouple systems and strongly couple systems. This fact suggests that selectively decentralized technique could be applied to many systems with wide-range of interconnection. Second, the converging time of selectively decentralized Q-learning grows much slower than the convergence time in centralized Q-learning. Due to the lack of space, we only draw the result when $N = 3$ and $N = 6$ to highlight the change in system dimensionality. As showed in figures 4.4 and 4.5, the centralized Q-learning does not converge within 100,000 iterations (or 2000 windows). From the characteristics of figure 4.2, we could still conclude that the centralized Q-learning in

figure 4.4 would converge if we continue the experiment with much larger number of iterations. Figures 4.6 and 4.7 show more details on how selectively decentralized Q-learning converges within the first few tens of windows.

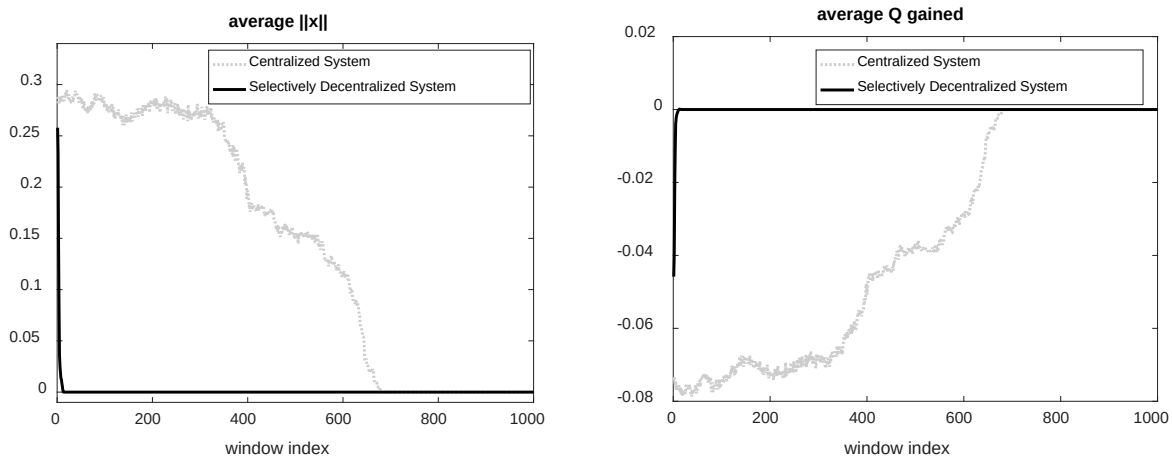

Figure 4.2. Comparison between centralized and selectively decentralized Q-learning in completely decentralized 3-subsystems.

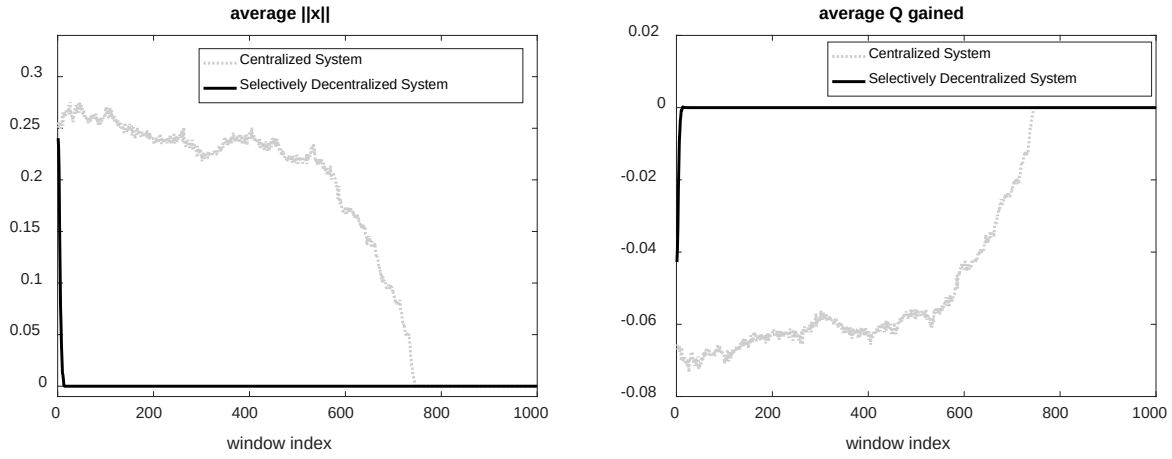

Figure 4.3. Comparison between centralized and selectively decentralized Q-learning in strongly couple ($\sigma = 0.5$) 3-subsystems.

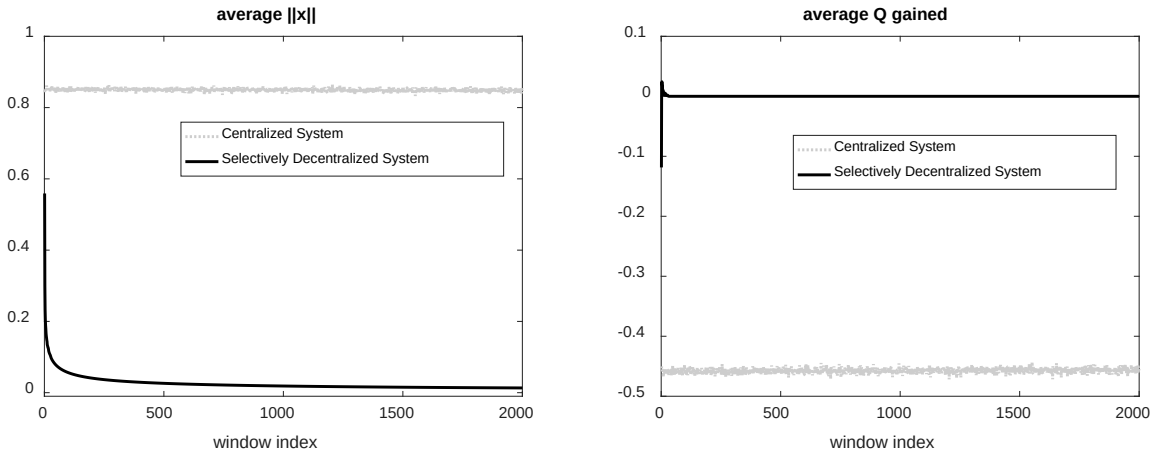

Figure 4.4. Comparison between centralized and selectively decentralized Q-learning in completely decouple 6-subsystems.

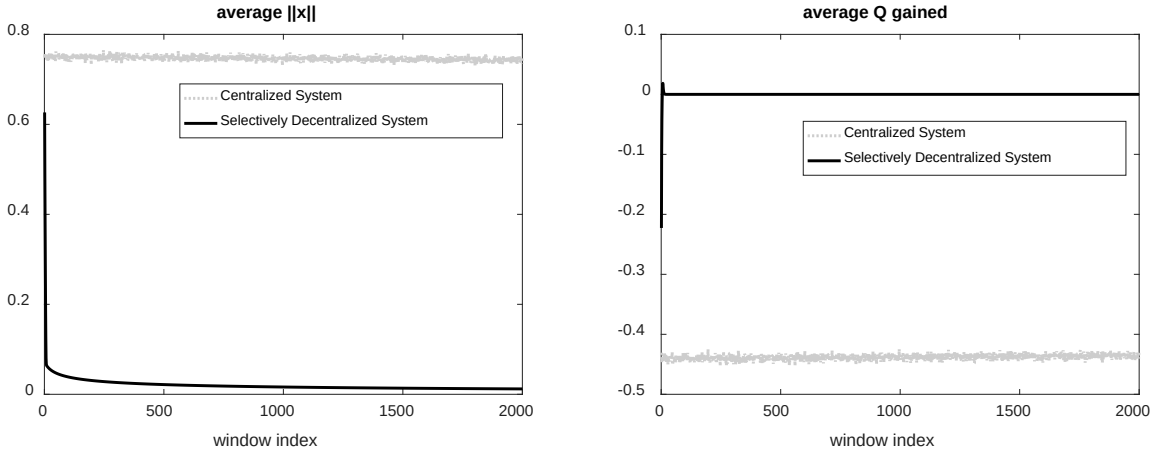

Figure 4.5. Comparison between centralized and selectively decentralized Q-learning in strongly couple ($\sigma = 0.5$) 6-subsystem.

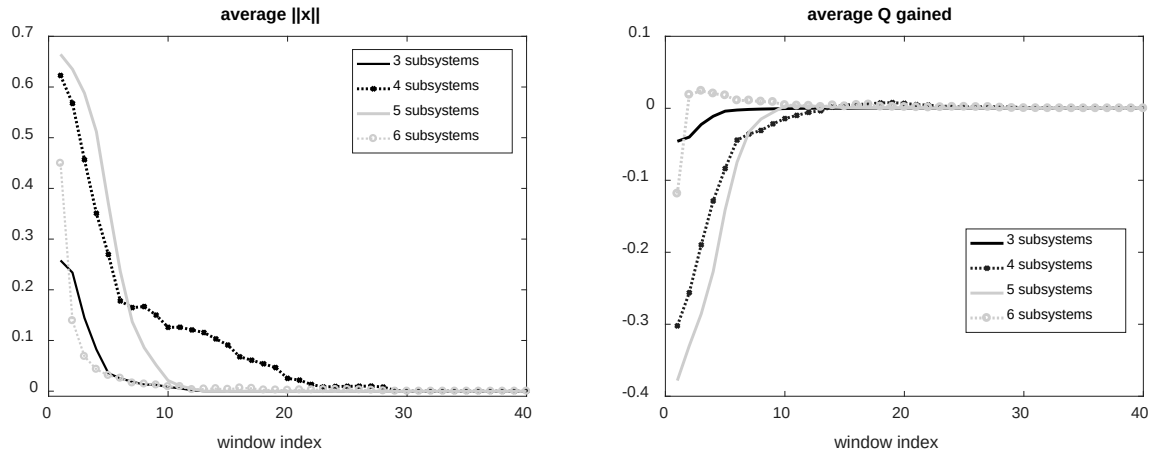

Figure 4.6. Convergence of selectively decentralized Q-learning in the first few of tens windows when the systems are completely decouple

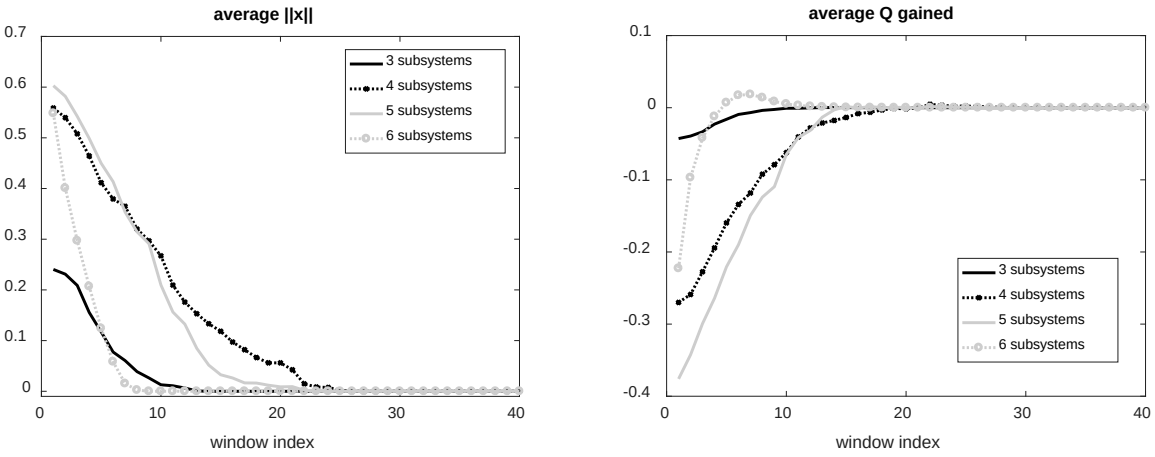

Figure 4.7. Convergence of selectively decentralized Q-learning in the first few of tens windows when the systems are strongly couple ($\sigma = 0.5$)

4.3.2 Switching among decentralization schemes

In Figure 4.8, we repeat all experiments in the previous section with $w = 1$ (the most frequent switching scenario) to show that selectively decentralized Q-learning will stop switching the 'best' decentralization scheme. Here, in order to compare with figure 4.6, we draw the average number of scheme switches for every 50 iterations/windows (to recall, in the previous section, we set $w = 50$). Comparing figure 4.6 and 4.7, in most of the cases, we observe that the point when number of scheme switches drop to 0 is earlier than the point when the selectively decentralized Q-learning

converges. This result may suggest that selectively decentralized Q-learning may learn the optimal communication policy during the optimal stabilization process.

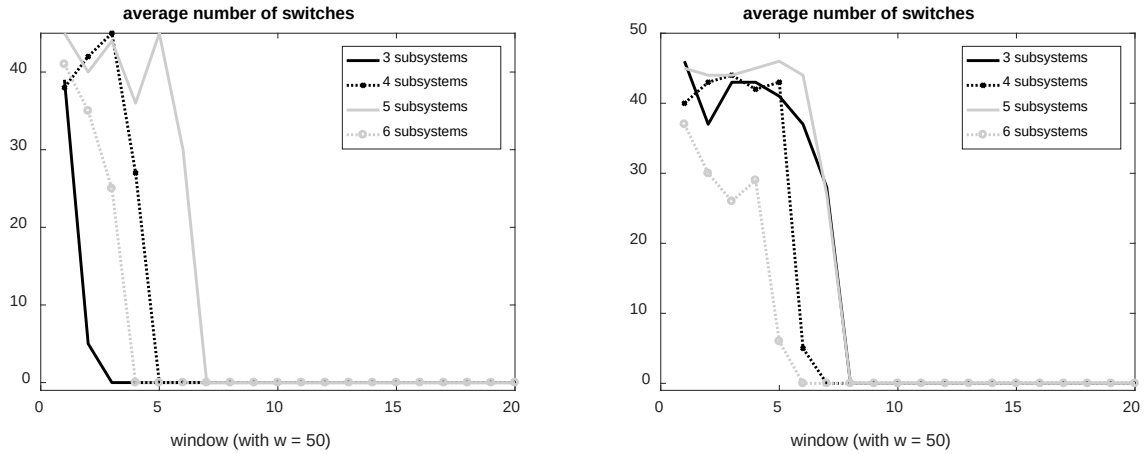

Figure 4.8. Average number of switches in selectively decentralized Q-learning. Left: completely decouple systems. Right: strongly couple ($\sigma = 0.5$) systems.

4.4 Discussions

Since the theoretical analysis of Q-learning, decentralized and distributed Q-learning mostly focuses on the existence of the optimal Q-value and the guarantee of reaching the optimal Q-value [26, 46, 88], we lack the theoretical explanation for the drastic superior converging speed of decentralized Q-learning. In this section, we explain the superior performance of selectively decentralized Q-learning from two points of view. First, as stated in the foundation of Q-learning [26], the convergence of Q-learning assumes that all of the state-action entries in the Q-table are visited infinitely. Therefore, in order to converge to the optimal Q, the Q-learning systems are supposed to spend time to explore the Q-table. In figures 4.2 and 4.3 where we show the convergence of centralized Q-learning, there are long periods where $||\mathbf{x}||$ and accumulate Q-gained change slowly. These periods may correspond to the exploration phases. Because the number of states, actions, and state-action entries grow exponentially with system dimensionality, decentralized Q-learning into smaller dimension may also improve the convergence exponentially

due to exponentially less search space. Second, selectively decentralized Q-learning proposes more search options than centralized Q-learning, which is another factor to improve the converging speed. In centralized Q-learning, a newly visited state has no prior information to estimate its Q-table entries. With the same state, in selectively decentralized Q-learning, the components of the state have higher chance to be visited by the subsystem learner (in different centralized states), which may reduce the effort to compute the optimal Q-value.

There are two major open questions in this chapter. First, although selectively decentralized Q-learning may offer exponential convergence measured by the number of data points/iterations, the number of decentralization schemes also grows exponentially with the dimensionality. Therefore, in practice, more refined techniques are needed to reduce the search of decentralization schemes. At this point, we believe that selectively decentralized Q-learning is practically useful because the best decentralization schemes stop switching after a few of tens windows (Figure 4.8). Second, we choose best decentralization scheme by the sum of subsystems' gained Q-values only because of the linearity in state-reward function, which is the main driver for Q-value update. However, there is no theoretical basis to support whether or not the different sum of subsystem gained Q-value in different decentralization scheme is comparable. There may exist more solid options for choosing the best decentralization scheme than cumulative gained Q-value.

5. SELECTIVELY DECENTRALIZED SYSTEM IDENTIFICATION

This chapter examines the impact of selective decentralization on system identification, which is the first step in model-based reinforcement learning. The better performance in identification is one of the fundamental reasons why selective decentralization improve the learning speed of the overall reinforcement learning – adaptive control problem. I show that selective decentralization can improve system identification in both linear and nonlinear system for most of the different subsystems' interconnection strength. I also address the identification converging speed, which has not been comprehensively studied before, and show that selective decentralization also outperforms the centralized system in converging speed. Overall, the improvement is more significant in nonlinear system identification. In linear system identification, at least the selective decentralization's performance is close to the centralization's performance.

In addition, compared to most of the state-of-the-art decentralized system identification techniques [89-91], I claim that the advantage of selective decentralization is the adaptability in switching and choosing the decentralization scheme to increase the identification performance. This advantage is important especially when the domain knowledge to precisely separate the entire system into subsystems is unknown or incomplete. By representing each dimension of the state vector as a subsystem, selective decentralization is able to learn the optimal decentralization scheme for identification, which may be used as an approximated method to detect the subsystems' components without domain knowledge.

This chapter is organized as follow. First, I present the problem statement and the pseudo code of selective decentralization for the system identification problem. Second, I review the identification

algorithms used in this chapter, for both linear and nonlinear cases. Third, I show some simulation results to demonstrate how selective decentralization could improve system identification.

5.1 Problem statement for selectively decentralized system-identification

5.1.1 Identification in unknown discrete-time invariant linear system

For linear system, we study the discrete-time invariant unknown system:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t\text{-}1) + \mathbf{B}\mathbf{u}(t\text{-}1) \quad (5.1)$$

where $\mathbf{x} \in \Re^N$ is the state vector, $\mathbf{u} \in \Re^M$ is the control input , $\mathbf{A} \in \Re^{N\times N}$ is the state-transition matrix, and $\mathbf{B} \in \Re^{N\times M}$ is the matrix of control unit. For identification problem, we assume that $\mathbf{A}$ is unknown. The objective is to find the approximation matrix $\widehat{\mathbf{A}} \in \Re^{N\times M}$ such that with the predicted state vector:

$$\hat{\mathbf{x}}(t) = \widehat{\mathbf{A}}\mathbf{x}(t\text{-}1) + \mathbf{B}\mathbf{u}(t\text{-}1) \quad (5.2)$$

the identification error

$$e(t) = |\mathbf{x}(t) - \hat{\mathbf{x}}(t)|^2 \quad (5.3)$$

approaches 0 as $t \to \infty$.

Let K be the number of subsystems in (5.1) with dimension $N_1, N_2, \ldots N_K$ such that $\sum_{i=1}^{K} N_i = N$. Decentralized identification computes $\widehat{\mathbf{A}}$ as the block-diagonal matrix

$$\widehat{\mathbf{A}} = \begin{bmatrix} \widehat{\mathbf{A}_1} & & & \\ & \widehat{\mathbf{A}_2} & & \\ & & \ddots & \\ & & & \widehat{\mathbf{A}_K} \end{bmatrix}$$

In the unknown system, we assume that we know the component of the system by the domain knowledge; but we do not know how the components interact with the others to form subsystems.

Therefore, we also aim to find the decentralized scheme of (5.1) in computing $\widehat{\mathbf{A}}$ such that $e(t)$ converges to 0 as quickly as possible.

5.1.2 Identification in unknown discrete-time invariant nonlinear system

For nonlinear system, we study the discrete-time invariant unknown system:

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (5.4)$$

where $\mathbf{x} \in \Re^N$ is the state vector, $\mathbf{u} \in \Re^m$ is the control input and $f \in \Re^{N+M} \to \Re^N$ is the nonlinear state transition function. We assume that f is completely unknown. The objective is to find the approximated nonlinear function $\hat{f}$ such that with the predicted state vector

$$\hat{\mathbf{x}}(t+1) = \hat{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad (5.5)$$

the identification error $e(t)$ approaches 0 as $t \to \infty$. We define the converging identification error and converging time concepts, which are the performance metrics, in section 5.3.

Similar to the linear system, let K be the number of subsystems in (5.5). Decentralized identification models $\hat{f}$ as follow

$$\hat{\mathbf{x}}(t+1) = \begin{bmatrix} \hat{\mathbf{x}}_1(t+1) \\ \hat{\mathbf{x}}_2(t+1) \\ \vdots \\ \hat{\mathbf{x}}_K(t+1) \end{bmatrix} = \hat{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} \hat{f}_1(\mathbf{x}_1(t), \mathbf{u}_1(t)) \\ \hat{f}_2(\mathbf{x}_2(t), \mathbf{u}_2(t)) \\ \vdots \\ \hat{f}_K(\mathbf{x}_k(t), \mathbf{u}_K(t)) \end{bmatrix} \quad (5.6)$$

5.1.3 Selective decentralization pseudo code

As a reminder in chapter 3, for the system of K component, B_K, the Bell's number of K [59], decentralization schemes cover all possible number of subsystems from 1 to K. A subsystem only uses its state and control variable to compute its own approximator. For example, with scheme $\{\{1, 2\}, \{3\}\}$, we have the format $\widehat{\mathbf{A}} = \begin{bmatrix} \widehat{\mathbf{A}}_{1,2} & \\ & \widehat{\mathbf{A}}_3 \end{bmatrix}$ for linear system and $\hat{f} = \begin{bmatrix} \hat{f}_{1,2} \\ \hat{f}_3 \end{bmatrix}$. In this

example, $\widehat{\mathbf{A}}_{1,2}$ and $\hat{f}_{1,2}$ are computed only using $\mathbf{x}_1$,$\mathbf{x}_2$, $\mathbf{u}_1$ and $\mathbf{u}_2$ according to formula (11) and backpropagation training algorithm, meanwhile $\widehat{\mathbf{A}}_3$ and $\hat{f}_3$ are computed only using $\mathbf{x}_3$ and $\mathbf{u}_3$.

Let Ω be the time-window size and w be the window index. Then the window w covers the discrete time index from $t = (w\text{-}1)\ \Omega + 1$ to $t = w\Omega$. Let $E(w)$ be the window-identification error at window w, which is the average of $e(t)$ from $t = (w\text{-}1)\ \Omega + 1$ to $t = w\Omega$. Let ε and γ be two small numbers for thresholds: ε decides the satisfactory identification error for termination and γ indicates whether or not the identification error no longer decreases. The pseudo code for selective decentralization is as follow

initialize b: the best decentralization scheme

for w from 1 to the maximum window index

Train approximator and compute $E(w)$ for $B(k)$ decentralization schemes

Select the decentralization scheme with the lowest $E(w)$ as b

if w > 1 and ($E(w) < \varepsilon$ or $|E(w) - E(w\text{-}1)| / |E(w)| < \gamma$)

return final identification error $E(w)$ and converging time $w\Omega$.

end if

end for

5.2 Reviews of system identification

5.2.1 Identification of linear time-invariant system

The theory for linear time-invariant system identification has been well-studied. The gradient decent is one of the most robust methods as shown in [8], which is summarized in equations (5.7-5.11). The gradient decent minimizes the second norm of the identification error

$$J = e(t)^2 = \left(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\right)^T \left(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\right) =$$

$$\left(\mathbf{x}(t) - \widehat{\mathbf{A}}\mathbf{x}(t-1) - \mathbf{B}\mathbf{u}(t-1)\right)^T \times \left(\mathbf{x}(t) - \widehat{\mathbf{A}}\mathbf{x}(t-1) - \mathbf{B}\mathbf{u}(t-1)\right) \quad (5.7)$$

Taking the derivative of J respect to $\widehat{\mathbf{A}}$ yields

$$\frac{\partial J}{\partial \widehat{\mathbf{A}}} = -\mathbf{x}(t-1)\left(\mathbf{x}(t) - \widehat{\mathbf{A}}\mathbf{x}(t-1) - \mathbf{B}\mathbf{u}(t-1)\right)^T \quad (5.8)$$

From the Newton-Raphson method, $\widehat{\mathbf{A}}$ could be iteratively updated as

$$\widehat{\mathbf{A}}(t) = \widehat{\mathbf{A}}(t-1) - \alpha \mathbf{H}(J, \widehat{\mathbf{A}}) \frac{\partial J}{\partial \widehat{\mathbf{A}}} \quad (5.9)$$

where $\mathbf{H}(J, \widehat{\mathbf{A}})$ is the Hessian matrix of J on $\widehat{\mathbf{A}}$ and α is the learning rate. A reasonable approximation of $\mathbf{H}(J, \widehat{\mathbf{A}})$ is

$$\mathbf{H}\left(J, \widehat{\mathbf{A}}\right) = \mathbf{x}(t-1)\mathbf{x}(t-1)^T + \mathbf{I} \quad (5.10)$$

where $\mathbf{I}$ is the identity matrix. Substituting (8) and (10) to (9) we have

$$\widehat{\mathbf{A}}(t) = \widehat{\mathbf{A}}(t-1) - \alpha \frac{\left(\mathbf{x}(t) - \widehat{\mathbf{A}}(\mathrm{t}-1)\mathbf{x}(t-1) - \mathbf{B}\mathbf{u}(t-1)\right)\mathbf{x}(t-1)^{\boldsymbol{T}}}{1 + \mathbf{x}(t-1)^{\boldsymbol{T}}\mathbf{x}(t-1)} =$$

$$\widehat{\mathbf{A}}(t-1) - \alpha \frac{\left(\mathbf{x}(t) - \hat{\mathbf{x}}(\mathrm{t})\right)\mathbf{x}(t-1)^{\boldsymbol{T}}}{1 + \mathbf{x}(t-1)^{\boldsymbol{T}}\mathbf{x}(t-1)} \quad (5.11)$$

Thus, (5.11) is could be executed iteratively, which is suitable for model identification and model-based system control problems.

5.2.2 Identification of nonlinear time-invariant systems

We use the three-layer feedforward neural network to identify *f* in the nonlinear system. Neural networks have been known for their capability to approximate a large and general class of nonlinear functions over compact domains. Theoretical foundation and application of neural network as such universal functional approximators in control systems can be found in [15, 76, 77]. We use the backpropagation training/learning algorithm for neural networks [92], in which $\{\mathbf{x}(t-1), \mathbf{u}(t-1)\}$ are presented at the input layer and $\hat{\mathbf{x}}(t)$ is computed at the output layer of the neural network identification model. In each training sample, the target is $\mathbf{x}(t)$. The remaining of this section is simply the recitation and more explanation for the neural network training presented in [92].

Mathematically, *i* be the node index at the input layer of $m + n$ nodes, *h* be the node index at the hidden layer of *d* nodes and *o* be the node index at the output layer of *m* nodes. Let $w_{IH}(i, h)$ be the input-hidden weight from input node *i* to hidden node *h*, Let $w_{HO}(h, o)$ be the input-hidden weight from hidden node *h* to output node *o*. From the input $\{\mathbf{x}(t-1), \mathbf{u}(t-1)\}$, the values at the hidden layer are computed as

$$H(h) = \sum_{i=1}^{m+n} w_{IH}(i,h) I(i) \quad (5.12)$$

where $H(h)$ denotes the hidden value at node *h* and $I(i)$ denotes the input value at node *i*. The estimated $\hat{\mathbf{x}}(t)$ at the output layer is computed as

$$O(o) = \sum_{h=1}^{d} w_{HO}(h,o) \times \sigma(H(h)) \quad (5.13)$$

Where $O(o)$ is the value at the o^{th} component of $\hat{\mathbf{x}}(t)$. Function σ, also called activation function, is defined as

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (5.14)$$

At each iteration t, starting with identification error at (5.3), which is also the error at the output layer, we use backpropagation [65] to train the neural network as follow. At the hidden layer

$$\frac{\partial e}{\partial w_{HO}(h,o)} = e_o(t)\times\sigma\big(H(h)\big) \quad (5.15)$$

in which $e_o(t)$ denotes the error at the o^{th} component of the output layer

$$e_o(t) = \mathbf{x}_o(t) - \hat{\mathbf{x}}_o(t) \quad (5.16)$$

The error at the hidden layer is carried from the error at the output layer by the hidden-output weight as follow, according to the chain rule

$$\frac{\partial e}{\partial w_{IH}(i,h)} = \big(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\big)^T \mathbf{w}_{HO}(h) \times \frac{\partial \sigma H(h)}{\partial w_{IH}(i,h)}$$

$$= \big(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\big)^T \mathbf{w}_{HO}(h) \times \sigma H(h) \times \big(1 - \sigma H(h)\big) \quad (5.17)$$

where $\mathbf{w}_{HO}(h)$ denotes the vector of all the hidden-output weights at hidden node h and $\big(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\big)$ is the vector of error at the output layer.

Thus, with the learning rate α, we update $w_{HO}(h, o)$ from (15) and (16) as

$$w_{HO}(h,o) = w_{HO}(h,o) + \alpha \frac{\partial e}{\partial w_{HO}(h,o)} = w_{HO}(h,o) + \alpha \times e_o(t)\times\sigma\big(H(h)\big) \quad (5.18)$$

and the input-hidden weights are updated as

$$w_{IH}(i,h) = w_{IH}(i,h) + \alpha \frac{\partial e}{\partial w_{IH}(i,h)} =$$

$$w_{IH}(i,h) + \alpha \times \big(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\big)^T \mathbf{w}_{HO}(h) \times \sigma H(h) \times \big(1 - \sigma H(h)\big) \quad (5.19)$$

5.3 Simulation results

5.3.1 Linear system identification

In this simulation, from (5.1), I setup systems of six dimensions for both **x** and **u** for the ease of decentralization. I choose k = 3, which means each subsystem covers two dimensions. The unknown transitional block matrix **A** is setup with real components {{1,2}, {3, 4}{5, 6}}, assumed to be the domain knowledge, as follow

$$\mathbf{A} = \begin{bmatrix} 0.6 & 0.4 & & & & \rho \\ 0.4 & 0.6 & & & & \\ & & 0.6 & 0.4 & & \\ & & 0.4 & 0.6 & & \\ \rho & & & & 0.6 & 0.4 \\ & & & & 0.4 & 0.6 \end{bmatrix} \qquad (5.20)$$

where the non-block entries of **A** is a random number between 0 and ρ. I choose **B** as the identity matrix. The control variables **u**(t), which is known by the system, is set randomly between -1 and 1. The initial **x**(0) is set to **1**. As shown in (20), σ decides the interconnection strength among system components. We call σ coupling parameter. For other parameters, we set α = 1.2, $\varepsilon = 10^{-3}$, $\gamma = 10^{-6}$ and Ω = 10. We run the identification process for at most 10000 iterations. Due to the randomness in **A** and **u**(t), we repeat the experimental process 100 times and report the average outcome for converging time and identification error.

Figure 5.1 and Figure 5.2 show that selective decentralization has superior performance when the coupling parameter is between 0 and 0.05, which may stand for the weakly coupled system. In these cases, selectively decentralized identification not only converges faster but also converges to lower identification error. The identification performance gap between selective decentralization and centralization decreases when the coupling parameter increases to 0.05. When

the coupling parameter is greater than 0.05, centralized identification starts showing superior performance. However, the selectively decentralized identification's performance is close to the centralized identification. To be more specific, Figure 5.3 and Figure 5.4 show more detail how the identification error converges to 0 after 300 windows, with the initial state vector $\mathbf{x}_1 = 1$.

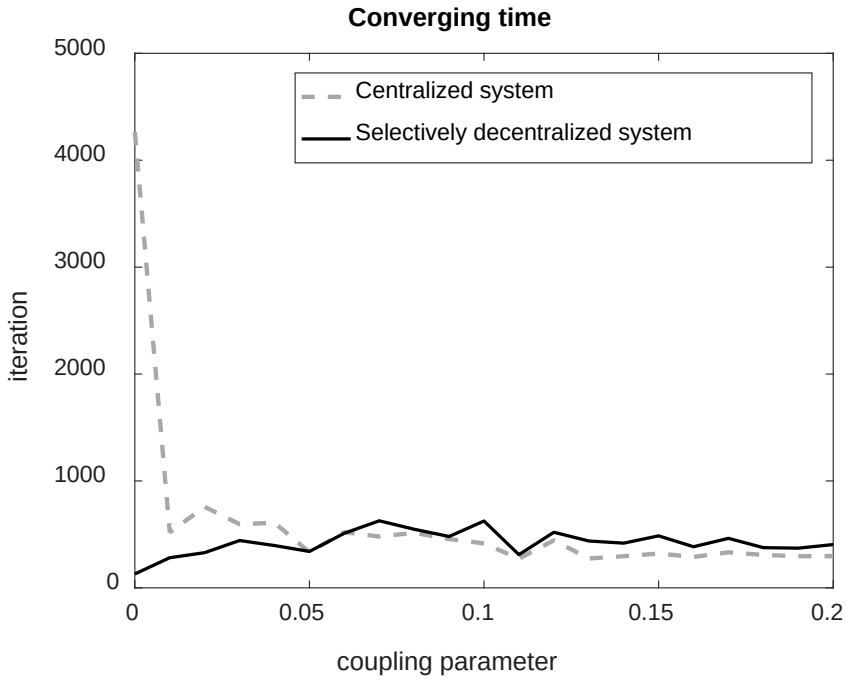

Figure 5.1. Comparison of converging time for identification between the centralized approach and the selectively decentralized approach in linear system

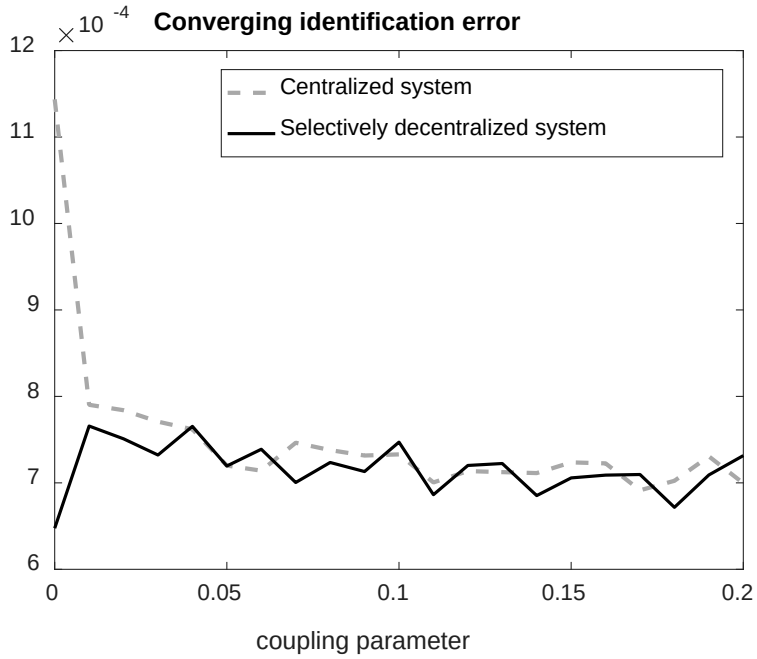

Figure 5.2. Comparison of converging identification error between the centralized model and the selectively decentralized model in linear system

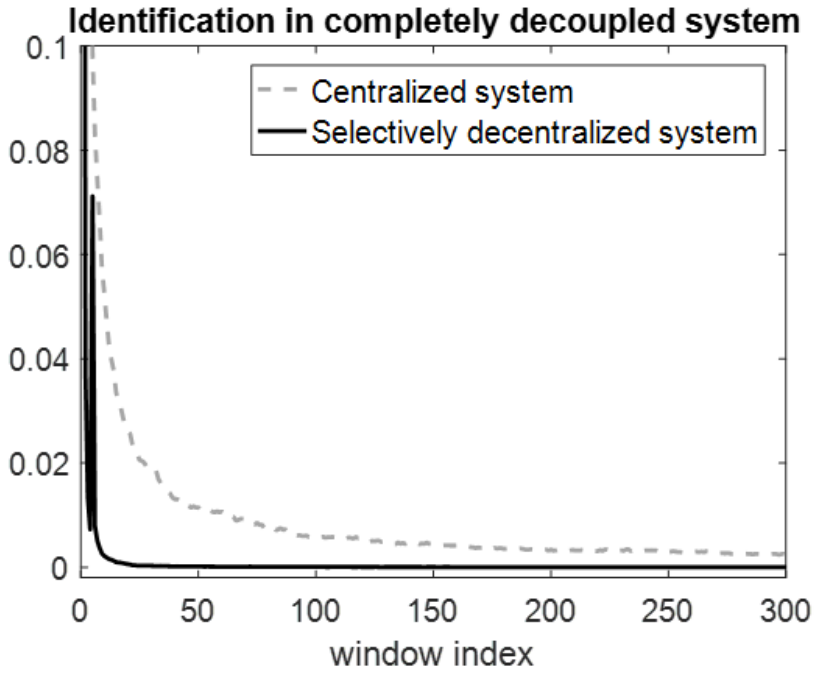

Figure 5.3. An example of how identification error converges to 0 with initial state $\mathbf{x}(0) = 1$ in the completely decoupled and linear systems ($\sigma = 0$)

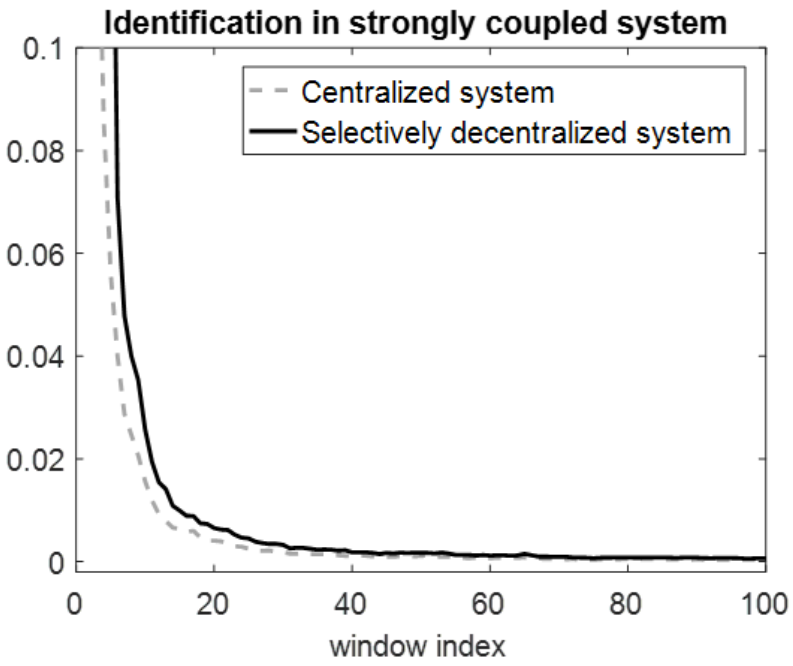

Figure 5.4. An example of how identification error converges to 0 with initial state **x**(0) = 1 in the strongly coupled and linear system ($\sigma = 0.2$)

5.3.2 Nonlinear system identification

In this simulation, we also setup systems of six dimensions for both **x** and **u** for the ease of decentralization. We also choose $K = 3$. With the same **A** matrix in the linear system simulation, we setup the nonlinear system as follow

$$x_i(t+1) = \frac{2\mathbf{A}_i\mathbf{x}(t) + u_i(t)}{1 + (\mathbf{A}_i\mathbf{x}(t) + u_i(t))^2} \quad \forall i \in [1,6] \quad (5.13)$$

where i is the state-control variable index, $\mathbf{A}_i$ is the i^{th} row of **A**. Here, the real subsystem components are {{1,2}, {3, 4},{5, 6}}. Similar to the linear system simulation, it can easily be seen that the coupling parameter σ decides the interconnection strength among system components. For identification, we setup three-layer neural networks in each group of subsystem $\{i_1, i_2, \ldots, i_l\}$ with $\{\mathbf{x}_{i1,i2,\ldots il}(t\text{-}1), \mathbf{u}_{i1,i2,\ldots il}(t\text{-}1)\}$ in the input layer, $\mathbf{x}_{i1,i2,\ldots il}(t)$ in the output layer and 50 nodes in the hidden layer. The learning rate for the neural networks is set to 0.5. For other parameters, we set $\varepsilon = 0.05$, $\gamma = 10^{-6}$ and $\Omega = 50$. Similar to the linear system case study, the initial state **x**(0) is set

to 1. We run the identification process for at most 10000 iterations. Due to the randomness in **A** and **u**(*t*), we repeat the experimental process 100 times and report the average outcome for converging time and identification error.

Figure 5.5 and Figure 5.6 show that for all of coupling parameters, selective decentralization outperforms centralizations in both identification converging speed and identification error. For all of the coupling parameters in our experiment, the centralized identification error does not reach the satisfactory threshold $\varepsilon = 0.05$ (it may reach the threshold after more iterations). Therefore, figure 5 shows the converging time of centralized identification close to 10000, which is the maximum number of iterations in our experiment. Figure 5.7 and Figure 5.8 show more detail how the identification error converges after 200 windows.

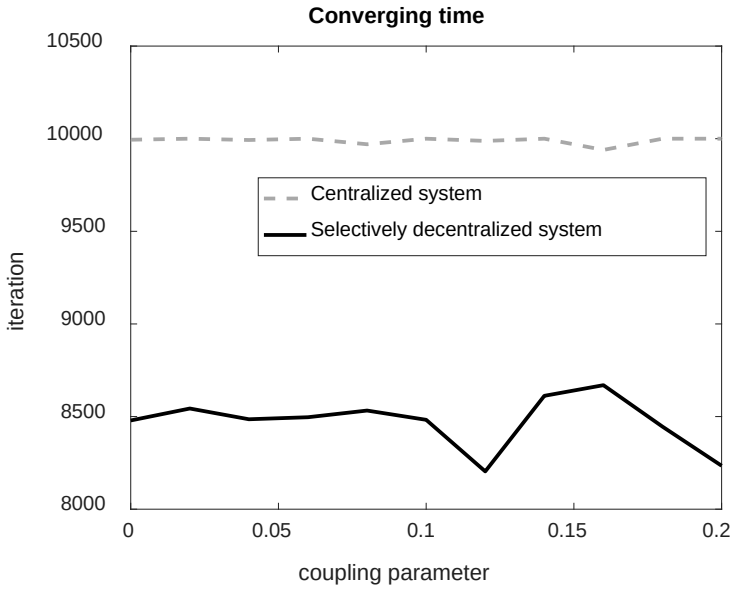

Figure 5.5. Comparison of converging time for system identification between the centralized approach and the selectively decentralized approach in nonlinear system

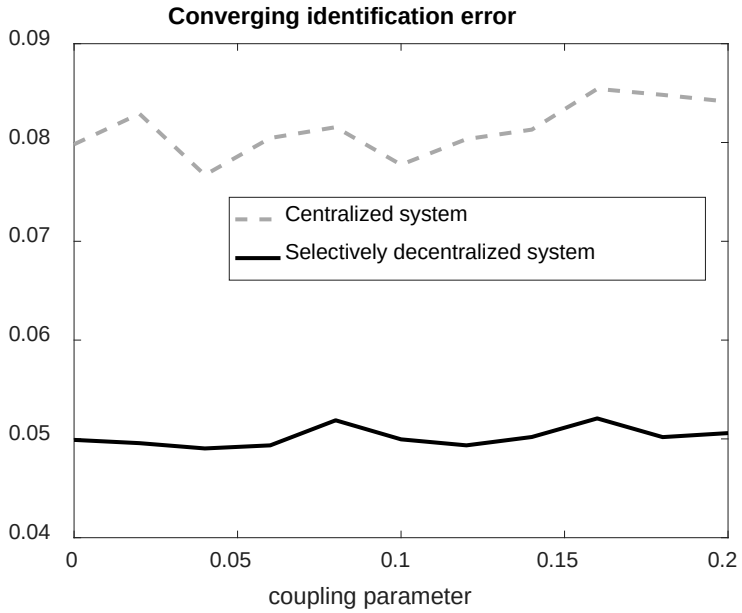

Figure 5.6. Comparison of converged identification error between the centralized model and the selectively decentralized model in a nonlinear system

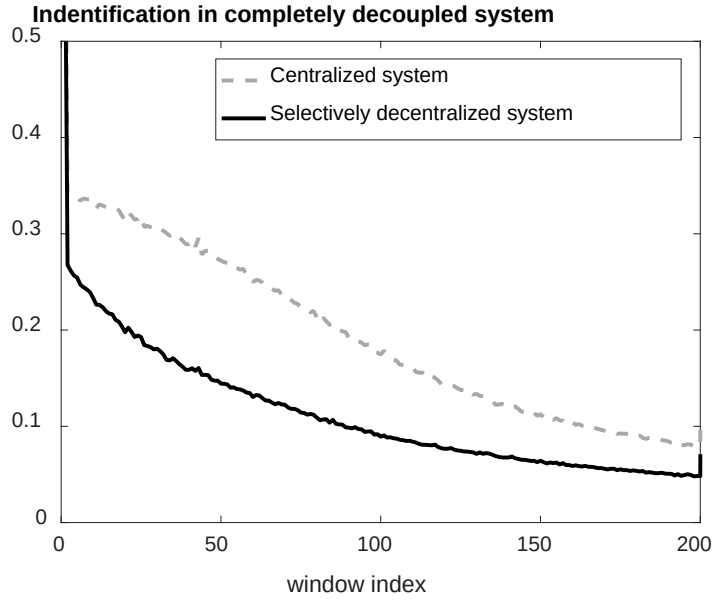

Figure 5.7. An example of how identification error converges with initial state $\mathbf{x}(0) = 1$ in the completely decoupled and nonlinear system case

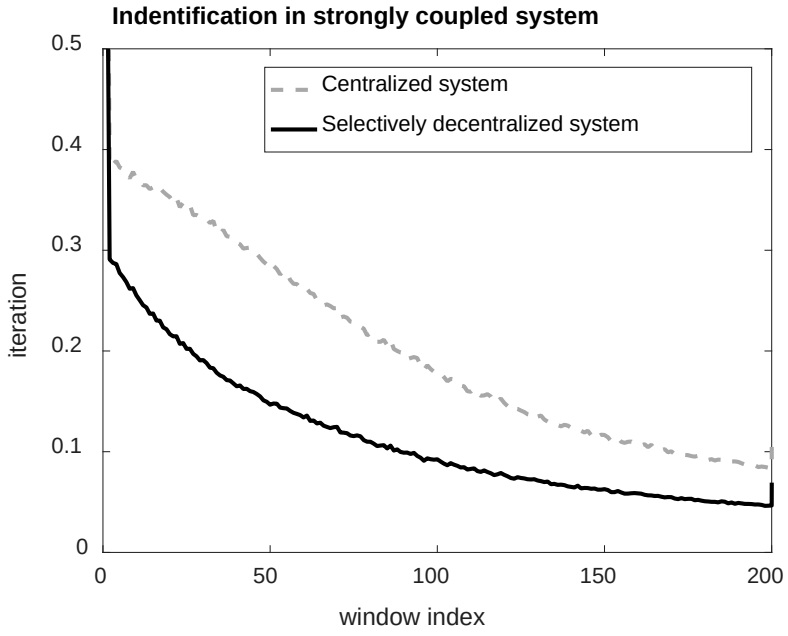

Figure 5.8. An example of how identification error converges with initial state **x**(0) = 1 in the strongly coupled and nonlinear system ($\sigma = 0.2$)

6. SELECTIVELY DECENTRALIZED LEARNING AND CONTROL WITH DISCRETIZED MDP

This chapter completes the impact of selective decentralization in improving model-based adaptive and optimal control, which is also equivalent to reinforcement learning. This chapter combines the success of selective decentralization in system identification, the well-known closed-form solution of Riccati equation in linear system, the discrete-MDP approximation (presented in chapter 2). We show that selective decentralization can improve the learning performance in both linear and nonlinear systems with several levels of interconnection among subsystems. Here, we measure the performance on the number of iterations, or samples, needed in learning. This measurement of performance is useful for problems in which the number of training samples is limited. In addition, we show that the discrete-MDP technique could help in learning nonlinear control problem in general form.

The content in this chapter is organized as follow. First, we rewrite the problem statement for the complete model-based reinforcement learning problem. Second, we present the two-phase framework to solve this problem, which includes the content in the previous chapters: selective decentralization, MDP, HJB equation and system identification. Third, we demonstrate the entire framework in some toy examples.

6.1 Problem statement for model-based reinforcement learning

In this chapter, we focus on discrete time, continuous-state, time-invariant system in the general format

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (6.1)$$

Where $\mathbf{x} \in \Re^N$ stands for the N dimensional bounded state vector, $\mathbf{u} \in \Re^M$ stands for the M dimensional bounded control unit, t stands for the iteration number, $\mathbf{x}(0)$ is given and $f: \Re^N \times \Re^M \to \Re^N$ is an continuously differentiable unknown function. Here, the symmetric boundaries $[-\chi, \chi]$ and $[-\mu, \mu]$ for all components of $\mathbf{x}$ and $\mathbf{u}$ are known. Let $p: \Re^N \to \Re$ and $q: \Re^M \to \Re$ be the two continuously semi-definite negative and differentiable reward functions with the following properties

$$p(\mathbf{x}_1) \leq p(\mathbf{x}_2) \Leftrightarrow \|\mathbf{x}_1\| \geq \|\mathbf{x}_2\| \text{ and } p(\mathbf{0}) = 0 \quad (6.2)$$

$$q(\mathbf{u}_1) \leq q(\mathbf{u}_2) \Leftrightarrow \|\mathbf{u}_1\| \geq \|\mathbf{u}_2\| \text{ and } q(\mathbf{0}) = 0 \quad (6.3)$$

where $\|\mathbf{x}\|$ denotes the second norm of $\mathbf{x}$. The main objective is to learn the control unit $\mathbf{u}$ such that

$$\mathbf{x}(t) \to 0, \mathbf{u}(t) \to 0 \text{ as } t \to \infty \quad (6.4)$$

To formulate an adaptive optimal control, or reinforcement learning problem, we convert the objective in (4) into a more formal control problem with discount factor $0 < \gamma \to 1$

$$J(\mathbf{x}_0) = \sum_{t=0}^{\infty} \gamma^t \left(p\big(\mathbf{x}(t)\big) + q\big(\mathbf{u}(t)\big)\right) \quad (6.5)$$

Thus, the goal is to optimize $J(\mathbf{x}_0)$. The function $J(\mathbf{x})$ defined in (6.5) is called the state value function. Since f is unknown, in the model-based approach, the intermediate goal is to find the approximated nonlinear function $\hat{f}$ such that with the predicted state vector

$$\hat{\mathbf{x}}(t+1) = \hat{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad (6.6)$$

the identification error

$$e(t) = \| \mathbf{x}(t) - \hat{\mathbf{x}}(t) \| \quad (6.7)$$

approaches 0 as $t \to \infty$.

6.2 Two-phase selective decentralized control framework

Figure 6.1 shows the design of the learning control system in this work with two phases: identification and control. In the identification phase, we train the neural networks to acquire the function approximators $\hat{f}$ from using <**x**(t), **u**(t)> as the input tuples and **x**(t+1) as the outputs. In the control phase, to compute the near-optimal control, we use DARE algorithm [61], and policy iteration algorithm for the nonlinear system after setting up the corresponding MDP [65]. Here, the window size parameter Ω decides how frequently we call the identification phase. In other words, Ω decides the number of <**x**(t), **u**(t), **x**(t+1)> tuples to train $\hat{f}$.

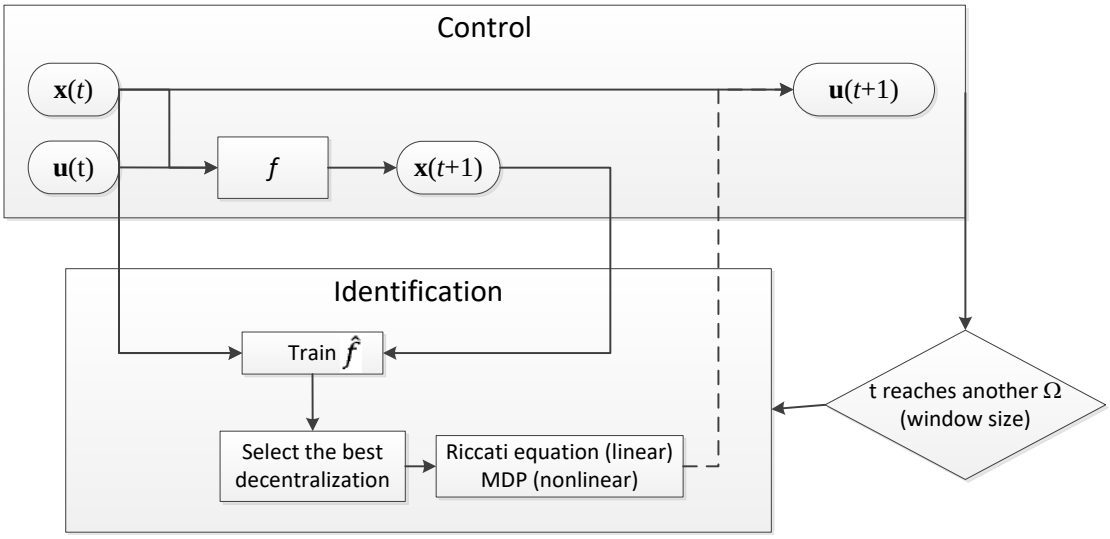

Figure 6.1. The learning design for selective decentralized reinforcement learning.

Let w be the window index. Then the window w covers the discrete time index from $t = (w\text{-}1)\,\Omega + 1$ to $t = w\Omega$. Let $E(w)$ be the window-identification error at window w, which is the average of $e(t)$ from $t = (w\text{-}1)\,\Omega + 1$ to $t = w\Omega$. Let ε and γ be two small numbers for thresholding. For the system of r components/agents, the number of decentralization scheme is $B(r)$, which is the r^{th} Bell's number [59]. The pseudo code for selective decentralization is as follow:

initialize b: the best decentralization scheme

for *w* from 1 to the maximum window index

***calculate* control policy using b** *b* (using Riccati equation for linear system and discrete-MDP for nonlinear system)

Train approximator and compute $E(w)$ for $B(r)$ decentralization schemes

Select the decentralization scheme with the lowest $E(w)$ as *b*

if w > 1 and ($E(w) < \varepsilon$ or $|E(w) - E(w\text{-}1)| / |E(w)| < \gamma$)

return

end if

end for

6.3 Simulation results

6.3.1 Linear system

In this simulation, we setup a system of 8-dimmension state and control variables with $r = 4$. The unknown transitional block matrix **A** is setup with real subsystem components {{1,2}, {3, 4},{5, 6}, {7,8}}as follow. With raw matrix $\widetilde{\mathbf{A}}$ matrix as

$$\widetilde{\mathbf{A}} = \begin{bmatrix} 0.7 & 0.3 & & & & & & \\ 0.2 & 0.8 & & & & & & \\ & & 0.23 & 0.77 & & & \sigma & \\ & & 0.4 & 0.6 & & & & \\ & \sigma & & & 0.5 & 0.5 & & \\ & & & & 0.35 & 0.65 & & \\ & & & & & & 0.9 & 0.1 \\ & & & & & & 0.15 & 0.85 \end{bmatrix} \quad (6.8)$$

where the non-block entries of $\widetilde{\mathbf{A}}$ are a random numbers between 0 and σ. To avoid numerical overflowing, we normalize $\widetilde{\mathbf{A}}$ into **A** such that **A** is a Markov matrix. The reward functions are $p(\mathbf{x}) = -\mathbf{x}^T\mathbf{x}$ and $q(\mathbf{u}) = -\mathbf{u}^T\mathbf{u}$. The discount reward factor in (5) is $\gamma = 0.9$. The initial control variables $\mathbf{u}(0)$ and state variable $\mathbf{x}(0)$ are set randomly between -1 and 1. As shown in (6.8), σ decides the

interconnection strength among system components. We call σ coupling parameter. We setup the completely decouple system by setting σ = 0 and the strongly couple system by σ = 0.1. We set **B** as the identity matrix. For identification, we set α = 0.5. At the starting point, we set **x**(0) as a vector of random numbers between -1 and 1. Because calculating $\hat{\mathbf{A}}$ using linear system identification (see chapter 5) is relatively simple, we set the window size Ω = 1. We repeat this setup 100 times since **A** and **x**(0) contains random parameters and report the mean statistics.

In Figures 6.2 and 6.3, we observe that the selectively decentralized system shows better control performance than the completely decentralized system and the centralized system. In these figures, we draw the y-axis in log scale since both **x** and **u** converges to 0 so quickly that the linear-scale plot could not show the difference. We use norm(x) and norm(u) to denote the second-norm of **x** and **u**, correspondingly. Clearly, after more than 30 iterations, both **x** and **u** in the completely decentralized system converge to 0 faster than they are in the completely decentralized system and the centralized system. At the first few iterations, the selectively decentralized system shows slightly poorer control performance. This may due to the complexity of the selectively decentralized system in identifying unknown **A**. In the other hands, as the systems are more coupled, we see that the performance gap between the decentralized systems and the centralize system is less.

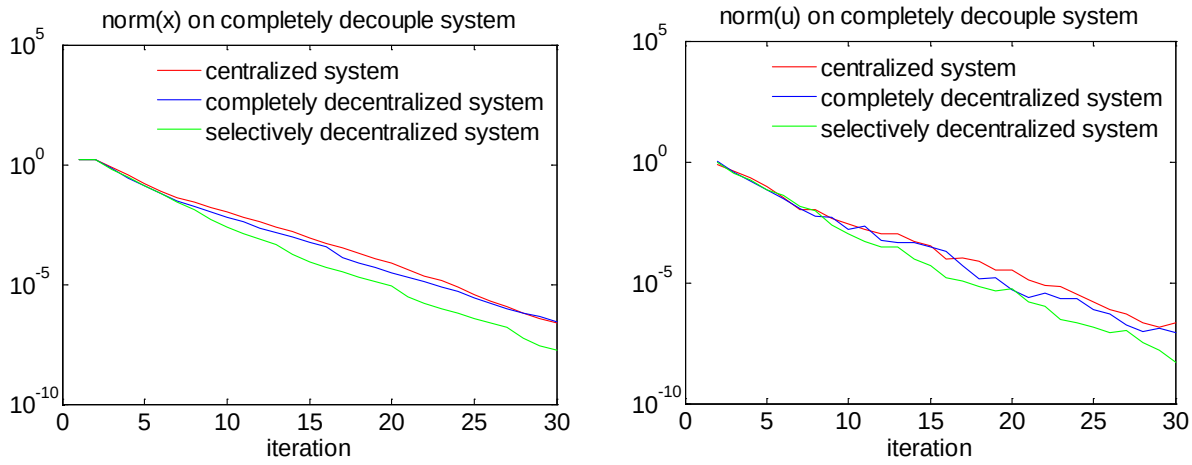

Figure 6.2. Comparison of control performance among the centralized system, the completely decentralized system and the selectively decentralized system when the system is linear and completely decoupled.

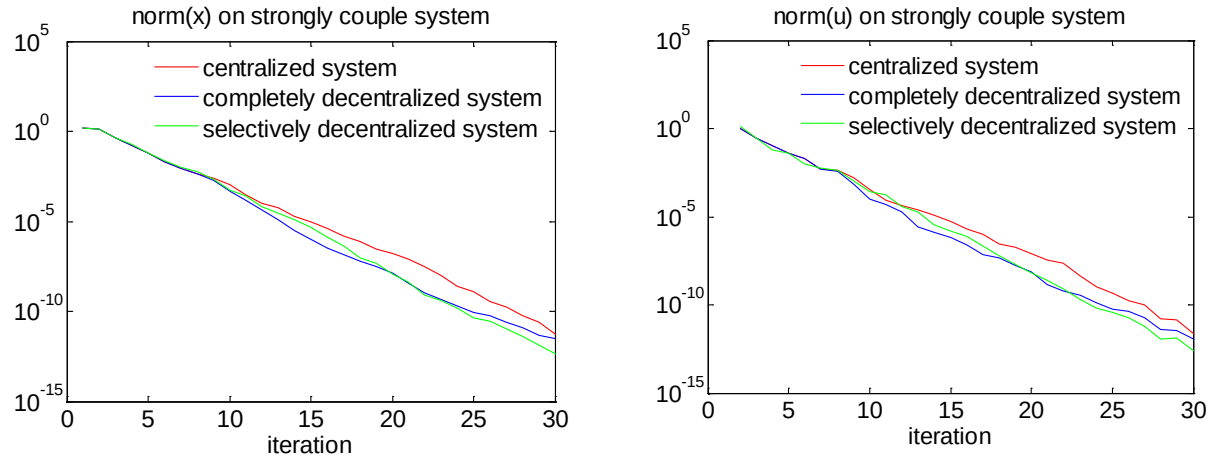

Figure 6.3. Comparison of control performance among the centralized system, the completely decentralized system and the selectively decentralized system when the system is linear and strongly coupled.

6.3.2 Nonlinear system

In this example, we choose the system

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) = sin\big(\mathbf{A}\mathbf{x}(t) + \mathbf{u}(t)\big) \quad (6.9)$$

where $\mathbf{x}$, $\mathbf{u} \in R^4$, matrix $\mathbf{A}$ is defined by normalizing $\widetilde{\mathbf{A}}$ into a Markov matrix where

$$\widetilde{\mathbf{A}} = \begin{bmatrix} 0.7 & 0.3 & & \sigma \\ 0.2 & 0.8 & & \\ & & 1 & \\ \sigma & & & 1 \end{bmatrix} \quad (6.10)$$

and the *sin* function is defined as

$$sin(\mathbf{x}) = \begin{bmatrix} \sin(x_1) \\ \vdots \\ \sin(x_n) \end{bmatrix} \quad (6.11)$$

and $\mathbf{x}(0) = \mathbf{0.2}$. Here, we assume that the boundary of **x** an **u** is known as $-0.2 \leq x_i, u_i \leq 0.2 \; \forall i \in [1,4]$ and the real subsystem component in (1) is {{1,2}, {3}, {4}}. The reward functions are $p(\mathbf{x}) = -\mathbf{x}^T\mathbf{x}$ and $q(\mathbf{u}) = -\mathbf{u}^T\mathbf{u}$. The discount reward factor in (6.5) is $\gamma = 0.9$.

For system approximation, we use a three-layer neural network with 30 hidden units, sigmoid activation function and backpropagation to train the neural network for $\hat{f}$. For each training step, we pass the training sample set $<\mathbf{x}(t), \mathbf{u}(t)>$ 2000 times. We set window size $\Omega = 50$ (figure 6.1). Similar to the linear system case study, we setup the completely decouple system by setting $\sigma = 0$ and the strongly couple system by $\sigma = 0.1$. In each state and control vector dimension, we divide the dimension into $G = 8$ regions, which makes the resolution threshold (see chapter 2) 0.05.

In Figures 6.4, we observe that the selectively decentralized system shows better control performance than the completely decentralized system and the centralized system. Similar to figures 6.2 and 6.3, we use norm(x) to denote the second-norm of **x**. Here, we observe that when the system is completely decouple, the centralized system converges to 0 significantly slower than the selectively decentralized system does. In addition, when the system is strongly couple, the centralized system fails to control.

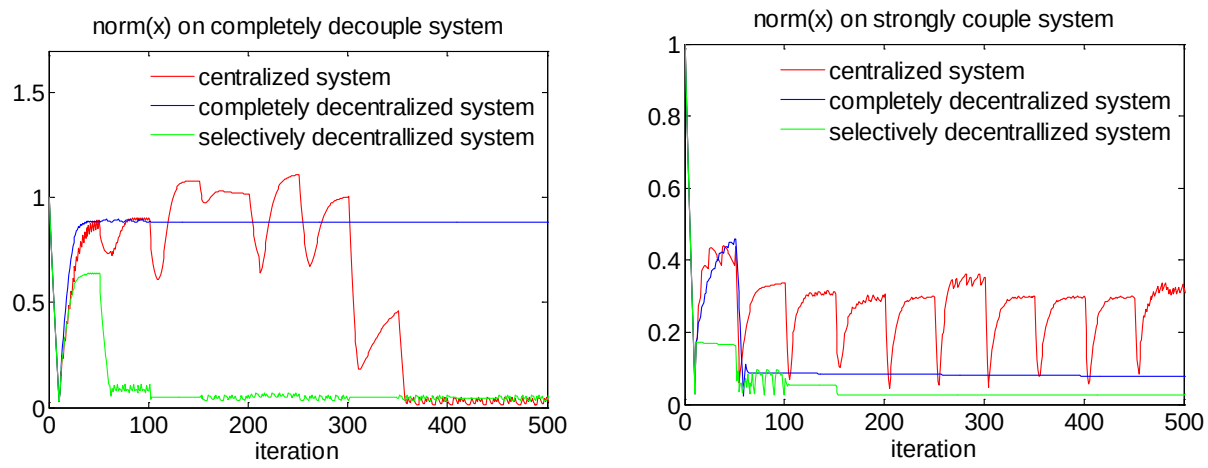

Figure 6.4. Comparison of control performance among the centralized system, the completely decentralized system and the selectively decentralized system when the system is nonlinear with different coupling.

6.4 Discussions

In this chapter, we show that selective decentralization can improve the learning performance in both linear and nonlinear systems with several levels of interconnection among subsystems. Here, we measure the performance on the number of iterations, or samples, needed in learning. This measurement of performance is useful for problems in which the number of training samples is limited. In addition, we show that the discrete-MDP technique could help in learning nonlinear control problem in general form.

There are several limitation in this chapter. First, the discretization thresholds need the distribution of the next state assuming that the current state and control vectors are uniformly distributed and may require a number of ad-hoc steps. Third, in selective decentralization, we still explore all possible decoupling scheme $B(k)$, which grows exponentially. However, since the selectively decentralized system converges faster than the centralized system in most of the cases, we believe that the heavily computational model-switching phase in the selective decentralized system will be relatively short. Therefore, the selectively decentralized system may be more computationally

efficient than the centralized system, which must run the learning algorithm in high dimensional data for long term.

7. MULTIDISCIPLINARY OPTIMIZATION IN DECENTRALIZED REINFORCEMENT LEARNING

In this chapter, we propose applying the **m**ulti**d**isciplinary **o**ptimization (MDO) idea in solving decentralized RL problems and demonstrates the capability of the MDO approach in several learning adaptive control toy examples. Fundamentally, the MDO approach, which could be categorized as a partial communication technique [48], is a compliment to the selective decentralization. Here, we use nonlinear system identification to approximate the unknown environment in the RL problem. Hence, in this chapter, we use the terms 'environment' and 'system' interchangeably. In the system identification step, each learning agent also identifies the impact of other agent's information on its learning performance, which is the central theme in MDO. From the identified model, each learning agent setups Markov decision process (MDP) to compute the action/control solution, which has been shown in [93]. In this step, we examine and compare the control solutions computed using both the MF and the IDF options. In addition, we examine the exchanged information among the agents, which is another characteristic to categorize the recent state-of-the-art MDO techniques. We focus on the question: how much the learning performance loss when the exchanged information among the agents is simplified. Then, we compare the learning performance of the MDO approach with the strictly decentralized approach, the selectively decentralized approach [93] and the centralized approach and discuss the advantage of the MDO in our learning examples.

Mathematically, MDO [94, 95], which has been intensively researched and applied in aerospace and engineering, could be a promising approach to tackle the first two challenges in decentralized RL. In MDO, the computational agents are well-defined and decomposed according to the domain-

knowledge of each discipline in a jointed optimization problem. For example, in aero-elastic optimization, there are two decentralized computational units: the aero dynamic units applies fluid dynamics law to manage the air-pressure on the aircraft wing and the structure unit applies the material law to manage the deflection and shape of the wing [38]. In the formulation step, there are two typical options. First in the *multidisciplinary feasible* (MF) option (which may also called 'boarder sense' option), each computational agent incorporate the information from the other agents in its own optimization function [38]. In addition, each agent only uses the optimization constrains from its own discipline. In this option, because the each agent includes the information from other agents in the optimization objective, the agent tends to approach closer to the global optimal solution even though the global problem is much more complex than its own capacity. Second, in the *individual discipline feasible* (IDF) option (which may also called the 'selfish' option), each computational agent only aims to optimize its own optimization function and uses the other agents' information as constrains. In this option, the agents tends to seek for local optimization; and the constrains from other agents will drive the local solutions to the global solution. The exchanged information among the agents could be preprocessed or transformed into simpler forms to reduce the complexity of the optimization in each agent. More literature details about MDO approach could be found in [96-98].

However, according to our best knowledge, MDO has not been widely applied in decentralized RL. In our opinion, there are two factors limiting the capability of the MDO in RL. First, the unknown nature and long-term goal of the RL problems implies that we could not get the optimization function in closed-form $J(\mathbf{x})$, where J denotes the optimization objectives in RL and $\mathbf{x}$ denotes the variables in the RL problem. Since MDO techniques rely on numerical methods -

especially the gradient methods - to solve the optimization problem, without a closed-form of $J(\mathbf{x})$ [94], most of the state-of-the-art MDO techniques are inapplicable. Second, even when the unknown nature of the RL problems could be solved by system identification methods [8], the closed-form solution for many of the Hamilton-Jacobi-Bellman equation, which is the central theory behind most of the RL problems, is very difficult to find.

This chapter is organized as follow. First, for the reading convenience, we restate the statements of the learning - adaptive control problem, as already mentioned in chapter 6. Second, we carefully state the assumption for the MDO learning agents. Third, we present the MDO learning agents using the same two-phase paradigm to selectively decentralized learning agents: identification and discrete-MDP control. And forth, we show the simulation results, which compare the performance among several approaches, including centralized learning, completely decentralized learning, selectively decentralized learning and MDO learning.

7.1 Problem statements

7.1.1 The learning adaptive control problem

In this chapter, we focus on discrete time, continuous-state, time-invariant system in the general format

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (7.1)$$

where $\mathbf{x} \in \Re^N$ stands for the N dimensional bounded state vector, $\mathbf{u} \in \Re^M$ stands for the M dimensional bounded control unit, t stands for the iteration number, $\mathbf{x}(0)$ is given and $f: \Re^N \times \Re^M \to \Re^N$ is an continuously differentiable unknown function. Here, the symmetric boundaries $[-\chi, \chi]$ and $[-\mu, \mu]$ for all components of $\mathbf{x}$ and $\mathbf{u}$ are known. Given $p: \Re^N \to \Re$ and $q: \Re^M \to \Re$ as the two

continuously semi-definite negative and differentiable reward functions with the following properties

$$p(\mathbf{x}_1) \leq p(\mathbf{x}_2) \Leftrightarrow \|\mathbf{x}_1\| \geq \|\mathbf{x}_2\| \text{ and } p(\mathbf{0}) = 0 \quad (7.2)$$

$$q(\mathbf{u}_1) \leq q(\mathbf{u}_2) \Leftrightarrow \|\mathbf{u}_1\| \geq \|\mathbf{u}_2\| \text{ and } q(\mathbf{0}) = 0 \quad (7.3)$$

where $\|\mathbf{x}\|$ denotes the second norm of $\mathbf{x}$. The main objective is to learn the control unit $\mathbf{u} = u(\mathbf{x})$ such that

$$\mathbf{x}(t) \to 0, \mathbf{u}(t) \to 0 \text{ as } t \to \infty \quad (7.4)$$

For optimally adaptive control, we convert the objective in (7.4) into the optimization objective with discount factor $0 < \gamma \to 1$

$$J(\mathbf{x}(0)) = \sum_{t=0}^{\infty} \gamma^t \left(p\big(\mathbf{x}(t)\big) + q\big(\mathbf{u}(t)\big)\right) \quad (7.5)$$

The equations (7.1-7.5) defines a Hamilton-Jacobi-Bellman equation. Since (7.1) is in general form, we assume that the closed-form solution for (7.1-7.5) is unknown.

7.1.2 The system identification problem statements

Since we use MDP, a model-based method, to compute $\mathbf{u}$, we need an approximation of the environment. The objective is to find the approximated nonlinear function $\hat{f}$ such that with the predicted state vector

$$\hat{\mathbf{x}}(t+1) = \hat{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad (7.6)$$

the identification error

$$e(t) = |\mathbf{x}(t) - \hat{\mathbf{x}}(t)|^2 \quad (7.7)$$

approaches 0 as $t \to \infty$.

7.2 Key assumptions for the MDO agents

To apply MDO, we assume that the system (7.1) could be decouple into multiple subsystems by domain knowledge. Each subsystem correspond to one learning agent. More importantly, we assume that each agent know precisely which components of **x** and **u** affecting its learning performance. In addition, we assume that each agent's control unit does not directly and instantly affect the **x** components of the other agents. In the other words,

$$\mathbf{x}(t+1) = \begin{bmatrix} \mathbf{x}_1\ (t+1) \\ \mathbf{x}_2\ (t+1) \\ \vdots \\ \mathbf{x}_K\ (t+1) \end{bmatrix} = f\ (\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} f_1\ (\mathbf{x}_1(t), C_1(\mathbf{x}_{i\neq 1}(t)), \mathbf{u}_1(t)) \\ f_2\ (\mathbf{x}_2(t), C_2(\mathbf{x}_{i\neq 2}(t)), \mathbf{u}_2(t)) \\ \vdots \\ f_K\ (\mathbf{x}_k(t), C_K(\mathbf{x}_{i\neq k}(t)), \mathbf{u}_K(t)) \end{bmatrix} \quad (7.8)$$

where k is the number of learning agents, i stands for agent index and C is the bounded communication function, which is known, among the agents. We assume that for all agents, C has the following property

$$\| C_j(\mathbf{x}_{i\neq j}(t)) \| \geq \| C_j(\mathbf{x'}_{i\neq j}(t)) \| \Leftrightarrow \| \mathbf{x}_{i\neq j}(t) \| \geq \| \mathbf{x'}_{i\neq j}(t) \| \quad (7.9)$$

In general, for agent j, $C_j(\mathbf{x}_{i\neq j}(t))$ should be simpler than $\mathbf{x}_{i\neq j}(t)$, such as having less dimension, to reduce the computational cost. In this chapter, we assume that the agents can freely exchange their state information. In the most complicated communication, one agent can send the exact state information to the other agents. From these assumption, we can rewrite the identification problem as

$$\mathbf{x}(t+1) = \begin{bmatrix} \mathbf{x}_1\ (t+1) \\ \mathbf{x}_2\ (t+1) \\ \vdots \\ \mathbf{x}_K\ (t+1) \end{bmatrix} = f\ (\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} f_1\ (\mathbf{x}_1(t), C_1(\mathbf{x}_{i\neq 1}(t)), \mathbf{u}_1(t)) \\ f_2\ (\mathbf{x}_2(t), C_2(\mathbf{x}_{i\neq 2}(t)), \mathbf{u}_2(t)) \\ \vdots \\ f_K\ (\mathbf{x}_K(t), C_K(\mathbf{x}_{i\neq K}(t)), \mathbf{u}_K(t)) \end{bmatrix} \quad (7.10)$$

The identification in (7.8) allowing solving the HJB equation (7.1)-(7.5) by both MDO's MF option and IDF option. For the MF option, each learning agent j has optimization function according to (7.5)

$$J(\mathbf{x}_j(0)) = \sum_{t=0}^{\infty} \gamma^t \left(p_j(\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t))) + q_j\left(\mathbf{u}_j(t)\right)\right) \quad (7.11)$$

Therefore, the MDP for agent j in the MF option has the form $[\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j})] \times [\mathbf{u}_j] \rightarrow [\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j})]$. For the IDF option, each agent will optimize

$$J(\mathbf{x}_j(0)) = \sum_{t=0}^{\infty} \gamma^t \left(p_j\left(\mathbf{x}_j(t)\right) + q_j\left(\mathbf{u}_j(t)\right)\right) \quad (7.12)$$

In this case, the MDP has the form $[\mathbf{x}_j] \times [\mathbf{u}_j] \rightarrow [\mathbf{x}_j]$. Because each MDP in IDF option does not cover the entire knowledge gained by identification, each agent may have multiple MDPs depending on the $C_j(\mathbf{x}_{i\neq j})$. We assume that the agents' reward functions p_j and q_j in (7.11) and (7.12) have the same properties to (7.2) and (7.3).

Finally, we define the centralized approach and the completely decentralized approach, which will be used to compare with the MDO approaches, as follow. The centralized approach is defined with $K = 1$. The completely decentralized approach is defined as $K > 1$ but $C_j(\mathbf{x}_{i\neq j}) = 0$, which implies no communication among the agents. The definition of the selectively decentralized approach could be found in [93].

7.3 Design of MDO learning agents with two phases

Similar to Figure 6.1, Figure 7.1 shows the design of the learning agent in this work with two phases: identification and control. In the identification phase, we train the neural networks to

acquire the function approximators $\hat{f}$. In the control phase, we use the discrete MDP method to compute **u**.

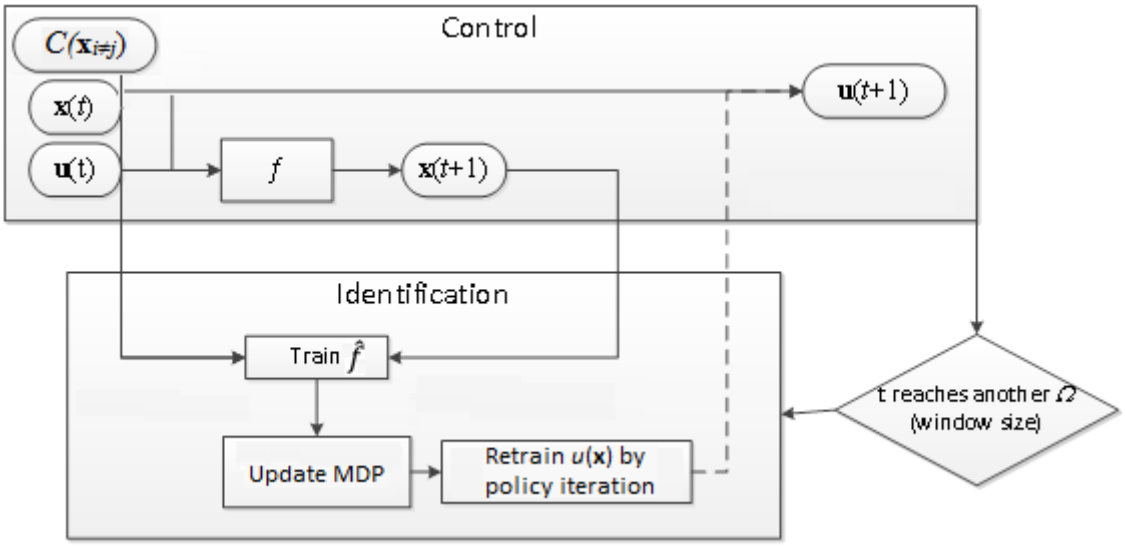

Figure 7. Two-phase design of the MDO learning agents.

7.3.1 MDO system identification

We use the three-layer feedforward neural network to approximate f as $\hat{f}$ in nonlinear system identification. Neural networks have been known for their capability to approximate a large and general class of nonlinear functions over compact domains. Theoretical foundation and application of neural network as such universal function approximators in control systems can be found in [15, 76, 77]. We use the backpropagation learning algorithm for neural networks [92]. For the MDO's MF option, $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t)), \mathbf{u}_j\ (t)\}$cis presented at the input layer, $\{\ \hat{\mathbf{x}}_j(t+1), \hat{C}(\mathbf{x}_{i\neq j}(t+1))\}$ is computed at the output layer of the neural network, and $\{\ \mathbf{x}_j(t+1), C_j(\mathbf{x}_{i\neq j}(t+1))\}$ is the target. In the other hand, for the IDF option, $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t)), \mathbf{u}_j\ (t)\}$ is presented at the input layer, $\{\ \hat{\mathbf{x}}_j(t+1)\}$ is computed at the output layer of the neural network, and $\{\ \mathbf{x}_j(t+1)\}$ is the target. Without MDO, the neural network layers could be set up as in [93].

7.3.2 Discrete MDP for MDO agents

In this chapter, we further develop the discretization method and MDP construction from the simulation in [93]. The theoretical assessment of the discrete MDP method to approximate the HJB equation's solution (2.1)-(2.5) has been presented in chapter 2. In this section, first, we would briefly present the discrete MDP method to solve (7.1)-(7.5) by the centralized and completely decentralized approach, which does not involve any MDO principles. Second, we would present the modification of the discrete MDP method to apply MDO principles in both the MF and the IDF options.

7.3.2.1 Discrete MDP for the centralized and completely decentralized approach

Fundamentally, the content of section 7.3.2.1 is the same to section 2.3 in chapter 2. For the convenience of reading, especially for the mathematical symbols, I rewrite the section as follow.

7.3.2.1.1 Discretize the state and action vectors

As defined in the problem statements, in this section, we ignore all of the communication among the agents $C(\mathbf{x}_{i\neq j})$. Let G be the number of intervals in each dimension of $\mathbf{x}$ and $\mathbf{u}$ for which we uniformly divide the dimension into small grids. Therefore, the entire state space is divided into G^n small hyper cubes with edge $\theta_x = 2\chi/G$. The control space is divided into G^m small hyper cubes with edge $\theta_u = 2\mu/G$. All points inside a hyper cube are discretely represented by the center of the hyper cube. Points on the border between two hyper cubes are represented by the center of the 'left' hypercube. Mathematically, the discretization process is described by the following formulas

$$\mathbf{x}[d] \to \theta_x + \chi/G \;\; \forall\, d \in [1,n] \;\; and\; \mathbf{x}[d] \in [\theta_x, \theta_x + 2\,\chi/G) \quad (7.13)$$

$$\mathbf{u}[d] \to \theta_u + \mu/G \;\; \forall\, d \in [1,m] \;\; and\; \mathbf{u}[d] \in [\theta_u, \theta_u + 2\,\mu/G) \quad (7.14)$$

where $\theta_x \in \{-\chi, -\chi + 2\chi/G, -\chi + 4\chi/G, ..., \chi - 2\chi/G\}$ and $\theta_u \in \{-\mu, -\mu + 2\mu/G, -\mu + 4\mu/G, ..., \mu - 2\mu/G\}$, which are the 'left' boundaries in the hyper cubes. We denote $\mathbf{x}_{dis}$ and $\mathbf{u}_{dis}$ as the discrete space and control vector of **x** and **u**, correspondingly. We also denote $(\mathbf{x}_{dis})$ and $(\mathbf{x}_{dis})$ as the set/supspace of points **x** and **u** whose discrete forms are $\mathbf{x}_{dis}$ and $\mathbf{u}_{dis}$, correspondingly.

With the discretization process in (7.11) and (7.12), it is important and easy to see that when *G* is odd, the zero vector **0** is one of the discrete space/control vectors. In the centralized MDP problem, the reward functions (7.2) and (7.3) become

$$p(\mathbf{x}) = p(\mathbf{x}_{\text{dis}}), q(\mathbf{u}) = q(\mathbf{u}_{\text{dis}}) \quad (7.15)$$

And the optimization goal becomes the MDP goal (7.5)

$$J(\mathbf{x}(0)) = \sum_{t=0}^{\infty} \gamma^t \left(p\left(\mathbf{x}_{\text{dis}}(t)\right) + q\left(\mathbf{u}_{\text{dis}}(t)\right) \right) \quad (7.16)$$

where $0 < \gamma < 1$ is the discount factor.

7.3.2.1.2 Setup the probabilistic transition function for the MDP

The MDP requires the probabilistic transition function as a matrix of $P(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$, which is the probability of reaching discrete state $\mathbf{x'}_{dis}$ when executing action $\mathbf{u}_{dis}$ at state $\mathbf{x}_{dis}$. We can apply the Monte Carlo method [74] to approximate $P(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$ as follow.

- Generate a large number of *S* points (**x**, **u**) following the uniform distribution in $(\mathbf{x}_{dis}) \times (\mathbf{u}_{dis})$.

- Count the number of points S_1 such that $\hat{f}(\mathbf{x}, \mathbf{u}) \in (\mathbf{x'}_{dis})$.

- Then $S_1/S \rightarrow P(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$ as $S \rightarrow \infty$.

We use the policy iteration algorithm [65] to compute the MDP solution. At every iteration, the action $\mathbf{u}_{dis}(t)$ is calculated as

$$\mathbf{u}_{\text{dis}}(t) = \underset{\mathbf{u}}{\text{argmax}} \sum_{\forall \mathbf{x}'\text{dis}} \text{P}(\mathbf{x}'_{\text{dis}} \mid \mathbf{x}_{\text{dis}}(t), \mathbf{u}) \quad (7.17)$$

And $R(\mathbf{x}_{\text{dis}})$ is updated after executing $\mathbf{u}_{\text{dis}}(t)$

$$R\big(\mathbf{x}_{\text{dis}}(t)\big) = p\big(\mathbf{x}_{\text{dis}}(t)\big) + q\big(\mathbf{u}_{\text{dis}}(t)\big) +$$

$$\gamma \sum_{\forall \mathbf{x}_{\text{dis}}'} P\big(\mathbf{x}_{\text{dis}}'(t+1) | \mathbf{x}_{\text{dis}}(t), \mathbf{u}_{\text{dis}}(t)\big) R\big(\mathbf{x}'_{\text{dis}}(t+1)\big) \quad (7.18)$$

Formula (7.13)-(7.18) are written for the centralized approach. They can be applied to each learning agent in the completely decentralized approach using the agent's local state, local action, local reward and local transition probability.

7.3.2.2 Discrete MDP method for the MDO approach

The central theme in this section is the discretization of communication $C_j(\mathbf{x}_{i\neq j})$. Let us call N_j, M_j and $N_{i\neq j}$ be the dimensionality of the $\mathbf{x}_j$, $\mathbf{u}_j$ and $C_j(\mathbf{x}_{i\neq j})$ in agent j. Since the system (7.1) and communication function is bounded, each dimension of $C_j(\mathbf{x}_{i\neq j})$ is bounded by $[-\lambda, \lambda]$. Since the communication sent by agent i to agent j should not be more complex than $\mathbf{x}_i$, we discretize each dimension of $C(\mathbf{x}_{i\neq j})$ into $L \leq G$ grids using the same method described in (7.13) and (7.14). We denote the discrete from of $C_j(\mathbf{x}_{i\neq j})$ by $C_j(\mathbf{x}_{i\neq j})_{\text{dis}}$.

For the MDO-MF option, since the identified model is in the form $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t)), \mathbf{u}_j(t)\} \rightarrow \{\mathbf{x}_j(t+1), C_j(\mathbf{x}_{i\neq j}(t+1))\}$, each agent j has one MDP model with dimensionality $(G^{N_j} \times N) \times G^{M_j} \times (G^{N_j} \times L^{N_{i\neq j}})$. The transition probability for the MDP could be setup similarly to the centralized approach, except the Mote-Carlo sampling should be done on $(\mathbf{x}_{j\text{dis}}) \times (C_j(\mathbf{x}_{i\neq j})_{\text{dis}}) \times (\mathbf{u}_{j\text{dis}})$ in the whole space.

For the MDO-IDF option, since the identified model is in the form $\{\mathbf{x}_j(t),\ C_j(\mathbf{x}_{i\neq j}(t)),\ \mathbf{u}_j\ (t)\} \rightarrow \{\mathbf{x}_j(t+1)\}$, the MDP in this option will have dimensionality $(G^{N_j}) \times G^{M_j} \times (G^{N_j})$. By rewriting the identification as

$$\mathbf{x}_j(t+1) \approx \hat{f}\left(\mathbf{x}_j(t), C_j\left(\mathbf{x}_{i\neq j}(t)\right), \mathbf{u}_j(t)\right) = \begin{cases} \hat{f}_{C_j\left(\mathbf{x}_{i\neq j}(t)\right)=\mathbf{c}_1}\left(\mathbf{x}_j(t), \mathbf{u}_j(t)\right) \\ \hat{f}_{C_j\left(\mathbf{x}_{i\neq j}(t)\right)=\mathbf{c}_2}\left(\mathbf{x}_j(t), \mathbf{u}_j(t)\right) \\ \vdots \\ \hat{f}_{C_j\left(\mathbf{x}_{i\neq j}(t)\right)=\mathbf{c}_\ldots}\left(\mathbf{x}_j(t), \mathbf{u}_j(t)\right) \end{cases} \quad (7.19)$$

where $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}\ldots$ are possible the discrete values of $C_j(\mathbf{x}_{i\neq j}(t))$, it is easy to see that each agent j will have $L^{N_{i\neq j}}$ MDP models, which could be indexed. When the agent receive $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t))\}$, it will look up the index of $C_j(\mathbf{x}_{i\neq j}(t))$ and choose the corresponding MDP to compute $\mathbf{u}(t)$.

7.3.3 The pseudo code for the MDO learning agent

Here, the window size parameter Ω decides how frequently we call the identification phase. The pseudo code for each agent is as follow

Initialize: $\hat{f}$ neural network with random weights.

Predefine the discretization parameters as in (7.13)-(7.16).

Construct the MDPs as showed in (7.17) and (7.19).

$u_j(\{\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j})\})$ / $u_j([\mathbf{x}_j])$: solution of the MDPs by policy iteration for the MF / IDF options.

For t from 2 to the maximum number of iterations

Receive and discretize $\{\mathbf{x}(t), C_j(\mathbf{x}_{i\neq j}(t))\}$ as in (7.13).

Compute $\mathbf{u}(t) = u(\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t))\})$ / $\mathbf{u}(t) = u(\mathbf{x}(t))$ according to the policy.

Add $\{\mathbf{x}(t\text{-}1), C_j(\mathbf{x}_{i\neq j}(t\text{-}1)), \mathbf{u}(t\text{-}1)\}$ and $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t))\}$ into the training set for future neural network training set.

if t % Ω = 0 // reach the end of the window, update the identification

Retrain $\hat{f}$.

Reconstruct the MDPs as showed in (7.17) and (7.19).

Recompute $u_j(\{\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j})\})$ / $u_j([\mathbf{x}_j])$ by policy iteration.

Clear the training set of the neural network.

Each agent could also keep track of the identification error to decide whether or not to keep collecting data point for training neural network and skip the identification phase.

7.4 Simulation results

In this section, we setup toy sinusoidal systems to demonstrate the capability of the MDO approach in RL. These systems has the form

$$\mathbf{x}(t) = \sin(\mathbf{A}\mathbf{x}(t\text{-}1)+\mathbf{u}(t\text{-}1)) \quad (7.20)$$

where **A** are 3×3 random Markov matrices such that all diagonal entries share the same value. For the ease of decentralization, we choose the system such that the dimensionalities of both **x** and **u** are the same. The vector sin function is defined from each dimension as

$$\sin(\mathbf{x}) = \begin{bmatrix} \sin(\mathbf{x}_1) \\ \sin(\mathbf{x}_2) \\ \sin(\mathbf{x}_3) \end{bmatrix} \quad (7.21)$$

In addition, each dimension of **x** and **u** is between -0.35 and 0.35. The non-diagonal entries are random numbers between 0 and coupling parameter σ. If the right side of (7.20) is beyond this range, the result will be scaled back to the nearest bound. We define the coupling parameter σ as

$$\sigma = \frac{\sum_{i\neq j} \mathbf{A}_{ij}}{\sum \mathbf{A}_{ij}} \quad \forall i,j \in [1,3] \quad (7.22)$$

In other words, σ is the ratio between the sum of non-diagonal entries in **A** and the sum of all entries in **A**. In our simulations σ ranges from 0 to 0.3. With σ = 0, **A** becomes the identity matrix

or the systems are completely decouple. The systems are more couple when σ increases. In this design, we assign three learning agent such that each agent is responsible for one dimension of **x** and **u**. The initial state **x**(0) is a vector of **0.2**. For equation (7.2) and (7.3), we choose $p(\mathbf{x}) = -\|\mathbf{x}\|^2$ and $q(\mathbf{u}) = -\|\mathbf{u}\|^2$ for both the centralized approach and the learning agents in the decentralized / MDO approaches. For equation (7.5), $\gamma = 0.9$. As the reminder, each learning agent does not know anything about **A** but may know all of the other information.

For identification, each agent has neural networks of 50 hidden layers. The input and output layer is as in section II.2. We use the window size of $\Omega = 50$ (figure 7.1). In each training round, the training data set is reused at most 1000 times (epoch) [99] to improve identification. The maximum number of iterations t is 5000.

For the MDO discretization, we suppose that each agent can send its full state information to the other agents, which means $C_j(\mathbf{x}_{i\neq j}) = \mathbf{x}_{i\neq j}\ \forall i, j$. Each agent j divides its $\mathbf{x}_j$ and $\mathbf{u}_j$ dimension into $G = 7$ grids (equation (7.13)-(7.14)). Therefore, the grid size in each dimension is 0.1. For the discretization of $C_j(\mathbf{x}_{i\neq j})$, we setup two scenarios:

- When the agents use the external information fully (with the same resolution) as it does for the internal $\mathbf{x}_j$ and $\mathbf{u}_j$, each dimension of $C_j(\mathbf{x}_{i\neq j})$ is divided into $G = L = 7$ grids. Therefore, in the MF option, each agent j has one MDP model of size $(7\times7^2) \times 7 \times (7\times7^2)$. In the IDF option, each agent j has 7^2 MDP models of size 7×7×7.

- When the agents use the external information less than (with less resolution) it does for the internal $\mathbf{x}_j$ and $\mathbf{u}_j$, each dimension of $C_j(\mathbf{x}_{i\neq j})$ is divided into $L = 5$ and $L = 3$ grids. Therefore,

in the MF option, each agent *j* has one MDP model of size $(7\times5^2) \times 7 \times (7\times5^2)$ and $(7\times3^2) \times 7 \times (7\times3^2)$. In the IDF option, each agent j has 5^2 and 3^2 MDP models of size 7×7×7.

For the completely decentralized, selectively decentralized and the centralized approach, each agent also divides each dimension of **x** and **u** into $G = 7$ grids.

7.4.1 The learning performance of MDO approach in stabilizing control system

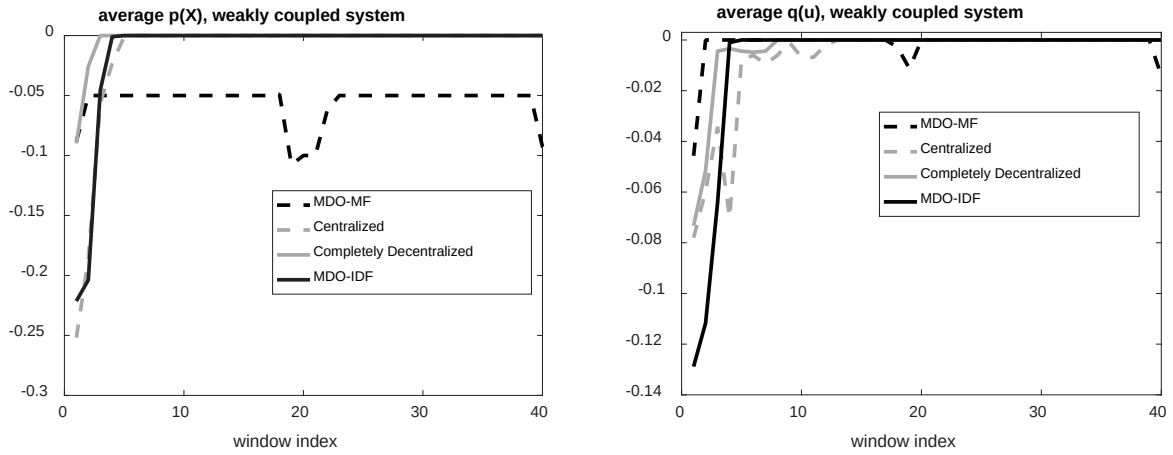

Figure 7.2. learning performance in $p(\mathbf{x})$ and $q(\mathbf{u})$ of the MDO approaches in weakly coupled system ($\sigma = 0.05$).

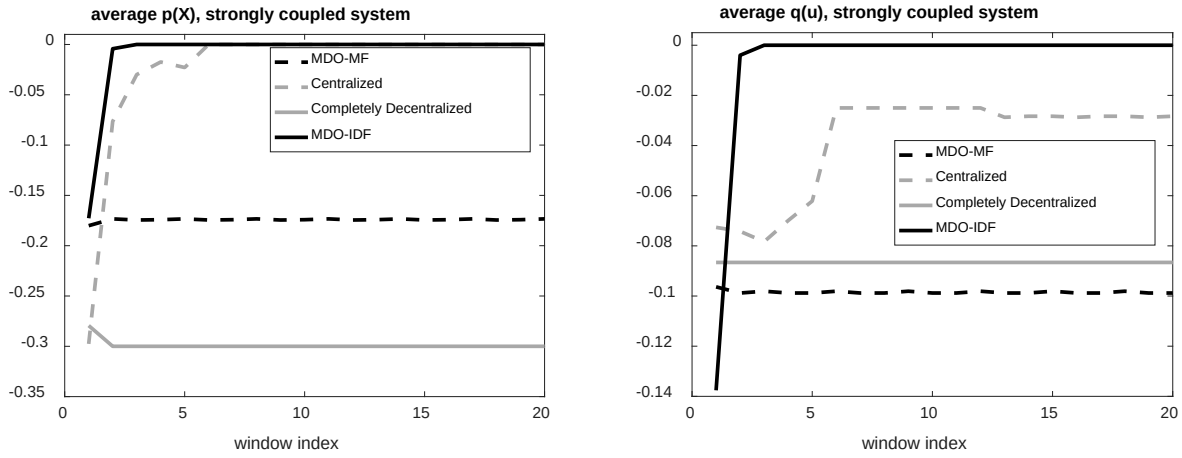

Figure 7.3. learning performance in $p(\mathbf{x})$ and $q(\mathbf{u})$ of the MDO approaches in strongly coupled system ($\sigma = 0.3$).

Figures 7.2 and 7.3 compare the learning performance among the MDO, the complete decentralization and the centralization approaches. Overall, the MDO-IDF outperforms both the centralization and the complete decentralization approach. In figure 3, where the centralization is expected to have advantage, the MDO-IDF minimize both $p(\mathbf{x})$ and $q(\mathbf{u})$ to 0. According to (7.2) and (7.3), this result implies that both **x** and **u** converges to 0. Meanwhile the centralization is able to minimize $p(\mathbf{x})$ to 0, but shows small oscillation on $q(\mathbf{u})$, which does not converge to 0. In the other hand, the MDO-MF is better in learning when the system is strongly decoupled, as showed in figure 7.2. However, even in this case, the MDO-MF still shows minor oscillation when both $p(\mathbf{x})$ and $q(\mathbf{u})$ approach 0.

7.4.2 Performance loss MDO-IDF when using resolution-less communication

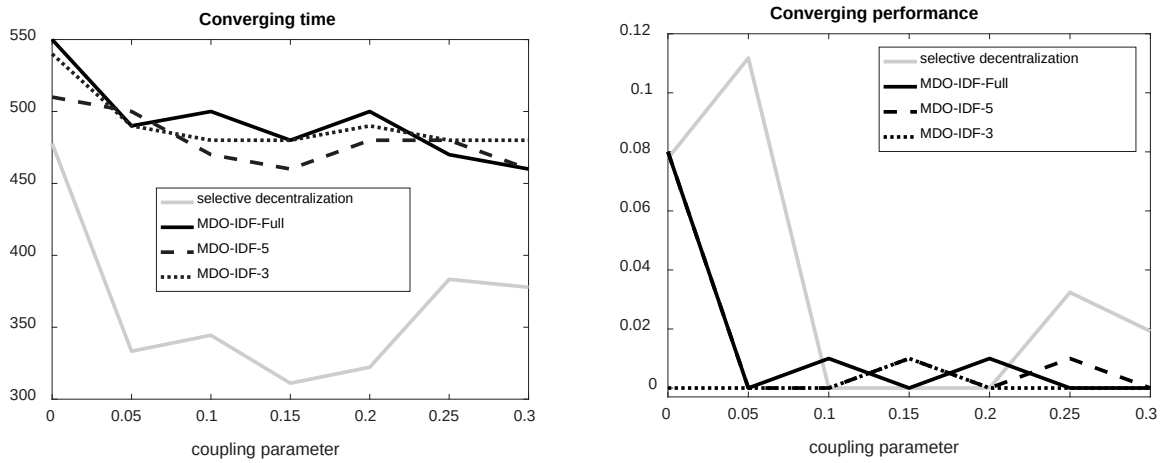

Figure 7.4. Converging time of the MDO-IDF approach with full and less resolution

In figure 7.4, we show that the learning performance of the MDO-IDF approach are similar when the communication $C_j(\mathbf{x}_{i\neq j})$ is used with less resolution, compared with the selective decentralization approach. In this figure, we denote MDO-IDF-Full as applying the IDF approach with $G = L = 7$, MDO-IDF-5 as $G = 7$ / $L = 5$ and MDO-IDF-3 as $G = 7$ / $L = 3$. Here, we address the performance of these approach by converging time, which is defined as $\Omega \times w$, in which w is

the window such that the average $p(\mathbf{x}) + q(\mathbf{u})$ at window w and w-1 are less than 0.001, or when the change of average $p(\mathbf{x}) + q(\mathbf{u})$ at w and w-1 is less than 0.001. Overall, for converging time, the selective decentralization converging time is still better than the MDO-IDF approach. However, at the converging window, the average $p(\mathbf{x}) + q(\mathbf{u})$ is closer to 0 in the MDO-IDF approach.

7.5 Discussions

In this chapter, we show the capability of MDO approaches in decentralized reinforcement learning. It is clear that from our experiement, the MDO-IDF option could successfully in control-and-stabilize learning problem; meanwhile, the MDO-MF option is not always successful. Mathematically, this fact could be explained by the divergence of $f(\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j}))$. In the MF option, $C_j(\mathbf{x}_{i\neq j})$ participates in the MDP construction in such a way that the Monte Carlo randomize samples of $C_j(\mathbf{x}_{i\neq j})$ through its range. When the system is more coupled, $f(\mathbf{x}_j, \mathbf{u}_j, C_j(\mathbf{x}_{i\neq j}))$ will be more diverge, even when $\mathbf{x}_j$ is already in the stable region. Therefore, the conditional probability $P(\mathbf{x}_j(t+1) \mid \mathbf{x}_j(t), \mathbf{u}_j(t)=\mathbf{0})$ such that both $\mathbf{x}_j(t+1)$ and $\mathbf{x}_j(t)$ are in the stable region is less when the system is more coupled. Thus, the utility value of $\mathbf{x}$ in the stable region become less; therefore, the agents see less 'motivation' to stablize. In the individual feasible option, the MDP is constructed by $f_{C_j(\mathbf{x}_{i\neq j})}(\mathbf{x}_j, \mathbf{u}_j)$ where $C_j(\mathbf{x}_{i\neq j})$ is fixed. Since the simulation examples satisfy that all of the agent could stabilize their state to 0 together, we do not see the diverse of $f_{C_j(\mathbf{x}_{i\neq j})}(\mathbf{x}_j, \mathbf{u}_j)$ when both $\mathbf{x}_j$ and $C_j(\mathbf{x}_{i\neq j})$ are in stable regions. In the other words, from philosophical perspective, in a cooperative task, trusting the behavior of the collaborators often lead to better results than doubting the incompetence or error from the collaborators.

Compared to the selective decentralization approach [93], which is also showed to outperform both the centralization and the complete decentralization, the MDO-IDF offer a complimentary technique to tackle the communication among the learning agents. In the selective decentralization approach, there exist a central agent deciding which communication scheme to be used for the agents to make decision. In the other words, selective decentralization is about model switching and the communication among all of the agents are not always free. In the MDO-IDF, the agents could freely send the state information to the others. In addition, each agent is responsible for its own communication: how to use the communication to compute the best action. Both of these methods show better performance than huge centralization and blind (completely) decentralization approach in many cases of system decoupling.

8. DECENTRALIZED LEARNING IN NOISY ENVIRONMENT

This section adds the system noise as another dimension of complexity in the learning and control problem for the selective decentralization framework. We want to answer the following questions. First, to what extend the selective decentralization could improve the system identification, compared to the centralized approach, given increasing level of noise? Second, to what extend the selective decentralization could stabilize the system faster than the centralized approach could, given increasing level of noise? Third, is there any trade-off among the learning performance, cost and the converging speed given the occurrence of noise? In this chapter, we answer these questions by experiments on all of the learning and control techniques presented in chapters 4-7. We also assess the impact of noise filtering techniques, in addition to selective decentralization, in improving the learning and control performance.

8.1 Experimental results without noise-filtering techniques

8.1.1 Linear system

This section applies the learning and control technique presented in chapter VI for linear system. For noise extension, we add the noise into the system as

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{r}(t) \quad (8.1)$$

where $\mathbf{x} \in \Re^N$ is the state vector, $\mathbf{u} \in \Re^N$ is the action (also called control) vector, $\mathbf{r} \in \Re^N$ is a random unknown noise vector with expected value of $\mathbf{0}$, $\mathbf{A} \in \Re^{N\times N}$ is the state-transition matrix, which is unknown, and $\mathbf{B} \in \Re^{N\times M}$ is a known semi positive-definite matrix. We assume that $\mathbf{r}$ are under a multivariate normal distribution. Here, we set $\mathbf{x}$ and $\mathbf{u}$ to have the same dimensionality for the ease of decentralization. For simulation, the matrix $\mathbf{A}$ is setup with underlying subsystem components {{1,2}, {3, 4},{5, 6}, {7,8}, {9, 10}} as follow:

$$\mathbf{A} = \begin{bmatrix} 0.68 & 0.32 & & & & & & \\ 0.20 & 0.80 & & & & & & \\ & & 0.25 & 0.75 & & & \sigma & \\ & & 0.44 & 0.56 & & & & \\ & & & & 0.50 & 0.50 & & \\ & & & & 0.41 & 0.59 & & \\ & & & & & & 0.85 & 0.15 \\ & & \sigma & & & & 0.15 & 0.85 \\ & & & & & & & \\ & & & & & & 0.35 & 0.65 \\ & & & & & & 0.67 & 0.33 \end{bmatrix} \quad (8.2)$$

where the non-block entries of **A** are a random numbers between 0 and σ, which is called coupling parameters. The initial control variables **u**(0) and state variable **x**(0) are set randomly between -1 and 1. As shown in (8.2), σ represents the underlying interconnection among the subsystems. We experiment with σ from 0 to 0.5; in the other word, from the completely decoupled system to the strongly coupled system. To avoid numerical overflowing, we normalize **A** into a Markov matrix in (8.1). We set **B** as the identity matrix. The learning objective is to maximize

$$\sum_{t=0}^{\infty} \gamma^t(-\mathbf{x}(t)^T\mathbf{x}(t) - \mathbf{u}(t)^T\mathbf{u}(t)) \quad (8.3)$$

Where γ = 0.9. For identification, we set α = 1 (formula 5.11). At the starting point, we set all elements of **x**(0) as random numbers between -1 and 1. Each noise element is randomly generated from normal distribution with mean of 0 and small standard deviation of from 0.01 to 0.05. Higher standard deviation implies more noise. For statistical purposes, we repeat the experiment 50 times for each choice of coupling parameter. We set the window size *w* = 1. For time index *t*, we terminate the experiment at *t* = 500. This choice is made based on the observation in [93], when the system is stabilized after the first tens iterations.

Figures 8.1 and 8.2 show that the selectively decentralized system shows better control performance than the centralized system. In these figures, we draw the result when the noise

standard deviation is 0.01. Figure 8.1 shows the result when (1) is completely decoupled (σ=0). Figure 8.2 shows the result when (1) is the most coupled in our experiments (σ=0.5). We use norm(**x**) and norm(**u**) to denote the second-norm of **x** and **u**. In these figures, the numbers for norm(**x**) and norm(**u**) are the average values of the 50 random repetitions. When the noise becomes large, as showed in figures 3 and 4, both the state and the action variate around the noise standard deviation. In this case, the difference between the centralized and selectively decentralized learning performance is marginal.

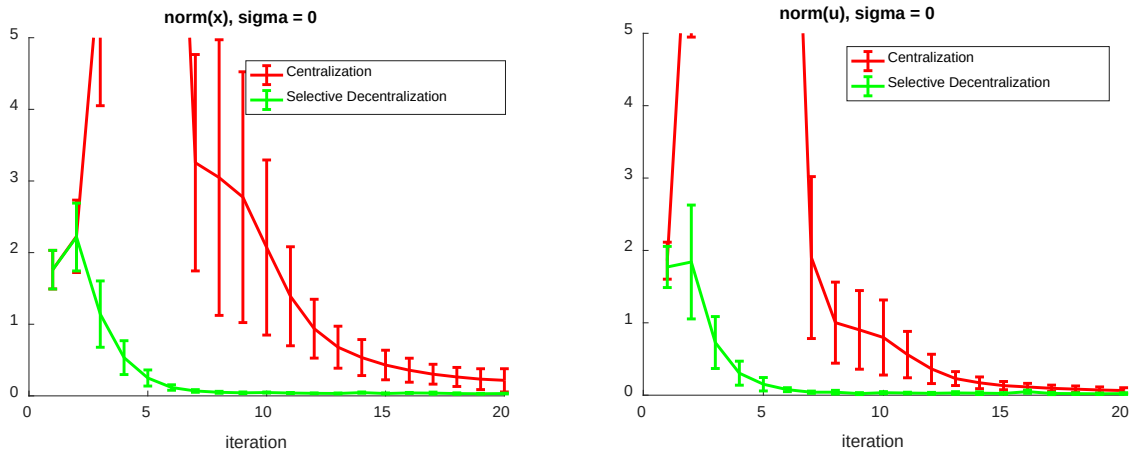

Figure 8.1. Comparison of learning performance between the centralized systems and the selectively decentralized systems when the systems are completely decoupled and linear (σ=0) in small noise scenario.

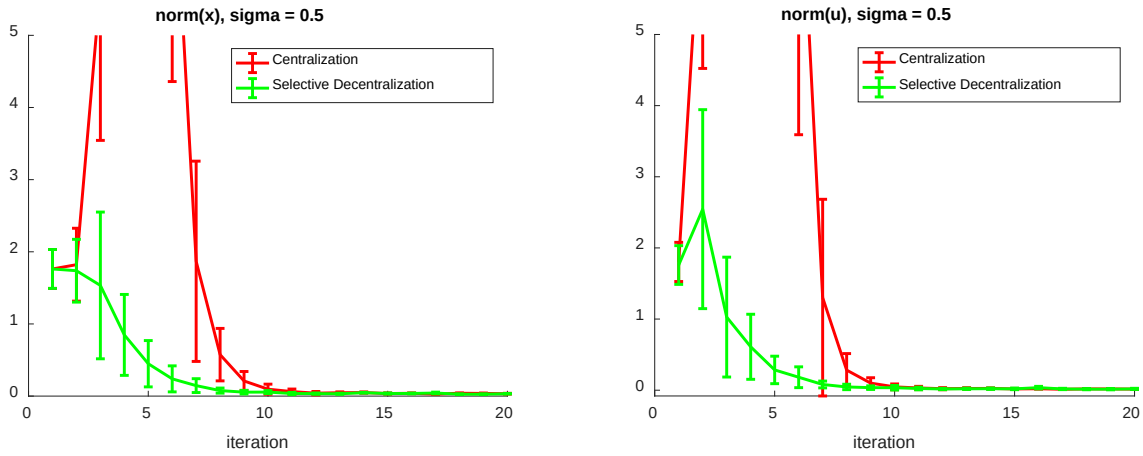

Figure 8.2. Comparison of learning performance between the centralized systems and the selectively decentralized systems when the systems are strongly coupled and linear (σ=0.5) in small noise scenario.

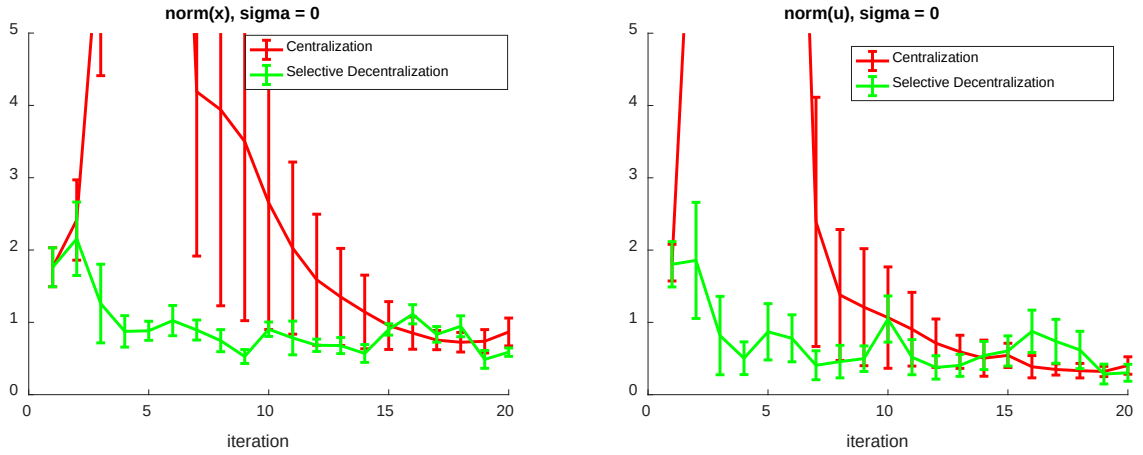

Figure 8.3. Comparison of learning performance between the centralized systems and the selectively decentralized systems when the systems is are completely decoupled and linear (σ=0) in large noise scenario

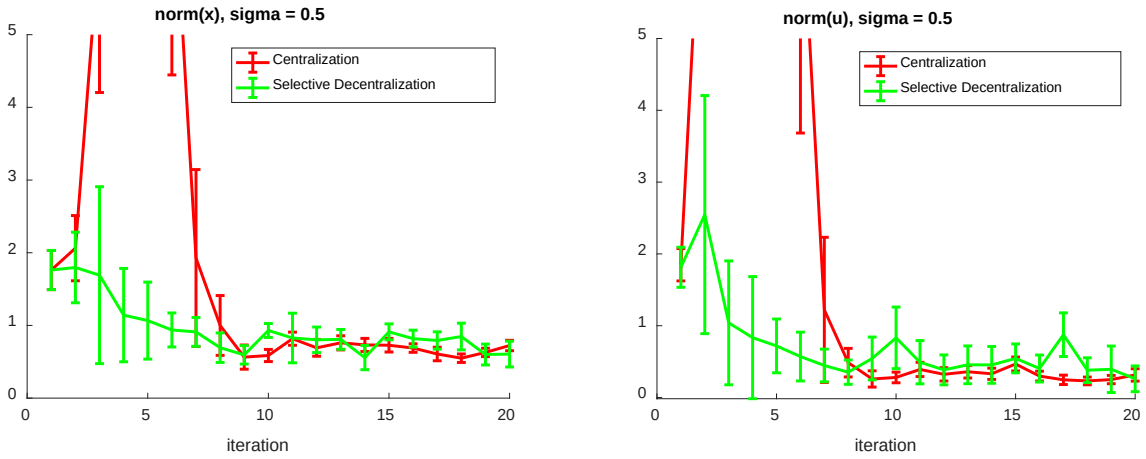

Figure 8.4. Comparison of learning performance between the centralized systems and the selectively decentralized systems when the systems is are strongly coupled and linear (σ=0.5) in large noise scenario

Notably, the centralization and selective decentralization can learn how to stabilize (8.1) despite the noise. However, we see significant gap between the performance of the centralization and the performance of selective decentralization. Selective decentralization stabilizes (8.1) faster. This gap tends to decrease when the systems are more coupled. Figure 8.5 shows the average number of iteration for the centralization selective decentralization to bring norm(x) less than 0.05. We observe that the selective decentralization always outperforms the other approaches regardless of the coupling parameters.

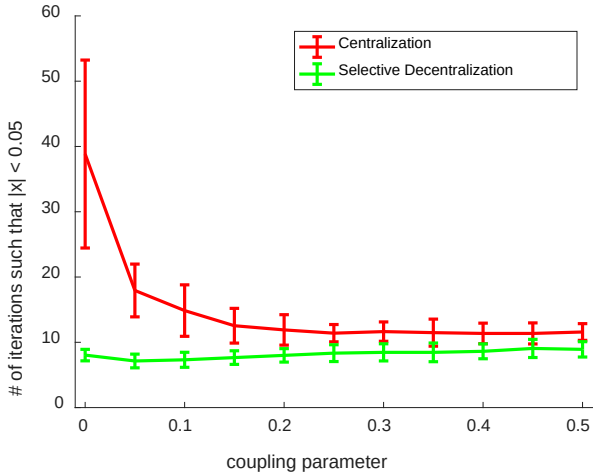

Figure 8.5. Number of iterations needed to bring norm(**x**) < 0.05 in small noise - linear system scenario

In figures 8.6 and 8.7, we show the learning performance of the selectively decentralized approach and the centralized approach when the noise increases. As mentioned, since each noise element is a random number with mean 0, higher noise standard deviation implies more noise. In figure 8.6, we measure the average identification error (6.7) at the last 50 iterations during the experiments as the 'identification error at the end of the experiment', and the largest identification error from t = 1 to t = 500 as the 'worst identification error'. Similar to figure 8.6, in figure 8.7, we measure the learning objective (6.5). Overall, after the experiment, we observe that the centralized approach is robust against increasing level of noise. Here, the convergent identification error and learning objective in the centralized approach do not degrade when more noise is introduced. Meanwhile, these metrics in the selectively decentralized approach deteriorate linearly with the increasing noise. However, the selectively decentralized approach still significantly outperforms the centralized approach when we look at the worst identification error and learning objective.

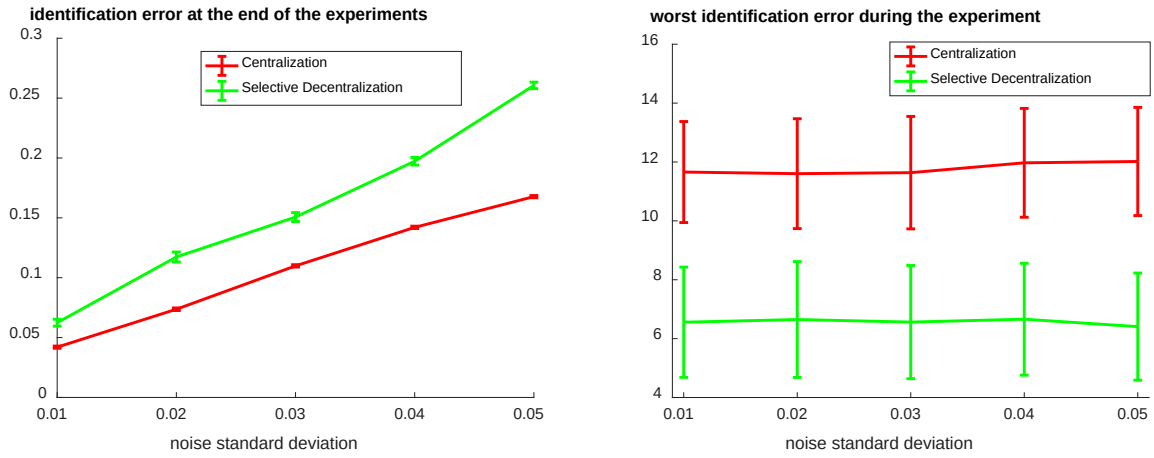

Figure 8.6. Comparison of identification errors between the selectively decentralized approach and the centralized approach given increasing noise level in linear system

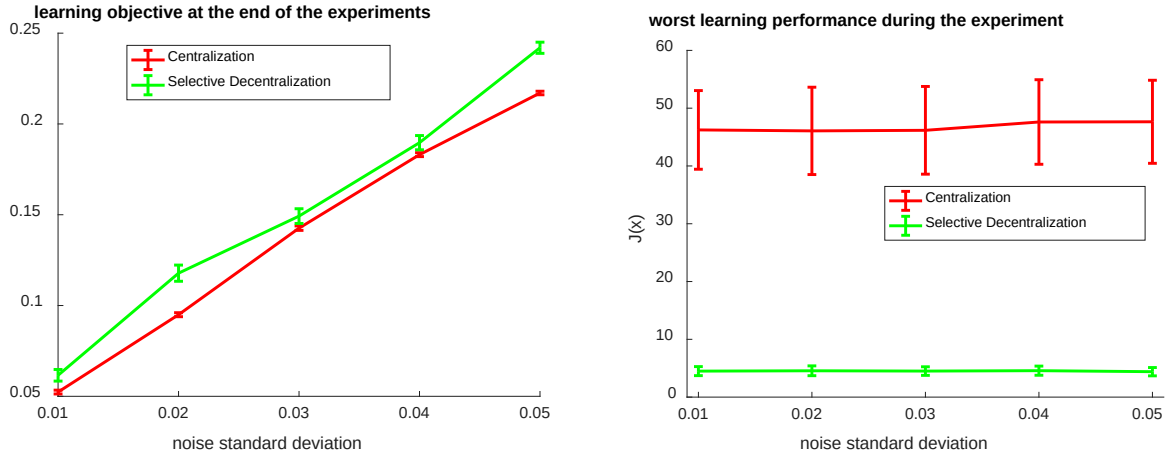

Figure 8.7. Comparison of learning performance $J(\mathbf{x})$ between the selectively decentralized approach and the centralized approach given increasing noise level in linear system

8.1.2 Nonlinear system with discrete-MDP approach

Similar to the linear system, for nonlinear system with the discrete-MDP approach, selective decentralization improves the learning performance in small noise scenario. The systems used in this section is similar (4.23-4.25) in chapter 4.

$$\mathbf{x}(t) = \sin(\mathbf{A}\mathbf{x}(t\text{-}1)) + \mathbf{u}(t) + \mathbf{r}(t) \quad (8.4)$$

where **A** are *3×3* random Markov matrices such that all diagonal entries share the same value. The non-diagonal entries are random number between 0 and the coupling parameter σ. The vector sin function is defined from each dimension as

$$\sin(\mathbf{x}) = \begin{bmatrix} \sin(\mathbf{x}_1) \\ \sin(\mathbf{x}_2) \\ \\ \sin(\mathbf{x}_3) \end{bmatrix} \quad (8.5)$$

The coupling parameter σ for the non-diagonal entries is defined as

$$\sigma = \frac{\sum_{i \neq j} \mathbf{A}_{ij}}{\sum \mathbf{A}_{ij}} \quad \forall i, j \in [1,3] \quad (8.6)$$

With $\sigma = 0$, **A** becomes the identity matrix or the systems are completely decouple. The systems are more couple when σ increases. The learning objective is to minimize (8.3). For state and action variables, all state **x** and action **u** components have the range between -0.5 and 0.5 and initial state vectors **x**(0) are vectors of 1. For discretization (2.10-2.11), we choose $M = N = 5$. Therefore, $\theta_x = \theta_u = 0.2$. Each noise element **r**(*t*) is randomly generated from normal distribution with mean of 0 and small standard deviation of from 0.01 to 0.05. Higher standard deviation implies more noise. For statistical purposes, we repeat the experiment 50 times for each choice of coupling parameter. We set the window size $w = 50$. For time index *t*, we terminate the experiment at $t = 10000$.

Figures 8.8 and 8.9 demonstrates the comparative performance between the selectively decentralized and the centralized approaches when the Gaussian noise standard deviation is 0.01. The selectively decentralized approach could bring both the state and the action vectors closer to 0, compared to the centralized approach could, when the systems are completely decoupled (figure 8.8) and strongly coupled (figure 8.9, with coupling parameter = 0.5).

However, when the noise increases, the gap of learning performance between the selectively decentralized and the centralized approach are narrower. Figures 8.10 and 8.11 shows typical example of the systems with large noise, when both x and u do not converge. In figure 8.12, we draw the learning goal, which is minimization of $J(\mathbf{x})$, when the experiments terminate. When the noise standard deviation reaches 0.04, we no longer see the improvement of selective decentralization, compared to centralization.

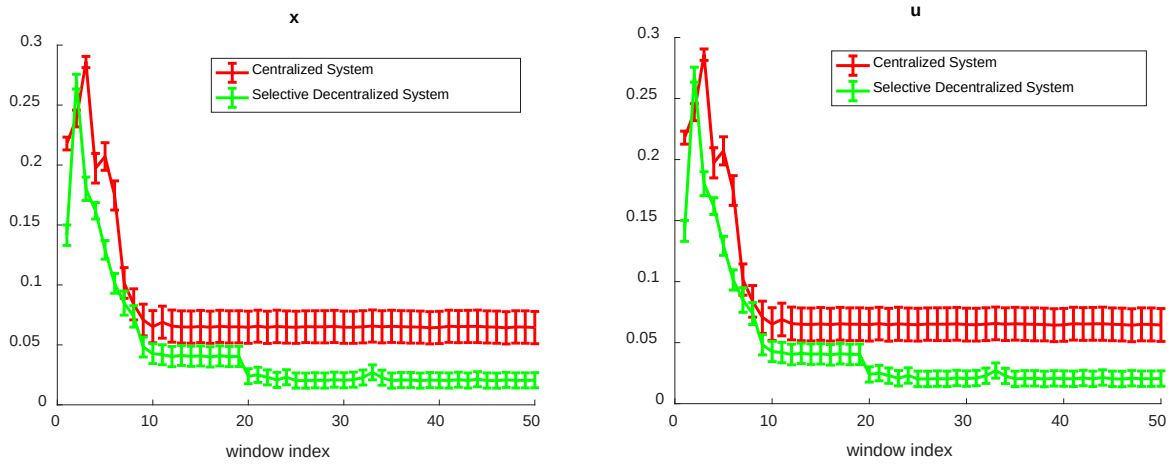

Figure 8.8. Comparison of learning performance between the discrete-MDP centralized systems and the selectively decentralized systems when the systems are completely decoupled and nonlinear in small noise scenario

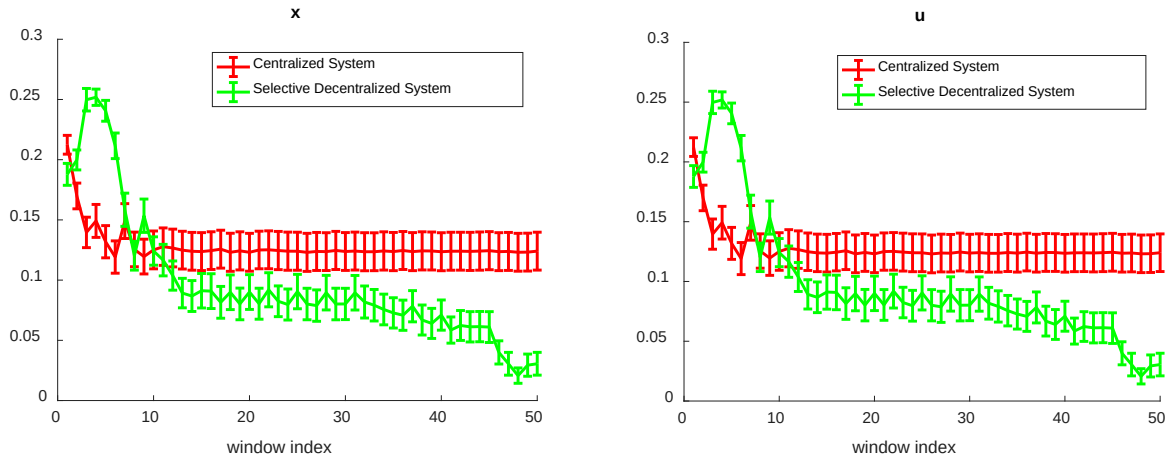

Figure 8.9. Comparison of learning performance between the discrete-MDP centralized systems and the selectively decentralized systems when the systems are strongly coupled and nonlinear in small noise scenario

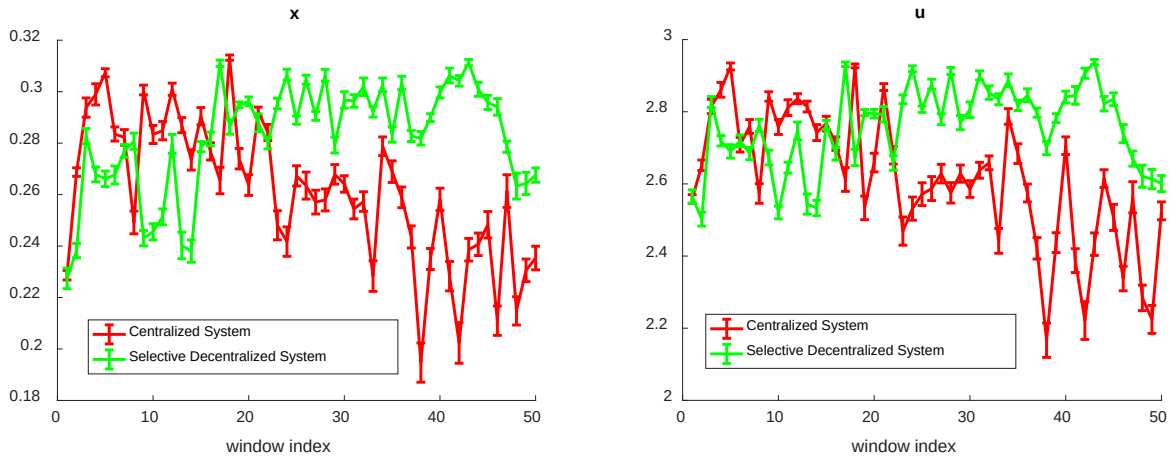

Figure 8.10. Comparison of learning performance between the discrete-MDP centralized systems and the selectively decentralized systems when the systems are completely decoupled and nonlinear in large noise scenario

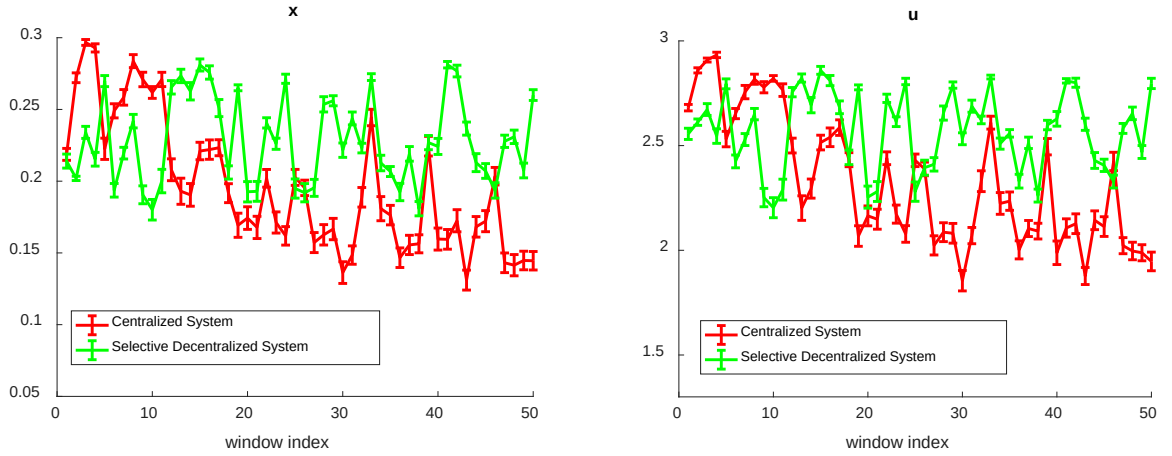

Figure 8.11. Comparison of learning performance between the discrete-MDP centralized systems and the selectively decentralized systems when the systems is are strongly coupled and nonlinear in large noise scenario

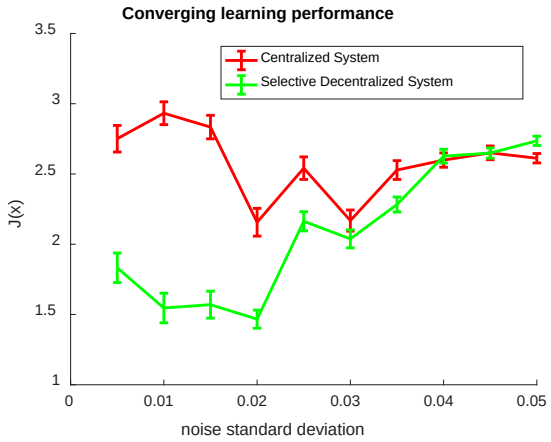

Figure 8.12. The converging learning performance of the selectively decentralized and the centralized discrete-MDP when the noise standard deviation increases

8.1.3 Q learning

In Q-learning, we use the same simulated system as showed in section 8.1.2. For discretization (4.3-4.4), we choose $G = 5$. Therefore, $\theta_x = \theta_u = 0.2$. For Q-learning parameters (4.5-4.8), we choose $r = 0.01$, $\alpha = 0.1$ and $\gamma = 0.9$. The window size is set as $w = 50$. For each choice of coupling parameters, we repeat the experiment 100 times.

It is clear that selective decentralization significantly improves the learning performance, compared to the centralized approach. This observation remains the same for all of the experiments. Figures 8.13 and 8.14 show typical example of how selective decentralization outperforms centralization. In figure 8.13, when the system is weakly coupled, selective decentralization could still stabilize state and control vectors close to 0; meanwhile, centralization fails to do the same task within the maximum number of allowed iteration. In figure 8.14, when the system are strongly coupled, both centralized and selectively decentralized Q-learning could stabilize the system. However, selective decentralization brings the system toward the zero-equilibrium point much faster.

In figure 8.17, we plot the minimization objective $J(\mathbf{x})$ for the learning process at the last 10 windows during the experiments. As expected, when the Gaussian noise standard deviation increase, it is less likely that the system could be stabilized. Figures 8.15 and 8.16 show that both **x** and **u** do not converge to 0 with large noise. In these two figures, the noise standard deviation is 0.1. Therefore, $J(\mathbf{x})$ increases with the noise standard deviation. However, selective decentralization achieves less $J(\mathbf{x})$.

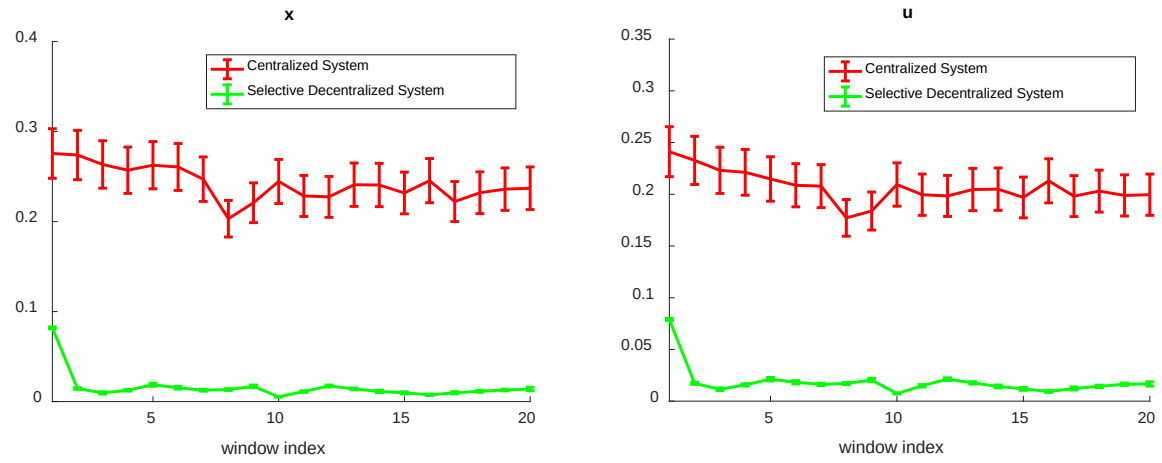

Figure 8.13. A typical example of how **x** and **u** converge in small-noise scenario in weakly coupled system with Q-learning; here, the Gaussian noise standard deviation is 0.005 and σ = 0.05

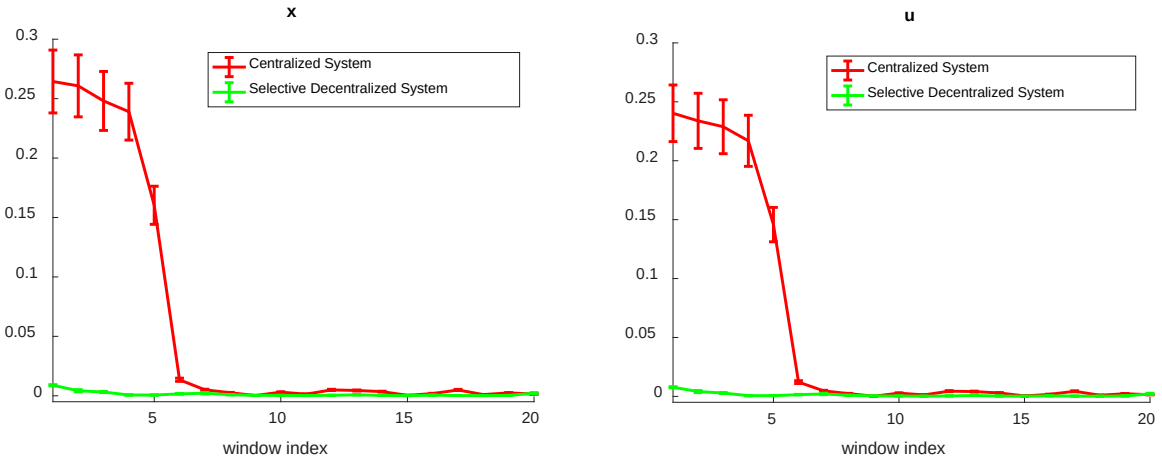

Figure 8.14. A typical example of how **x** and **u** converge in small-noise scenario in strongly coupled system Q-learning; here, the Gaussian noise standard deviation is 0.005 and σ = 0.5

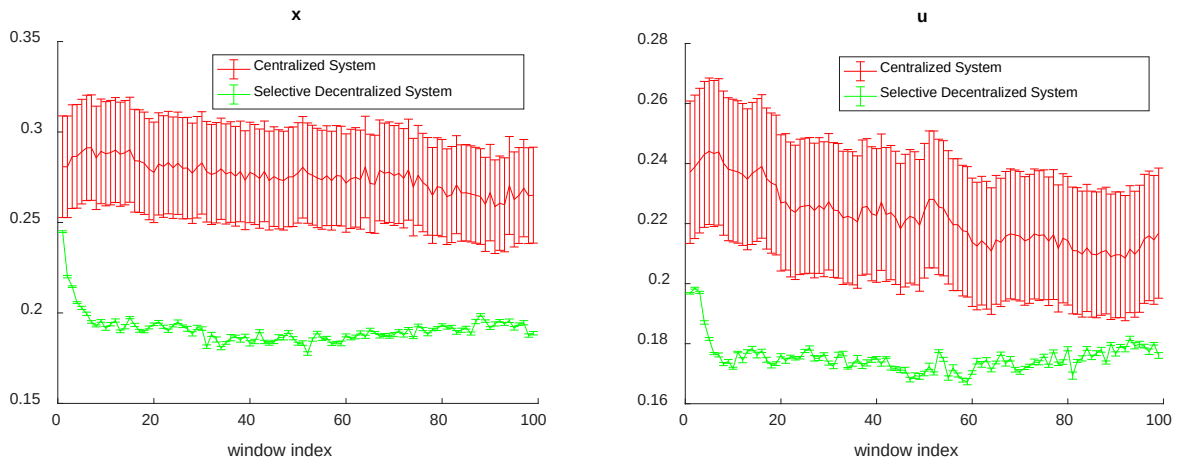

Figure 8.15. A typical example of how **x** and **u** converge in small-noise scenario in weakly coupled system with Q-learning; here, the Gaussian noise standard deviation is 0.05 (large noise) and $\sigma = 0.05$

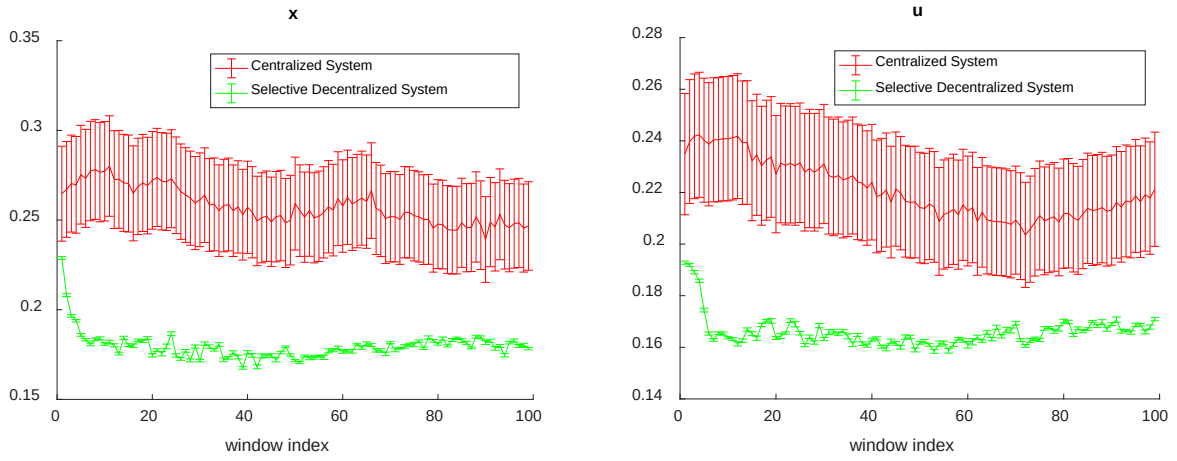

Figure 8.16. A typical example of how **x** and **u** converge in small-noise scenario in strongly coupled system with Q-learning; here, the Gaussian noise standard deviation is 0.05 (large noise) and $\sigma = 0.15$

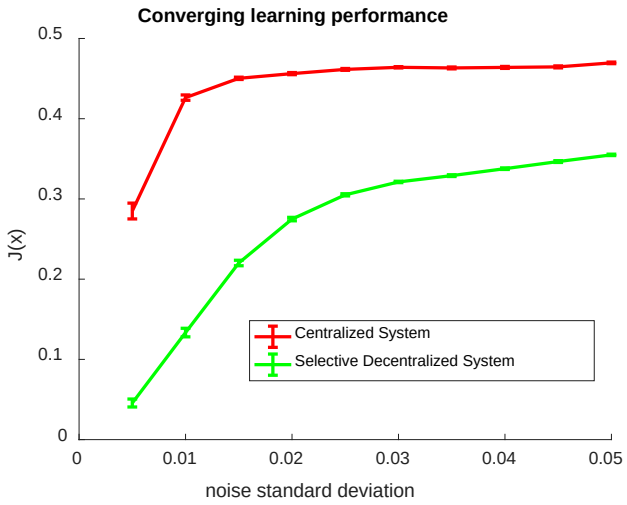

Figure 8.17. The converging learning performance of the selectively decentralized and the centralized Q-learning when the noise standard deviation increases

8.1.4 MDO

The systems used in MDO noise simulation is the same to the systems in sections 8.1.2 and 8.1.3 for discretized MDP and Q-learning. In MDO, we only implemented the MDO-IDF approach, which is showed to outperform the MDO-IF approach in chapter 7. For MDO agent discretization (chapter 7), all MDO agents discretize its state and action dimension into 7 regions. The other parameters for MDO are identical to the simulation showed in section 7.4.

Compared to the centralized approach, the MDO shows less learning performance when the noise level is low. However, when the noise level increases, MDO performance starts approaching and eventually outperforms the centralized performance, as showed in Figure 8.22. Figures 8.18 and 8.19 show typical examples of the MDO performance in low-noise scenario. Interestingly, in these figures, both MDO and the centralized approach could converge **x** to relatively similar levels. This phenomena also appears when the noise level increases. Therefore, we conclude that the performance of MDO is mostly decided by the action vector **u**. Figures 8.20 and 8.21 show that

with large noise, similar to the other approaches, both **x** and **u** do not converge. In these two figures, the noise standard deviation is 0.1.

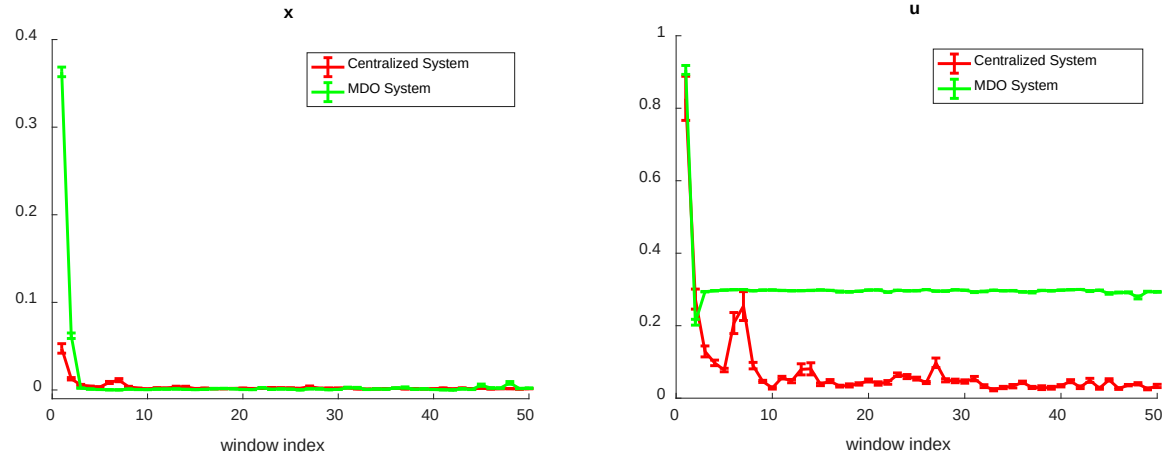

Figure 8.18. A typical example of how **x** and **u** converge in small-noise scenario in weakly coupled system with MDO; here, the Gaussian noise standard deviation is 0.005 (small noise) and $\sigma = 0.05$

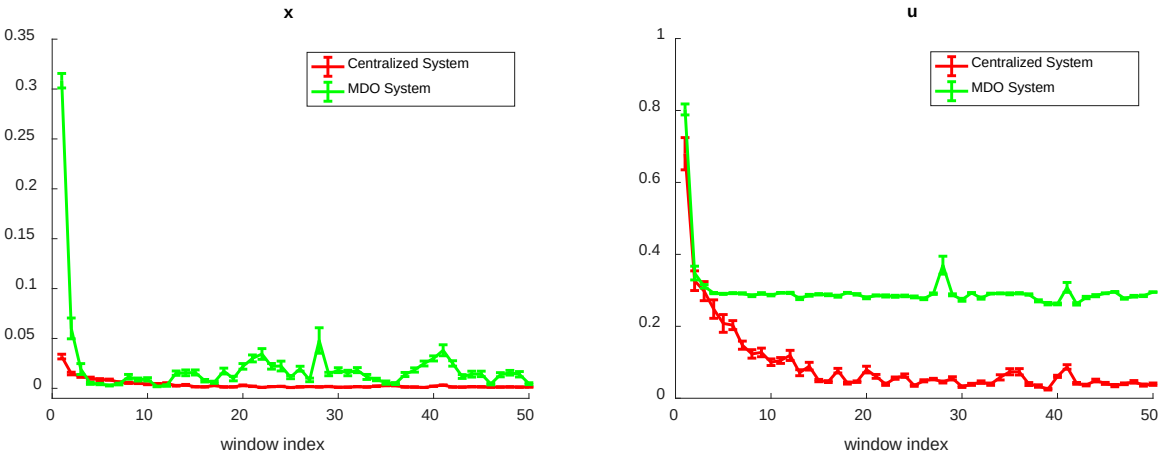

Figure 8.19. A typical example of how **x** and **u** converge in small-noise scenario in strongly coupled system with MDO; here, the Gaussian noise standard deviation is 0.005 and $\sigma = 0.5$

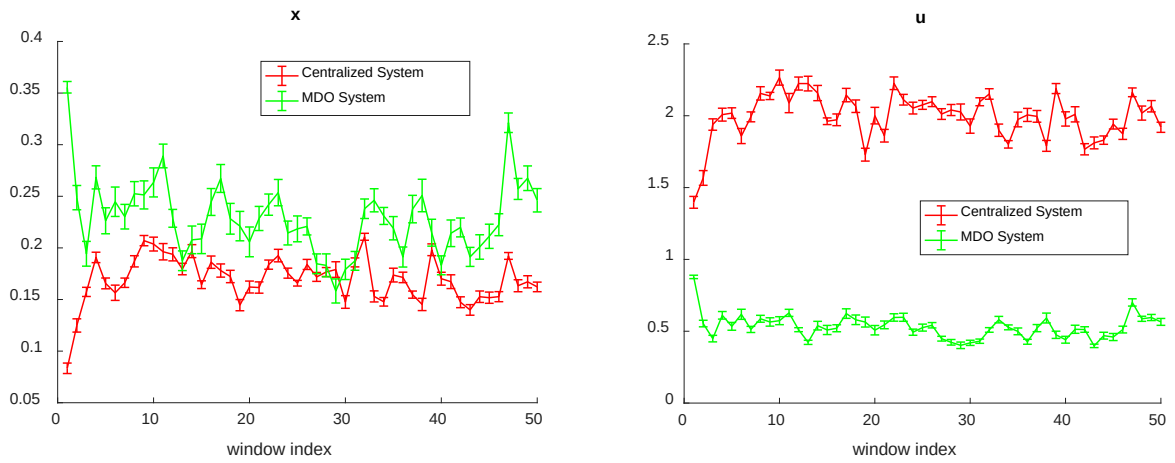

Figure 8.20. A typical example of how **x** and **u** do not converge in small-noise scenario in weakly coupled system with MDO; here, the Gaussian noise standard deviation is 0.05 (large noise) and $\sigma = 0.05$

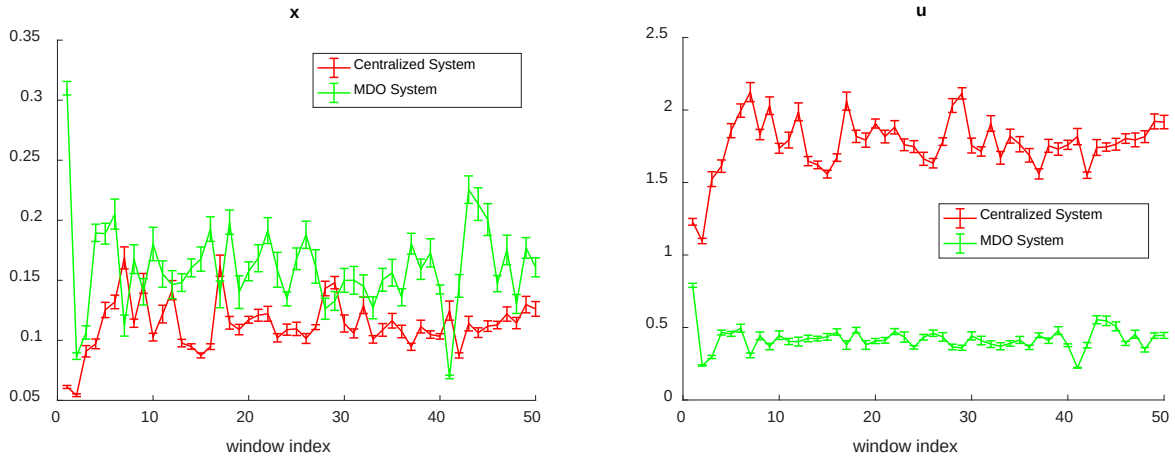

Figure 8.21. A typical example of how **x** and **u** do not converge in small-noise scenario in strongly coupled system with MDO; here, the Gaussian noise standard deviation is 0.05 (large noise) and $\sigma = 0.25$

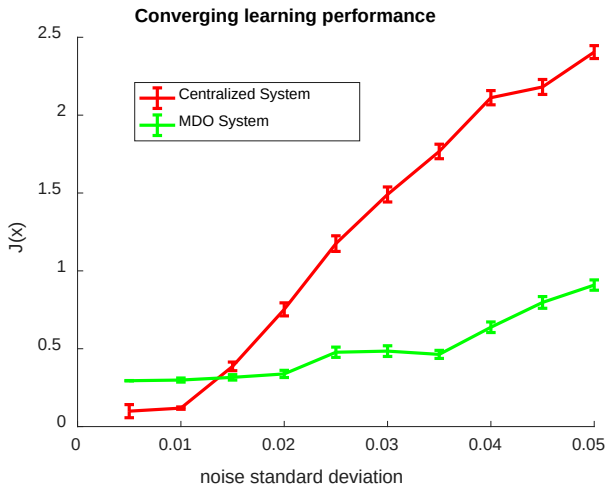

Figure 8.22. The converging learning performance of the MDO and the centralized system when the noise standard deviation increases

8.2 Discussions

This chapter confirms the robustness of the decentralized approaches, including selective decentralization and MDO, in learning and control noisy systems. For all of the experiments showed in section I, we observe that by the end of the experiment, the decentralized approaches reach lower value of the minimization objective $J(\mathbf{x})$. In addition, we still see that selective decentralization has better converging speed, compared to the centralized approach, as showed in the linear and Q-learning systems. These finding are accordant to the claim that the decentralized approaches are less susceptible to uncertain system parameters [40], which is the noise in this chapter.

For model-based technique, our result suggests that the learning performance $J(\mathbf{x})$ is worse when the noise level increase primarily due to the higher system identification error. Figures 8.4 and 8.5 strongly demonstrates this fact in the linear system case. Therefore, noise-filtering techniques should be applied in combination with the selective decentralization approach to reduce system identification error.

9. CASE-STUDIES

This chapter demonstrates the application of selectively decentralized reinforcement learning in two real-world case-studies. In the first case, I apply the linear-quadratic-regulator algorithm (chapter 6) in stabilizing the mass-spring system [100], which is the building block for automatic braking system. Here, each mass is a subsystem characterized by its location and velocity, which compose the mass state. Each mass could move horizontally on both direction, and could apply force horizontally to speed up or slow down its motion (control). These masses connect to each other by several springs. When some mass drifts away from its default resting position, the springs' elastic forces trigger all masses moving. In this case, the objective is to compute each mass' controlling force needed to stop the masses' motion and bring them back to the resting positions, with consideration of conserving the controlling forces. In the second case, we hypothesize that the gene expression in cancer is directed by a model-able biological system, which could be formulated and solve by existing control system technologies. In complex diseases, such as cancer, we observe that many genes express abnormally, either overexpression or underexpression. Therefore, the common practice in drug and treatment design is to reverse the expression of abnormally expressed genes so that the gene expressions return back to the balance, or 0 level. This practice could be easily formulated by existing reinforcement learning and control system theory. The treatment, such as drug, could be considered as a sub-optimal solution to control the biological system. In all of these case-studies, we show that the selectively decentralized approaches could improve the overall learning-and-controlling performance, compared to the centralized approach.

9.1 Learning to control the mass-spring system

9.1.1 System formulation

Figure 9.1 demonstrates the mass-spring system of three masses landing and moving horizontally. Masses (measured by kg) m_1 and m_3 connect to the fixed wall by springs (measured by elasticity constant unit kg/m^2) k_1 and k_3. Mass m_2 stands between m_1 and m_3, and connects to the other masses by springs k_{12} and k_{23}. The resting positions of m_1, m_2 and m_3 are p_1, p_2 and p_3, correspondingly. Without the loss of generalization in the theory and result, suppose that m_1, m_2 and m_3 stay at q_1,q_2 and q_3 such that k_1, k_3 are compressed and k_{12}, k_{23} are stretched as the Figure 9.1 (lower) show.

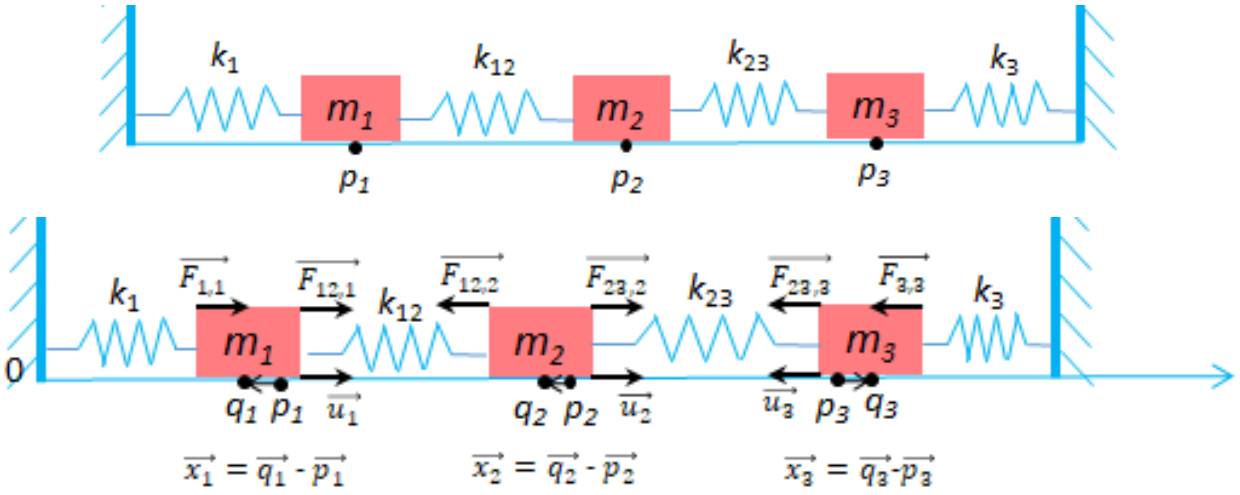

Figure 9.1. The 3-mass mass-spring system: (upper) at the resting positions; (lower) the forces applying on these masses (in bold and black arrows) when the masses are not at the resting positions

Analyzing the forces action on each mass, we have

- Mass m_1 has: force $\overrightarrow{F_{1,1}}$ caused by the compressed k_1 pushing to the right, force $\overrightarrow{F_{12,1}}$ caused by the stretched k_{12} pushing to the right, and individual control force $\overrightarrow{u_1}$ pushing to the right in order to return to the resting point p_1.
- Mass m_2 has: force $\overrightarrow{F_{12,2}}$ caused by the stretched k_{12} pushing to the left, force $\overrightarrow{F_{23,2}}$ caused by the stretched k_{23} pushing to the right, and individual control force $\overrightarrow{u_2}$ pushing to the right

in order to return to the resting point p_2.

Mass m_3 has: force $\overrightarrow{F_{3,3}}$ caused by the compressed k_3 pushing to the left, force $\overrightarrow{F_{23,3}}$ caused by the stretched k_{23} pushing to the left, and individual control force $\overrightarrow{u_3}$ pushing to the left in order to return to the resting point p_3.

Writing the second Newton's law vector-equations [101] for these masses, we have

$$\begin{cases} m_1\overrightarrow{a_1} = \overrightarrow{F_{1,1}} + \overrightarrow{F_{12,1}} + \overrightarrow{u_1} \\ m_2\overrightarrow{a_2} = \overrightarrow{F_{12,2}} + \overrightarrow{F_{23,2}} + \overrightarrow{u_2} \\ m_3\overrightarrow{a_3} = \overrightarrow{F_{3,3}} + \overrightarrow{F_{23,3}} + \overrightarrow{u_3} \end{cases} \quad (9.1)$$

Where $\overrightarrow{a_1}, \overrightarrow{a_2}$ and $\overrightarrow{a_{13}}$ stand for the accelerations of m_1, m_2 and m_3, correspondingly of. Applying Hooke's law for elastic spring [102], we can write (9.1) as

$$\begin{cases} m_1\overrightarrow{a_1} = -k_1(\overrightarrow{q_1} - \overrightarrow{p_1}) - k_{12}[(\overrightarrow{q_1} - \overrightarrow{p_1}) - (\overrightarrow{q_2} - \overrightarrow{p_2})] + \overrightarrow{u_1} \\ m_2\overrightarrow{a_2} = -k_{12}[(\overrightarrow{q_1} - \overrightarrow{p_1}) - (\overrightarrow{q_2} - \overrightarrow{p_2})] + -k_{23}[(\overrightarrow{q_3} - \overrightarrow{p_3}) - (\overrightarrow{q_2} - \overrightarrow{p_2})] + \overrightarrow{u_2} \\ m_3\overrightarrow{a_3} = -k_3(\overrightarrow{q_3} - \overrightarrow{p_{13}}) - k_{23}[(\overrightarrow{q_3} - \overrightarrow{p_{31}}) - (\overrightarrow{q_2} - \overrightarrow{p_2})] + \overrightarrow{u_3} \end{cases} \quad (9.2)$$

We can denote the displacement vectors as $\overrightarrow{x_1} = (\overrightarrow{q_1} - \overrightarrow{p_1}), \overrightarrow{x_2} = (\overrightarrow{q_2} - \overrightarrow{p_2}), \overrightarrow{x_3} = (\overrightarrow{q_3} - \overrightarrow{p_3})$. Also, choosing the reference axis horizontally from left to right, we have

$$\begin{cases} m_1a_1 = -k_1x_1 + k_{12}(x_2 - x_1) + u_1 \\ m_2a_2 = k_{12}(x_2 - x_1) + k_{23}(x_2 - x_3) + u_2 \\ m_3a_3 = -k_3x_3 + k_{23}(x_2 - x_3) + u_3 \end{cases} \quad (9.3)$$

In (9.3), the positive value x implies that vector $\vec{x}$ has direction from left to right, and the negative value x implies that vector $\vec{x}$ has direction from right to left. Let v_1, v_2 and v_3 denote the velocity of m_1, m_2 and m_3, correspondingly. From (9.3), we can write

$$\begin{cases} m_1 \dfrac{d(v_1)}{dt} = -k_1 x_1 + k_{12}(x_2 - x_1) + u_1 \\ v_1 = \dfrac{d(x_1)}{dt} \\ m_2 \dfrac{d(v_2)}{dt} = k_{12}(x_2 - x_1) + k_{23}(x_2 - x_3) + u_2 \\ v_2 = \dfrac{d(x_2)}{dt} \\ m_3 \dfrac{d(v_3)}{dt} = -k_3 x_3 + k_{23}(x_2 - x_3) + u_3 \\ v_3 = \dfrac{d(x_3)}{dt} \end{cases} \tag{9.4}$$

Discretize (9.4) by small interval Δt, we have

$$\begin{cases} m_1 \dfrac{v_1(t+1) - v_1(t)}{\Delta t} = -k_1 x_1(t) + k_{12}(x_2(t) - x_1(t)) + u_1(t) \\ v_1(t) = \dfrac{x_1(t+1) - x_1(t)}{\Delta t} \\ m_2 \dfrac{v_2(t+1) - v_2(t)}{\Delta t} = k_{12}(x_2(t) - x_1(t)) + k_{23}(x_2(t) - x_3(t)) + u_2(t) \\ v_2(t) = \dfrac{x_2(t+1) - x_2(t)}{\Delta t} \\ m_3 \dfrac{v_3(t+1) - v_3(t)}{\Delta t} = -k_3 x_3(t) + k_{23}(x_2(t) - x_3(t)) + u_3(t) \\ v_3(t) = \dfrac{x_3(t+1) - x_3(t)}{\Delta t} \end{cases}$$

$$\Leftrightarrow \begin{cases} x_1(t+1) = x_1(t) + \Delta t \times v_1(t) \\ v_1(t+1) = -\dfrac{\Delta t}{m_1}(k_1 + k_{12}) \times x_1(t) + v_1(t) + k_{12}\dfrac{\Delta t}{m_1} x_2(t) + \dfrac{\Delta t}{m_1} u_1(t) \\ x_2(t+1) = x_2(t) + \Delta t \times v_2(t) \\ v_2(t+1) = \dfrac{-k_{12}\Delta t}{m_2} \times x_1(t) + \dfrac{(k_{12} + k_{23})\Delta t}{m_2} \times x_2(t) \\ \quad + v_2(t) - \dfrac{k_{23}\Delta t}{m_2} \times x_3(t) + \dfrac{\Delta t}{m_2} u_2(t) \\ x_3(t+1) = x_3(t) + \Delta t \times v_3(t) \\ v_3(t+1) = k_{23}\dfrac{\Delta t}{m_3} x_2(t) - \dfrac{\Delta t}{m_3}(k_3 + k_{23}) \times x_3(t) + v_3(t) + \dfrac{\Delta t}{m_3} u_3(t) \end{cases} \tag{9.5}$$

Therefore, with $\mathbf{x} = \begin{bmatrix} x_1 \\ v_1 \\ x_2 \\ v_2 \\ x_3 \\ v_3 \end{bmatrix}$ and $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$, system (9.1) is linearly equivalent to

$$\mathbf{x}(t+1) = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ -\frac{\Delta t}{m_1}(k_1 + k_{12}) & 1 & k_{12}\frac{\Delta t}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ \frac{-k_{12}\Delta t}{m_2} & 0 & \frac{(k_{12}+k_{23})\Delta t}{m_2} & 1 & -\frac{k_{23}\Delta t}{m_2} & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & k_{23}\frac{\Delta t}{m_3} & 0 & -\frac{\Delta t}{m_3}(k_3 + k_{23}) & 1 \end{bmatrix} \mathbf{x}(t)$$

$$+ \begin{bmatrix} 0 & 0 & 0 \\ \frac{\Delta t}{m_1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{\Delta t}{m_2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta t}{m_3} \end{bmatrix} \mathbf{u}(t) \quad (9.6)$$

Bringing these masses toward the resting position implies that $v_1 = v_2 = v_3 = 0$ and displacement $x_1 = x_2 = x_3 = 0$, or $\mathbf{x} = 0$. In addition, by conserving the control unit, we have the learning goal by minimizing

$$\sum_{t=1}^{\infty} \gamma^t \left(\mathbf{x}(t)^T \mathbf{P} \mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{Q} \mathbf{u}(t) \right) \quad (9.7)$$

Where **P** and **Q** are positive definite matrices. From (9.6) and (9.7), we can compute the control unit to (9.1) by applying the algorithms showed in chapter 6.

9.1.2 Experiment

I setup the experiment with the following parameters. The masses are $m_1 = m_2 = m_3 = 1$ (kg). The spring elastic constants are $k_1 = k_3 = 1$ (kg/m^2), $k_{12} = k_{23} = 0.5$ (kg/m^2). The small time interval for linearization is $\Delta t = 0.01$ (s). The discount factor in equation (9.7) is 0.9. **P** and **Q** are identity matrices with 6 and 3 dimensions, correspondingly. Initially, the displacements are $x_1 = -0.5$ (m), $x_2 = -0.3$ (m) and $x_3 = 0.2$ (m), and the initial velocities for these masses are $v_1 = v_2 = v_3 = 0$ (m/s). The learning algorithm is the same to section 6.3.1 (chapter 6), with learning rate $\alpha = 0.05$.

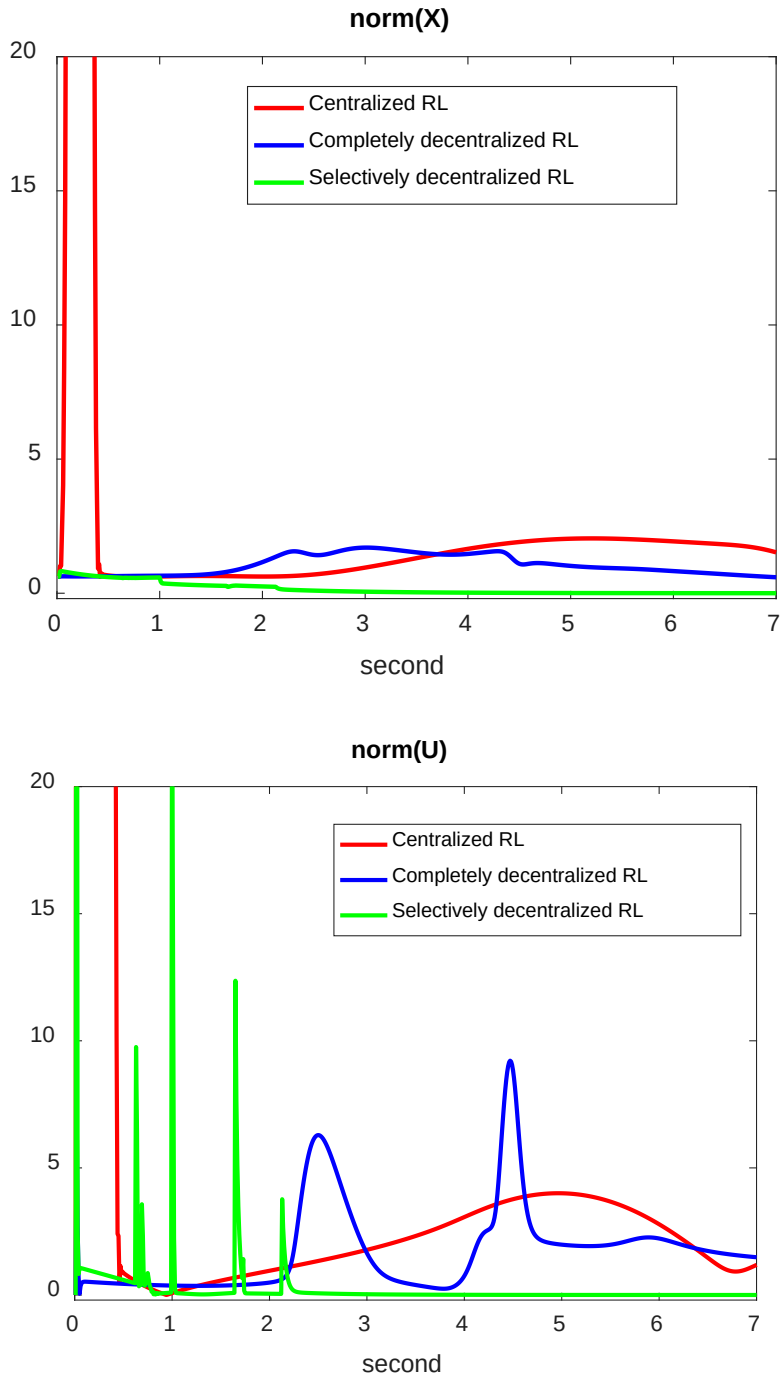

Figure 9.2. Learning and control performance of the approaches: centralized reinforcement learning (RL), completely decentralized RL and selectively decentralized RL; top figure: state trajectory; bottom figure: control trajectory

Figure 9.2 shows that the selectively decentralized approach outperforms the centralized and the completely decentralized approaches in stabilizing the system. Both the completely decentralized

RL and the centralized RL fails to stabilize the system. The completely decentralized RL could bring the masses closer to the resting point. In the other hands, the selectively decentralized RL stabilizes the system within 3 seconds, by bringing the masses toward the resting positions and stop the masses movement.

9.2 Potential application in drug discovery / repositioning

9.2.1 Motivation of applying reinforcement learning in drug repositioning

Drug repurposing (also called drug repositioning) has become one of the most active areas in pharmacology since last decade [103] because this approach could significantly reduce the cost and time to invent a new treatment. One of the key reasons for low productivity in traditional drug development is the lack of systematic evaluation of additional indications [104], which may lead to unexpected side effects and low efficacy. Briefly, drug repurposing finds new indications for known drugs and compounds [105] to reduce the risk of failure and shorten time of discovery, because it does not need the time to discover and test the new chemical compounds. Drug repurposing applies modern computational techniques to digitalize genomic [106], bioinformatics, chemical informatics [107] and patients' individual health records [108] to offer more systematic evaluation of the chemical compound before entering the laboratory testing and clinical trial steps. Prominent successful examples for drug repurposing include Viagra, Avastin and Rituxan [104].

The application of reinforcement learning into system biology and drug repurposing is promising from the following points. From the mathematical system-model-control-based point of view, there exist a mechanism regulating the gene expression profile. In the healthy condition, the gene expression stays in the stable equilibrium region such that $\mathbf{x}(t) = f(\mathbf{x}(t\text{-}1)) \approx \mathbf{x}(t\text{-}1)$, where f indicates the expression-regulating mechanism computed from data integration, $\mathbf{x}$ stands for expression and

t stands for time. The essential components of the mechanism could be retrieved, although containing uncompleted and possibly wrong and conflicting information, from the system biology data sources [109-115]. In the disease state, the critical gene expression strays outside the stable region. In this case, without a control (treatment), the expression will be unbounded. The system control algorithms aim to find the sequence of control-treatment that optimally stabilize the expression back to the original equilibrium point, such as linear control [116, 117], nonlinear control [118, 119], adaptive neural network [120, 121]. By comparing the real drug treatments with the optimal control-treatment (also called hypo-treatment), we can evaluate the potential efficacy of the drug before being repurposed. These points demonstrate three reasons why decentralized reinforcement learning could become a solution in drug repurposing. First, the repurposing problem could be transformed to an adaptive learning and control problem. Second, the repurposing problem contains the unknown nature, which is the fundamental nature of reinforcement learning. Third, the system biology itself is large if count by number of genes involving in a disease or process, and contains underlying modules for decentralization.

However, applying mathematical system modeling and control in drug repurposing is still in very early steps. There are three key challenges in applying system control approach. First, it is difficult to quantify the gene expression and real drug treatment, as there is very little literature discussing the ‘normal range’ of each gene’s expression. Second, constructing a comprehensive and accurate mathematical model to simulate the gene expression change is complicated due to the diversity of gene-gene interaction mechanisms, mutation, and under-discovered data. Third, the biological systems are known for large scale for system control: there may be from hundreds to thousands of genes of interest in a specific disease or biological process. These challenges are similar to the

challenges which selectively decentralized reinforcement learning aims for. Therefore, I would like to take explore applying selectively decentralized reinforcement learning and control in drug repurposing. Most of the content in section 9.2 has been presented my paper at [122], with details on data set, system setup and biological insights in Appendix A.

9.2.2 Overall ideas of drug repositioning based on reinforcement learning and control

Figure 9.3 shows the overall ideas on repurposing framework from the system modelling and control points. The framework integrates three types of data. First, from the Disease-specific expression profile, I quantified the expression as the system initial condition vector, where each vector elements specified whether the corresponding gene gene was overexpressed (red), underexpressed (green) or normally expressed (white). Second, from the protein-protein interaction database, I built the mathematical system model in order to apply the system-control algorithm. The red arrows implies activative; and the green arrow implies inhibitive interactions. Third, from the chemical-protein interaction data, I quantified the treatment vector for each drug for later ranking. Using the initial condition vector and the mathematical model, I computed the optimal hypo-treatment. By mapping the pattern of the optimal hypo-treatment and the drugs' treatment vectors, I could rank the drugs and suggest repurposed drugs.

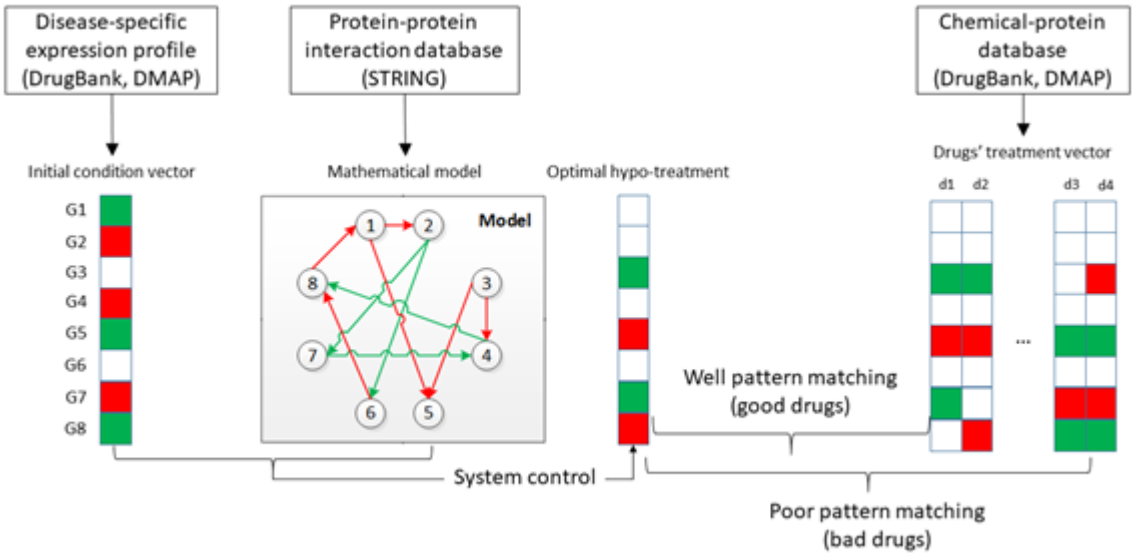

Figure 9.3. Overview of RL-system control-based drug repurposing frameworks. Red squares: overexpressed genes/ drug's activation. Green squares: under expressed genes / drug's inhibition. Red arrow: activated protein-protein interaction. Green arrows: inhibited protein-protein interaction.

The key principle in applying system control to evaluate drugs' therapy relies in the following assumption: in disease condition, the gene expressions are derived away from the balanced level of 0. Therefore, a good treatment should reverse the gene expressions in disease condition and stabilize the expressions to the balance level. In addition, based on system biology literature [123], I assume that there exists a model governing the gene expressions, which allows us to model the expression using time-series perspective

$$\mathbf{x}(t) = f(\mathbf{x}(t\text{-}1), \mathbf{u}(t\text{-}1)) \quad (9.8)$$

where $\mathbf{x} \in \Re^N$ stands for the quantified gene expression of N genes, $\mathbf{u} \in \Re^N$ stands for the quantified treatment and t is the iteration and f is the arbitrary function controlling the expression change. The initial $\mathbf{x}(0)$ is the quantified gene expression in disease condition. In this thesis, I choose a linear model for f.

$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t\text{-}1) + \mathbf{u}(t\text{-}1) \quad (9.9)$$

I chose the linear model because not only it is simple but also it has equilibrium point at the origin: if $\mathbf{x}(t\text{-}1) = \mathbf{u}(t\text{-}1) = 0$ then $\mathbf{x}(t) = 0$. This fact implies that when the gene expressions are already at the balance level, treatment is no longer needed. In addition, it is easier to setup a linear system with stability [124]

$$\text{If } \|\mathbf{x}(0)\| < \varepsilon \text{ and } \mathbf{u} = 0 \text{ then } \|\mathbf{x}(t)\| < \varepsilon \ \forall t \quad (9.10)$$

where $\|\mathbf{x}\|$ stands for the second norm of $\mathbf{x}$ and ε is an arbitrary small number. This fact implies the self-adjustment of the gene expression at the control level. I setup matrix $\mathbf{A}$ from quantification of protein-protein mechanism of interactions (section 9.II.3). With temporal matrix $\mathbf{A}^*$ as the result of section III.1

$$\mathbf{A}^*(i,j) = \begin{cases} -1 \text{ if protein } i \text{ inhibits protein } j \\ 1 \text{ if protein } i \text{ activates protein } j \\ 0 \text{ otherwise} \end{cases} \quad (9.11)$$

Let λ be the eigenvalue of $\mathbf{A}^*$ with the largest magnitude. By setting up $\mathbf{A}$ as

$$\mathbf{A} = (1/\lambda)\mathbf{A}^* \quad (9.12)$$

We can guarantee the stability of system (9.12) [124]. From this point, we can see that the problem could be solved by applying the algorithms for linear system showed in chapter 6.

In system control practice, since $\mathbf{u}(t)$ often converges to 0 quickly [125], the first treatment vector $\mathbf{u}(0)$ often plays the most important role in optimally stabilizing the system (9.9). Therefore, we can consider $\mathbf{u}(0)$ as the optimal hypo-treatment. We compare the similarity between the real drug treatment and the hypo-treatment as the therapeutic score $T(d)$ for each drug d as follow

$$T_d = \frac{\left|\mathbf{u}_d^T \text{sign}\left(\mathbf{u}(0)\right)\right|}{\left|\text{abs}(\mathbf{u}_d)^T \text{abs}\left(\text{sign}\left(\mathbf{u}(0)\right)\right)\right|} \quad (9.13)$$

where abs stand for the absolute value function. The numerator $|\mathbf{u}_d^T \text{sign}(\mathbf{u}(0))|$ is the matching function between drug *d* and the optimal hypo-treatment, which is incremented when $\mathbf{u}_d(i)$ and $\mathbf{u}(0)(i)$ are non-zero analog, and decremented when $\mathbf{u}_d(i)$ and $\mathbf{u}(0)(i)$ are opposite.

9.2.3 Setup the system for drug repositioning from Breast Cancer-omics data

This section briefly summarizes one example on my collaborators and I could integrate multi-types of omics data to setup the learning and control system for drug repositioning, with Breast Cancer as the case-study disease. More details related to this section could be found at Appendix A.

We built an integrated breast cancer specific pathway model that accelerates drug discovery by having more disease specific proteins and coverage than any other breast cancer specific pathway. This completes the fully curated pathway model (M), which is verified on WikiPathway (http://www.wikipathways.org/index.php/Pathway:WP1984). We optimized the pathway M to increase the simulation quality based on domain knowledge drug-protein interaction data (M^*). For general information, our breast cancer pathway model M^* contains 228 proteins and 481 protein-protein interactions and enables repurposing 63 drugs [122].

For drug-target information, we used the drug list suggested by Huang et al [126] as the initial drug list. With this initial drug list, we us shared target, shared side effect and similar chemical structure method to expand the drug list using the following databases [127-129]. The drug list after this expansion contained 82 drugs. We manually curated the target information for these 82 drugs and removed drugs having none or ambiguous target information. There are 68 drugs having clear target information as can be seen in [122]. Among these 68 drugs, 63 drugs reach more than 20 effectors via the pathway model and are selected in the experiment. Among these 63 drugs, in

this chapter, I only focus on 23 drugs from the following two categories. The first category (D1) includes 16 drugs approved by the FDA for breast cancer, which are considered as 'positive' drugs. The second category (or D2') contains 7 drug withdrawn from Breast Cancer treatment, which are considered as 'negative' drugs.

For gene expression profile, we chose significantly expressed gene genes after applying expression analysis on dataset GSE10886 from Gene Expression Omnibus (GEO) database. GSE10886 is among the largest and most comprehensive Breast Cancer microarray in GEO. After the latest update in January 2013, GSE10886 has 226 samples and includes samples from both ER+ and ER- subtypes. We discretize the gene expression into +1 for overexpression, -1 for underexpression and 0 otherwise. The discrete gene expression will be used as the initial state vector $\mathbf{x}(0)$ in the control algorithm later.

9.2.4 Selectively decentralized approach improve the capability of detecting therapeutic drugs for Breast Cancer

In this section, I focus on the computational results of the centralized and selectively decentralized RL in a simpler task: classifying the drugs that are approved or rejected/withdrawn from Breast Cancer treatment. For any drug repositioning technique, this is the fundamental task to validate the potential capability of the technique. Only when the repositioning technique achieves good performance in classifying approved versus rejected/withdrawn drugs, the technique could be confidently used in suggesting new treatment. As showed in section 9.2.2, the T_d score in formula (9.13), derived from RL techniques (chapter 6), is the metric for scoring the drugs. More results, especially on biological and pharmacological impacts, could be found in Appendix A. The methodological details, which contains mostly the techniques in data collection and preprocessing,

could be found at [122]. I omit these methodological details because they are long and irrelevant to the main contribution of this thesis.

Figures 9.4 and 9.5 shows that the selectively decentralized approach significantly improved the performance in classifying drugs. Here, to partition the system (10-12) to subsystem, we use the fast modularity clustering algorithm [130] to detect 9 subsystems. The improvement is significant in Breast Cancer ER+ subtype (10% AUC increase).

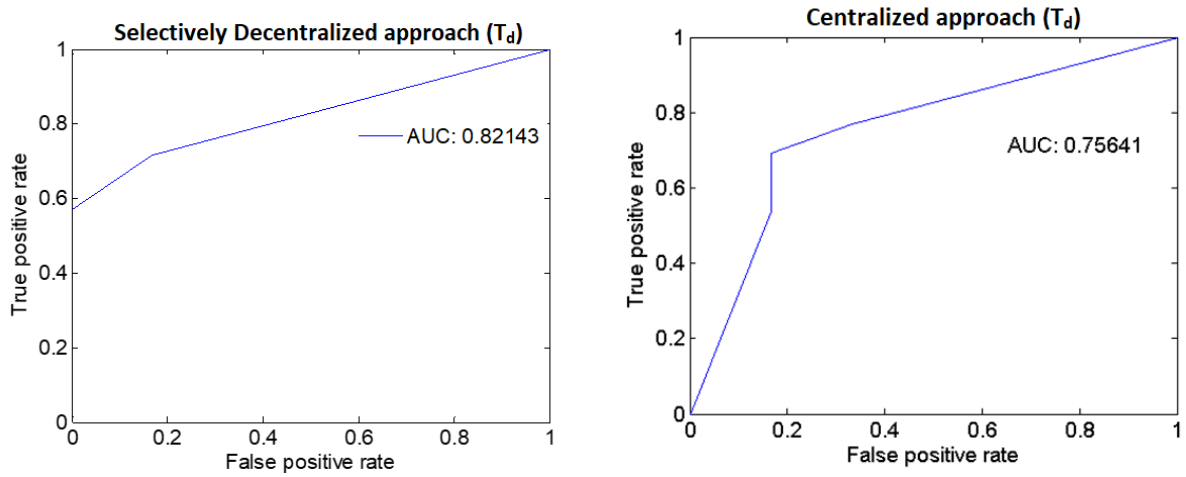

Figure 9.4. Comparison between the selectively decentralized and the centralized RL approach in classifying drugs for Breast Cancer ER+ subtype disease.

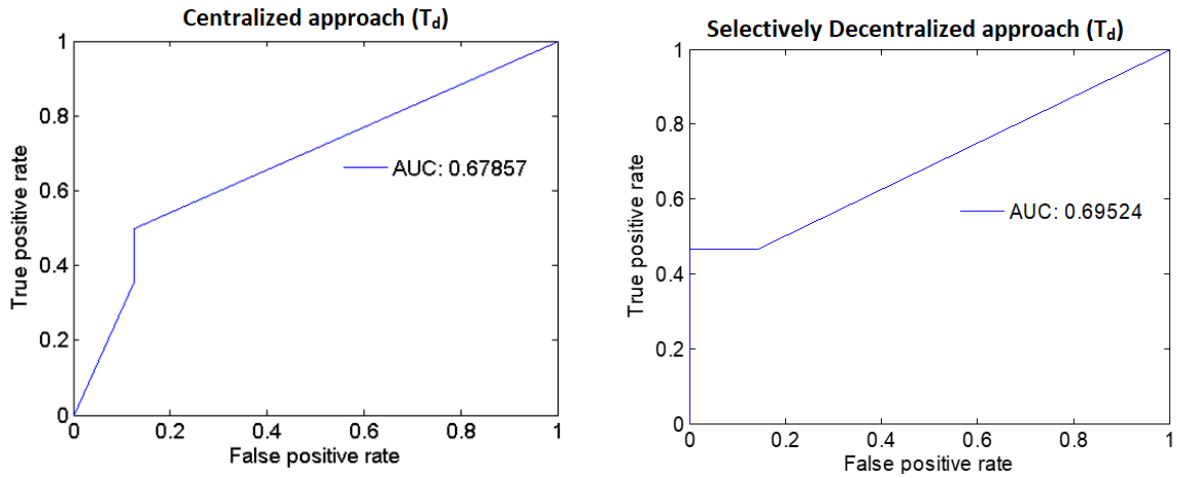

Figure 9.5. Comparison between the selectively decentralized and the centralized RL approach in classifying drugs for Breast Cancer ER- subtype disease.

10. CONCLUSIONS

In this thesis, we have proposed a new approach for decentralized reinforcement learning: selective decentralization, and showed how this approach improves the learning performance, compared to the centralized and completely decentralized approaches. The selectively decentralized approach tackle the four major challenges in decentralized learning as follow. For the first two questions of ‘when’ and ‘to whom’ a single agent should communicate, a central coordinator would answer these questions based on maximizing the learning fitness (i.e. the best identification error, the fastest cumulative learning Q-value) instead of maximizing the learning goal. For the third question of ‘sharing information’, the agents are assumed to freely send its whole state and action to the others. For the fourth question of ‘using shared information’, the agents could discretize the shared information to reduce the computational resources for this information. In addition, the selective decentralization is also able to learn the optimal communication scheme among the learning agents without any prior knowledge of communicative structure. This learning capability is also another innovation of the selectively decentralized approach.

The thesis also tackle some challenges in single-agent and general reinforcement learning. First, the selectively decentralized approach could easily incorporate a large number of well-known reinforcement learning techniques, from model-based learning to model-free learning. This flexibility is one of the innovation in the thesis, compared to most of the state-of-the-art decentralized learning, according to the best of our knowledge. Second, the thesis shows that applying MDP could solve larger scope of nonlinear reinforcement learning problems, compared to most of the state-of-the-art techniques focusing on problems in feedback-linearizable format. In addition, we first explore and verify the impact of the MDO approach in reinforcement learning.

Since the theoretical analysis of reinforcement learning and decentralized and distributed Q-learning mostly focuses on the existence of the optimal policy, we lack the theoretical explanation for the superior converging speed of selectively decentralized learning. Therefore, taking Q-learning as a typical example, [26, 46, 88], we try to explain the superior performance of selectively decentralized learning from two points of view. First, as stated in the foundation of Q-learning [26], the convergence of Q-learning assumes that all of the state-action entries in the Q-table are visited infinitely. Therefore, in order to converge to the optimal Q, the Q-learning systems are supposed to spend time to explore the Q-table. In figures 1 and 2 of chapter 4 where we show the convergence of centralized Q-learning, there are long periods where $||\mathbf{x}||$ and accumulate Q-gained change slowly. These periods may correspond to the exploration phases. Because the number of states, actions, and state-action entries grow exponentially with system dimensionality, decentralized Q-learning into smaller dimension may also improve the convergence exponentially due to exponentially less search space. Second, selectively decentralized Q-learning proposes more search options than centralized Q-learning, which is another factor to improve the converging speed. In centralized Q-learning, a newly visited state has no prior information to estimate its Q-table entries. With the same state, in selectively decentralized Q-learning, the components of the state have higher chance to be visited by the subsystem learner (in different centralized states), which may reduce the effort to compute the optimal Q-value.

The ‘art’ in this thesis is choosing the criteria for switching communication scheme. From our points of view, as showed in this thesis, the criteria should be specific to the problems and techniques which selective decentralization incorporates. For model-bases learning techniques, since the estimation of the dynamic changes in the environment is critical for the learning

performance, we choose system identification as the only criterion. For Q-learning, which is a model-free technique, we choose best decentralization scheme by the sum of subsystems' gained Q-values only because of the linearity in state-reward function, which is the main driver for Q-value update. However, there is no theoretical basis to support whether or not the different sum of subsystem gained Q-value in different decentralization scheme is comparable. There may exist more solid options for choosing the best decentralization scheme than cumulative gained Q-value. Similar to Q-learning, due to the difficulty to theoretically prove the switching criteria, this important point should be carefully examined by the scientist performing the experiment.

The outcomes in this thesis reflex and could be easily understood from the philosophical points of view. First, the 'trial and error' paradigm is showed in learning the optimal communication scheme. At the beginning, the central coordinator frequently switches among different communication schemes. Then, as the overall learning performance improves, the central coordinator decides less switching and eventually stops switching the communication scheme before all of the learning agents learn the stable policy. This 'trial and error' paradigm is typically demonstrated in selectively decentralized Q-learning. Second, at least in a cooperative task, trusting the behavior of the collaborators often lead to better results than doubting the incompetence or error from the collaborators. As showed in the MDO results, when a learning agent 'doubts' the performance other agents and makes the decision considering all of the possible scenarios from the others, the learning agent may not be able to learn the optimal policy. However, when an agent believes that the others at least do not do worse, if not doing better, the agent eventually learn the optimal policy faster than a centralized agent does.

There are several limitations in this thesis. First, for MDP approximation, the discretization thresholds need the distribution of the next state assuming that the current state and control vectors are uniformly distributed and may require a number of ad-hoc steps. Third, in selective decentralization, we still explore all possible decoupling scheme $B(k)$, which grows exponentially. However, since the selectively decentralized system converges faster than the centralized system in most of the cases, we believe that the heavily computational model-switching phase in the selective decentralized system will be relatively short. Therefore, the selectively decentralized system may be more computationally efficient than the centralized system, which must run the learning algorithm in high dimensional data for long term. Forth, due to the lack of collaboration opportunities, the case-studies presented in this thesis are limited to a well-known problem, which has been thoroughly studied, and to an unexplored problem of pharmacology, where the domain knowledge may or may not sufficient to ensure the quality of the modelling and controlling the system.

From the innovations and limitations in this thesis, we believe that the following points are still opened to explore in future works

1. The theoretical converging time for reinforcement learning and decentralized learning should be fully addressed.

2. The converging policy of decentralized learning should be further studied. What are the relationships, or mappings, between the decentralized policy and the centralized policy, and do they converge to the same point?

3. Techniques to reduce the search place for optimal communication schemes should be applied to avoid exponential computational time.

4. The selective decentralization should be applied in more real-world learning problems. Two promising areas for application are bioinformatics - system biology: personalized medicine and automatic vehicle control. Automatic vehicle control and robotics have been among the main applications for reinforcement learning for decades. System biology with large number of gene and undiscovered biological knowledge suggests that decentralized reinforcement learning could be a promising approach.

APPENDIX. DECOST FRAMEWORK: REINFORCEMENT LEARNING – CONTROL SYSTEM APPLICATION IN DRUG REPURPOSING

This appendix contains more details on the dataset, system setup and biological insights for section 9.2, where I present applying Reinforcement learning and control system in drug repurposing.

Biological insights

Therapeutic scores for Breast Cancer Drugs

From the Integrated Breast Cancer Pathway [131] on Wikipathway (section III.1) and the Breast Cancer drug list, we queried 222 drug-protein interactions for the drugs' treatment vectors (Supplemental Table S2). Supplemental Table S3 contains the initial condition vector from GEO2R expression analysis.

Figure A1 shows that the T_d score is able to give appropriate ranking for most of the well-known therapeutic drugs and suggest candidate drugs for repurposing in Breast Cancer ER-positive case. T_d score reflexes the difference between the D1 and D2 drugs with receiver operator characteristic [132] area under the curve (AUC) of 0.76. We did not setup training set and test set for classification because the model construction and T_d calculation does not need the drug categories. The T_d scores for D1 drugs in Breast Cancer ER-negative case are relatively lower than the scores for ER-positive case (Figure A2). Using T_d for classifying D1 and D2 drugs yields AUC of 0.68. In fact, clinical trials and literature have showed several drugs which are effective in ER-positive treatment but show little or no impact in ER-negative treatment. For example, Tamoxifen (T_d ER-positive: 0.294, T_d ER-negative: 0.176), which is a selective estrogen receptor modulator, does not prevent ER-negative Breast Cancer, when the estrogen receptor genes do not express [133, 134].

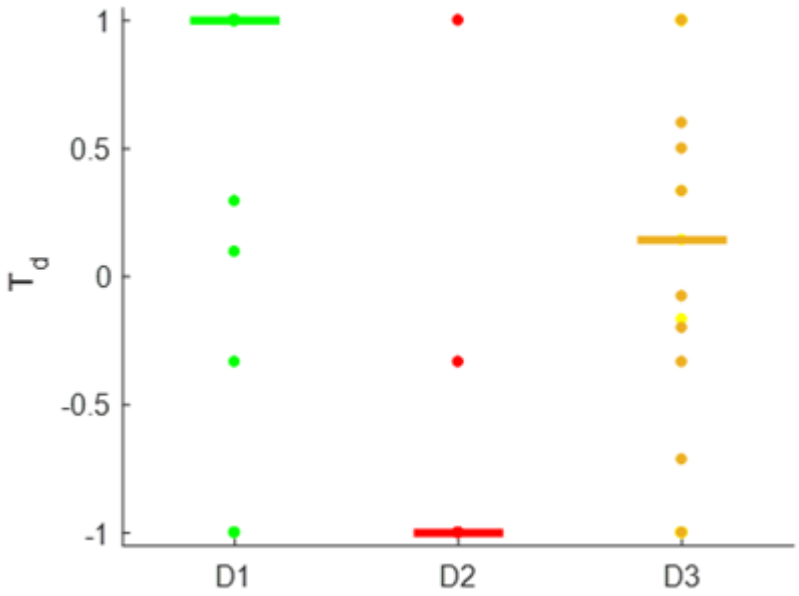

Figure A1. T_d score in Breast Cancer, ER-positive subtype; the horizontal bars in each group stand for median value of T_d

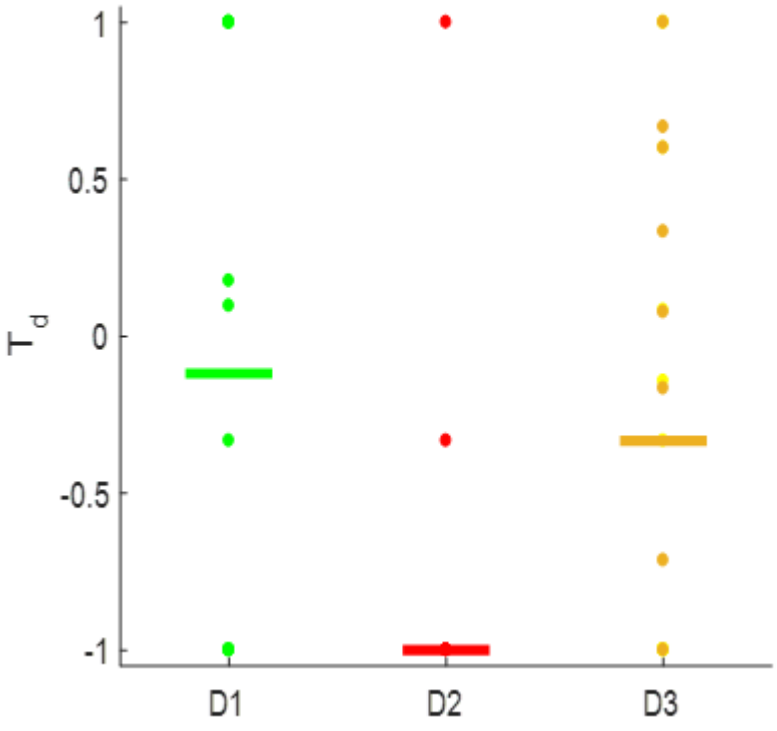

Figure A2. T_d score in Breast Cancer, ER-negative subtype; the horizontal bars in each group stand for median value of T_d

Potential drugs for Breast Cancer studies and biological insights

From the T_d scores for D3 drugs, our framework suggests 8 drugs (Erbitux, Flutamide, Medrysone, Methylprednisolone, Norethindrone, Prednisolone, Prednisonea and Vandetanib) with high potential efficacy in Breast Cancer ER+ drug repurposing. Significantly, these drugs do not directly target Estrogen receptor, which is the most well-known approach in Breast Cancer ER+ drug design. Tamoxifen is a typical example of Breast Cancer drugs which slows cancer process by blocking estrogen hormone receptors, preventing hormones from binding to them. About 80% of all breast cancers are ER+: the cancer cells grow in response to the hormone estrogen [135]. About 65% of the ER+ cases grow in response to another hormone, progesterone [136]. Tumors in ER/PR-positive cases are much more likely to respond to hormone therapy than tumors that are ER/PR-negative. ER+ breast cancer entirely depends on the estrogen for growth and propagation involving genomic and non-genomic pathways. Epidermal growth factor receptor (EGFR) is a receptor found on both normal and tumor cells that is important for cell growth [137]. ER-positive (ER+) drugs recommended for repurposing in this framework block the activities and growth of EGFR. These drugs show different mechanism of action with the common objective of the inhibition of the growth of cancerous cells. By adjusting and modifying the known biases of the interactomic networks, our procedure would help to reveal the therapeutic effect of drugs along with effective treatments.

For Breast Cancer ER- case, our framework suggests Daunorubicin and Donepezil as the repurposing candidates. These drugs are independent of estrogen and usually inhibit the cell growth by either interacting with DNA or inhibiting Cholinesterases. Daunorubicin interacts with DNA by intercalation and inhibition of macromolecular biosynthesis [138]. This inhibits the progression of the enzyme topoisomerase II, and thereby stopping the process of replication.

Donepezil is in a class of cholinesterase inhibitor that improves mental function and fatigue in cancer. The current research focused on recent large-scale efforts to systematically find repositioning candidates and elucidate individual disease mechanisms in cancer [139]. Personalized medicine and repositioning both aim to improve the productivity of current drug discovery pipelines. Standard drug discovery strategies can also lead to repositioning opportunities. D1, D2, and D3 drugs found to potently modulate the desired activity are repositioning candidates.

REFERENCES

1. Fujio, F.F., *Doraemon (Episode 1)*. 1999, Hanoi: Kimdong Publishing House.
2. Kozo, K., *Doreamon: Scary Rice Cake*, in *Doreamon*. 2005, TV Asahi: Tokyo.
3. Russell, S. and P. Norvig, *Artificial Intelligence: A Modern Approach, Third Edition*. 2010, New Jersey: Pearson.
4. Anomyous. *Ozma (Final Fantasy IX)*. 2017; Available from: http://finalfantasy.wikia.com/wiki/Ozma_(Final_Fantasy_IX).
5. Kmoch, H., *The Game of the Century.* Chess Review, 1956: p. 374.
6. Bradtke, S.J. and M.O. Duff, *Reinforcement learning methods for continuous-time Markov decision problems.* Advances in neural information processing systems, 1995: p. 393-400.
7. Kaelbling, L.P., M.L. Littman, and A.W. Moore, *Reinforcement learning: A survey.* Journal of artificial intelligence research, 1996. **4**: p. 237-285.
8. Keesman, K.J., *System Identification: an Introduction*. Advanced Textbooks in Control and Signal Processing, ed. M.J. Grimble and M.A. Johnson. 2011, London: Springer-Verlag.
9. Nelles, O., *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. 2013: Springer Science & Business Media.
10. Pillonetto, G., et al., *Kernel methods in system identification, machine learning and function estimation: A survey.* Automatica, 2014. **50**(3): p. 657-682.
11. Werbos, P.J. *Neural networks for control and system identification*. in *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on*. 1989. IEEE.
12. Sutton, R.S., A.G. Barto, and R.J. Williams, *Reinforcement learning is direct adaptive optimal control.* IEEE Control Systems, 1992. **12**(2): p. 19-22.
13. Bertsekas, D.P., *Dynamic programming and optimal control 3rd edition, volume II.* Belmont, MA: Athena Scientific, 2011.
14. Bellon, J., *Riccati Equations in Optimal Control Theory.* 2008.
15. Abu-Khalaf, M. and F.L. Lewis, *Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach.* Automatica, 2005. **41**(5): p. 779-791.
16. Saridis, G.N. and C.-S.G. Lee, *An approximation theory of optimal control for trainable manipulators.* Systems, Man and Cybernetics, IEEE Transactions on, 1979. **9**(3): p. 152-159.
17. Beard, R.W., G.N. Saridis, and J.T. Wen, *Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation.* Automatica, 1997. **33**(12): p. 2159-2177.
18. Huang, C.-S., S. Wang, and K. Teo, *Solving Hamilton—Jacobi—Bellman equations by a modified method of characteristics.* Nonlinear Analysis: Theory, Methods & Applications, 2000. **40**(1): p. 279-293.
19. Al-Tamimi, A., F.L. Lewis, and M. Abu-Khalaf, *Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2008. **38**(4): p. 943-949.
20. Liu, D. and Q. Wei, *Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems.* IEEE Transactions on Neural Networks and Learning Systems, 2014. **25**(3): p. 621-634.

21. Seiffertt, J., S. Sanyal, and D.C. Wunsch, *Hamilton–Jacobi–Bellman equations and approximate dynamic programming on time scales.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2008. **38**(4): p. 918-923.
22. Sutton, L.K.R. *Model-based reinforcement learning with an approximate, learned model*. in *Proc. Yale Workshop Adapt. Learn. Syst*. 1996.
23. Kamalapurkar, R., P. Walters, and W.E. Dixon, *Model-based reinforcement learning for approximate optimal regulation.* Automatica, 2016. **64**: p. 94-104.
24. Polydoros, A.S. and L. Nalpantidis, *Survey of Model-Based Reinforcement Learning: Applications on Robotics.* Journal of Intelligent & Robotic Systems, 2017: p. 1-21.
25. Strehl, A.L., et al. *PAC model-free reinforcement learning*. in *Proceedings of the 23rd international conference on Machine learning*. 2006. ACM.
26. Watkins, C.J. and P. Dayan, *Q-learning.* Machine learning, 1992. **8**(3-4): p. 279-292.
27. Kiumarsi, B., et al., *Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics.* Automatica, 2014. **50**(4): p. 1167-1175.
28. Albus, J.S., *A new approach to manipulator control: The cerebellar model articulation controller (CMAC).* Journal of dynamic systems, measurement and control, 1975. **97**(3): p. 220-227.
29. Ormoneit, D. and Ś. Sen, *Kernel-based reinforcement learning.* Machine learning, 2002. **49**(2): p. 161-178.
30. Berry, D.A. and B. Fristedt, *Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)*. 1985: Springer.
31. Sallans, B. and G.E. Hinton, *Reinforcement learning with factored states and actions.* Journal of Machine Learning Research, 2004. **5**(Aug): p. 1063-1088.
32. Crawford, D., et al., *Reinforcement Learning Using Quantum Boltzmann Machines.* arXiv preprint arXiv:1612.05695, 2016.
33. Ammar, H.B., et al. *Automatically mapped transfer between reinforcement learning tasks via three-way restricted boltzmann machines*. in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2013. Springer.
34. Lerman, L., et al. *Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis)*. in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. 2015. Springer.
35. Powell, W.B., *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. 2007: John Wiley & Sons.
36. Szepesvári, C. *The asymptotic convergence-rate of Q-learning*. in *Proceedings of the 10th International Conference on Neural Information Processing Systems*. 1997. MIT Press.
37. Wender, S., *Integrating Reinforcement Learning into Strategy Games*. 2009: Diplomarbeit, The University of Auckland, New Zealand.
38. Cramer, E.J., et al., *Problem formulation for multidisciplinary optimization.* SIAM Journal on Optimization, 1994. **4**(4): p. 754-776.
39. Busoniu, L., B. De Schutter, and R. Babuska. *Decentralized reinforcement learning control of a robotic manipulator*. in *2006 9th International Conference on Control, Automation, Robotics and Vision*. 2006. IEEE.
40. Ioannou, P.A., *Decentralized adaptive control of interconnected systems.* Automatic Control, IEEE Transactions on, 1986. **31**(4): p. 291-298.
41. Dillenbourg, P., *Collaborative learning: Cognitive and computational approaches. advances in learning and instruction series*. 1999: ERIC.

42. Friedrich, H., et al., *Learning and communication in multi-agent systems.* Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments, 1997: p. 259-275.
43. Bui, H.H., D. Kieronska, and S. Venkatesh. *Learning other agents' preferences in multiagent negotiation*. in *Proceedings of the national conference on artificial intelligence*. 1996.
44. Narendra, K., N. Oleng, and S. Mukhopadhyay, *Decentralised adaptive control with partial communication.* IEE Proceedings-Control Theory and Applications, 2006. **153**(5): p. 546-555.
45. Arslan, G. and S. Yüksel, *Decentralized Q-Learning for Weakly Acyclic Stochastic Dynamic Games.*
46. Arslan, G. and S. Yuksel, *Decentralized Q-Learning for Stochastic Teams and Games.* IEEE Transactions on Automatic Control, 2016.
47. Teacy, W.L., et al. *Decentralized Bayesian reinforcement learning for online agent collaboration*. in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. 2012. International Foundation for Autonomous Agents and Multiagent Systems.
48. Shah, P. and P.A. Parrilo, *H2-Optimal Decentralized Control Over Posets: A State-Space Solution for State-Feedback.* IEEE Transactions on Automatic Control, 2013. **58**(12): p. 3084-3096.
49. Hua, C. and S.X. Ding, *Decentralized networked control system design using T–S fuzzy approach.* IEEE Transactions on fuzzy systems, 2012. **20**(1): p. 9-21.
50. Ranjbar-Sahraei, B., et al., *A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems.* IEEE Transactions on Industrial Electronics, 2012. **59**(8): p. 3124-3134.
51. Mahajan, A., *Optimal decentralized control of coupled subsystems with control sharing.* IEEE Transactions on Automatic Control, 2013. **58**(9): p. 2377-2382.
52. Han, Z. and K.S. Narendra, *New concepts in adaptive control using multiple models.* Automatic Control, IEEE Transactions on, 2012. **57**(1): p. 78-89.
53. Narendra, K.S. and J. Balakrishnan, *Improving transient response of adaptive control systems using multiple models and switching.* Automatic Control, IEEE Transactions on, 1994. **39**(9): p. 1861-1866.
54. Narendra, K.S. and S. Mukhopadhyay, *To communicate or not to communicate: A decision-theoretic approach to decentralized adaptive control*, in *American Control Conference (ACC), 2010*. 2010, IEEE.
55. Battistelli, G., et al., *Model-free adaptive switching control of time-varying plants.* IEEE Transactions on Automatic Control, 2013. **58**(5): p. 1208-1220.
56. Liu, W., et al., *Decentralized multi-agent system-based cooperative frequency control for autonomous microgrids with communication constraints.* IEEE Transactions on Sustainable Energy, 2014. **5**(2): p. 446-456.
57. Bian, T., Y. Jiang, and Z.-P. Jiang, *Decentralized adaptive optimal control of large-scale systems with application to power systems.* IEEE Transactions on Industrial Electronics, 2015. **62**(4): p. 2439-2447.
58. Gao, J., L. Dou, and P. Su. *Multi-model switching control of hypersonic vehicle with variable scramjet inlet based on adaptive neural network*. in *Intelligent Control and Automation (WCICA), 2016 12th World Congress on*. 2016. IEEE.

59. Rota, G.-C., *The number of partitions of a set.* The American Mathematical Monthly, 1964. **71**(5): p. 498-504.
60. Bellman, R., *Dynamic programming. princeton, nj: Princeton universitypress.* BellmanDynamic Programming, 1957.
61. Lancaster, P. and L. Rodman, *Algebraic riccati equations*. 1995: Clarendon press.
62. Ben-Israel, A., *Dynamic Programming & Optimal Control.*
63. Liberzon, D., *Calculus of variations and optimal control theory: a concise introduction*. 2012: Princeton University Press.
64. Lewis, F.L. and V.L. Syrmos, *Optimal control*. 1995: John Wiley & Sons.
65. Russell;, S. and P. Norvig, *Artificial Intelligence A Modern Approach*. 3rd ed. 2010, New Jersey: Prentice Hall.
66. Munos, R. and A.W. Moore, *Variable resolution discretization for high-accuracy solutions of optimal control problems.* Robotics Institute, 1999: p. 256.
67. Munos, R. and A. Moore, *Variable resolution discretization in optimal control.* Machine learning, 2002. **49**(2-3): p. 291-323.
68. Kharroubi, I., N. Langrené, and H. Pham, *A numerical algorithm for fully nonlinear HJB equations: an approach by control randomization.* Monte Carlo Methods and Applications, 2014. **20**(2): p. 145-165.
69. Puterman, M.L., *Markov decision processes: discrete stochastic dynamic programming*. 2014: John Wiley & Sons.
70. Lovejoy, W.S., *A survey of algorithmic methods for partially observed Markov decision processes.* Annals of Operations Research, 1991. **28**(1): p. 47-65.
71. Chang, H.S., *Decentralized learning in finite Markov chains: revisited.* IEEE Transactions on Automatic Control, 2009. **54**(7): p. 1648-1653.
72. Vrancx, P., K. Verbeeck, and A. Nowé, *Decentralized learning in markov games.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2008. **38**(4): p. 976-981.
73. Ng, A.Y., *Shaping and policy search in reinforcement learning*. 2003, University of California, Berkeley.
74. Bishop, C.M., *Pattern Recognition.* Machine Learning, 2006: p. 537-541.
75. Bellman, R., *On the theory of dynamic programming.* Proceedings of the National Academy of Sciences, 1952. **38**(8): p. 716-719.
76. Funahashi, K.-I., *On the approximate realization of continuous mappings by neural networks.* Neural networks, 1989. **2**(3): p. 183-192.
77. Miller, W.T., P.J. Werbos, and R.S. Sutton, *Neural networks for control*. 1995: MIT press.
78. Nguyen, T. and S. Mukhopadhyay, *Two-phase selective decentralization to improve reinforcement learning systems with MDP.* AI Communications, (31): p. 319-337.
79. Liu, K. and Q. Zhao, *Distributed learning in multi-armed bandit with multiple players.* IEEE Transactions on Signal Processing, 2010. **58**(11): p. 5667-5681.
80. Matignon, L., G.J. Laurent, and N. Le Fort-Piat. *Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams*. in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. 2007. IEEE.
81. Lauer, M. and M. Riedmiller. *An algorithm for distributed reinforcement learning in cooperative multi-agent systems*. in *In Proceedings of the Seventeenth International Conference on Machine Learning*. 2000. Citeseer.

82. Galindo-Serrano, A. and L. Giupponi, *Distributed Q-learning for aggregated interference control in cognitive radio networks.* IEEE Transactions on Vehicular Technology, 2010. **59**(4): p. 1823-1834.
83. Morozs, N., et al. *Distributed Q-learning based dynamic spectrum management in cognitive cellular systems: Choosing the right learning rate*. in *Computers and Communication (ISCC), 2014 IEEE Symposium on*. 2014. IEEE.
84. Narayanan, V. and S. Jagannathan, *Distributed adaptive optimal regulation of uncertain large-scale interconnected systems using hybrid Q-learning approach.* IET Control Theory & Applications, 2016. **10**(12): p. 1448-1457.
85. Nguyen, T. and S. Mukhopadhyay, *Selectively Decentralized Q-Learning*, in *IEEE International Conference on Systems, Man, and Cybernetics*. 2017: Bannf, Canada.
86. team, I.B. *I2B2 Overview*. 2016; Available from: https://www.i2b2.org/about/index.html.
87. Melo, F.S., *Convergence of Q-learning: A simple proof.* Institute Of Systems and Robotics, Tech. Rep, 2001: p. 1-4.
88. Sastry, P., V. Phansalkar, and M. Thathachar, *Decentralized learning of Nash equilibria in multi-person stochastic games with incomplete information.* IEEE Transactions on systems, man, and cybernetics, 1994. **24**(5): p. 769-777.
89. Karakaşoğlu, A., S.I. Sudharsanan, and M.K. Sundareshan, *Identification and decentralized adaptive control using dynamical neural networks with application to robotic manipulators.* Neural Networks, IEEE Transactions on, 1993. **4**(6): p. 919-930.
90. Kim, J. and J.P. Lynch, *Autonomous decentralized system identification by Markov parameter estimation using distributed smart wireless sensor networks.* Journal of Engineering Mechanics, 2011. **138**(5): p. 478-490.
91. Ucinski, D., *Optimal measurement methods for distributed parameter system identification*. 2004: CRC Press.
92. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, *Learning representations by back-propagating errors.* Cognitive modeling, 1988. **5**(3): p. 1.
93. Nguyen, T. and S. Mukhopadhyay, *Identification and Optimal Control of Large-scale System Using Selective Decentralization*, in *IEEE International Conference on Systems. Man and Cybernetics*. 2016: Budapest.
94. Martins, J.R. and A.B. Lambe, *Multidisciplinary design optimization: a survey of architectures.* AIAA journal, 2013. **51**(9): p. 2049-2075.
95. Alexandrov, N.M. and M.Y. Hussaini, *Multidisciplinary design optimization: State of the art*. Vol. 80. 1997: SIAM.
96. Deb, K., *Current trends in evolutionary multi-objective optimization.* International Journal for Simulation and Multidisciplinary Design Optimization, 2007. **1**(1): p. 1-8.
97. Alexandrov, N.M. and R.M. Lewis, *Analytical and computational aspects of collaborative optimization for multidisciplinary design.* AIAA journal, 2002. **40**(2): p. 301-309.
98. Balling, R.J. and J. Sobieszczanski-Sobieski, *Optimization of coupled systems-a critical overview of approaches.* AIAA journal, 1996. **34**(1): p. 6-17.
99. INC, M. *Train: Train Neural Network*. 2017 [cited 2017.
100. Young, H.D., R.A. Freedman, and A.L. Ford, *Elastic Potential Energy*, in *University Physics*. 2008, Pearson Education Inc.: San Francisco. p. 222-230.
101. Kleppner, D. and R. Kolenkow, *An introduction to mechanics*. 2013: Cambridge University Press.

102. Hooke, R., *De Potentia Restitutiva, or of Spring Explaining the Power of Springing Bodies, vol. 1678.* London, UK: John Martyn: p. 23.
103. Oprea, T.I., et al., *Drug Repurposing from an Academic Perspective.* Drug Discov Today Ther Strateg, 2011. **8**(3-4): p. 61-69.
104. Dudley, J.T., T. Deshpande, and A.J. Butte, *Exploiting drug-disease relationships for computational drug repositioning.* Brief Bioinform, 2011. **12**(4): p. 303-11.
105. Gupta, S.C., et al., *Cancer drug discovery by repurposing: teaching new tricks to old dogs.* Trends Pharmacol Sci, 2013. **34**(9): p. 508-17.
106. Power, A., A.C. Berger, and G.S. Ginsburg, *Genomics-enabled drug repositioning and repurposing: insights from an IOM Roundtable activity.* JAMA, 2014. **311**(20): p. 2063-4.
107. Bisson, W.H., *Drug repurposing in chemical genomics: can we learn from the past to improve the future?* Curr Top Med Chem, 2012. **12**(17): p. 1883-8.
108. Xu, H., et al., *Validating drug repurposing signals using electronic health records: a case study of metformin associated with reduced cancer mortality.* Journal of the American Medical Informatics Association, 2014: p. amiajnl-2014-002649.
109. Law, V., et al., *DrugBank 4.0: shedding new light on drug metabolism.* Nucleic Acids Res, 2013.
110. Andersson, M.L., et al., *Evaluation of usage patterns and user perception of the drug-drug interaction database SFINX.* Int J Med Inform, 2015. **84**(5): p. 327-33.
111. Kuhn, M., et al., *STITCH 3: zooming in on protein-chemical interactions.* Nucleic Acids Res, 2012. **40**(Database issue): p. D876-80.
112. Chen, J., R. Pandey, and T.M. Nguyen, *HAPPI-2: a Comprehensive and High-quality Map of Human Annotated and Predicted Protein Interactions.* BMC Genomics, 2017.
113. Szklarczyk, D., et al., *STRING v10: protein-protein interaction networks, integrated over the tree of life.* Nucleic Acids Res, 2015. **43**(Database issue): p. D447-52.
114. Baxevanis, A.D., *Searching Online Mendelian Inheritance in Man (OMIM) for information on genetic loci involved in human disease.* Curr Protoc Hum Genet, 2012. **Chapter 9**: p. Unit 9 13 1-10.
115. Barrett, T., et al., *NCBI GEO: archive for functional genomics data sets--update.* Nucleic Acids Res, 2013. **41**(Database issue): p. D991-5.
116. Willems, J., *Least squares stationary optimal control and the algebraic Riccati equation.* IEEE Transactions on Automatic Control, 1971. **16**(6): p. 621-634.
117. Chen, M.Z., et al., *Stabilizing solution and parameter dependence of modified algebraic Riccati equation with application to discrete-time network synchronization.* IEEE Transactions on Automatic Control, 2016. **61**(1): p. 228-233.
118. Bardi, M. and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. 2008: Springer Science & Business Media.
119. Falcone, M. and R. Ferretti, *Semi-Lagrangian Approximation Schemes for Linear and Hamilton—Jacobi Equations*. 2013: SIAM.
120. Rovithakis, G.A. and M.A. Christodoulou, *Adaptive control of unknown plants using dynamical neural networks.* Systems, Man and Cybernetics, IEEE Transactions on, 1994. **24**(3): p. 400-412.
121. Tong, S., et al., *Adaptive neural network output feedback control for stochastic nonlinear systems with unknown dead-zone and unmodeled dynamics.* IEEE transactions on cybernetics, 2014. **44**(6): p. 910-921.

122. Nguyen, T.M., et al., *DeCoSTT: A New Approach in Drug Repurposing from Control System Theory.* Frontiers in Pharmacology, 2018. **9**: p. 583.
123. Alberghina, L., *Systems biology: definitions and perspectives*. Vol. 13. 2007: Springer Science & Business Media.
124. Chui, C.K. and G. Chen, *Linear Systems and optimal control*. Vol. 18. 2012: Springer Science & Business Media.
125. Bemporad, A., et al., *The explicit linear quadratic regulator for constrained systems.* Automatica, 2002. **38**(1): p. 3-20.
126. Huang, H., et al., *Predicting Drug Efficacy Based on the Integrated Breast Cancer Pathway Model*, in *2011 IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*. 2011: San Antonio, TX p. 42-45.
127. Zhu, F., et al., *Update of TTD: Therapeutic Target Database.* Nucleic Acids Res, 2010. **38**(Database issue): p. D787-91.
128. Gunther, S., et al., *SuperTarget and Matador: resources for exploring drug-target relationships.* Nucleic Acids Res, 2008. **36**(Database issue): p. D919-22.
129. Campillos, M., et al., *Drug target identification using side-effect similarity.* Science, 2008. **321**(5886): p. 263-6.
130. Newman, M.E., *Fast algorithm for detecting community structure in networks.* Physical review E, 2004. **69**(6): p. 066133.
131. Ibrahim, S., et al., *Integrated Breast Cancer Pathway (Homo sapiens)*. 2015, Wikipathway.org: Wikipathway.org.
132. Hanley, J.A. and B.J. McNeil, *The meaning and use of the area under a receiver operating characteristic (ROC) curve.* Radiology, 1982. **143**(1): p. 29-36.
133. Uray, I.P. and P.H. Brown, *Chemoprevention of hormone receptor-negative breast cancer: new approaches needed.* Recent Results Cancer Res, 2011. **188**: p. 147-62.
134. Fabian, C.J., *The what, why and how of aromatase inhibitors: hormonal agents for treatment and prevention of breast cancer.* Int J Clin Pract, 2007. **61**(12): p. 2051-63.
135. Bulut, N. and K. Altundag, *Does estrogen receptor determination affect prognosis in early stage breast cancers?* Int J Clin Exp Med, 2015. **8**(11): p. 21454-9.
136. Hefti, M.M., et al., *Estrogen receptor negative/progesterone receptor positive breast cancer is not a reproducible subtype.* Breast Cancer Res, 2013. **15**(4): p. R68.
137. Herbst, R.S., *Review of epidermal growth factor receptor biology.* Int J Radiat Oncol Biol Phys, 2004. **59**(2 Suppl): p. 21-6.
138. Momparler, R.L., et al., *Effect of adriamycin on DNA, RNA, and protein synthesis in cell-free systems and intact cells.* Cancer Res, 1976. **36**(8): p. 2891-5.
139. Bruera, E., et al., *Donepezil for cancer fatigue: a double-blind, randomized, placebo-controlled trial.* J Clin Oncol, 2007. **25**(23): p. 3475-81.

VITA

Thanh Nguyen received his B.S. in Computer Science from Indiana University Purdue University Indianapolis (IUPUI) in 2012. He started his PhD in Computer Science at IUPUI in 2013 and completed the PhD degree in 2018. After graduation, he joined the Informatics Institute at the University of Alabama at Birmingham (UAB) as a research fellow.

Education

- 2011: Minor in Mathematics, IUPUI.
- 2012: B.S. in Computer Science, IUPUI.
- 2018: Ph.D in Computer Science, IUPUI

Honors, Awards, Fellowships

- 2007: Vietnam 4th rank honor for excellent student in History (top 150 of the nation).
- 2008: Hanoi 4th rank honor for excellent student in Chemistry (top 100 of the province).
- 2009: Vietnamese governmental scholarship for undergraduate studying oversea.
- 2010-2012: IUPUI School of Science Scholar's List Recipients.
- 2011-2012: IUPUI Multidisciplinary Undergraduate Research Institute Recipients.
- 2013: IUPUI Dept. of Computer Science Gersting Award.

Research and Training Experience

- 2011-2012: IUPUI Multidisciplinary Undergraduate Research Institute: Image processing and Bioinformatics.
- 2011-2012: Undergraduate research (independent study): Intelligent Systems.

- 2012-2015: Volunteer researcher at Indiana Center for System Biology and Personalized Medicine.
- 2013-2016: Research Assistant in Computer Science, IUPUI Department of Computer and Information Science.
- 2014-2017: Visiting Scholar in Bio-Health Informatics, Wenzhou Medical University, Zhejiang, China (summer internship).
- 2017: Teaching Assistant, IUPUI Department of Computer and Information Science.
- 2017-2018: Visiting Scholar, the Informatics Institute at UAB.

Professional Experience

- 2010-2012: Math Tutor, IUPUI Mathematics Assistant Center.
- 2015-2016: Consultant, MedeoLinx LLC, Nanjing, China (summer).

Skills

- Computer science
  + primary areas: artificial intelligence, machine learning, data mining, algorithm development.
  + secondary areas: database management, distributed computing, bioinformatics.
- Programming efficiency: C++, Java, Python, Matlab.
- Math: linear algebra, optimization, system modeling and control.
- Fast learning speed.
- Leadership in both academic and industry environment.
- Mentoring.
- Suitable for multi-disciplinary and translational research.

PUBLICATIONS

The following papers are the direct results from this thesis

1. Nguyen, T. and S. Mukhopadhyay, *Identification and Optimal Control of Large-scale System Using Selective Decentralization*, in *IEEE International Conference on Systems. Man and Cybernetics*. 2016: Budapest.
2. Nguyen, T. and S. Mukhopadhyay. *Selective decentralization to improve reinforcement learning in unknown linear noisy systems*. in *Intelligent and Evolutionary Systems (IES), 2017 21st Asia Pacific Symposium on*. 2017. IEEE.
3. Nguyen, T. and S. Mukhopadhyay, *Selectively Decentralized Q-Learning*, in *IEEE International Conference on Systems, Man, and Cybernetics*. 2017: Bannf, Canada.
4. Nguyen, T. and S. Mukhopadhyay, *Multidisciplinary Optimization in Decentralized Reinforcement Learning*, in *16th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 2017: Cancun, Mexico.
5. Nguyen, T. and S. Mukhopadhyay, *Two-phase selective decentralization to improve reinforcement learning systems with MDP.* AI Communications, (31): p. 319-337..
6. Nguyen, T., Muhammad , S.A., Ibrahim , S., Ma L., Guo , J. and Bai, B., *DeCoST: A New Approach in Drug Repurposing from Control System Theory*, Frontiers in Pharmacology, 2018. **9**: p. 583.

The other works published during the graduate program

7. Chen, J., R. Pandey, and T.M. Nguyen, *HAPPI-2: a Comprehensive and High-quality Map of Human Annotated and Predicted Protein Interactions.* BMC Genomics, 2017.
8. Huang, H., et al., *DMAP: a connectivity map database to enable identification of novel drug repositioning candidates.* BMC Bioinformatics, 2015. **16 Suppl 13**: p. S4.
9. Yue, Z., et al., *PAGER: constructing PAGs and new PAG-PAG relationships for network biology.* Bioinformatics, 2015. **31**(12): p. i250-7.
10. Fang, X., et al. *Use Cases for Public Health Data Visualization*. in *2013 Workshop on Visual Analytics in Healthcare*. 2013. Washington, DC.
11. Xia, Y., et al. *Data Exploration of a Notifiable Condition Detector System*. in *2013 Workshop on Visual Analytics in Healthcare*. 2013. Washington, DC.
12. Palakai, M., et al. *Detecting Comorbidity of Chlamydia from Clinical Reports*. in *2013 Workshop on Visual Analytics in Healthcare*. 2013. Washington, DC.
13. Nguyen, T., et al., *An integrated machine learning and network analysis to discover marker clinical measurement in lung cancer*, in *Genomic Medicine*. 2015: Ho Chi Minh City.
14. Cao, N., et al., *Predictive and Preventive Models for Diabetes Prevention using Clinical Information in Electronic Health Record*, in *IEEE International Conference on Bioinformatics and Biomedicine*. 2015: Washington DC.

15. Muhammad, S.A., et al., *Cellular Signaling Pathways in Insulin Resistance-Systems Biology Analyses of Microarray Dataset Reveals New Drug Target Gene Signatures of Type 2 Diabetes Mellitus.* Frontiers in physiology, 2017. **8**: p. 13.
16. Chen, J.Y., et al. *Towards constructing "Super Gene Sets" regulatory networks*. in *Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on*. 2016. IEEE.
17. Muhammad, S.A., et al., *Simulation Study of cDNA Dataset to Investigate Possible Association of Differentially Expressed Genes of Human THP1-Monocytic Cells in Cancer Progression Affected by Bacterial Shiga Toxins* Frontiers in Microbiology, 2018.
18. Yue, Z., et al., *"Super Gene Set" Causal Relationship Discovery from Functional Genomics Data.* IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2018.