2015

Goal Representation Adaptive Dynamic Programming for Machine Intelligence

Zhen Ni

University of Rhode Island, nizhen10@gmail.com

Follow this and additional works at: https://digitalcommons.uri.edu/oa_diss

Recommended Citation

Ni, Zhen, "Goal Representation Adaptive Dynamic Programming for Machine Intelligence" (2015). *Open Access Dissertations*. Paper 351.
https://digitalcommons.uri.edu/oa_diss/351

This Dissertation is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Dissertations by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons-group@uri.edu.

GOAL REPRESENTATION ADAPTIVE DYNAMIC PROGRAMMING FOR
MACHINE INTELLIGENCE

BY
ZHEN NI

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL
ENGINEERING

UNIVERSITY OF RHODE ISLAND

2015

DOCTOR OF PHILOSOPHY DISSERTATION

OF

ZHEN NI

APPROVED:

Dissertation Committee:

Major Professor Haibo He

Yan Sun

Lisa DiPippo

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2015

ABSTRACT

This dissertation is focused on a general purpose new framework for machine intelligence based on adaptive dynamic programming (ADP) design. This research is significantly important for developing self-adaptive intelligent system that are highly robust and fault-tolerant to uncertain and unstructured environments. Generally, there are two key components toward building truly self-adaptive systems: fundamental understanding of brain intelligence and complex engineering designs. This dissertation will focus on general purpose computational intelligence methodologies from a biological inspired perspective, and develop a new self-learning machine intelligent system online over time. Furthermore, this new approach will also be explored on wide critical engineering applications.

Specifically, a new framework, named “goal representation adaptive dynamic programming (GrADP)”, is proposed and introduced in this dissertation. It is regarded as the foundation of building intelligent systems through internal reward learning, goal representation and state-action association. Unlike the traditional ADP design with an action network and a critic network, this new approach integrates an additional network, called the reference (or goal) network, such that to build a general internal reinforcement signal. Unlike the traditional fixed or pre-defined reinforcement learning signal, this new design can adaptively update the internal reinforcement representation over time and thus facilitate the system’s learning and optimization to accomplish the ultimate goals.

The original contribution of this research is to integrate an adaptive goal representation design into ADP framework rather than engineering hand-crafted reward functions in literature. This is the first time that the reward signal is presented in a general mapping function by the observation of system variables over time. This is also an important step towards a general purpose self-adaptive

learning system based on ADP designs. Generally, ADP family has three major categories: heuristic dynamic programming (HDP), dual heuristic dynamic programming (DHP), and globalized dual heuristic dynamic programming (GDHP). In this research, goal representation principle has been integrated into each design, and verified with promising optimization and learning results. To this end, goal representation heuristic dynamic programming (GrHDP), goal representation dual heuristic dynamic programming (GrDHP), and goal representation globalized dual heuristic dynamic programming (Gr-GDHP), are successfully proposed and developed as a new GrADP family. Further studies of GrADP approaches from toy problems to real-world applications have been provided in comparison with several other classical control and reinforcement learning approaches. The rigorous mathematical analysis and stability assurance have also been provided to address the convergence and boundedness issues, which are the theoretical assurance for this new integrated design. In summary, this is the first time that the new GrADP design framework has been proposed and described explicitly with its family members. The numerical simulation verification, engineering applications and also theoretical results are provided to study each of the new architecture design from different viewpoints.

ACKNOWLEDGMENTS

First and foremost, I would like to express my earnest gratitude to my advisor, Prof. Haibo He, for his continuous and invaluable, guidance, help and support through my Ph.D. study at University of Rhode Island (URI). I still remember his words five years ago: “to be an independent researcher”, which has been the motto in my mind all the way to today. It has been my great honor to be one of his Ph.D. students and work with him on quite many exciting research topics. He not only delivered cutting-edge knowledge in the field to me, but also educated me to be a creative and committed person. His vision towards frontier research directions inspired me to be a self-motivated researcher and keep moving forward in my academic career. His mentorship was paramount in shaping various aspects of my professional career, and his personality was also a remarkable role model in my life. Without those, I would not be so proud of myself as of today.

I also want to sincerely thank for Prof. Yan Sun, Prof. Richard Vaccaro, Prof. Lisa DiPippo, and Prof. P. V. August for serving as my dissertation defense committee. Precious feedback and comments in this process provided me a different view of my research work and myself, which helped this dissertation to be in better shape. Also thanks for Prof. Jinyu Wen and Dr. Danil Prokhorov for close collaborations and suggestions all the way along this research direction.

Special thanks are conveyed to my teammates in Computational Intelligence and Self-Adaptive System (CISA) group in URI throughout my Ph.D. period: Xiangan, Yufei, Jing, Jun, Bo, Chris, Chaoxu, Siyao, Lu, Xiao, and many other visiting scholars. Discussions with them provided me lots of inspirations and motivations. Also thanks for my close friends Xiaorong, Quan, Fu, Yuhong, Yihai and Daxian, who also made my life in URI colorful and memorable.

My final heartfelt acknowledgement must go to my beloved parents and sister

for their encouragement. Without them, this work would never come true. Many thanks for their continuous support and love. I would also like to share my every single step forward and achievement all together with them.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	iv
TABLE OF CONTENTS	vi
LIST OF TABLES	xi
LIST OF FIGURES	xiii
CHAPTER	
1 Introduction	1
1.1 Motivations and Inspirations	1
1.2 Significance of Self-Adaptive System Designs	2
1.3 Research Objectives	3
1.4 Dissertation Organization	4
2 Background and Literature Discussion	6
2.1 Introduction	6
2.2 Markov Decision Process (MDP) and Reinforcement learning (RL)	6
2.3 Adaptive Dynamic Programming (ADP) Family	8
2.3.1 Model-Based and Model-Free Designs	8
2.3.2 Heuristic Dynamic Programming (HDP)	9
2.3.3 Dual Heuristic Dynamic Programming (DHP)	10
2.3.4 Globalized Dual Heuristic Dynamic Programming (GDHP)	11
2.4 Summary	12

	Page
3 A New Internal Goal Representation Framework	13
3.1 Introduction	13
3.2 A Three-Network ADP Architecture	13
3.3 Learning and Optimization Algorithms	15
3.3.1 Learning in Goal Network	17
3.3.2 Learning in Critic Network	19
3.4 Simulation Studies	20
3.4.1 Cart-Pole Balancing Example	21
3.4.2 Triple-Link Inverted Pendulum Balancing Example . . .	23
3.5 Summary	33
4 Goal Representation Design for Heuristic Dynamic Program-	
ming (GrHDP) Architecture	34
4.1 Introduction	34
4.2 Design of GrHDP for Tracking Control	34
4.2.1 Design of Dual-Critic Network	35
4.2.2 Design of Tracking Filter	37
4.2.3 Design of Reinforcement Signals	39
4.3 Association and Implementation	39
4.3.1 Learning and Optimization in Dual Critic Network . . .	40
4.3.2 Interaction in Action Network and Tracking Filter	46
4.4 Simulation Studies	48
4.4.1 Tracking Control Problem with Disturbance	49
4.4.2 Adaptive Signal Tracking Control Problem	52
4.4.3 Real-Time Tracking Control for Ball and Beam System .	53

	Page
4.4.4 Virtual Reality Demonstration with Unknown Disturbances	58
4.5 Summary	59
5 Goal Representation Design for Dual Heuristic Dynamic Programming (GrDHP) Architecture	61
5.1 Introduction	61
5.2 Design of the GrDHP Framework	61
5.3 Learning Process of the GrDHP Design	65
5.3.1 Design of Goal Representation in HDP	66
5.3.2 Design of General Utility Representation in DHP	67
5.3.3 Online Learning in Critic Network	68
5.3.4 Online Learning in Action Network	70
5.4 Simulation Studies	70
5.4.1 Ball and Beam Balancing Example	71
5.4.2 Triple-link Inverted Pendulum Balancing Example	74
5.5 Discussion of Advanced GrADP Designs	75
5.6 Summary	81
6 Hierarchical Goal Representation Heuristic Dynamic Programming Design	82
6.1 Introduction	82
6.2 Design of Hierarchical HDP Structure	82
6.2.1 Learning and Adaptation in Hierarchical Goal Networks	84
6.2.2 Learning and Adaptation in the Critic Network	86
6.2.3 Learning and Adaptation in the Action Network	87
6.3 Simulation Studies	87

	Page
6.3.1 Experiment Configuration and Parameters	87
6.3.2 Simulation Results and Analysis	89
6.4 Summary	93
7 Applications: From Toy Problems to Real-World Applications	97
7.1 Introduction	97
7.2 Maze Navigation Application	97
7.2.1 Maze Navigation Learning Algorithm Description	97
7.2.2 Maze Navigation Environment Setup	99
7.2.3 Simulation Studies and Analysis	101
7.3 Smart Grid Application	105
7.3.1 Multimachine Power System Environment Setup	109
7.3.2 Simulation Studies and Analysis	110
7.4 Summary	115
8 Conclusions and Future Research Directions	121
8.1 Conclusions	121
8.2 Original Contributions	121
8.3 Future Research Directions	123
LIST OF REFERENCES	125
APPENDIX	
A Pseudo Code of GrHDP based Tracking Control Algorithm .	139
B Stability Analysis for GrHDP-based Tracking Control	143
C Boundedness Theoretical Results of Maze Navigation Example	152

	Page
C.1 Problem Formulation	152
C.2 Analysis of GrADP based Learning Approach	155
D Abbreviations	161
BIBLIOGRAPHY	162

LIST OF TABLES

Table		Page
3.1	Summary of the parameters used in cart-pole balancing task . .	21
3.2	Performance evaluation on case I: cart-pole balancing task. The 2nd and the 3rd columns are with our proposed method, while the 4th and the 5th columns are the results from existing approach	22
3.3	System constants for the triple link inverted pendulum problem	25
3.4	Summary of the parameters used in triple-link inverted pendulum balancing task	28
3.5	Performance evaluation on case II: triple-link balancing task. The 2nd and the 3rd columns are with our proposed method, while the 4th and the 5th columns are the results from existing approach	29
4.1	Summary of the parameters used in the simulation study one .	49
4.2	Summary of the parameters used in the simulation study two .	56
4.3	Summary of total tracking error for the simulation cases	57
5.1	Comparison of the statistical simulation results on the triple-link inverted pendulum balancing task with the DHP and the GrDHP approaches.	77
6.1	Summary of the parameters used in the simulation	88
6.2	Simulation results on ball-and-beam balancing task. The 1st column is with the noise type. The 2nd column is with <i>Algorithm0</i> , while the 3rd column is with <i>Algorithm1</i> and the 4th column is with <i>Algorithm2</i> . The number of trials and standard deviation are calculated based on the successful runs . . .	91
7.1	Summary of the parameters used in the GrHDP/HDP controller. The notations are kept the same as those in Chapter 3	110

Table		Page
7.2	Summary of the parameters used in the GrHDP/HDP controller. The notations are kept the same as those in Chapter 3	113
7.3	The admittance matrices between each line/bus in the three-machine nine-bus system.	118
7.4	The per unit value for the parameters of each generator/bus in the system.	119
7.5	The sequence of load fluctuation applied in load A	119

LIST OF FIGURES

Figure		Page
2.1	The schematic diagram of typical model-free HDP structure. . .	10
2.2	The schematic diagram of typical DHP structure.	11
2.3	The schematic diagram of typical GDHP structure.	12
3.1	The proposed ADP architecture with internal goal representation.	14
3.2	Parameters adaptation and tuning based on backpropagation .	16
3.3	Design of the three networks with nonlinear neural network: (a) Reference network design; (b) Critic network design	17
3.4	Definition of notation used in the system equations for the triple link inverted pendulum benchmark	23
3.5	Statistics of the successful runs. (a) Number of required trials for the successful runs. (b) Histogram the statistics.	29
3.6	Typical trajectory on the triple-link inverted pendulum balancing task. (a) The position x of the cart. (b) The 1st joint angle of the triple link pendulum. (c) The 2st joint angle of the triple link pendulum. (d) The 3st joint angle of the triple link pen- dulum. (e) The velocity of the cart. (b) The angular velocity of the 1st joint angle of the triple link pendulum. (c) The an- gular velocity of the 2st joint angle of the triple link pendulum. (d) The angular velocity of the 3st joint angle of the triple link pendulum.	30
3.7	Typical histogram on the triple-link inverted pendulum balancing task. (a) The histogram of position x of the cart. (b) The histogram of the 1st joint angle of the triple link pendulum. (c) The histogram of the 2st joint angle of the triple link pendu- lum. (d) The histogram of the 3st joint angle of the triple link pendulum.	31
3.8	Typical trajectory of cost-to-go and control action signal on the triple-link inverted pendulum balancing task. (a) The cost-to-go signal on the task. (b) The control action signal on the task. . .	32

Figure		Page
4.1	Architecture design of the proposed HDP approach with a tracking filter.	35
4.2	Description of the dual critic network.	36
4.3	Description of the tracking filter.	37
4.4	Learning schematic in dual critic network.	40
4.5	(a) The neural network structure of the reference network; (b) The neural network structure of the critic network.	42
4.6	(a) The neural network structure of the action network; (b) The control action generator.	48
4.7	The typical tracking performance with our proposed approach. .	50
4.8	The typical tracking performance with HDP approach	51
4.9	The typical tracking performance with our proposed approach. .	53
4.10	The typical tracking performance with HDP approach.	54
4.11	A schematic diagram of ball-and-beam system	55
4.12	The typical tracking performance with our proposed approach. .	56
4.13	The typical tracking performance with HDP approach	57
4.14	The schematics of virtual reality demonstration platform.	59
4.15	The typical tracking performance with the proposed approach in VR/Simulation platform	59
5.1	The conceptual diagram of the GrDHP architecture. Solid arrows refer to the signal paths, while the dash arrows refer to the backpropagation paths.	62
5.2	The learning flowchart in the GrDHP design. The neural networks are represented with their corresponding weights and these weights are carried on throughout the learning process. The subscript number under each parameter (e.g., r) refers to time step.	63

Figure		Page
5.3	The proposed architecture of the GrDHP design. Solid/Bolded arrows refer to the signals/vectors and dash arrows refer to the weights tuning paths. $\frac{\partial r(t)}{\partial x(t)}$ and $\frac{\partial r(t)}{\partial u(t)}$ refer to the partial derivatives of the external reward/utility w.r.t the system state $x(t)$ and the control action $u(t)$, respectively. The block DER implements (5.9).	65
5.4	Comparison of the statistical results on the ball-and-beam balancing task with both the GrDHP and the DHP approaches. . .	72
5.5	Typical trajectories of the state vectors in the first 2000 time steps in a typical successful trial.	73
5.6	The weights evolution of the goal network in the first 1000 steps in a typical successful trial.	74
5.7	Typical trajectories of the angles in a successful trial under the condition of noise free.	75
5.8	Typical trajectories of the λ (corresponding to the angles in Fig.5.7) in a successful trial under the condition of noise free. . .	76
5.9	The typical trajectory of the control force (Newton) applied on the cart in a successful trial under the condition of noise free. . .	78
5.10	The histogram of the control force (Newton) applied on the cart (corresponding to the trajectory in Fig.5.9).	79
6.1	The schematic of hierarchical HDP structure	83
6.2	Cascading weights tuning path inside the goal generator	84
6.3	The typical trajectory of x_1 and x_2 with Algorithm 2	89
6.4	The histogram of x_1 in a typical successful run with Algorithm 2	90
6.5	The typical trajectory of x_3 and x_4 with Algorithm 2	91
6.6	The typical trajectory of the control action and the total cost-to-go signal with Algorithm 2	92
6.7	The typical trajectory of internal goal signals with Algorithm 2	93
6.8	The histogram of the internal goals with Algorithm 2	94

Figure		Page
6.9	The boxplot of the required number of trials in 100 random runs with Algorithm 0, Algorithm 1 and Algorithm 2.	95
6.10	The typical trajectory of control action with Algorithm 2 under 5% uniform noise on the actuator	95
6.11	The typical trajectory of the state vectors x_1 and x_2 with Algorithm 2 under 5% uniform noise on the sensor of the position of the ball	96
7.1	Simulation setup: flowchart of the GrHDP approach on maze navigation problem. Value table is updated at the end of each trial. The learning process will be terminated when the trial number reaches the maximum trial number.	100
7.2	Learning curves with GrHDP, HDP, Sarsa(λ) and Q-learning approaches in 16 * 16 maze navigation. GrHDP approach shows the fastest learning speed and lowest final sum of square error than the other three approaches.	103
7.3	Surface plot of the value table learned by GrHDP approach on maze navigation problem (16*16). x and y axis refer to the coordinates of the agent while z axis refers to the J-value. . . .	104
7.4	Diagram of 3-D maze (5*5*5) navigation benchmark. The goal locates at the upper-right corner of the maze and the agent needs to try 6 directions before it can learn the policy.	105
7.5	Learning curves with GrHDP, HDP, Sarsa(λ) and Q-learning approaches in 5*5*5 maze navigation. GrHDP approach shows the fastest learning speed and lowest final sum of square error than the other three approaches.	106
7.6	The schematic diagram of the three-machine nine-bus power system. The dot lines show the observations from each generator. The ADP controller provides three supplementary control signals for three generators respectively based on these (delayed) measurements.	108

Figure		Page
7.7	The schematic diagram for a single generator. The ADP and PSS controller will be connected into the closed-loop if S1 switches to 1 or 2. The dot arrows show the observed signals from the generator and the grid, while the solid arrows show the signal paths.	109
7.8	The data flowchart of GrHDP controller with neural network implementation.	111
7.9	Damping performance on $\Delta\omega_{12}$ under three-phase-ground fault. Four approaches are compared on the same environment settings.	114
7.10	Damping performance on $\Delta\omega_{13}$ under three-phase-ground fault. Four approaches are compared on the same environment settings.	115
7.11	Damping performance on $\Delta\omega_{23}$ under three-phase-ground fault. Four approaches are compared on the same environment settings.	116
7.12	Comparison of the control performance on the output active power Pe_1 under both -5% and $+5\%$ step changes	117
7.13	Comparison of the control performance on the output active power Pe_2 under both -5% and $+5\%$ step changes.	117
7.14	Comparison of the control performance on the output active power Pe_3 under both -5% and $+5\%$ step changes.	118
7.15	The output active power Pe_1 under the load fluctuations. The GrHDP approach is compared against the other three approaches under the same sequential load disturbances.	119
7.16	The supplementary control signals u_1 , u_2 and u_3 provided by GrHDP approach in this sequential load disturbances scenario. .	120

CHAPTER 1

Introduction

1.1 Motivations and Inspirations

Brain intelligence and animal intelligence are very important biological inspiration to develop truly self-adaptive systems to such a level of intelligence in certain perspectives [1, 2, 3]. Although many important fundamental researches and critical engineering applications have been successfully developed, there is still a long way to achieve the general-purpose intelligent machine in an engineering way. One of the key challenges is how to design the intelligent systems to have the capacity to learn, predict and optimize over time, in order to achieve the ultimate goals [4, 5, 6, 7, 8]. In this dissertation, a new data-driven framework based on adaptive dynamic programming (ADP) will be proposed to help the intelligent system to learning to optimize online in the uncertain and unstructured environment. A further hierarchical data-driven framework will also be provided to mimic brain intelligence to help achieve the long-term goal through multi-stage internal short-term goals from a biological viewpoint.

In the traditional ADP and reinforcement learning (RL) design, the reinforcement signal feedback is usually set as a fixed formula, such as a binary value or the quadratic functions. These are defined according to designers' past experience and the prior knowledge of the systems under control. For instance, in two important editorial research books [9, 10], many researchers and professors are designing the reward feedback based on their knowledge of the system, including several complex engineering problems. Other research works [11, 12, 13, 14] connected this reward design with the control problem and thus define it as a quadratic function (or the integration in a continuous time format). Although promising results have been reported, there is still an opportunity to address this issue from a more general

way [15, 16, 17].

To this end, I am motivated to introduce a general-purpose reinforcement signal representation that is applicable for general systems without any prior knowledge and past experience. This general reinforcement signal feedback design will be able to help providing an adaptive internal goal guidance online over time, in order to best fit the online operation environment. For example, if the objectives change over time, the reinforcement signal feedback should also be adjusted to fit the new objective [18, 19, 20]. Or, if there are some disturbances or noises during the operation, the reinforcement signal feedback should also be adjusted to provide the best representation. More importantly, I am inspired by biological systems that have multi-level and multi-stage internal goals to accomplish, in order to achieve the long-term goal [21, 22, 23]. For instance, if an animal wants to survive in a hostile environment, it needs to consider the trade-off between getting the food and escaping from a hunter. Motivated by such observations, a multi-level reinforcement signal representation is proposed to build a value system that can facilitate the development of hierarchical internal goal representation. The learning and association in both top-down and bottom-up pathways to support this intelligent decision-making process is also discussed here. Based on all these literature search and biological inspiration, a new adaptive learning framework for machine intelligence based on adaptive dynamic programming is introduced in this dissertation.

1.2 Significance of Self-Adaptive System Designs

This is the first time that a self-adaptive reinforcement signal design has been proposed and demonstrated in the community of ADP and RL. The significance of the results provided in this dissertation is summarized as following:

- This research proposes a new self-adaptive learning architecture for machine

intelligence based on adaptive dynamic programming technique. The online learning for the internal reinforcement signal is conducted with environment uncertainties over time. This architecture is a very important step ahead for the general-purpose machine intelligent system designs.

- Multi-stage hierarchical reinforcement signal representation has been proposed to handle the multi-stage short-term goals in the biological systems, in order to accomplish long-term objectives. This is the first time that multi-stage reinforcement signals design have been proposed based on adaptive dynamic programming architecture.
- The reinforcement signal representation, also called goal representation, is fully applied to the existing designs in adaptive dynamic programming family. It is thus called goal representation adaptive dynamic programming (GrADP) design family. Such designs have been applied in several critical engineering applications to mimic certain level of human brain and animal behaviors.

1.3 Research Objectives

The objective of this dissertation is focused on designing general-purpose intelligent systems that are capable to learn to optimize the decision-making process in an unknown environment, based on the biological inspired multi-stage internal goal representations. It is very important to justify the objectives of this integrated framework via different types of ADP architectures onto the various critical engineering applications:

- The goal representation design could be able to learn proper internal reward through the interaction with the environment adaptively, rather than a fixed reward formula all the way over time.
- This designed system could be able to learn near-optimal control policy in

unknown environment under disturbances, noises and uncertainties. The control policy could be better fitted through the online adjustment.

- The designed hierarchical goal representation could be able to provide multi-stage internal reward signals for the intelligent systems, to mimic certain-level of intelligence in biological systems.
- The proposed new framework could be able to be implemented in scalable hardware embedded systems, including FPGA board and GPU board, bring such intelligence into real-world applications.

1.4 Dissertation Organization

This dissertation will be organized as following:

Chapter 2 provides the background of my research and literature review in current community. It further provides the introduction Markov decision process, reinforcement learning, and the ADP design family.

Chapter 3 focuses on a new internal goal representation design based on the traditional ADP architecture. An additional reference/goal network has been added into this structure, and the performance is verified through two balancing benchmarks.

Chapter 4 discusses a new tracking control scheme based on the dual-critic ADP architecture (i.e., critic and reference networks). The design and implementation of dual-critic network, action network and tracking filter are presented. The comparative performance is also provided via several tracking examples.

Chapter 5 presents the integration of a general utility function representation onto the DHP design. The explicit internal utility function is provided by the goal network and thus its derivatives can be adaptively adjusted. The learning capacity of the proposed DHP design is improved and demonstrated via the commonly used

balancing benchmarks.

Chapter 6 further provides the integration of multi-level goal representation design onto HDP architecture, which is also called hierarchical HDP design. This hierarchical design is demonstrated with the significant control improvement in comparison with classical HDP and goal representation HDP designs from certain perspectives.

Chapter 7 provides the applications of the proposed goal representation adaptive dynamic programming design family from toy problems (maze navigation), to real-world applications (multi-machine power system stability and control).

Chapter 8 concludes the dissertation and also discusses the future directions of this self-adaptive learning framework based on adaptive dynamic programming.

CHAPTER 2

Background and Literature Discussion

2.1 Introduction

Markov decision process (MDP) is a long-standing research topic in decision making models in stochastic process [24, 25, 26]. Value function (or state-action pair) is usually used to evaluate how good it is of the agent for a given state. If the state and the action spaces are finite, then it is called finite MDP, which is particularly important to the theory of reinforcement learning [27, 28, 16].

In this chapter, the background of both MDP and reinforcement learning will be introduced. The design architectures and differences of both model-based and model-free ADP will be discussed. I will also discuss the current literature review of the ADP designs among three major categories, i.e., heuristic dynamic programming, dual heuristic dynamic programming and globalized dual heuristic dynamic programming, as the background of this research.

2.2 Markov Decision Process (MDP) and Reinforcement learning (RL)

A Markov decision process is denoted as a tuple X, U, r, P , where X is the state space, U is the action space, P is the transition probability and r is the reward feedback [29, 2, 30]. For instance, given any state and action, x and u , the probability of each possible next state x' is defined as

$$P_{xx'}^u = \Pr\{x_{t+1} = x' | x_t = x, u_t = u\}. \quad (2.1)$$

While the expected value of the next reward is defined as

$$R_{xx'}^u = E\{r_{t+1} | x_t = x, u_t = u, x_{t+1} = x'\}. \quad (2.2)$$

Solving a reinforcement learning task means finding a policy that achieves the maximum reward feedback in the long-run. There is always at least one policy

that is better or equal than any other policies, and this is call optimal optimal policy [31, 32, 33]. In literature, π is defined as policy and π^* is denoted as the optimal policy (although there may be more than one). The value of a state x under a policy π is defined as

$$V^\pi(x) = E_\pi\{R_t|x_t = x\} = E_\pi\left\{\sum_{k=0}^{\infty}\{\{\gamma^k r_{k+1}|x_t = x\}\}\right\}. \quad (2.3)$$

where E_π is the expected value given that the agent follows policy π . The objective in this process is to find the optimal policy π^* so that to achieve the optimal $V^{\pi^*}(x)$ as

$$V^*(x) = \max_{\pi} V^\pi(x) = \max_u \sum_{x'} \left\{ P_{xx'}^u \{ R_{xx'}^u + \gamma V^{\pi^*}(x') \} \right\}. \quad (2.4)$$

In optimal control area [34, 35, 2], Bellman’s optimality principle suggests that an optimal policy can be built for the “tail subproblem” involving the last stage and extended backward until that the optimal strategy is built for the entire process. With the notations in (2.1) and (2.2), Bellman’s optimality equation for Q^* can be written as

$$Q^*(x, u) = \sum_{x'} P_{xx'}^u \left[R_{xx'}^u + \gamma \max_{u'} Q^*(x', u') \right]. \quad (2.5)$$

where $Q^*(x, u)$ refers to the value function of the current state x and $Q^*(x', u')$ refers to the value function of the possible next state x' . Equations (2.4) and (2.5) are actually two forms of Bellman optimality presentations.

In past decades, RL, especially Q-learning and temporal difference (TD) learning, has been employed to solve Bellman’s equation in MDP. For instance, in [36], a robust reinforcement learning approach, basically Q-learning approach, was proposed to help the agent to find the optimal control policy with minimum cost. In addition, “Dyna-Q” learning architecture was later introduced in [37]. This architecture can be integrated with trial-and-error (reinforcement) learning and execution-time planning, into a single process operation alternately on the world

and on a learned model of the world. Furthermore, $TD(\lambda)$ was also developed to improve the convergence speed on solving MDP problems in [38, 39, 24, 40, 41]. Meanwhile, ADP has demonstrated the capability to find the optimal control policy over time and solve the Bellman’s equation in a principle way. High-level understanding of ADP [42, 43, 44, 45, 46, 47], implied that ADP approaches could be able to learn and optimize the control policy over time, and find the solution for Bellman’s optimality equation efficiently. Various ADP architectures, including heuristic dynamic programming (HDP), dual heuristic dynamic programming (DHP), and globalized dual heuristic dynamic programming (GDHP) (together with their action-dependent (AD) versions), have been proposed in [48, 29] to seek the optimal policy over time. It has also been demonstrated that HDP has the similar learning and association principle with the Q-learning algorithm.

2.3 Adaptive Dynamic Programming (ADP) Family

2.3.1 Model-Based and Model-Free Designs

In this ADP design family, there are usually two major architectures: model-based ADP architecture and model-free ADP architecture. In the model-based architecture, a model-network is used to predict the future system variables and the corresponding future value function (or future derivatives of value function). Moreover, this model network is also used to connect action and critic networks during the back-propagation process. In the model-free architecture, the temporal difference error will be achieved by the current time step and the previous time step. In this case, there is no need to use the model network to predict for the future system variables (or the future value function). The back-propagation process is thus simplified without any usage of model network. Such model-free HDP has also been regarded as the equivalence to the classical Q-learning approach. More importantly, both versions have been used for various real-world applications with

successful results, and also been applied on the ADP design family. For instance, the online model-free HDP was developed in [49, 50, 51], where the authors took the advantages of the potential scalability of the adaptive critic designs and the intuitiveness of Q-learning. It is also an online learning scheme that simultaneously updates the value function and the control policy. The model-based HDP was also proposed with rigorous convergence proof to solve the optimal control problem for discrete-time nonlinear systems [52]. For model-based DHP/GDHP design, the authors in [12, 53] introduced that the efficient learning can be achieved with different weights for different error terms on the auto-lander helicopter problem. In [54, 55, 56], the authors also demonstrated the convergence analysis for model-based DHP/GDHP in terms of cost function and control law.

2.3.2 Heuristic Dynamic Programming (HDP)

Heuristic dynamic programming is the most basic architecture in the ADP family [57, 58]. There are usually two networks in model-free heuristic dynamic programming design, as presented in Fig. 2.1. An action network is used to provide the control action to the system, and a critic network is used to evaluate the control performance over time. For example, the action network will generate the control action u based on the observation of the system variables x . The critic network will evaluate the performance of this control policy based on the reinforcement signal feedback r from the environment. Meanwhile, the value function J will be approximated by the critic network. As presented in Fig. 2.1, the objective function of critic network will be provided by the temporal difference between current step and previous step in Bellman's equation, denoted as

$$e_c(t) = \alpha J(t) - [J(t-1) - r(t)] \quad (2.6)$$

The objective function for the action network is to minimize the total cost (or maximize the total reward), denoted as

$$e_a(t) = J(t) - U_c \quad (2.7)$$

where U_c is the ultimate (expected) cost function and J function is expected to approach this expected value.

Figure 2.1. The schematic diagram of typical model-free HDP structure.

2.3.3 Dual Heuristic Dynamic Programming (DHP)

Dual heuristic dynamic programming belongs to the advanced ADP design [59, 60]. As presented in Fig. 2.2, there are still one action network and one critic network in the design, and the action network is used to generate control actions. Yet, the critic network is used to approximate the derivatives of the value function λ rather than value function itself. The evaluation criteria is said to be more accurate in this case. Usually, there is a model network in DHP design. This model network is required to be trained offline based on the input and output data from the system, and then applied online to predict the future system variables (e.g., $\hat{x}(t+1)$ and $\hat{u}(t+1)$). In the backward process, the model network is also used as a bridge to connect the weights propagation from critic network to action network. Note that, the partial derivatives of value function is defined as $\lambda = [\frac{\partial J(t)}{\partial x(t)} \quad \frac{\partial J(t)}{\partial u(t)}]$. Thus the error function of critic network is defined as

$$e_c(t) = \frac{\partial J(t)}{\partial Y(t)} - \alpha \frac{\partial J(t+1)}{\partial Y(t)} - \frac{\partial r(t)}{\partial Y(t)} \quad (2.8)$$

where $Y(t) = [x(t) \quad u(t)]^T$. The error function of action network can be directly obtained from λ as

$$e_a(t) = \alpha \frac{\partial \hat{J}(t+1)}{\partial u(t)} \quad (2.9)$$

In recent literature, model-free DHP has also been proposed with finite difference technique on various balancing examples [61]. This approach has also been demonstrated with efficient computational time cost.

Figure 2.2. The schematic diagram of typical DHP structure.

2.3.4 Globalized Dual Heuristic Dynamic Programming (GDHP)

Globalized dual heuristic dynamic programming is the most advanced ADP design [59, 60]. As seen from Fig. 2.3, it has very similar structure with DHP, i.e., it has an action network to generate control action, yet a critic network to approximate both the value function and its derivatives (one may see that there are two dash lines to). Model network is also usually applied in GDHP design, as it is used to predict the future system variables and the corresponding future value function (with its derivatives). Though GDHP is regarded to be the most accurate ADP learning control approach, it generally has more computational cost and complicated learning algorithms than those with HDP and DHP. The error function of critic (action) network is the error combination of the previously introduced HDP and DHP designs, and thus the learning algorithm is more complicated. It is always a tradoff to choose between advanced ADP designs (i.e., DHP and GDHP)

and basic ADP design (i.e., HDP) for a specific system.

Figure 2.3. The schematic diagram of typical GDHP structure.

2.4 Summary

This chapter presents the background knowledge and literature discussion for adaptive dynamic programming and reinforcement learning, which are both deeply indebted to the idea of Markov decision processes from the field of optimal control. The background of learning and association (i.e., state-action pair) in decision-making process is provided. The fundamental principles of reinforcement learning and adaptive dynamic programming are also discussed. Three major architectures in adaptive dynamic programming are introduced to solve the Bellman's equation and find the optimal control policies.

In the rest of this dissertation, new ADP architectures are proposed based on the presented knowledge and principle in this chapter, and improved learning control results from toy problems to real-world applications will also be provided.

CHAPTER 3

A New Internal Goal Representation Framework

3.1 Introduction

In this chapter, a novel adaptive dynamic programming architecture with three neural networks, i.e., an action network, a critic network, and a reference network, is developed with internal goal-representation technique for online learning and optimization [62, 63]. Unlike the traditional ADP design with two neural networks (i.e., an action network and a critic network), this approach integrates the third neural network, called the reference network, into the actor-critic design framework. The motivation is to build a general-purpose (internal) reinforcement signal to facilitate learning and optimization overtime [6, 7, 5]. The detailed design procedure and its associated learning algorithm are provided to explain how the effective learning and optimization can be achieved. Furthermore, the learning control performance on two commonly used balancing benchmarks are also presented.

3.2 A Three-Network ADP Architecture

Fig. 3.1 shows the proposed ADP architecture with goal representation network [62, 63]. Compared to the existing ADP architectures [64, 65, 66, 67], the key idea is to integrate another neural network, the *reference network*, to provide the internal reinforcement signal (internal goal representation) $s(t)$. By introducing such a reference network to represent the system's internal goal, this architecture provides a new way to adaptively estimate the internal reinforcement signal instead of crafted by hands. This is the most important contribution of this work when compared to the existing ADP designs. From a mathematical point of view, this new architecture presents two major differences compared with that of the

existing ADP designs. First, the critic network has one more additional input $s(t)$ from the reference network. Second, the optimization error function and learning in the reference network and critic network are different: The error function of the reference network is related to the primary reinforcement signal $r(t)$, whereas for the critic network, it is related to the internal reinforcement signal $s(t)$. Both of these characteristics will change the parameters tuning and adaptation. Note that another characteristic of this method is that it shares the advantage of no requirement of a system model to predict the future system state, as proposed in the important ADP architecture in [68, 69]. This means similar to the architecture in [68], the proposed approach also stores the previous cost-to-go value to obtain the temporal difference for training at any time instance. This enables the online learning, association, and optimization over time.

Figure 3.1. The proposed ADP architecture with internal goal representation.

3.3 Learning and Optimization Algorithms

From Fig. 3.1 one can see, there are three paths to tune the parameters of the three types of networks. The action network in this architecture is similar to the classic ADP approach to indirectly backpropagate the error between the desired ultimate objective U_c and the J function from the critic network [68][70]. Therefore, the error function $E_a(t)$ used to update the parameters in the action network can be defined as (path 1 in Fig. 3.1):

$$e_a(t) = J(t) - U_c(t); \quad E_a(t) = \frac{1}{2}e_a^2(t). \quad (3.1)$$

The key of this architecture relies on the learning and adapting process for the reference network and critic network. As the primary reinforcement signal $r(t)$ is presented to the reference network, the secondary (also called internal) reinforcement signal $s(t)$ is adapted to provide a more informative internal reinforcement representation to the critic network, which in turn is used to provide a better approximation of the $J(t)$. In this way, the primary reinforcement signal $r(t)$ is in a higher hierarchical level and can be a simple binary signal to represent “good” or “bad”, or “success” or “failure”, while the secondary reinforcement signal $s(t)$ can be a more informative continuous values for improved learning and generalization performance. Therefore, the error function $E_f(t)$ used to update the parameters in the reference network can be defined as (path 2 in Fig. 3.1):

$$e_f(t) = \alpha J(t) - [J(t-1) - r(t)]; \quad E_f(t) = \frac{1}{2}e_f^2(t). \quad (3.2)$$

Once the reference network outputs the $s(t)$ signal, it will be used as an input to the critic network, and also used to define the error function to adjust the parameters of the critic network (path 3 in Fig. 3.2).

$$e_c(t) = \alpha J(t) - [J(t-1) - s(t)]; \quad E_c(t) = \frac{1}{2} e_c^2(t). \quad (3.3)$$

In this architecture, the chain backpropagation rule is used for training and adaptation of the parameters of all three networks. Fig. 3.2 shows the three backpropagation paths used to adapt the parameters in the three networks.

Figure 3.2. Parameters adaptation and tuning based on backpropagation

In this figure, the optimization error functions for the action network E_a , reference network E_f , and critic network E_c are defined in equations (3.1), (3.2) and (3.3), respectively. Therefore, chain backpropagation can be calculated through the three data paths as highlighted in Fig. 3.2. Briefly speaking, the high level conceptual calculation on this can be summarized as follows.

Path 1 For action network:

$$\frac{\partial E_a(t)}{\partial w_a(t)} = \frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u(t)} \frac{\partial u(t)}{\partial w_a(t)} \quad (3.4)$$

Path 2 For reference network:

Figure 3.3. Design of the three networks with nonlinear neural network: (a) Reference network design; (b) Critic network design

$$\frac{\partial E_f(t)}{\partial w_f(t)} = \frac{\partial E_f(t)}{\partial J(t)} \frac{\partial J(t)}{\partial s(t)} \frac{\partial s(t)}{\partial w_f(t)} \quad (3.5)$$

Path 3 For critic network:

$$\frac{\partial E_c(t)}{\partial w_c(t)} = \frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial w_c(t)} \quad (3.6)$$

3.3.1 Learning in Goal Network

Fig. 3.3(a) shows the reference network used in this design with a 3-layer nonlinear architecture (with 1 hidden layer). To calculate the backpropagation, we first need to define the reference network output $s(t)$ as follows.

$$s(t) = \frac{1 - \exp^{-k(t)}}{1 + \exp^{-k(t)}}. \quad (3.7)$$

$$k(t) = \sum_{i=1}^{N_h} w_{f_i}^{(2)}(t) y_i(t), \quad (3.8)$$

$$y_i(t) = \frac{1 - \exp^{-z_i(t)}}{1 + \exp^{-z_i(t)}}, \quad i = 1, \dots, N_h \quad (3.9)$$

$$z_i(t) = \sum_{j=1}^{n+1} w_{f_i, j}^{(1)}(t) x_j(t), \quad i = 1, \dots, N_h \quad (3.10)$$

Where z_i is the i th hidden node input of the reference network and y_i is the corresponding output of the hidden node, k is the input to the output node of the reference network before the sigmoid function, N_h is the number of hidden neurons of the reference network, and $(n+1)$ is the total number of inputs to the reference network including the action value $u(t)$ from the action network.

To apply the backpropagation rule, one can refer to Fig. 3.2 and equations (3.2) and (3.5). Specifically, since the output $s(t)$ is an input to the critic network, backpropagation can be applied here through the chain rule (path 2) to adapt the parameters W_f . This procedure is illustrated as follows.

(1) $\Delta w_f^{(2)}$: Reference network weight adjustment for the hidden to the output layer.

$$\Delta w_{f_i}^{(2)} = \eta_f(t) \left[-\frac{\partial E_f(t)}{\partial w_{f_i}^{(2)}(t)} \right] \quad (3.11)$$

$$\begin{aligned} \frac{\partial E_f(t)}{\partial w_{f_i}^{(2)}(t)} &= \frac{\partial E_f(t)}{\partial J(t)} \frac{\partial J(t)}{\partial s(t)} \frac{\partial s(t)}{\partial k(t)} \frac{\partial k(t)}{\partial w_{f_i}^{(2)}(t)} \\ &= \alpha e_f(t) \cdot \sum_{i=1}^{N_h} \left[w_{c_i}^{(2)}(t) \frac{1}{2} (1 - p_i^2(t)) w_{c_i, n+2}^{(1)}(t) \right] \cdot \frac{1}{2} (1 - (s(t))^2) \cdot y_i(t) \end{aligned} \quad (3.12)$$

(2) $\Delta w_f^{(1)}$: Reference network weight adjustments for the input to the hidden layer.

$$\Delta w_{f_i, j}^{(1)} = \eta_f(t) \left[-\frac{\partial E_f(t)}{\partial w_{f_i, j}^{(1)}(t)} \right] \quad (3.13)$$

$$\begin{aligned}
\frac{\partial E_f(t)}{\partial w_{f_i, j}^{(1)}(t)} &= \frac{\partial E_f(t)}{\partial J(t)} \frac{\partial J(t)}{\partial s(t)} \frac{\partial s(t)}{\partial k(t)} \frac{\partial k(t)}{\partial y_i(t)} \frac{\partial y_i(t)}{\partial z_i(t)} \frac{\partial z_i(t)}{\partial w_{f_i, j}^{(1)}(t)} \\
&= \alpha e_f(t) \cdot \sum_{i=1}^{N_h} [w_{c_i}^{(2)}(t) \frac{1}{2} (1 - p_i^2(t)) w_{c_i, n+2}^{(1)}(t)] \cdot \frac{1}{2} (1 - (s(t))^2) \cdot w_{f_i}^{(2)}(t) \\
&\quad \cdot \frac{1}{2} (1 - y_i^2(t)) \cdot x_j(t)
\end{aligned} \tag{3.14}$$

Once the reference network provides the secondary reinforcement signal $s(t)$ to the critic network, one can adapt the parameters in the critic network.

3.3.2 Learning in Critic Network

Fig. 3.3(b) shows the critic network used in our current design with a 3-layer nonlinear architecture (with 1 hidden layer). To calculate the backpropagation, we first need to define the critic network output $J(t)$ as follows.

$$J(t) = \sum_{i=1}^{N_h} w_{c_i}^{(2)}(t) p_i(t), \tag{3.15}$$

$$p_i(t) = \frac{1 - \exp^{-q_i(t)}}{1 + \exp^{-q_i(t)}}, \quad i = 1, \dots, N_h \tag{3.16}$$

$$q_i(t) = \sum_{j=1}^{n+2} w_{c_i, j}^{(1)}(t) x_j(t), \quad i = 1, \dots, N_h \tag{3.17}$$

Where q_i and p_i are the input and output of the i th hidden node of the critic network, respectively, and $(n + 2)$ is the total number of inputs to the critic network including the action value $u(t)$ from the action network and the secondary reinforcement signal $s(t)$ from the reference network.

By applying chain backpropagation rule (path 3), the procedure of adapting parameters in the critic network is summarized as follows.

(1) $\Delta w_c^{(2)}$: Critic network weight adjustments for the hidden to the output layer.

$$\Delta w_{c_i}^{(2)} = \eta_c(t) \left[-\frac{\partial E_c(t)}{\partial w_{c_i}^{(2)}(t)} \right] \quad (3.18)$$

$$\frac{\partial E_c(t)}{\partial w_{c_i}^{(2)}(t)} = \frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial w_{c_i}^{(2)}(t)} = \alpha e_c(t) \cdot p_i(t) \quad (3.19)$$

(2) $\Delta w_c^{(1)}$: Critic network weight adjustments for the input to the hidden layer.

$$\Delta w_{c_i, j}^{(1)} = \eta_c(t) \left[-\frac{\partial E_c(t)}{\partial w_{c_i, j}^{(1)}(t)} \right] \quad (3.20)$$

$$\frac{\partial E_c(t)}{\partial w_{c_i, j}^{(1)}(t)} = \frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial p_i(t)} \frac{\partial p_i(t)}{\partial q_i(t)} \frac{\partial q_i(t)}{\partial w_{c_i, j}^{(1)}(t)} = \alpha e_c(t) \cdot w_{c_i}^{(2)}(t) \cdot \frac{1}{2}(1 - p_i^2(t)) \cdot x_j(t) \quad (3.21)$$

3.4 Simulation Studies

The proposed three-network algorithm has been implemented on a cart-pole balancing problem, which is the same as that in [68]. The ultimate goal here is to control the force applied on the cart to move it either left or right to keep the balance of the single pole mounted on the cart. The system function of the model is described as following:

$$\partial^2 \theta / \partial t^2 = \frac{g \sin \theta + \frac{\cos \theta [-F - ml\ddot{\theta}^2 \sin \theta + \mu_c \text{sgn}(\dot{x})]}{m_c + m} - \frac{\mu_p \dot{\theta}}{ml}}{l(\frac{4}{3} - \frac{m \cos \theta^2}{m_c + m})} \quad (3.22)$$

$$\partial^2 x / \partial t^2 = \frac{F + ml[\ddot{\theta}^2 \sin \theta - \ddot{\theta} \cos \theta] - \mu_c \text{sgn}(\dot{x})}{m_c + m} \quad (3.23)$$

where the acceleration $g = 9.8m/s^2$, the mass of the cart $m_c = 1.0kg$, the mass of the pole $m = 0.1kg$, half-pole length $l = 0.5m$, the coefficient of friction of the cart $\mu = 0.0005$ and the coefficient of friction the pole $\mu_p = 0.000002$. The force

F applied to the cart is either 10 Newtons or -10 Newtons, and the sgn function in equation (3.23) is defined as following:

$$\text{sgn}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \quad (3.24)$$

The state vector in this system model is

$$Q = [x \quad \theta \quad \dot{x} \quad \dot{\theta}] \quad (3.25)$$

In current case study, the same criteria as those in [68] are adopted to evaluate the performance of our control approach. That is to say a pole is considered fallen when the angular is outside the range of $[-12^\circ, 12^\circ]$ or the cart is beyond the range of $[-2.4, 2.4]m$. Note that F , which applied to the cart, is binary while the control action $u(t)$, which fed into critic network, is continuous value.

3.4.1 Cart-Pole Balancing Example

In order to evaluate the statistic performance of our proposed approach, 100 independent runs are set to this task with different initial state conditions. Specifically, the angular and angular velocity of the pole in each of these initial states are uniformly generated within $[-0.1^\circ, 0.1^\circ]$ and $[-0.5, 0.5] * 180/\pi \text{ rad/s}$ respectively, while the position and velocity of the cart are both 0. We hope these different initial conditions will provide a comprehensive understanding of our approach.

Table 3.1. Summary of the parameters used in cart-pole balancing task

Para.	$l_c(0)$	$l_a(0)$	$l_r(0)$	$l_c(f)$	$l_a(f)$	$l_r(f)$	*
value	0.3	0.3	0.3	0.001	0.001	0.001	*
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
value	80	100	50	0.05	0.005	0.05	0.95

The parameters in our simulation are summarized in Table 3.1 and the notations are defined as following:

$l_c(0)$: initial learning rate of the critic network;

$l_a(0)$: initial learning rate of the action network;

$l_r(0)$: initial learning rate of the reference network;

$l_c(k)$: learning rate of the critic network which is decreased by 0.05 every 5 time step until it reach $l_c(f)$ and stay thereafter;

$l_a(k)$: learning rate of the action network which is decreased by 0.05 every 5 time step until it reach $l_a(f)$ and stay thereafter;

$l_r(k)$: learning rate of the reference network which is decreased by 0.05 every 5 time step until it reach $l_r(f)$ and stay thereafter;

N_c : internal cycle of the critic network;

N_a : internal cycle of the action network;

N_r : internal cycle of the reference network;

T_c : internal training error threshold for the critic network;

T_a : internal training error threshold for the action network;

T_r : internal training error threshold for the reference network;

For comparative study, the proposed approach and the ADP approach in [68]

Table 3.2. Performance evaluation on case I: cart-pole balancing task. The 2nd and the 3rd columns are with our proposed method, while the 4th and the 5th columns are the results from existing approach

Noise type	Success rate	# of trial	Success rate	# of trial
Noise free	100 %	13.7	100 %	6
Uniform 5% a.*	100 %	16.1	100 %	8
Uniform 10% a.	100 %	20.6	100 %	14
Uniform 5% s.†	100 %	12.6	100 %	32
Uniform 10% s.	100 %	14.4	100 %	54
Gaussian $\sigma^2(0.1)$ s.	100 %	15.0	100 %	164
Gaussian $\sigma^2(0.2)$ s.	100 %	21.3	100 %	193

* a. : actuators are subject to the noise

† s. : sensors are subject to the noise

are tested under the same parameter setting. These results are summarized in Table 3.2. For fair comparison, the noise disturbances are also added in the simulation. From these results, the proposed approach can provide competitive results, especially under the uniform noise and Gaussian noise on sensors. Note that under noise free condition and uniform noise on actuator (both 5% and 10%), the approach in [68] can provide better results. This might indicate the proposed three-network ADP approach is more robust and can work effectively under relatively large level of noises.

3.4.2 Triple-Link Inverted Pendulum Balancing Example

The three-network approach mentioned above is now tested on triple-link inverted pendulum, which is unstable with multivariables and exhibits non-negligible nonlinearities. This kind of pendulum is frequently used to evaluate the performance of new control strategies. Here we consider the same system model as in [68]. Fig.3.4 shows a schematic diagram depicting the notations used.

Figure 3.4. Definition of notation used in the system equations for the triple link inverted pendulum benchmark

The nonlinear dynamic equations of this system can be expressed as:

$$F(q) \frac{d^2 q}{dt^2} = -G(q, \frac{dq}{dt}) \frac{dq}{dt} - H(q) + L(q, u) \quad (3.26)$$

where

$$F(q) = \begin{bmatrix} A_1 & A_2 \cos(\theta_1) & A_3 \cos(\theta_2) & A_4 \cos(\theta_3) \\ A_9 \cos(\theta_1) & A_{10} & A_{11} \cos(\theta_1 - \theta_2) & A_{12} \cos(\theta_1 - \theta_3) \\ A_{18} \cos(\theta_2) & A_{19} \cos(\theta_1 - \theta_2) & A_{20} & A_{21} \cos(\theta_2 - \theta_3) \\ A_{28} \cos(\theta_3) & A_{29} \cos(\theta_1 - \theta_3) & A_{30} \cos(\theta_2 - \theta_3) & A_{31} \end{bmatrix} \quad (3.27)$$

$$G(q, \frac{dq}{dt}) = \begin{bmatrix} A_5 & A_6 \sin(\theta_1) \dot{\theta}_1 & A_7 \sin(\theta_2) \dot{\theta}_2 & A_8 \sin(\theta_3) \dot{\theta}_3 \\ 0 & A_{13} & A_{14} \sin(\theta_1 - \theta_2) \dot{\theta}_2 + A_{15} & A_{16} \sin(\theta_1 - \theta_3) \dot{\theta}_3 \\ 0 & A_{22} \sin(\theta_1 - \theta_2) \dot{\theta}_2 + A_{23} & A_{24} & A_{25} \sin(\theta_2 - \theta_3) \dot{\theta}_3 + A_{26} \\ 0 & A_{33} \sin(\theta_1 - \theta_3) \dot{\theta}_1 & A_{35} \sin(\theta_2 - \theta_3) \dot{\theta}_2 + A_{36} & A_{32} \end{bmatrix} \quad (3.28)$$

$$q = \begin{bmatrix} x \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \quad (3.29)$$

$$H(q) = \begin{bmatrix} 0 \\ A_{17} \sin(\theta_1) \\ A_{27} \sin(\theta_2) \\ A_{34} \sin(\theta_3) \end{bmatrix} \quad (3.30)$$

$$L(q, u) = \begin{bmatrix} K_s u - \text{sgn}(x) \mu_x A_{37} \\ -\text{sgn}(\theta_1) \mu_1 A_{38} \\ -\text{sgn}(\theta_2) \mu_2 A_{39} \\ -\text{sgn}(\theta_3) \mu_3 A_{40} \end{bmatrix} \quad (3.31)$$

Note that all the μ here is the Coulomb friction coefficient for links and is not linearizable. In this simulation, $\mu_x = 0.07$, $\mu_1 = \mu_2 = \mu_3 = 0.003$, and A_i are all available in Table 3.3. The parameters in Table 3.3 are defined in the following:

L_1 : 0.43m, total length of the 1st link;

L_2 : 0.33m, total length of the 2nd link;

Table 3.3. System constants for the triple link inverted pendulum problem

Constant	Value	Constant	Value
A_1	$M + m_1 + m_2 + m_3$	A_2	$m_1 l_1 + (m_2 + m_3)L_1$
A_3	$m_2 l_2 + m_3 L_2$	A_4	$m_3 l_3$
A_5	C_c	A_6	$-m_1 l_1 - (m_2 + m_3)L_1$
A_7	$-(m_2 l_2 + m_3 L_2)$	A_8	$-m_3 l_3$
A_9	$m_1 l_1 + (m_2 + m_3)L_1$	A_{10}	$I_1 + m_2 l_1^2 + (m_2 + m_3)L_1^2$
A_{11}	$(m_2 l_2 + m_3 L_2)L_1$	A_{12}	$m_3 l_3 L_1$
A_{13}	$C_1 + C_2$	A_{14}	$(m_2 l_2 + m_3 L_2)L_1$
A_{15}	$-C_2$	A_{16}	$m_3 l_3 L_1$
A_{17}	$-g(m_1 l_1 + m_2 L_1 + m_3 L_1)$	A_{18}	$m_2 l_2 + m_3 L_2$
A_{19}	$(m_2 l_2 + m_3 L_2)L_1$	A_{20}	$I_2 + m_3 L_2^2 + m_2 l_2^2$
A_{21}	$m_3 l_3 L_2$	A_{22}	$-(m_2 l_2 + m_3 L_2)L_1$
A_{23}	$-C_2$	A_{24}	$C_2 + C_3$
A_{25}	$m_3 l_3 L_2$	A_{26}	$-C_3$
A_{27}	$-g(m_2 l_2 + m_3 L_2)$	A_{28}	$m_3 l_3$
A_{29}	$m_3 l_3 L_1$	A_{30}	$m_3 l_3 L_2$
A_{31}	$I_3 + m_3 l_3^2$	A_{32}	C_3
A_{33}	$-m_3 l_3 L_1$	A_{34}	$-gm_3 l_3$
A_{35}	$-m_3 l_3 L_2$	A_{36}	$-C_3$
A_{37}	1.3	A_{38}	0.506
A_{39}	0.219	A_{40}	0.568

$L_3 : 0.13m$, total length of the 3rd link;
 $l_1 : 0.37m$, length from mount joint to the center of gravity of 1st link;
 $l_2 : 0.3m$, length from 1st joint to the center of gravity of 2nd link;
 $l_3 : 0.05m$, length from 2nd joint to the center of gravity of 3rd link;
 $m_1 : 0.4506kg$, mass of the 1st link;
 $m_2 : 0.219kg$, mass of the 2nd link;
 $m_3 : 0.0568kg$, mass of the 3rd link;
 $M : 1.014kg$, mass of the whole cart;
 $g : 9.8m/s^2$, acceleration of gravity;
 $I_1 : 0.0042kgm^2$, mass moment of inertia of the 1st link about its center of gravity;
 $I_2 : 0.0012kgm^2$, mass moment of inertia of the 2nd link about its center of gravity;
 $I_3 : 0.00010609kgm^2$, mass moment of inertia of the 3rd link about its center of gravity;
 $C_c : 5.5Nms$, dynamic friction coefficient between the cart and the track;
 $C_1 : 0.00026875Nms$, dynamic friction coefficient for the 1st link;
 $C_2 : 0.00026875Nms$, dynamic friction coefficient for the 2nd link;
 $C_3 : 0.00026875Nms$, dynamic friction coefficient for the 3rd link;

In this case, the only control unit u (in voltage), generated by the action network, is converted into force by an analog amplifier (with gain $K_s = 24.7125$ Newtons/volt) to the DC servo motor. Each link here only rotates in a vertical plane, and the sample time interval is chosen to be $5ms$. In order to better show the performance of the proposed three network algorithm with Runge-Kutta methods,

the system equations are transformed into the state-space forms as follows:

$$\dot{Q}(t) = f(Q(t), u(t)) \quad (3.32)$$

$$f(Q(t), u(t)) = \begin{bmatrix} \mathbf{0}_{4 \times 4} & \mathbf{I}_{4 \times 4} \\ \mathbf{0}_{4 \times 4} & -F^{-1}(Q(t))G(Q(t)) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{4 \times 1} \\ -F^{-1}(Q(t))[H(Q(t)) - L(Q(t), u(t))] \end{bmatrix} \quad (3.33)$$

and

$$Q = [x \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \dot{x} \quad \dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3] \quad (3.34)$$

Specific physical meanings for these eight state variables are illustrated in the Fig.3.4. They are:

- a) x , position of the cart on the track;
- b) θ_1 , vertical angle of the 1st link joint to the cart;
- c) θ_2 , vertical angle of the 2nd link joint to the 1st link;
- d) θ_3 , vertical angle of the 3rd link joint to the 2nd link;
- e) $\dot{x}$, cart velocity;
- f) $\dot{\theta}_1$, angular velocity of θ_1 ;
- g) $\dot{\theta}_2$, angular velocity of θ_2 ;
- h) $\dot{\theta}_3$, angular velocity of θ_3 .

Environment Setup

In this experimental setup, the constraints for the triple-linked inverted pendulum are: 1) The cart track extends 1.0 meter to both sides from the center point; 2) The voltage applied to the motor should be within $[-30, 30]V$; 3) Each link angle should be within the range of $[-20^\circ, 20^\circ]$ with respect to the vertical axis. Here, condition 2) is guaranteed by using a sigmoid function. While for the other two conditions, if either one fails or both fail, the system will be provided with an external reinforcement signal $r = -1$ at the moment of failure, otherwise $r = 0$

all the time. Based on this external reinforcement signal, the three-network ADP approach will automatically and adaptively develop an internal reinforcement signal to facilitate the learning and optimization process over time. For performance assessment, the same criteria as in [68] are adopted for this triple-linked inverted pendulum in this case study.

One hundred runs in this current study are conducted. Similarly as cart-pole problem, for different runs, here I will use different initial starting states. Specifically, I set the three angles and angle velocity of the triple links to be uniformly within the range of $[-1^\circ, 1^\circ]$ and $[-0.50, 0.50] * 180/\pi \text{ rad/s}$, respectively. As for x and $\dot{x}$, their initial states are set to zero. The critic network is chosen as a 10-20-1 multi-layer perceptron (MLP) neural network structure [71, 72]. That is to say there are 10 input neurons, 20 hidden layer neurons, and 1 output neuron in this neural network. The action neural network is chosen as 8-14-1 MLP structure and the reference network is set as 9-14-1 MLP structure. The learning parameters such as learning rate, internal cycle, and internal training error threshold for the action network, reference network, and critic network are presented in Table 3.4.

Table 3.4. Summary of the parameters used in triple-link inverted pendulum balancing task

Para.	$l_c(0)$	$l_a(0)$	$l_r(0)$	$l_c(f)$	$l_a(f)$	$l_r(f)$	*
value	0.3	0.3	0.3	0.001	0.001	0.001	*
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
value	80	100	50	0.05	0.005	0.05	0.95

Simulation Analysis

Summary of the simulation results with different noise conditions are presented in Table 3.5. Comparing with those in [68], the proposed three-network ADP architecture can achieve competitive results in this case as well. To observe how the proposed approach performs under this task, here I present a snapshot of the

Figure 3.5. Statistics of the successful runs. (a) Number of required trials for the successful runs. (b) Histogram the statistics.

statistics of different runs under noise free condition. Fig. 3.5(a) shows the number of required trials for each successful run, and Fig. 3.5(b) shows the corresponding histogram information. The three angles and angle velocities of the triple links are set to be uniformly within the range of $[-1^\circ, 1^\circ]$ and $[-0.50, 0.50] * 180/\pi \text{ rad/s}$, respectively, and the initial states of x and $\dot{x}$ are set to zero.

Table 3.5. Performance evaluation on case II: triple-link balancing task. The 2nd and the 3rd columns are with our proposed method, while the 4th and the 5th columns are the results from existing approach

Noise type	Success rate	# of trial	Success rate	# of trial
Noise free	99 %	571.4	97 %	1194
Uniform 5% a.	99 %	596.9	92 %	1239
Uniform 10% a.	99 %	673.1	84 %	1852
Uniform 5% s.	99 %	620.1	89 %	1317
Uniform 10% s.	99 %	657.9	80 %	1712
Gaussian $\sigma^2 (0.1)$ s.	80 %	1170.4	85 %	1508
Gaussian $\sigma^2 (0.2)$ s.	50 %	1372.2	76 %	1993

To further analyze how the proposed ADP structure can accomplish the control task, Fig. 3.6(a) to 3.6(h) show a typical trajectory on the task for all the

Figure 3.6. Typical trajectory on the triple-link inverted pendulum balancing task. (a) The position x of the cart. (b) The 1st joint angle of the triple link pendulum. (c) The 2st joint angle of the triple link pendulum. (d) The 3st joint angle of the triple link pendulum. (e) The velocity of the cart. (b) The angular velocity of the 1st joint angle of the triple link pendulum. (c) The angular velocity of the 2st joint angle of the triple link pendulum. (d) The angular velocity of the 3st joint angle of the triple link pendulum. 30

Figure 3.7. Typical histogram on the triple-link inverted pendulum balancing task. (a) The histogram of position x of the cart. (b) The histogram of the 1st joint angle of the triple link pendulum. (c) The histogram of the 2st joint angle of the triple link pendulum. (d) The histogram of the 3st joint angle of the triple link pendulum.

state variables under noise free condition, namely the the position x of the cart (3.6(a)), the first, second, and third joint angle of the triple link pendulum (3.6(b) to 3.6(d)), the velocity of the cart 3.6(e), and the angular velocity of the first, second, and third joint angle of the triple link pendulum (3.6(f) to 3.6(h)). The corresponding histogram information for the position x and three joint angle information are also shown in Fig. 3.7(a) to Fig. 3.7(d). All these results clearly indicate that the proposed ADP approach can effectively control the system to achieve desired states during the online learning process.

Furthermore, since the main objective of the reference network is to provide

(a)

(b)

Figure 3.8. Typical trajectory of cost-to-go and control action signal on the triple-link inverted pendulum balancing task. (a) The cost-to-go signal on the task. (b) The control action signal on the task.

an internal reinforcement signal to facilitate the learning and optimization in the ADP structure, I further analyze how the J value and control action u looks like in this case. Fig. 3.8(a) shows a snapshot of the convergence of the J value during the learning process, and Fig. 3.8(b) shows another snapshot of the control action u during a typical successful run. Both figures also clearly demonstrate that our proposed approach can effectively accomplish the control performance in this case.

3.5 Summary

A three-network ADP architecture with an action network, a critic network, and a reference network, for adaptive learning, control, and optimization is presented here. The key idea of this approach is the introduction of a new reference network (will be called goal network in the remaining of this dissertation) to develop internal goal-representation to facilitate learning and optimization. It provides an effective way to adaptively and automatically build the internal goal representations for the intelligent systems. A detailed design architecture and learning algorithm is presented, followed by detailed simulation analysis on two benchmark tasks (i.e., balancing a cart-pole model and a triple-link inverted pendulum model) to demonstrate the effectiveness of our approach. In the next chapter, I am going to further demonstrate its adaptive learning mechanism in an tracking control problem.

CHAPTER 4

Goal Representation Design for Heuristic Dynamic Programming (GrHDP) Architecture

4.1 Introduction

A “Dual” critic network technique is integrated into the ADP architecture, and this new architecture is called goal representation heuristic dynamic programming (GrHDP) design [73, 74]. Specifically, an alternative choice rather than crafting the reinforcement signal manually from priori knowledge is proposed. The overall adaptive learning performance has been tested on two tracking control benchmarks with a tracking filter. For comparative studies, the tracking performance with the typical HDP is also presented to justify the improved performance. Furthermore, a virtual reality (VR) platform is provided to demonstrate the real-time simulation under different disturbance situations. Detailed Lyapunov stability analysis for the proposed approach is presented to support the proposed structure from a theoretical point of view (see Appendix A).

4.2 Design of GrHDP for Tracking Control

The schematic diagram of this proposed idea is presented in Fig.4.1. The action network is kept the same as that in [68, 51]. While for the critic network, I integrate with one reference network (also called goal network) and therefore there are two networks in the dual-critic network block as presented in Fig.4.2. The tracking filter is added to show the performance on tracking control problem. The following of this section will introduce the dual critic network block and the tracking filter, respectively.

Figure 4.1. Architecture design of the proposed HDP approach with a tracking filter.

4.2.1 Design of Dual-Critic Network

The motivation of this dual critic network (goal representation) design is in two-fold: one is to provide an internal goal guidance for the critic network; the other one is to help approximate the cost-to-go adaptive, since the internal reinforcement signal works as one of the input vectors for the critic network.

From the system-level view in Fig.4.1, we can see that the parameters in dual critic network block can not only be tuned by external signal, but also be adjusted by itself. Specifically, the reference (goal) network in the top of the block is tuned by the error function with the external reward signal, while the critic network in the bottom of the block is tuned by the error function with the internal reinforcement signal. As presented in Fig.4.2, the reference network observes a regular reward signal $r(k)$ (usually a binary value) from external environment and provides the critic network with a detailed internal reinforcement signal $s(k)$ (usually a contin-

Figure 4.2. Description of the dual critic network.

uous value) by justifying the system state vectors $X(k)$ and control action $u(k)$. In order to approximate the value function $J(k)$ well, the critic network keeps the same inputs as the reference network in addition with the internal signal $s(k)$. Moreover, $s(k)$ also contributes in the error function of the critic network as the dash line showed inside the block. Since the $s(k)$ can be automatically adjusted according to the state vectors $X(k)$ and the control action $u(k)$, we regard it as an adaptive reinforcement signal. In summary, the key idea of this “*dual critic*” design is to use the reference network to automatically and adaptively generate the internal goal signal, rather than hand-crafted in the traditional HDP approaches, to guide the decision-making process for the optimal action at any time instance to accomplish the final goal. This reference network can also actively interact with the critic network and action network, either directly (for critic network) or indirectly (for action network), to support the action selection in a principled way.

Figure 4.3. Description of the tracking filter.

4.2.2 Design of Tracking Filter

In order to demonstrate the improvement of the proposed dual-critic (GrHDP) controller, I would like to test it on nonlinear tracking control problem with a tracking filter as presented in Fig.4.1. The inner structure of the tracking filter is presented in Fig.4.3, which was motivated from [75] and later developed in [51, 76].

The nonlinear system function is defined in a general form as that.

$$x(k+1) = f(x(k)) + u(k) + d(k) . \quad (4.1)$$

where $f(x(k))$ is the nonlinear system function, $x(k) = [x_1(k) \ x_2(k) \ x_3(k) \ \dots \ x_n(k)]$ and $x_i(k) \in \mathfrak{R}$ is the state value for the i th dimension at time instance k . $u(k)$ is the control action. And $d(k)$ is the disturbance bounded in $[-d_m, d_m]$, where d_m is a constant value.

The nonlinear system function $f(x(k))$ is assumed unknown in the simulation and can be approximated by the action network here. The approximation value is denoted as $\hat{f}(x(k))$, which works as one of the inputs of the filter. Moreover, the inputs of the filter also include the current state vector $X(k)$ and the desired trajectory value $x_d(k)$ and $x_d(k+1)$ as presented in Fig.4.3. The error $e(k)$ is defined with the difference between the current state value and the desired value

as follows.

$$e(k) = x(k) - x_d(k). \quad (4.2)$$

and the filtered tracking error $\bar{e}(k)$ is defined as

$$\bar{e}(k) = e(k) + \lambda_1 e_{n-1}(k) + \dots + \lambda_{n-1} e_1(k). \quad (4.3)$$

Where $e_{n-1}(k), \dots, e_1(k)$ are the past error values, which mean that $e_{n-i}(k) = e_n(k-i)$, $i = 0, 1, \dots, n-1$, $n \in \mathfrak{R}$. For brevity, we define $\Lambda = [\lambda_{n-1}, \lambda_{n-2}, \dots, \lambda_1]$, where $\lambda_i \in \mathfrak{R}$. Therefore, (4.3) can be re-written as

$$\bar{e}(k) = [\Lambda \ I]e(k). \quad (4.4)$$

Again, we can rewrite (4.3) for time instance $k+1$ as,

$$\bar{e}(k+1) = e(k+1) + \lambda_1 e_{n-1}(k+1) + \dots + \lambda_{n-1} e_1(k+1). \quad (4.5)$$

Substitute (4.1) into (4.5), we will get

$$\begin{aligned} \bar{e}(k+1) &= f(x(k)) - x_d(k+1) + \lambda_1 e_{n-1}(k+1) + \dots \\ &\quad + \lambda_{n-1} e_1(k+1) + u(k) + d(k) \\ &= f(x(k)) - x_d(k+1) + \lambda_1 e_n(k) + \dots \\ &\quad + \lambda_{n-1} e_2(k) + u(k) + d(k). \end{aligned} \quad (4.6)$$

Similar as that in [51, 76], we define that the control sequence

$$\begin{aligned} u(k) &= x_d(k+1) - \hat{f}(x(k)) + k_v \bar{e}(k) - \lambda_1 e_n(k) - \\ &\quad \dots - \lambda_{n-1} e_2(k). \end{aligned} \quad (4.7)$$

Substitute (4.7) into (4.6), we will get that

$$\bar{e}(k+1) = K_v \bar{e}(k) - \tilde{f}(x(k)) + d(k). \quad (4.8)$$

where $\tilde{f}(x(k))$ is the nonlinear system function approximation error given by

$$\tilde{f}(x(k)) = f(x(k)) - \hat{f}(x(k)). \quad (4.9)$$

and K_v is the gain value. Assuming that $\tilde{f}(x(k))$ is bounded, the system will be stable if $0 < K_{vmax} < 1$, where K_{vmax} is the maximum eigenvalue of K_v [76].

4.2.3 Design of Reinforcement Signals

As it is mentioned above, there are two types of reinforcement signals in our proposed approach. One is the external reinforcement signal which comes from the environment, and the other one is internal reinforcement signal which comes from the reference network and works as an internal goal that guides the system's behavior specifically.

External reinforcement signal $r(k)$ is defined according to the current filtered tracking error $\bar{e}(k)$

$$r(k) = [r_1(k) \ r_2(k) \ , \dots, \ r_m(k)] \in \Re^m. \quad (4.10)$$

with

$$r_i(k) = \begin{cases} 0, & \text{if } \|\bar{e}_i(k)\| \leq c \\ -1, & \text{if } \|\bar{e}_i(k)\| \geq c \end{cases}, i = 1, 2, 3, \dots, m. \quad (4.11)$$

where $\|\cdot\|$ represents Euclidean vector 2-norm and c is the constant threshold for the filtered tracking error. The binary value “0” represents “good” for the tracking performance while “-1” means “poor”.

The internal reinforcement signal $s(k)$ is the output of reference network bounded in $[-1, 1]$ and can be adaptively adjusted as the system states change. At the feed-forward stage, the internal signal works as one of the inputs for the critic network. At the feed-backward stage, the parameters in the critic network are tuned by the error function with $s(k)$. Therefore, the internal reinforcement signal $s(k)$ closely connects the reference network and the critic network as a whole block.

4.3 Association and Implementation

In this section, the procedures to implement the proposed GrHDP (dual critic network HDP) approach with a tracking filter is presented. A brief pseudo-code (Algorithm 1) is provided for the implementation steps. Multilayer perceptron

(MLP) neural network has been one of the most popular techniques to approximate the nonlinear function in the ADP community [77, 12, 78]. Thus, in this design, MLP is used for the neural network implementation.

4.3.1 Learning and Optimization in Dual Critic Network

Compared with the typical HDP design in [68], the learning and optimization of the critic network here is associated with the reference network as presented in Fig.4.4. At the forward stage, the reference network obtains the inputs of the state vector $X(k)$ and the control action $u(k)$ and provides an internal reinforcement signal $s(k)$ for the critic network. Then the critic network updates the cost-to-go signal $J(k)$. At the backward stage, the reference network will first be tuned by the error function (4.15) with $r(k)$ and the updated cost-to-go signal $J(k)$. After this is done, the reference network will provide the updated internal reinforcement signal $s(k)$ for the critic network, which will then be adjusted with the error function (4.12). The learning process will repeat until the terminal conditions are satisfied. Pseudo-code in *Algorithm1* shows exactly this learning procedure.

Figure 4.4. Learning schematic in dual critic network.

The error function of the critic network is defined as follows.

$$e_c(k) = \alpha J(k) - [J(k-1) - s(k)]; \quad E_c(k) = \frac{1}{2} e_c^2(k). \quad (4.12)$$

where

$$J(k) = \omega_c^{(1)}(k) \cdot \phi(\omega_c^{(2)}(k) \cdot x_c(k)). \quad (4.13)$$

and $\omega_c^{(1)}(k)$ and $\omega_c^{(2)}(k)$ refer the weights of input to hidden layer and hidden to output layer in critic network, respectively. $x_c(k)$ is the input vector of the critic network and it contains the state vector $X(k)$, the control action $u(k)$ and the internal signal $s(k)$. ϕ is for sigmoid function that refines the output into the range of $[-1, 1]$. And

$$s(k) = \phi\left(\omega_r^{(1)}(k) \cdot \phi(\omega_r^{(2)}(k) \cdot x_r(k))\right). \quad (4.14)$$

and $\omega_r^{(1)}(k)$ and $\omega_r^{(2)}(k)$ refer the weights of input to hidden layer and hidden to output layer in reference network, respectively. $x_r(k)$ is the input vector of reference network and it contains the state vector $X(k)$ and the control action $u(k)$.

The error function of the reference network is defined as follows.

$$e_r(k) = \alpha J(k) - [J(k-1) - r(k)]; \quad E_r(k) = \frac{1}{2} e_r^2(k). \quad (4.15)$$

Given that the state vector $X(k)$ has n elements and the control action $u(k)$ is a single control unit, the inputs for the reference network and the critic network will be $(n+1)$ and $(n+2)$, as presented in Fig.4.5(a) and Fig.4.5(b), respectively. Chain backpropagation rule is employed for the neural networks to learn and adapt their weights.

1) *Reference Network* : The reference network is introduced here to provide an internal goal representation for the critic network. The internal goal $s(k)$ is defined as follows:

$$s(k) = \frac{1 - \exp^{-l(k)}}{1 + \exp^{-l(k)}}. \quad (4.16)$$

Figure 4.5. (a) The neural network structure of the reference network; (b) The neural network structure of the critic network.

$$l(k) = \sum_{i=1}^{N_{rh}} w_{r_i}^{(2)}(t) y_i(k), \quad (4.17)$$

$$y_i(k) = \frac{1 - \exp^{-z_i(k)}}{1 + \exp^{-z_i(k)}}, \quad i = 1, \dots, N_{rh} \quad (4.18)$$

$$z_i(k) = \sum_{j=1}^{n+1} w_{r_i, j}^{(1)}(t) x_{rj}(k), \quad i = 1, \dots, N_{rh} \quad (4.19)$$

Where z_i is the input of the i th hidden node and y_i is the corresponding output of this hidden node after the sigmoid function, l is the input to the output node, N_{rh} is the number of hidden neurons in the reference network, and x_{rj} is the input vector of the reference network, which has $(n + 1)$ input nodes as presented in Fig.4.5(a).

The procedure of backpropagation rule applied to the reference network is illustrated as follows.

(i) $\Delta w_r^{(2)}$: Reference network weights adjustment from hidden layer to output layer.

$$\Delta w_{r_i}^{(2)}(k) = \eta_r(k) \left[-\frac{\partial E_r(k)}{\partial w_{r_i}^{(2)}(k)} \right] \quad (4.20)$$

where $\eta_r(k)$ is the learning rate of the reference network at time instance k , and

$$\begin{aligned} \frac{\partial E_r(k)}{\partial w_{r_i}^{(2)}(k)} &= \frac{\partial E_r(k)}{\partial J(k)} \frac{\partial J(k)}{\partial s(k)} \frac{\partial s(k)}{\partial l(k)} \frac{\partial l(k)}{\partial w_{r_i}^{(2)}(k)} \\ &= \alpha e_r(k) \cdot \frac{1}{2} (1 - (s(k))^2) \cdot y_i(k) \\ &\quad \cdot \sum_{i=1}^{N_{rh}} [w_{c_i}^{(2)}(k) \frac{1}{2} (1 - p_i^2(k)) w_{c_{i, n+2}}^{(1)}(k)] \end{aligned} \quad (4.21)$$

(ii) $\Delta w_r^{(1)}$: Reference network weights adjustment from input layer to hidden layer.

$$\Delta w_{r_{i, j}}^{(1)}(k) = \eta_r(k) \left[-\frac{\partial E_r(k)}{\partial w_{r_{i, j}}^{(1)}(k)} \right] \quad (4.22)$$

$$\begin{aligned} \frac{\partial E_r(k)}{\partial w_{r_{i, j}}^{(1)}(t)} &= \frac{\partial E_r(k)}{\partial J(k)} \frac{\partial J(k)}{\partial s(k)} \frac{\partial s(k)}{\partial l(k)} \frac{\partial l(k)}{\partial y_i(k)} \frac{\partial y_i(k)}{\partial z_i(k)} \frac{\partial z_i(k)}{\partial w_{r_{i, j}}^{(1)}(k)} \\ &= \alpha e_r(k) \cdot \frac{1}{2} (1 - y_i^2(k)) \cdot x_{rj}(k) \cdot \frac{1}{2} (1 - (s(k))^2) \\ &\quad \cdot w_{r_i}^{(2)}(k) \cdot \sum_{i=1}^{N_{rh}} [w_{c_i}^{(2)}(k) \frac{1}{2} (1 - p_i^2(k)) w_{c_{i, n+2}}^{(1)}(k)] \end{aligned} \quad (4.23)$$

Once the internal goal $s(k)$ is updated in reference network, we can adapt the weights tuning in the critic network.

2) *Critic Network* : In literature, the critic network is applied to approximate the cost function and its inputs normally contain state vector $X(k)$ and control unit $u(k)$. Here we add one more input with the internal goal $s(k)$ and hope that $s(k)$ can provide the critic network with detailed goal representation that contributes to the system's decision making. The cost-to-go signal $J(k)$ is defined as follows.

$$J(k) = \sum_{i=1}^{N_{ch}} w_{c_i}^{(2)}(k) p_i(k), \quad (4.24)$$

$$p_i(k) = \frac{1 - \exp^{-q_i(k)}}{1 + \exp^{-q_i(k)}}, \quad i = 1, \dots, N_{ch} \quad (4.25)$$

$$q_i(k) = \sum_{j=1}^{n+2} w_{c_i, j}^{(1)}(k) x_{cj}(k), \quad i = 1, \dots, N_{ch} \quad (4.26)$$

Where q_i and p_i are the input and output of the i th hidden node in the critic network, respectively, and x_{cj} is inputs vector of the critic network with $n + 2$ nodes as presented in Fig.4.5(b).

The procedure of backpropagation rule applied to the critic network is provided as follows.

(i) $\Delta w_c^{(2)}$: Critic network weights adjustment from hidden layer to output layer.

$$\Delta w_{c_i}^{(2)}(k) = \eta_c(k) \left[-\frac{\partial E_c(k)}{\partial w_{c_i}^{(2)}(k)} \right] \quad (4.27)$$

where $\eta_c(k)$ is the learning rate of the critic network at time instance k .

$$\frac{\partial E_c(k)}{\partial w_{c_i}^{(2)}(k)} = \frac{\partial E_c(k)}{\partial J(k)} \frac{\partial J(k)}{\partial w_{c_i}^{(2)}(k)} = \alpha e_c(k) \cdot p_i(k) \quad (4.28)$$

(ii) $\Delta w_c^{(1)}$: Critic network weights adjustment from input layer to hidden layer.

$$\Delta w_{c_i, j}^{(1)}(k) = \eta_c(k) \left[-\frac{\partial E_c(k)}{\partial w_{c_i, j}^{(1)}(k)} \right] \quad (4.29)$$

$$\begin{aligned} \frac{\partial E_c(k)}{\partial w_{c_i, j}^{(1)}(k)} &= \frac{\partial E_c(k)}{\partial J(k)} \frac{\partial J(k)}{\partial p_i(k)} \frac{\partial p_i(k)}{\partial q_i(k)} \frac{\partial q_i(k)}{\partial w_{c_i, j}^{(1)}(k)} \\ &= \alpha e_c(k) \cdot w_{c_i}^{(2)}(k) \cdot \frac{1}{2} (1 - p_i^2(k)) x_{cj}(k) \end{aligned} \quad (4.30)$$

[Algorithm 1]: ***Outline of Implementation of Dual Critic Network HDP***

/ * $\mathbf{s} \Leftarrow \text{RefNet}(\mathbf{x}, \mathbf{u}, \mathbf{w}_r)$, internal goal representation with the reference network;

RefNet: the reference network;

$\mathbf{w}_r$: weights of the *RefNet*;

$\mathbf{s}$: internal goal signal;

$\mathbf{J} \Leftarrow \text{CritNet}(\mathbf{x}, \mathbf{u}, \mathbf{s}, \mathbf{w}_c)$, total cost-to-go signal approximated by the critic network;

CritNet: the critic network;

$\mathbf{w}_c$: weights of the *CritNet*;

$\mathbf{J}$:total cost-to-go signal, the output of the critic network; */

- 1) **while** $\neg \text{TerminalContion}$ **do**
 - 2) Initiate $\mathbf{x}$, $\mathbf{w}_r$, $\mathbf{w}_c$;
 - 3) **repeat**
 - 4) Obtain the updated action $\mathbf{u}$ and apply to the system;
 - 5) Obtain the updated state $\mathbf{X}$ and immediate external reward r ;
 - 6) Update the error function E_r based on (4.15);
 - 7) Employ backpropagation rules (4.20)-(4.23) to minimize E_r till the *TerminalContion1*;
 - 8) Update the error function E_c based on (4.12);
 - 9) Employ backpropagation rules (4.27)-(4.30) to minimize E_c till the *TerminalContion2*;
 - 10) **until** *CurrentState* $\notin$ *threshold*
 - 11) **end while**
-

4.3.2 Interaction in Action Network and Tracking Filter

The control action is generated by the tracking filter as expressed in Fig.4.6(b). The action network here is to approximate the nonlinear system function $f(x(k))$ and the approximation error $\tilde{f}(x(k))$ is added in the error function of the action network. Note that $\tilde{f}(x(k))$ can not be obtained directly, since $f(x(k))$ is assumed unknown. The model network is commonly used in the ADP designs for tracking control [79, 80]. The same technique is applied to predict the state vector $\hat{x}(k+1)$ and get $\hat{e}(k+1)$. Then $\tilde{f}(x(k))$ can be got from equation 4.9.

The error function of the action network is defined as follows.

$$e_a = J(k) + \tilde{f}(k); \quad E_a(k) = \frac{1}{2}e_a^2(k). \quad (4.31)$$

The nonlinear system function $\hat{f}(x(k))$ can be obtained as follows.

$$\hat{f}(x(k)) = \frac{1 - \exp(-v(k))}{1 + \exp(-v(k))}. \quad (4.32)$$

$$v(k) = \sum_{i=1}^{N_{ah}} w_{a_i}^{(2)}(k)g_i(k). \quad (4.33)$$

$$g_i(k) = \frac{1 - \exp(-h_i(k))}{1 + \exp(-h_i(k))}. \quad i = 1, \dots, N_{ah} \quad (4.34)$$

$$h_i(k) = \sum_{j=1}^n w_{a_{i,j}}^{(1)}(k)x_{aj}(k). \quad i = 1, \dots, N_{ah} \quad (4.35)$$

where h_i and g_i are the input and output of the i th hidden node in the action network, v is the input for the output node. $\hat{f}$ is the output of the action network, N_{ah} is the total number of the hidden nodes in the action network. And x_{aj} is the input vector of the action network presented in Fig.4.6(a). Note that the weights tuning of the action network should consider the tracking filter as well.

(i) $\Delta w_a^{(2)}$: Action network weights adjustment from hidden layer to output layer.

$$\Delta w_{a_i}^{(2)}(k) = \eta_a(k) \left[-\frac{\partial E_a(k)}{\partial w_{a_i}^{(2)}(k)} \right] \quad (4.36)$$

where $\eta_a(k)$ is the learning rate of the action network at time instance k . And

$$\frac{\partial E_a(k)}{\partial w_{a_i}^{(2)}(k)} = \frac{\partial E_a(k)}{\partial J(k)} \frac{\partial J(k)}{\partial u(k)} \frac{\partial u(k)}{\partial v(k)} \frac{\partial v(k)}{\partial w_{a_i}^{(2)}(k)} \quad (4.37)$$

$$= e_a(k) \left[-\frac{1}{2} (1 - \hat{f}^2(k)) \right] g_i(k) \cdot \left\{ \sum_{i=1}^{N_{ch}} w_{c_i}^{(2)}(k) \left[\frac{1}{2} (1 - p_i^2(k)) \right] w_{c_{i,(n+1)}}^{(1)}(k) \right\}$$

(i) $\Delta w_a^{(1)}$: Action network weights adjustment from input layer to hidden layer.

$$\Delta w_{a_{i,j}}^{(1)}(k) = \eta_a(k) \left[-\frac{\partial E_a(k)}{\partial w_{a_{i,j}}^{(1)}(k)} \right] \quad (4.38)$$

$$\frac{\partial E_a(k)}{\partial w_{a_{ij}}^{(1)}(k)} = \frac{\partial E_a(k)}{\partial J(k)} \frac{\partial J(k)}{\partial u(k)} \frac{\partial u(k)}{\partial v(k)} \frac{\partial v(k)}{\partial g_i(k)} \cdot \frac{\partial g_i(k)}{\partial h_i(k)} \frac{\partial h_i(k)}{\partial w_{a_{ij}}^{(1)}(k)} \quad (4.39)$$

$$= \sum_{l=1}^{N_{ah}} \left(w_{c_l}^{(2)}(k) w_{c_{l,(n+1)}}^{(1)}(k) \left[\frac{1}{2} (1 - p_l^2(k)) \right] \right) e_a(k) \left\{ w_{a_i}^{(2)}(k) x_{aj}(k) \left[\frac{1}{2} (1 - \hat{f}^2(k)) \right] \left[\frac{1}{2} (1 - g_i^2(k)) \right] \right\} \quad (4.40)$$

Similar to [68, 77], the normalization of the weights will be employed during the learning and adaptation for all the networks used here. The weights are confined into proper range by

$$\mathbf{w}_r(k+1) = \frac{\mathbf{w}_r(k) + \Delta \mathbf{w}_r(k)}{a}, \quad \{a = \max(|a_{ij}|) | \forall a_{ij} \in \mathbf{w}_r(k) + \Delta \mathbf{w}_r(k)\} \quad (4.41)$$

Figure 4.6. (a) The neural network structure of the action network; (b) The control action generator.

$$\mathbf{w}_c(k+1) = \frac{\mathbf{w}_c(k) + \Delta \mathbf{w}_c(k)}{b},$$

$$\{b = \max(|b_{ij}|) | \forall b_{ij} \in \mathbf{w}_c(k) + \Delta \mathbf{w}_c(k)\} \quad (4.42)$$

$$\mathbf{w}_a(k+1) = \frac{\mathbf{w}_a(k) + \Delta \mathbf{w}_a(k)}{c},$$

$$\{c = \max(|c_{ij}|) | \forall c_{ij} \in \mathbf{w}_a(k) + \Delta \mathbf{w}_a(k)\} \quad (4.43)$$

4.4 Simulation Studies

In this section, two numerical simulations are conducted based on the same system function. The motivation is to compare the tracking performance between the proposed approach and the typical HDP approach in [51], which is originally from [68]. The system function is defined with the general nonlinear form as follows.

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= f(x(k)) + u(k) + d(k) \end{aligned} \quad (4.44)$$

where

$$f(x(k)) = -\frac{4}{11} \cdot \left(\frac{x_1}{1+x_2^2} \right) + \frac{2}{5} x_2(k) \quad (4.45)$$

where $f(x(k))$ is assumed to be unknown in the tracking process. Instead, the approximation value $\hat{f}(x(k))$ can be obtained from the action network. And $d(k)$ is the disturbance here.

4.4.1 Tracking Control Problem with Disturbance

The objective for this example is to track the sinusoid signal with some harmonic signals on x_2 . The desired signal function is defined as $x_{2d} = \sin(\omega kT)\cos(2\omega kT + \tau)$, where $\omega = 0.2\text{rad/s}$ and $\tau = \pi/2$. Set the sample interval $T = 50\text{ms}$ and the total simulation time here to be 150s . In literature, people normally add noise or disturbance in the simulation to see how robustness of the proposed approach can be, like in [76, 40, 46]. The similar techniques as that in [76] are adopted. Disturbance $d(k) = 1.5$ is introduced at $k = 1200$, corresponding to $t = 60\text{s}$. Otherwise, it is set to be 0. The input vector X is defined as $X_{in}(k) = [e(k-1) \ e(k) \ x_{2d}(k-1) \ x_{2d}(k)]$, where $e(k) = x_2(k) - x_{2d}(k)$. The structures of the action network, reference network and critic network are $4 - 4 - 1$ (i.e. the network has 4 input nodes, 4 hidden nodes and 1 output node), $5 - 4 - 1$, and $6 - 4 - 1$, respectively. The parameters we used in the simulation are summarized as in Table 4.1.

Table 4.1. Summary of the parameters used in the simulation study one

Para.	η_c	η_a	η_r	Kv	λ	c	*
value	$5e-3$	$8e-3$	$2.5e-3$	0.1	0.2	$1.5e-3$	*
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
value	40	150	50	$1e-4$	$1e-5$	$1e-5$	0.95

where

α : discount factor;

c : threshold on mean square error;

η_c : initial learning rate of the critic network;

η_a : initial learning rate of the action network;

η_r : initial learning rate of the reference network;

N_c : internal cycle of the critic network;

N_a : internal cycle of the action network;

N_r : internal cycle of the reference network;

T_c : internal training error threshold for the critic network;

T_a : internal training error threshold for the action network;

T_r : internal training error threshold for the reference network;

The learning rates will drop once the tracking performance is “good” over time. Specifically, the mean square error (MSE) as expressed in (4.46) is first compared with a certain threshold c

$$MSE = \frac{1}{N} \sum_{i=1}^N (x - x_d)^2 \quad (4.46)$$

where x is the state vector, x_d is the desired tracking signal, and N is a preset integer.

If $MSE < threshold$, then the learning rate will be divided by a certain number and the new *threshold* can be calculated by dividing another certain number correspondingly. This kind of evaluation will repeat during the whole process of the tracking control. The detailed implementation are presented in *Algorithm2* from line 24 to 32.

Figure 4.7. The typical tracking performance with our proposed approach.

Figure 4.8. The typical tracking performance with HDP approach

For comparative study, the simulation with both approaches under the same parameters and environment settings are provided. The weights in the neural networks used in both approaches are randomly selected from $[-1, 1]$. The starting point of the state vector is $(0, 1.5)$, which is the same for both approaches. The typical tracking performance with both the proposed approach and the typical HDP approach are presented in Fig.4.7 and Fig.4.8, respectively. From Fig.4.7, it is clear to see that the tracking signal (solid line) can exactly follow the desired signal (dash line) within 1s. In addition, the tracking signal can also quickly go back to the “right track” after the disturbance at 60s. On the other hand, the tracking signal with the typical HDP approach can only follow the desired signal after 10s. Moreover, the tracking signal need more time (13s) to go back to track the desired signal after the disturbance. From this example, the proposed approach not only shows faster learning process than the typical HDP approach, but also shows better robustness for disturbance.

4.4.2 Adaptive Signal Tracking Control Problem

In order to show the adaptiveness of the proposed approach, another numerical example is conducted to track the signal that would change from saw signal to square signal and finally to sinusoid signal. The same system function and the environment settings are used as above, except for $d(k)$ is set to be white Gaussian noise with a standard deviation of 0.005 for all k . The objective is to track the desired signal with the state vector x_2 . The desired tracking signal is defined as

$$x_{2d} = \begin{cases} A \cdot (1 - |t - \frac{T_0}{2}|), & 0 < t < T_0 \\ A \cdot (1 - |t - \frac{3T_0}{2}|), & T_0 < t < 2T_0 \\ -A, & 2T_0 < t < 3T_0 \text{ or } 4T_0 < t < 5T_0 \\ A, & 3T_0 < t < 4T_0 \text{ or } 5T_0 < t < 6T_0 \\ A \cdot \sin(\omega t), & 6T_0 < t < 7.5T_0 \end{cases} \quad (4.47)$$

where $t = kT$, k is the step number and T is the sample time ($T = 50ms$). And $A = 0.95$ is the amplitude of the signal. $T_0 = 40s$ is taken as the time internal that each signal last (i.e. the signal will change after T_0).

The difficulty of this task is that the controller need to learn to track the desired signal, which will change over time, under the white Gaussian noise. Fig.4.9 shows the typical tracking performance with the proposed approach and one can clearly see the good transient tracking performance when the signal changes. While in Fig. 4.10, the HDP controller takes about 50s to learn to follow the saw signal and also spends much time to learn when the desired signal change to a rectangular signal. In addition, this traditional controller also requires about 20s to catch up after that the desired signal changes to sinusoid signal. This phenomenon indicates that the proposed approach shows better adaptiveness on this tracking problem than the typical HDP approach under noisy condition.

Figure 4.9. The typical tracking performance with our proposed approach.

4.4.3 Real-Time Tracking Control for Ball and Beam System

Instead of testing on two numerical cases, the proposed approach is tested on a continues benchmark of ball-and-beam tracking problem [81, 82]. There are many versions of this benchmark and here the model in Fig.4.11 is adopted. The system contains a long beam that can be tilted by a servo or electric motor with a ball rolling back and forth on the top of the beam. In this system, the driver is located in the center of the beam. The angle of the beam to the horizontal axis is measured by an incremental encoder and the position of the ball can be obtained with the cameras mounted on the top of system. This proposed approach will learn to track the desired signal with the position of the ball.

From [81], the motion equations from Lagrange equation is obtained as following:

$$(m + \frac{I_b}{r^2})\ddot{x}' + (mr^2 + I_b)\frac{1}{r}\ddot{\alpha} - mx'\dot{\alpha}^2 = mg(\sin \alpha) \quad (4.48)$$

$$[m(x')^2 + I_b + I_\omega]\ddot{\alpha} + (2m\dot{x}'x' + bl^2)\dot{\alpha} + Kl^2\alpha + (mr^2 + I_b)\frac{1}{r}\ddot{x}' - mgx'(\cos \alpha) = ul(\cos \alpha) \quad (4.49)$$

where

Figure 4.10. The typical tracking performance with HDP approach.

$m : 0.0162kg$, the mass of the ball;

$r : 0.02m$, the roll radius of the ball;

$I_b : 4.32 \times 10^{-5}kg \cdot m^2$, the inertia moment of the ball;

$b : 1N_s/m$, the friction coefficient of the drive mechanics;

$l : 0.48m$, the radius of force application;

$l_\omega : 0.5m$, the radius of beam;

$K : 0.001N/m$, the stiffness of the drive mechanics;

$g : 9.8N/kg$, the gravity;

$I_\omega : 0.14025kg \cdot m^2$, the inertia moment of the beam;

u : the force of the drive mechanics;

In order to simplify the system model function, it is defined that $x_1 = x'$ represents the position of the ball, $x_2 = \dot{x}'$ represents the velocity the ball, $x_3 = \alpha$ is the angle of the beam with respect to the horizontal axis, and $x_4 = \dot{\alpha}$ is the angular velocity of the beam. Therefore, the state vector can be defined as $X = [x_1 \ x_2 \ x_3 \ x_4]$. In this way, the system function (4.48) and (4.49) can be transformed

Figure 4.11. A schematic diagram of ball-and-beam system

into the following forms:

$$\left(m + \frac{I_b}{r^2}\right)\dot{x}_2 + (mr^2 + I_b)\frac{1}{r}\dot{x}_4 = mx_1x_4^2 + mg(\sin x_3) \quad (4.50)$$

$$\begin{aligned} (mr^2 + I_b)\frac{1}{r}\dot{x}_2 + [mx_1^2 + I_b + I_\omega]\dot{x}_4 &= (ul + mgx_1) \cos x_3 \\ &\quad - (2mx_2x_1 + bl^2)x_4 - Kl^2x_3 \end{aligned} \quad (4.51)$$

To be more clear, it is written in equations (4.50) and (4.51) with the specific value of all the parameters mentioned above into the approximate nonlinear state-space equations as following:

$$\dot{x}_1 = x_2 \quad (4.52)$$

$$\dot{x}_2 = 1.717 \sin(x_3) \quad (4.53)$$

$$\dot{x}_3 = x_4 \quad (4.54)$$

$$\dot{x}_4 = -0.241x_4 + 0.157x_1 \cos(x_3) + 0.5 \cos(x_3) \cdot u \quad (4.55)$$

The objective is to track the sinusoid signal $x_{1d} = 0.1 \sin(\omega t)$ with the position of ball (x_1), where $\omega = 0.1$. This task requires the controller to not only keep the balance of the ball on the beam, but also track the desired signal using the position of the ball (x_1). That is to say, if x_1 is out of bound ($[-0.48, 0.48]m$), or

x_3 exceed the angular velocity tolerance ($[0.24, 0.24]rad/s$), we will reset the ball to the initial starting point ($[0 \ 0 \ 0 \ 0]$). Since the learning process of the neural network is continuous, it is assumed that the weights can be carried on when the task is reset. Interested readers may also find that this is a continuous-time benchmark rather the discrete-time case above. As discussed in existing literature [68, 77, 62], the continuous time system model can be called by *ode45* function in Matlab with the step size $0.02s$. The parameters used in this case are summarized in the Table 4.2, and 5% uniform noise is also added on the sensor of x_1 to show the tracking performance under noisy condition.

Table 4.2. Summary of the parameters used in the simulation study two

Para.	η_c	η_a	η_r	Kv	λ	c	*
value	0.02	0.01	0.05	0.05	0.1	$5e-5$	*
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
value	100	200	200	$1e-4$	$1e-5$	$1e-5$	0.95

Figure 4.12. The typical tracking performance with our proposed approach.

Fig.4.12 and Fig.4.13 present the tracking performance with the proposed approach and the typical HDP approach, respectively. Fig.4.12 clearly shows that the ball is out of bound at the very beginning, but can quickly tracking the desired

Figure 4.13. The typical tracking performance with HDP approach

signal within one period. The oscillation of the tracking signal (solid line) in the first period shows the learning process of the controller with the proposed approach. While for Fig.4.13, one can see that the ball is out of bound for many times before it can track the desired signal. In other words, the controller spends the whole first period to learn to control the ball and it learns after about 80s. The simulation results show that the controller with our proposed approach has better noise tolerance than that with the typical HDP approach.

To provide more accurate assessment of the tracking performance, I summarize the quantitative measurements in terms of tracking error for the first two examples and the third example (i.e., the ball and beam benchmark). The evaluation function in [79] is defined as $PER = \sum_0^N e^T(k)e(k)$, where $e(k)$ is the tracking error in Fig.4.3 and N refers the number of step in the simulation. The PER for 4.4.1, 4.4.2 and 4.4.3 are summarized in Table 4.3.

Table 4.3. Summary of total tracking error for the simulation cases

number	<i>example1</i>	<i>example2</i>	<i>ballandbeam</i>
GrHDP	7.454	9.292	3.761
HDP	223.9	712.8	54.49

From the results in this table, the proposed GrHDP (dual-critic design) can achieve much lower PER (total tracking error) [65, 83, 84] than that with typical HDP approach. The results also confirm that the proposed structure with the informative and adaptive reinforcement signal can outperform the typical HDP structure in terms of tracking accuracy.

4.4.4 Virtual Reality Demonstration with Unknown Disturbances

The proposed approach is further applied for the algorithm on virtual reality (VR) environment to show real-time simulation of the approach with the interaction of environment. VR can enable powerful human-computer interactions, and it is interesting to observe how the proposed algorithm works on the real time simulation without the requirement of setting up of the real physical system [85, 86, 87, 88]. Here I would like to demonstrate the tracking performance of the proposed approach on the ball and beam benchmark [89, 90, 91]. Also the disturbances are added to see how robust can the proposed approach perform.

The VR platform is developed as in Fig.4.14, where one can see that the ball and beam system is in the center of the scene and the state vectors (x_1 and x_3) are displayed in the upper-left table of the figure. In order to be more realistic, I add a disturbance option (upper-center) in the simulation. That is to say that the user can apply whatever disturbance between $-8N$ to $8N$ onto the ball whenever they want, and the corresponding force will be applied to the system and also displayed on the upper-right table.

Real time simulation result on the tracking problem with our proposed approach is presented in Fig.4.15, where one can clearly see the online learning process. In other words, the first period of signal is like a distorted sin wave while the signal in second period almost track the desired sin signal. At about $64s$, we added a $2.1N$ disturbance on the ball and we can see that the controller spent about two

periods to learn to get back to the right track.

Figure 4.14. The schematics of virtual reality demonstration platform.

Figure 4.15. The typical tracking performance with the proposed approach in VR/Simulation platform

4.5 Summary

This Chapter introduces a novel ADP structure that integrates a reference (goal) network into the original critic network, and thus formulates a dual critic network design. The reference network provides the critic network with an internal

goal representation, which helps to approximate the total cost-to-go signal over time. Unlike the predefined reward signal from the (external) environment, this internal goal signal can be adjusted adaptively with regard to the system state and the control action online. Compared with the typical HDP design under the same simulation environment settings, the proposed GrHDP approach can achieve better tracking performance in terms of learning time and the accumulated tracking error. A real-time simulation is also demonstrated in a VR platform. In the next chapter, the goal representation technique is further applied on advanced ADP designs.

CHAPTER 5

Goal Representation Design for Dual Heuristic Dynamic Programming (GrDHP) Architecture

5.1 Introduction

A general utility function representation (also called reward or reinforcement signal in the previous chapters) is proposed to provide the required derivable and adjustable utility function for the dual heuristic dynamic programming (DHP) design. Traditionally, the DHP design requires to define the engineering designed utility function offline and conduct the derivation manually [59, 92, 93]. Here, the goal network is used to provide this general mapping between the system states and the derivatives of the utility function automatically. With this proposed architecture, the required derivatives of the utility function can be directly obtained from the goal network. Thus, this architecture is called goal presentation DHP (GrDHP) design [94, 95]. The control performance of both the GrDHP and the traditional DHP approaches is provided under the same environment and parameter settings. The statistical simulation results and the snapshot of the system variables are presented to demonstrate the improved learning and controlling performance.

5.2 Design of the GrDHP Framework

The conceptual diagram of the GrDHP approach is presented in Fig.5.1. The dash line in the middle is used to distinguish the typical DHP architecture from the proposed goal representation network. From this figure, one can see that the action network observes the system states and provides the control action for the dynamic system/environment. Compared with the traditional DHP design, the goal network can directly provide the *required* derivatives of the utility function

for the critic network, which will in turns, help the system's decision making process in the action network.

In this design, we apply a model network to predict the future system states. Subsequently, the predicted partial derivatives of both the value function and the internal goal can be obtained at the current time step. Therefore, we can achieve the temporal difference (TD) errors for the derivatives of both the value function and the internal goal signal between the current time step and the future time step. In addition, the model network is also used to connect the critic/goal and the action networks, so that the backpropagation paths can be realized. We note that the building and the training of model network are the same as that in literature [96, 12, 65].

Figure 5.1. The conceptual diagram of the GrDHP architecture. Solid arrows refer to the signal paths, while the dash arrows refer to the backpropagation paths.

Fig. 5.2 is provided to show the learning process of the GrDHP design, where λ represents the partial derivatives of the value function and g represents the partial

Figure 5.2. The learning flowchart in the GrDHP design. The neural networks are represented with their corresponding weights and these weights are carried on throughout the learning process. The subscript number under each parameter (e.g., r) refers to time step.

derivatives of the internal utility function. The signal flowing steps are described as following:

1. At $t = 1$, x_1 is observed from the system/environment and u_1 can be obtained through the action network. Then λ_1 and g_1 are obtained through the critic network and the goal network, respectively;
2. The predicted system state $\hat{x}_2$ is provided by the model network, with the input x_1 and u_1 ;
3. The subsequently predicted $\hat{\lambda}_2$ and $\hat{g}_2$ are obtained through the critic and the goal networks, respectively;
4. The external reward/utility r_1 is given based on the system state. If the system fails, another run will be started from step 1);

5. The temporal difference errors are obtained between time step 1 and 2 (e.g., λ_1 and $\hat{\lambda}_2$). The weights in the goal, the critic and the action networks are updated accordingly;
6. At $t = 2$, x_2 is observed from the output of the system. The neural network weights are carried on from the last time step to this time step;
7. The signal u_2 , x_2 and their estimated values for the next step are obtained through the similar paths as those in $t = 1$. The weights are updated through the temporal difference errors accordingly;
8. The learning process will be terminated when the time step satisfies the maximum requirement.

In the GrHDP design [62, 4], the goal network is introduced to provide an internal reinforcement/goal signal for the critic network. In contrast with the traditional HDP design, this internal reinforcement signal can be adaptively tuned over time rather than the fixed discrete reward values (“0” or “-1”) or other predefined reward functions. The theoretical assurance and the stability analysis of this GrHDP design were investigated and demonstrated in [73, 97, 74]. Following this trend, the goal network is integrated into the DHP design. Within this GrDHP structure, the goal network is used to approximate the *required* partial derivatives of the internal reinforcement/utility signal for the critic network. Meanwhile, the critic network is still employed to approximate the partial derivatives of the value function. The same as that in previous chapters, the outputs of the goal network are feeded into the critic network, so that they can help the approximation of the derivatives of the value function over time. In this design, a neural network is used to implement the general utility function representation (i.e., the goal network). Moreover, the derivatives of utility function (i.e., the outputs of the goal network)

Figure 5.3. The proposed architecture of the GrDHP design. Solid/Bolded arrows refer to the signals/vectors and dash arrows refer to the weights tuning paths. $\frac{\partial r(t)}{\partial x(t)}$ and $\frac{\partial r(t)}{\partial u(t)}$ refer to the partial derivatives of the external reward/utility w.r.t the system state $x(t)$ and the control action $u(t)$, respectively. The block DER implements (5.9).

are updated over time. Note that although the goal and the critic network look similar, they are conducting different approximation and online learning process.

5.3 Learning Process of the GrDHP Design

Fig. 5.3 presents the schematic diagram of this GrDHP design. The error function and the weights tuning path for each neural network are included in the figure. The inputs for the critic network include the outputs from the goal network, so that the goal network and the critic network can be closely connected. The outputs for the critic network are still the partial derivatives of the value function with respect to (w.r.t.) the system states and the control action. In the following of this section, the designs of the goal, the critic and the action networks are provided with the error function and the weights tuning rule, respectively.

5.3.1 Design of Goal Representation in HDP

Goal representation design has been introduced into the ADP field in recent years. It has been demonstrated in Chapter 4 and 3 that an integrated reference network (also called the goal network) could help to improve the control performance, compared with the traditional HDP design. The reference/goal network is assigned to learn from the external reinforcement signal and provides the critic network with detailed internal reinforcement signal. In addition, this internal reinforcement signal can be adaptively tuned by the goal network over time. More discussions and simulation studies about this internal reinforcement signal are provided in [73, 97, 98]. GrHDP is thus named for this integration of the goal network onto the traditional HDP framework.

Among these papers, the internal reinforcement signal is generally defined as $s(t) = f(x(t), u(t))$, and the value function is defined as $J(t) = f(x(t), u(t), s(t))$. x and u are the system state and the control action, respectively. s represents the internal goal, which is also the output of the goal network. In the model-free HDP design, the error function of the goal network is defined as

$$e_g(t) = \alpha s(t) - [s(t-1) - r(t)] \quad (5.1)$$

and the error function of the critic network is defined as

$$e_c(t) = \alpha J(t) - [J(t-1) - s(t)] \quad (5.2)$$

where α is the discount factor. The objective for the action network is to provide the control action that can minimize the total cost over time. We note that equations (5.1) and (5.2) are based on the model-free GrHDP design. While for the model-base GrDHP design, the error functions are usually based on the temporal difference errors between t and $t+1$.

5.3.2 Design of General Utility Representation in DHP

To keep the consistence of the previous chapters, the similar terminology is used here. The goal network is adopted to approximate the required partial derivatives of the internal reinforcement signal w.r.t the system variables and the control action. These partial derivatives can directly contribute to the error function of the critic network. The outputs of the goal network are defined as

$$\mathbf{g}(t) = \frac{\partial s(t)}{\partial \mathbf{Y}(t)} = \begin{bmatrix} \frac{\partial s(t)}{\partial \mathbf{x}(t)} & \frac{\partial s(t)}{\partial \mathbf{u}(t)} \end{bmatrix}. \quad (5.3)$$

where $\mathbf{Y}(t) = [x(t) \quad u(t)]^T$. Thus, $\mathbf{g}(t)$ can also be written as

$$\mathbf{g}(t) = [\mathbf{g}^x(t) \quad \mathbf{g}^u(t)]. \quad (5.4)$$

where $\mathbf{g}^x(t) = [\frac{\partial s(t)}{\partial x_1(t)} \quad \frac{\partial s(t)}{\partial x_2(t)} \quad \dots \quad \frac{\partial s(t)}{\partial x_M(t)}]$ and $\mathbf{g}^u(t) = [\frac{\partial s(t)}{\partial u_1(t)} \quad \frac{\partial s(t)}{\partial u_2(t)} \quad \dots \quad \frac{\partial s(t)}{\partial u_N(t)}]$. M and N refer to the dimensions of the x and u , respectively. The error function for the goal network is defined as

$$\mathbf{e}_g(t) = \frac{\partial s(t)}{\partial \mathbf{Y}(t)} - \alpha \frac{\partial s(t+1)}{\partial \mathbf{Y}(t)} - \frac{\partial r(t)}{\partial \mathbf{Y}(t)} \quad (5.5)$$

and

$$\mathbf{E}_g(t) = \frac{1}{2} \mathbf{e}_g^T(t) \mathbf{e}_g(t) \quad (5.6)$$

Equation (5.5) is actually obtained from equation (5.1) (by taking the partial derivatives w.r.t. to the input $\mathbf{Y}$). Here the $s(t)$ signal is the same internal reinforcement signal as that in [73, 97]. $\partial(\cdot)/\partial \mathbf{Y}(t)$ is a vector including partial derivatives of the scalar $(\cdot)$ w.r.t. the components in $\mathbf{Y}$. The online learning of the goal network is to minimize the squared error $\mathbf{E}_g(t)$ by updating the weights with the gradient descent method as

$$\boldsymbol{\omega}_g(t+1) = \boldsymbol{\omega}_g(t) + \Delta \boldsymbol{\omega}_g(t) \quad (5.7)$$

$$DER = \begin{bmatrix} \frac{\partial \hat{x}(t+1)}{\partial u(t)} \cdot \frac{\partial u(t)}{\partial x(t)} + \frac{\partial \hat{x}(t+1)}{\partial x(t)} & \frac{\partial \hat{x}(t+1)}{\partial u(t)} \\ \frac{\partial \hat{u}(t+1)}{\partial \hat{x}(t+1)} \cdot \left(\frac{\partial \hat{x}(t+1)}{\partial u(t)} \cdot \frac{\partial u(t)}{\partial x(t)} + \frac{\partial \hat{x}(t+1)}{\partial x(t)} \right) & \frac{\partial \hat{u}(t+1)}{\partial \hat{x}(t+1)} \cdot \frac{\partial \hat{x}(t+1)}{\partial u(t)} \end{bmatrix}. \quad (5.9)$$

where

$$\begin{aligned} \Delta \omega_g(t) &= \eta_g \left[-\frac{\partial \mathbf{E}_g(t)}{\partial \omega_g(t)} \right] \\ &= \eta_g \left[-\left(\frac{\partial s(t)}{\partial \mathbf{Y}(t)} - \alpha \frac{\partial s(t+1)}{\partial \mathbf{Y}(t)} - \frac{\partial r(t)}{\partial \mathbf{Y}(t)} \right) \right. \\ &\quad \left. \cdot \frac{\partial s^2(t)}{\partial \mathbf{Y}(t) \partial \omega_g(t)} \right] \end{aligned} \quad (5.8)$$

where ω_g and η_g refer to the weights and the learning rate in the goal network, respectively. The model network is employed to predict the future system state, so that the $\hat{u}(t+1)$ and the $\hat{\mathbf{g}}(t+1)$ can be obtained subsequently. The forward temporal difference error can thus be achieved to update the weights in the goal network. In Fig.5.3, the block “DER” is adopted to realize the error function for both the goal and the critic networks and it is defined as in equation (5.9). Therefore, one can realize $\frac{\hat{s}(t+1)}{\mathbf{Y}(t)}$ in (5.5) with matrix calculation as

$$\frac{\partial \hat{s}(t+1)}{\partial \mathbf{Y}(t)} = \hat{\mathbf{g}}(t+1) \cdot DER. \quad (5.10)$$

Substituting (5.10) into (5.5), the weights updating rules can be obtained in (5.8) and thus the learning in the goal network can be conducted.

5.3.3 Online Learning in Critic Network

As discussed in Chapter 4, the critic network was designed to approximate the discounted total internal reward/cost-to-go and the error function for the critic network was defined with TD error backward (i.e., with time instance t and $t-1$) as (5.2). The third term in this error function is internal reward signal s rather than external reward signal r in literature.

For the GrDHP design here, the similar error function is defined for the critic

network with TD error forward (i.e., with time instance t and $t + 1$) as

$$\mathbf{e}_c(t) = \frac{\partial J(t)}{\partial \mathbf{Y}(t)} - \alpha \frac{\partial J(t+1)}{\partial \mathbf{Y}(t)} - \frac{\partial s(t)}{\partial \mathbf{Y}(t)} \quad (5.11)$$

and

$$\mathbf{E}_c(t) = \frac{1}{2} \mathbf{e}_c^T(t) \mathbf{e}_c(t) \quad (5.12)$$

The online learning of the critic network is to minimize the squared error $\mathbf{E}_c(t)$ by updating the weights with the gradient descent as

$$\boldsymbol{\omega}_c(t+1) = \boldsymbol{\omega}_c(t) + \Delta \boldsymbol{\omega}_c(t) \quad (5.13)$$

where

$$\begin{aligned} \Delta \boldsymbol{\omega}_c(t) &= \eta_c \left[-\frac{\partial \mathbf{E}_c(t)}{\partial \boldsymbol{\omega}_c(t)} \right] \\ &= \eta_c \left[-\left(\frac{\partial J(t)}{\partial \mathbf{Y}(t)} - \alpha \frac{\partial J(t+1)}{\partial \mathbf{Y}(t)} - \frac{\partial s(t)}{\partial \mathbf{Y}(t)} \right) \right. \\ &\quad \left. \cdot \frac{\partial J^2(t)}{\partial \mathbf{Y}(t) \partial \boldsymbol{\omega}_c(t)} \right] \end{aligned} \quad (5.14)$$

where $\boldsymbol{\omega}_c$ and η_c refer to the weights and the learning rate of the critic network, respectively. The outputs for the critic network are defined as

$$\boldsymbol{\lambda}(t) = \frac{\partial J(t)}{\partial \mathbf{Y}(t)} = \begin{bmatrix} \frac{\partial J(t)}{\partial \mathbf{x}(t)} & \frac{\partial J(t)}{\partial \mathbf{u}(t)} \end{bmatrix} = [\boldsymbol{\lambda}^x(t) \quad \boldsymbol{\lambda}^u(t)]. \quad (5.15)$$

where $\boldsymbol{\lambda}^x(t) = [\frac{\partial J(t)}{\partial x_1(t)} \quad \frac{\partial J(t)}{\partial x_2(t)} \quad \dots \quad \frac{\partial J(t)}{\partial x_M(t)}]$ and $\boldsymbol{\lambda}^u(t) = [\frac{\partial J(t)}{\partial u_1(t)} \quad \frac{\partial J(t)}{\partial u_2(t)} \quad \dots \quad \frac{\partial J(t)}{\partial u_N(t)}]$.

As the first and the third term in (5.11) can be directly obtained from the critic network and the goal network respectively, here I only provide the formula to obtain the second term in (5.11). The second term in (5.11) can be calculated in the similar way using block “DER”. Thus, the $\frac{\partial \hat{J}(t+1)}{\partial \mathbf{Y}(t)}$ can be obtained as

$$\frac{\partial \hat{J}(t+1)}{\partial \mathbf{Y}(t)} = \hat{\boldsymbol{\lambda}}(t+1) \cdot DER. \quad (5.16)$$

Substituting (5.16) into (5.14), the weight updating rules in the critic network can also be completed.

5.3.4 Online Learning in Action Network

In the traditional DHP design, the error function of the action network is usually defined as $e_a(t) = \frac{\partial r(t)}{\partial u(t)} + \alpha \frac{\partial \hat{J}(t+1)}{\partial u(t)}$ [99, 12, 100, 101]. Since the goal network is used to provide the partial derivative of the internal goal function with respect to the input vector, $\frac{\partial r(t)}{\partial u(t)}$ can be replaced with part of the output from goal network as $\frac{\partial s(t)}{\partial u(t)}$. Therefore, the error function for action network is defined as

$$\mathbf{e}_a(t) = \frac{\partial s(t)}{\partial u(t)} + \alpha \frac{\partial \hat{J}(t+1)}{\partial u(t)} \quad (5.17)$$

$$\mathbf{E}_a(t) = \frac{1}{2} \mathbf{e}_a^T(t) \mathbf{e}_a(t) \quad (5.18)$$

and the gradient descent method is used to tune the weights in the action network as

$$\boldsymbol{\omega}_a(t+1) = \boldsymbol{\omega}_a(t) + \Delta \boldsymbol{\omega}_a(t) \quad (5.19)$$

where

$$\begin{aligned} \Delta \boldsymbol{\omega}_a(t) &= \eta_a \left[-\frac{\partial \mathbf{E}_a(t)}{\partial \boldsymbol{\omega}_a(t)} \right] \\ &= \eta_a \left[-\left(\frac{\partial s(t)}{\partial u(t)} + \alpha \frac{\partial \hat{J}(t+1)}{\partial u(t)} \right) \right. \\ &\quad \left. \cdot \frac{\partial \hat{J}^2(t+1)}{\partial u(t) \partial \boldsymbol{\omega}_a(t)} \right] \end{aligned} \quad (5.20)$$

where $\boldsymbol{\omega}_a$ and η_a refer to the weights and the learning rate of the action network, respectively. For $\frac{\partial s(t)}{\partial u(t)}$ in (5.17), it can be directly obtained from the goal network as

$$\frac{\partial s(t)}{\partial u(t)} = \mathbf{g}^u(t). \quad (5.21)$$

Since $\frac{\partial \hat{J}(t+1)}{\partial u(t)}$ can be obtained from (5.16), the weights tuning in the action network is completed.

5.4 Simulation Studies

In this section, two well-known study cases are conducted for both the GrDHP and the traditional DHP approaches. For fair comparisons, both approaches are

tested based on the same environment and parameter settings. The external reward/utility function is set as the quadratic form that $r = x^T Q x + u^T R u$, where Q and R are the identity matrices with appropriate dimensions. The goal network will learn the mapping (goal representation) between the system state and the partial derivatives of the internal utility function.

5.4.1 Ball and Beam Balancing Example

The system function of the ball and beam and the parameters used in this study are identical as those in Chapter 4. The simulations are based on 100 runs with a maximum of 1000 consecutive trials in each run. It would be considered successful if the last trial (trial number less than 1000) of the run lasted 10000 time steps. That is to say that, if the controller is unable to learn to balance the ball on the beam for 10000 time steps for any of the 1000 consecutive trials, this run is regarded as unsuccessful. The ball is considered fallen if the x_1 is running out of $[-0.48m, 0.48m]$ or if x_3 is running out of $[-0.24rad, 0.24rad]$. A starting force $F = 10N$ is applied on the system at $t = 1$. The ball will be balanced on the beam with the control force provided by the action network thereafter. In addition, the initial values of x_1 and x_3 are uniformly distributed in $[-0.2m, 0.2m]$ and $[-0.15rad, 0.15rad]$, respectively. Different uniform noises are added to the system state (i.e., observed system state) as follows: let x be the observed state variable. If the noise level is 5%, then the sampled state value that fed into the learning controller should be $x + 0.05x * random(-1, 1)$.

The multi-layer perceptron (MLP) structure for the action, the critic and the goal networks are 4-8-1 (i.e., the neural network has 4 input nodes, 8 hidden nodes and 1 output node), 10-18-5, 5-10-5, respectively. The weights in the goal, the critic and the action networks are updated until they satisfy with their internal criteria N_g/T_g , N_c/T_c , N_a/T_a , respectively. For example, the action network is

updated at most N_a cycles in each time step until the squared error (objective function in (5.18)) is tuned under the threshold T_a . In this case study, we set $N_g = 80$, $N_c = 80$ and $N_c = 100$ and the squared error thresholds are set as $T_g = T_c = T_a = 1e-6$. The discounted factor is set as $\alpha = 0.95$. The learning rate for the action network is initialized as $\eta_a(0) = 0.3$ and will be dropped 0.05 every 10 steps. The value of this learning rate is reset to 0.005 if it is a non-positive value after dropping. It is assumed that the goal and the critic networks have the same settings as those in the action network. Moreover, we use the same training method for the goal, the critic and the action networks and test their final learned policies online [102, 101].

Figure 5.4. Comparison of the statistical results on the ball-and-beam balancing task with both the GrDHP and the DHP approaches.

Simulation results are presented in Fig. 5.4. Both the GrDHP and the DHP approaches can obtain 100% successful rate to balance the ball on the beam under various noise conditions. Comparing with the DHP approach, the GrDHP approach requires less average number of trials to success under the same noise setting. This may indicate that the GrDHP approach can improve the control performance on this task in terms of the required average number of trials. In order

Figure 5.5. Typical trajectories of the state vectors in the first 2000 time steps in a typical successful trial.

to numerically analyze the learning time for both approaches, we measure the time cost for the backpropagation calculation in each successful run/trial (i.e., we only count the backpropagation calculation time in the neural networks in the successful run/trial and take the average value). The GrDHP approach has three types of neural network and requires $0.0102ms$ to generate each action per time step, while the DHP approach has two types of neural network and needs $0.0090ms$ per action generation (the simulations are conducted in Matlab 2013a based on Sun Server with 16GB memory, Intel Xeon CPU, 3.60GHz). Fig.5.5 shows the trajectories of the state vectors in a typical successful trial with the GrDHP approach. That the amplitude of the $x_1(m)$ and $x_3(rad)$ gradually converge to zero shows the learning and controlling during the first 2000 time steps of the simulation. In addition, Fig.5.6 indicates the learning process in the goal network and also shows the convergence of the weights in a typical successful trial. From the simulation results, the GrDHP approach shows promising performance in the learning and controlling process.

Figure 5.6. The weights evolution of the goal network in the first 1000 steps in a typical successful trial.

5.4.2 Triple-link Inverted Pendulum Balancing Example

The triple-link inverted pendulum system model is the same as that in Chapter 3. The simulations are based on 100 runs with a maximum of 3000 consecutive trials in each run. It would be considered successful if the last trial (trial number less than 3000) of the run lasted 10000 time steps. This indicates that if the controller is unable to learn to balance the triple-link inverted pendulum on the cart for 10000 time steps for any of the 3000 consecutive trials, this run is regarded as unsuccessful. The pendulum is considered fallen if the x is running out of $[-0.1m, 0.1m]$ or if any of θ is running out of $[-0.35rad, 0.35rad]$. The MLP structures for the goal, the critic and the action network are 9-14-9 (i.e., the neural network has 9 input nodes, 14 hidden nodes and 9 output node), 18-24-9, and 8-14-1, respectively. The only control u generated by the action network is converted into a force by an analog amplifier through a conversion gain K_s (in Newton/volt). In this simulation, $K_s = 24.7125$ N/V. We assume that each link could only rotate in the vertical plane and the position sensor is fixed to the top of each link. For the other parameter settings, we keep the same as those in section 5.4.1.

Figure 5.7. Typical trajectories of the angles in a successful trial under the condition of noise free.

Simulation results are summarized in Table 5.1. Under different types of noise, the GrDHP approach can outperform the DHP approach in terms of both the successful rate and the required average number of trials. The typical trajectories of the angles in a successful trial under the condition of noise free are provided in Fig.5.7. The steady-state of three joint angles can be obtained after learning in a few hundred steps. The corresponding λ values are provided in Fig.5.8. The typical trajectory of the control force (Newton) applied on the cart is provided in Fig.5.9, and the histogram of this control force is also presented in Fig.5.10. The simulation results presented in this case study have again demonstrated the control capabilities of the GrDHP design. It is worth to point out that the angle variations are significantly smaller than those using the HDP design in [68].

5.5 Discussion of Advanced GrADP Designs

Goal representation adaptive dynamic programming (GrADP) design has been proposed and studied with various simulation examples and many real applications in the smart grid area [4, 103, 62, 73, 97, 104]. Specifically, in the goal representation HDP design, the goal network is introduced to provide the secondary/internal

Figure 5.8. Typical trajectories of the λ (corresponding to the angles in Fig. 5.7) in a successful trial under the condition of noise free.

reinforcement signal $s(t)$ (also known as the internal goal signal in later publications) for the critic network. This s signal can provide a more informative and specific internal reinforcement representation for the critic network, and thus help the value function approximation in the learning process. We have investigated the reasons for this ADP design to work better than other ADP designs, with information of higher-order derivatives better approximated being one of the keys for the improved performance. We have implemented the hierarchical HDP design with three goal networks, and conducted the simulation on the ball and beam balancing benchmarks in [105, 106]. We compared the performance among the traditional HDP, GrHDP and the hierarchical HDP designs under the same environment settings (i.e., noise conditions). The hierarchical HDP design actually achieve the best performance in terms of both the successful rate and the required average number of trials. We also did the t-test for the required average number of trials, and the hierarchical HDP design showed significant improvement with high confidence level. The stability analysis and mathematical foundation of this GrHDP controller has been studied and investigated in [73, 74]. The conditions

Table 5.1. Comparison of the statistical simulation results on the triple-link inverted pendulum balancing task with the DHP and the GrDHP approaches.

	DHP		GrDHP	
Noise type	SR	Mean	SR	Mean
Noise free	100 %	20.3	100 %	11.6
Uniform 5% a.*	80 %	169.8	98 %	62.7
Uniform 10% a.	43 %	203.4	95 %	89.9
Uniform 5% x.†	79 %	130.1	91 %	60.0
Uniform 10% x.	58 %	197.7	86 %	144.6

a.* : actuator sensor noise

x.† : position sensor noise

SR : successful rate

Mean : required average number of trials

and constraints for the key parameters used in the learning and optimization process were provided and justified. Furthermore, the Robbins-Monro algorithm has been adopted to analyze the theoretical characteristics for the GrHDP design on maze navigation examples. Faster convergent speed with respect to the sum of squared errors were observed, comparing to three other traditional reinforcement learning algorithms [97, 98]. A more rigorous convergence analysis of both the internal reward signal and the value function is under investigation.

In this GrDHP design, the derivatives $\frac{\partial r}{\partial Y}$ are indeed needed to be specified when implementing this ADP architecture. However, the function s and its derivatives g (i.e., $g = \frac{\partial s}{\partial Y}$) are generated internally within the GrDHP design, and those are used to drive the critic network learning in the absence of any r or $\frac{\partial r}{\partial Y}$. Moreover, since the goal network is built with the mapping between the system state and the internal reinforcement signal (or its derivatives), this s (or g) signal could be adaptively tuned over time. We have observed that in a simple cart-pole inverted pendulum balancing process, the reward could always be zero (“0” for success and “-1” for failure) as long as the inverted pendulum is balanced (no matter where the cart is). Comparing with the cart close to the boundary of the track, the cart

Figure 5.9. The typical trajectory of the control force (Newton) applied on the cart in a successful trial under the condition of noise free.

in the origin is more satisfactory. However, in both cases, the traditional ADP controller can only receive zero reward. We have demonstrated that our proposed GrHDP can provide an informative non-zero internal reward signal when the inverted pendulum is balanced with the cart not at the origin [105]. This specific non-zero internal reward signal helps to drive the learning process of the critic network.

In this chapter, we follow our previous work and build the general mapping between the system states and the required derivatives of the utility function. The relationship for this general mapping can be described as

$$\frac{\partial s(t)}{\partial \mathbf{Y}(t)} = F_g(x(t), u(t), w_g(t)) \quad (5.22)$$

where F_g refers to the neural network mapping in the goal network, and $\frac{\partial s(t)}{\partial \mathbf{Y}(t)}$ refers to the partial derivatives of the internal reinforcement signal s w.r.t the input $\mathbf{Y}$. From (5.22), one can also see that $\frac{\partial s(t)}{\partial \mathbf{Y}(t)}$ could be adaptively adjusted when the weights w_g is updated according to the objective function in (5.6). In this design, we integrate such a general mapping (i.e., the goal network) into the DHP design with the motivation to help the system's decision making process.

Figure 5.10. The histogram of the control force (Newton) applied on the cart (corresponding to the trajectory in Fig.5.9).

From a mathematical viewpoint, there are two major differences for this GrDHP design comparing to the traditional DHP design. First, the optimization error function for the critic network is different: the error function for the critic network is related to the derivatives of the internal reward/utility function $\frac{\partial s(t)}{\partial \mathbf{Y}(t)}$, which is provided by the goal network within the GrDHP structure; second, the outputs of the goal network are set as the inputs for the critic network, so that the goal network could support the approximation of the derivatives of the value function by the critic network. The idea is to use a neural network mapping to build such an internal reward/utility representation through the association and anticipation in the learning process. Generally, the use of the derivatives as the optimization criterion, rather than the optimization criterion itself, is regarded as being more critical to seek the optimal solution [12, 107, 108]. In the traditional DHP design, the required derivatives of the utility function are obtained from the pre-defined utility function, which is domain-oriented. It has a potential problem that a pre-defined utility function relies on the prior experience and can not be adjusted over time. The proposed GrDHP approach has an important advantage

over the traditional DHP design in terms of the required partial derivatives of the utility function. In our proposed design, approximation of these derivatives results in *direct* outputs from the goal network within the GrDHP structure. Moreover, the outputs of the goal network can be adaptively adjusted over time. Working on a rigorous proof of the advantages is currently in progress, but interested readers can refer to our previous theoretical works on GrHDP for justification [73, 74, 97] and refer to Chapter 5.4 for experimental demonstration.

Among the recent publications on GDHP design, one of the simplest implementation has been proposed as taking the combinations/advantages of HDP and DHP designs. That is, the critic network is used to approximate both the value function and its derivatives. The objective function for the critic network has thus been assigned as the combinations of both designs [12, 107, 108]. It is also possible to assign different weighted factor for these two objective functions to tune the critic network over time. In [54, 42, 55], it is compared with the performance of GDHP, DHP and HDP approaches on numerical examples, and also addressed the convergence of both the value function and its derivatives. Following by previous designs in Chapter 3 and Chapter 4, it is very possible to introduce the goal representation design into GDHP architecture. In this Gr-GDHP design, the goal network can not only provide the internal reward signal, but also its partial derivatives for the critic network. So that the critic network can obtain an adaptive (internal) reward within the Gr-GDHP structure, and also realize its objective function with the direct output (i.e., required partial derivatives of the internal reward signal) of the goal network. It is expected that such advanced ADP design can achieve the most accurate learning control results.

5.6 Summary

This chapter provides an advanced ADP design, namely GrDHP design. The previously developed approach featuring the goal representation network is first provided, as there is a strong connection between the GrHDP and GrDHP designs. Then the error functions for the goal, the critic and the action networks, are provided respectively. Two case studies demonstrate successful control performance of GrDHP approach in comparison with the existing DHP design. Further research work on Gr-GDHP is also pointed out as the critical direction to complete such GrADP design family.

CHAPTER 6

Hierarchical Goal Representation Heuristic Dynamic Programming Design

6.1 Introduction

This is an advanced research work from Chapter 3. It is interesting to see how multiple goal representation works, and what is the mechanism between each goal network (internal goal signals) [105, 106]. In this chapter, a hierarchical structure of goal networks is integrated into HDP design. The multi-goal networks not only interact with themselves as a whole unit, but also contribute to the value function approximation in critic network. The control performance is evaluated on the ball-and-beam balancing benchmark under noise-free and various noisy conditions. Simulation results demonstrate the significant improvement of this hierarchical HDP approach.

6.2 Design of Hierarchical HDP Structure

The schematic of the hierarchical HDP design is presented in Fig.6.1, where one can see that the typical model-free action dependent HDP is maintained the same as in [68]. The main contribution is to introduce the goal generator with hierarchical neural networks to cascade the external reinforcement signal and provide the critic network with hopefully improved internal goal representation of the external reinforcement signal. The goal generator can critique the system's behaviors and the control action, and then generate the adjustable internal goal signal automatically. In the following subsections, the learning and adaptation of the goal generator networks, the critic network and the action network are presented respectively.

Figure 6.1. The schematic of hierarchical HDP structure

Figure 6.2. Cascading weights tuning path inside the goal generator

6.2.1 Learning and Adaptation in Hierarchical Goal Networks

From Fig.6.2, one can see that the parameters in the goal generator networks are adjusted independently. The top goal generator network l will first learn to approximate the discounted total future reward-to-go based on r , and then provide the goal generator network $l - 1$ with the updated internal reinforcement signal s_l . In this way, r provides a top-down guidance for the s_l . The internal goal s_l should follow the “guidance” that the external reinforcement signal r provides. When $m = 1$, the internal goal s_1 will follow the guidance of s_2 through the goal generator network 1, and pass the goal information to the critic network. The goal generator networks form a cascade to represent the external reinforcement r internally. Here we would like to note that the input of the goal generator network l only contains the state vectors and the control value, while the other goal generator network m ($1 \leq m < l$) takes the state vectors, control action and the internal goal s_{m+1} as the inputs.

Consider the top neural network l in the goal generator first. The output of this network is to approximate the discounted total future reward. Specifically, it

approximates $R_l(t)$ at time instance t

$$R_l(t) = r(t+1) + \alpha r(t+2) + \alpha^2 r(t+3) + \dots \quad (6.1)$$

where $R_l(t)$ is the future accumulative reward/cost-to-go value at the time instance t , α is the discount factor ($0 < \alpha < 1$) for the infinite Markov decision process (MDP), and $r(t+1)$ is the external reinforcement signal value at time $t+1$. The goal generator networks m ($1 \leq m < l$) are to approximate the total future internal reward/goal signal s_{m+1} from the above goal generator network $m+1$.

The signal s_l is to approximate the R_l expressed in (6.1), the error of this goal generator network can be defined as

$$e_{r_l}(t) = \alpha s_l(t) - [s_l(t-1) - r(t)] \quad (6.2)$$

and the objective function is defined as

$$E_{r_l}(t) = \frac{1}{2} e_{r_l}^2(t) \quad (6.3)$$

The weights tuning rule for the goal generator network l is chosen as the gradient descent rule as

$$\omega_{r_l}(t+1) = \omega_{r_l}(t) + \Delta \omega_{r_l}(t) \quad (6.4)$$

where

$$\begin{aligned} \Delta \omega_{r_l}(t) &= l_{r_l} \left[-\frac{\partial E_{r_l}(t)}{\partial \omega_{r_l}(t)} \right] \\ &= l_{r_l} \left[-\frac{\partial E_{r_l}(t)}{\partial s_l(t)} \frac{\partial s_l(t)}{\partial \omega_{r_l}(t)} \right] \end{aligned} \quad (6.5)$$

For the goal generator network m ($1 \leq m < l$), it is designed to approximate R_m defined in (6.6) with s_{m+1}

$$R_m(t) = s_{m+1}(t+1) + \alpha s_{m+1}(t+2) + \alpha^2 s_{m+1}(t+3) + \dots \quad (6.6)$$

Therefore, the error function of goal generator network m can be defined as

$$e_{r_m}(t) = \alpha s_m(t) - [s_m(t-1) - s_{m+1}(t)], \quad (6.7)$$

and the objective function is defined as

$$E_{r_m}(t) = \frac{1}{2}e_{r_m}^2(t) \quad (6.8)$$

The weights tuning rule for the goal generator network m is also chosen as the gradient descent rule as

$$\omega_{r_m}(t+1) = \omega_{r_m}(t) + \Delta\omega_{r_m}(t) \quad (6.9)$$

where

$$\begin{aligned} \Delta\omega_{r_m}(t) &= l_{r_m} \left[-\frac{\partial E_{r_m}(t)}{\partial \omega_{r_m}(t)} \right] \\ &= l_{r_m} \left[-\frac{\partial E_{r_m}(t)}{\partial s_m(t)} \frac{\partial s_m(t)}{\partial \omega_{r_m}(t)} \right] \end{aligned} \quad (6.10)$$

6.2.2 Learning and Adaptation in the Critic Network

Unlike the regular critic network in the typical HDP design in [68], the inputs of the critic network here not only include the system state vectors and the control action, but also contain the internal goal signal s_1 . In this way, the total cost-to-go J is more closely associated with this informative goal/reinforcement signal than before. The error function of the critic network here is defined as follows

$$e_c(t) = \alpha J(t) - [J(t-1) - s_1(t)], \quad (6.11)$$

and the objective function is defined as

$$E_c(t) = \frac{1}{2}e_c^2(t) \quad (6.12)$$

The weights updating rule for the critic network is chosen as the gradient descent rule as

$$\omega_c(t+1) = \omega_c(t) + \Delta\omega_c(t) \quad (6.13)$$

where

$$\begin{aligned} \Delta\omega_c(t) &= l_c \left[-\frac{\partial E_c(t)}{\partial \omega_c(t)} \right] \\ &= l_c \left[-\frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial \omega_c(t)} \right] \end{aligned} \quad (6.14)$$

6.2.3 Learning and Adaptation in the Action Network

The weights tuning in the action network is similar as that in [68]. The error is between the desired ultimate objective U_c and the J function as defined in (6.15)

$$e_a(t) = J(t) - U_c(t), \quad (6.15)$$

The objective function for the action network is defined in (6.16)

$$E_a(t) = \frac{1}{2}e_a^2(t) \quad (6.16)$$

The weights tuning rule for the action network is chosen as the gradient descent rule as

$$\omega_a(t+1) = \omega_a(t) + \Delta\omega_a(t) \quad (6.17)$$

where

$$\begin{aligned} \Delta\omega_a(t) &= l_a \left[-\frac{\partial E_a(t)}{\partial \omega_a(t)} \right] \\ &= l_a \left[-\frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u(t)} \frac{\partial u(t)}{\partial \omega_a(t)} \right] \end{aligned} \quad (6.18)$$

6.3 Simulation Studies

The evaluation is in two-aspect: one is with only one goal generator network (or the three-network architecture as discussed in Chapter 3), which is defined as *Algorithm1*; the other is with multiple (three) goal generator networks, which is defined as *Algorithm2*. The motivation is to test these two algorithms, together with the typical HDP in [68] (without any goal generator network), which is defined as *Algorithm0*. These three algorithms are tested and compared on the ball and beam balancing problem in the same simulation environment. The ball-and-beam system is a popular laboratory model as described in Chapter 4.4.3.

6.3.1 Experiment Configuration and Parameters

In the implementation, multi-layer perceptron (MLP) structure for all the neural networks is used. As the control system has 4 state vectors, the action

network is set with 4-6-1 structure (i.e., 4 input neurons, 6 hidden layer neurons, and 1 output neuron) and the critic network is set with 5-6-1 structure. The top goal generator network l is with 5-6-1 structure, while the other goal generator networks are with 6-6-1 structure. The parameters used in the experiment are summarized in Table 6.1, and the notation is defined as follows:

Table 6.1. Summary of the parameters used in the simulation

Para.	$l_c(0)$	$l_a(0)$	$l_r(0)$	$l_c(f)$	$l_a(f)$	$l_r(f)$	*
value	0.3	0.3	0.3	0.005	0.005	0.005	*
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
value	80	100	50	0.05	0.005	0.05	0.95

We keep all the goal generator networks with the same learning rate, error threshold and the maximum internal iteration cycle number in the simulation. For the ACD approach in [68], we set the parameters the same as that in this table, except the terms belonging to the goal generator networks.

All simulation results presented in this experiment are based on 100 runs with random initial neural network weights. The initial conditions of the the ball-and-beam system are set up as follows: The ball position (x_1) and the angle of the beam with respect to the horizontal axis (x_3) are uniformly distributed in the range of $[-0.2m, 0.2m]$ and $[-0.15rad, 0.15rad]$, respectively, and the ball velocity (x_2) and the angular velocity (x_4) are set to be zero. For fair comparison, in each run we also set the neural network initial weights and initial conditions of the beam and ball system to be the same for all three methods discussed here. The objective of the task is to keep balancing the ball on the beam for a certain period of time. Specifically, each run consists of a maximum of 1000 trials, and a trial will be considered successful if it can maintain the balance of the ball for 10,000 time steps (the ball remains on the beam and the angle of the beam with respect to the horizontal axis is under the maximum value). In simulations, the Euler

integration method is used with the fixed step size of $0.02s$. The range of beam is $[-0.48m, 0.48m]$ and the range of the angle of the beam to the horizontal axis is $[-0.24rad, 0.24rad]$. The external reinforcement signal is set to be “0” if the ball is on the beam and the angle of the beam to the horizontal axis is within the range, otherwise it is set to be “-1”, which means “failure” and we should start a new trial.

6.3.2 Simulation Results and Analysis

Figure 6.3. The typical trajectory of x_1 and x_2 with Algorithm 2

Fig.6.3 shows a typical trajectory of the position of the ball (x_1) and the velocity of the ball (x_2) in a successful run under the noise-free condition for *Algorithm2*. From this figure one can see that ball starts at a random position and rolls forth and back at the early stage. As the system continues to learn to control the ball, the trajectory of x_1 is like a typical damping sinusoid wave, which converges as time goes by. The variation of the x_1 is also shown in Fig.6.4, which indicates that the ball is always around the center point under proper control. Fig.6.5 shows the angle of the beam with respect to the horizontal axis (x_3) and the beam angular velocity (x_4), which also clearly shows that the control system

Figure 6.4. The histogram of x_1 in a typical successful run with Algorithm 2

balances the ball quickly.

Fig.6.6 shows the typical trajectory of the control action and the total cost-to-go, both of which indicate how the system learns to appropriately adjust the force to balance the task with the minimum cost. The internal goal s_3 , s_2 , and s_1 , together with the external reward r in a typical successful run are shown in Fig.6.7, which shows that the internal goal signals $s_1 - s_3$ are the damping sinusoid signals rather than the zero value of r all the way in this trial. Once again, the zero value of r means the ball is on the beam and the angle of the beam to the horizontal axis is within the range . Further observations indicate that the internal goal signals are with different phases, which may suggest that the internal goals are trying to fit the total future cost and provide the networks below with a more refined goal representation. The variation of $s_1 - s_3$ are also presented in Fig.6.8, indicating that there are some variances in the goal signals.

Table.6.2 demonstrates the successful rate, the required average number of trials to learn the balancing task and its associated standard deviation for the three approaches tested in 100 random runs. For the required average number of trials,

Figure 6.5. The typical trajectory of x_3 and x_4 with Algorithm 2

Table 6.2. Simulation results on ball-and-beam balancing task. The 1st column is with the noise type. The 2nd column is with *Algorithm0*, while the 3rd column is with *Algorithm1* and the 4th column is with *Algorithm2*. The number of trials and standard deviation are calculated based on the successful runs

Noise type	<i>Algorithm0</i>			<i>Algorithm1</i>			<i>Algorithm2</i>		
	Success rate	# of trial	σ	Success rate	# of trial	σ	Success rate	# of trial	σ
Noise free	98 %	43.4	71.9	100 %	21.9	29.3	100 %	13.5	20.0
Uniform 5% a.*	98 %	46.5	59.2	98 %	21.3	30.4	99 %	17.6	44.8
Uniform 5% x.†	96 %	65.3	113.7	100 %	23.8	77.2	100 %	16.2	18.4

σ : standard deviation

* a. : actuators are subject to the noise

† x. : sensors of positions are subject to the noise

we will only count the first successful balancing trial (10000 steps of balancing) in the each run. In this table, the 1st column indicates the noise types under which the algorithms are tested; the 2nd column presents the statistical results of the successful runs with *Algorithm0*; the 3rd column and the 4th column present the statistical results of the successful runs with our proposed *Algorithm1* and *Algorithm2*, respectively.

Under the condition of noise free, one can clearly see that both of our proposed approaches achieve higher successful rate with lower average trial number and lower standard deviation than those of the baseline *Algorithm0*. And *Algorithm2* can obtain better results than *Algorithm1* in terms of the average number of trials and the standard deviation. Also the boxplot of the required number of trials is

Figure 6.6. The typical trajectory of the control action and the total cost-to-go signal with Algorithm 2

presented in 100 random runs with the three algorithms under the noise-free conditions in Fig.6.9. Here the ANOVA analysis for the statistical results is conducted among *Algorithm0*, *Algorithm1* and *Algorithm2*. The average number of trials required to learn the balancing task with *Algorithm2* is significantly different from that of *Algorithm0/Algorithm1*, with the confidence level is 99.99%/98.21% (i.e., $p = 7.25e - 5/p = 0.0179$), respectively.

The 5% uniform noise is added on the actuator (u) and the sensor of position of the ball (x_1) respectively. While the actuator is under 5% uniform noise, one can see that our proposed *Algorithm1* and *Algorithm2* can both obtain the lower average number of trials and the standard deviation than those of *Algorithm0*. Fig.6.10 clearly shows that the control value (with *Algorithm2*) now is not as smooth as that in Fig.6.6. Also, the ANOVA analysis is conducted for the statistical results between *Algorithm0* and *Algorithm2*. The results show that *Algorithm2* can obtain significantly different average number of trial compared with that of *Algorithm0* in 99.8% (i.e. $p = 0.002$) confidence. While the sensor of the ball posi-

Figure 6.7. The typical trajectory of internal goal signals with Algorithm 2

tion is under 5% uniform noise, one can also see that *Algorithm1* and *Algorithm2* can both obtain lower average number of trial and standard deviation than that with *Algorithm0*. Fig.6.11 shows the typical trajectories of x_1 and x_2 . The control task becomes complicated since the observed state vector x_1 is not as smooth as that in Fig.6.3. Similarly, the ANOVA analysis is conducted for the statistical results with *Algorithm0* and *Algorithm2* here. The confidence level 99.99% (i.e. $p = 1.76e - 5$) is obtained that *Algorithm2* can achieve statistically significant improvement compared with that of *Algorithm0*.

6.4 Summary

A new hierarchical goal representation architecture is introduced based on the previous GrHDP design. A hierarchical structure of goal networks is used to interpret the external reinforcement signal into adaptive and informative internal goals signals, and feed such signals into critic network. This approach is tested against the GrHDP and the typical HDP approach on the same balancing example. The statistical simulation results demonstrate the improved learning control performance with a high confidence level. This is a preliminary study of the goal

Figure 6.8. The histogram of the internal goals with Algorithm 2

representation design. Next, the further simulation studies from toy problems and real-world applications will be provided.

Figure 6.9. The boxplot of the required number of trials in 100 random runs with Algorithm 0, Algorithm 1 and Algorithm 2.

Figure 6.10. The typical trajectory of control action with Algorithm 2 under 5% uniform noise on the actuator

Figure 6.11. The typical trajectory of the state vectors x_1 and x_2 with Algorithm 2 under 5% uniform noise on the sensor of the position of the ball

CHAPTER 7

Applications: From Toy Problems to Real-World Applications

7.1 Introduction

In this chapter, the applications of GrHDP are provided from maze navigation (toy problem) to smart grid (real-world) applications. The maze navigation example is a typical Markov decision process and can be perfectly solved by reinforcement learning and adaptive dynamic programming [109, 110, 111]. Here, 2-D maze navigation examples and 3-D maze navigation example are all provided and the performance of GrHDP approach is compared among typical HDP, SARSA(λ) and Q-learning approaches [97, 98].

Furthermore, this GrHDP approach has also been used as a nonlinear optimal control on the multi-machine power system. Compared with the conventional control approaches, the GrHDP controller conducts adaptive learning control and assumes unknown of the power system model. For fair comparative studies, the damping control performance is also compared with typical HDP and power system stabilizer (PSS). Simulation results verify that the investigated GrHDP control approach can achieve the improved performance in terms of the transient stability and robustness under various fault conditions [112].

7.2 Maze Navigation Application

7.2.1 Maze Navigation Learning Algorithm Description

The implementation details among GrHDP, HDP, Sarsa(λ) and Q-learning approaches are provided as follows:

- **Q-learning**

The Q-learning algorithm is one of earliest RL algorithms to find a reliable way to estimate training values for Q, given only a sequence of immediate

reward r spread out over time. Here we implement the Q learning algorithm based on [113] to build the Q-value table for the maze navigation problem. We set discount parameter γ to be 0.95.

- **Sarsa(λ)**

Temporal-difference (TD) learning is a combination of the Monte Carlo and dynamic programming idea. TD can learn directly from raw experience without a model of environment's dynamics and also update estimates without waiting to the final stage. Here we implement one of the typical algorithms in TD learning, namely Sarsa(λ) based on [27] to build state-action pairs for the agent in the maze navigation. The parameters setting are as: $\gamma = 0.95$, $\lambda = 0.9$, and $\alpha = 0.4$.

- **HDP**

Online model-free HDP is one of the typical ADP approach proposed in [68]. The initial learning parameters are set as: $\eta_c = 0.005$ and $\eta_a = 0.01$, where η_c and η_a refer to the learning rate of critic network and action network, respectively. The stopping criteria are: $N_c = 20$, $N_a = 30$, $T_c = 1e - 4$ and $T_a = 1e - 4$. That is to say, the learning process of critic/action network will be terminated either if the error drops into the threshold T_c/T_a or the iteration number meets the threshold N_c/N_a . The definitions of all these parameters are the same as those in Chapter 3.

- **GrHDP**

The GrHDP is implemented according to the key pseudo-code listed in Algorithm 1. The initial parameters for the goal network are: $\eta_g = 0.012$, $T_g = 1e - 4$ and $N_g = 25$. For fair comparison, we also ensure that the GrHDP and HDP start with the same initial weights between $[-0.3, 0.3]$.

All other parameters are kept the same as that in the HDP approach.

[Algorithm 1]: ***GrHDP on value table learning***

- 1) For each state-action pair (x, u) , initialize the table entry $J(x, u)$ to zero
 - 2) Observe the current state x
 - 3) Do forever
 - 4) Obtain action u , decide the direction with winner-take-all (WTA), and execute it;
 - 5) Receive an immediate reward r , and observe the new state x' ;
 - 6) Obtain internal goal s and value function J ;
 - 7) Update the weights in goal, critic and action networks according to formulas in Chapter 3.3;
 - 10) Update the table entry for $J(x, u)$;
 - 11) $x \leftarrow x'$;
-

7.2.2 Maze Navigation Environment Setup

In this simulation, the instant reward between any two state x and x' is denoted as $r(x, x')$. Assume that there are N possible states in the maze and the probabilities $P_{xx'}$ in (2.1) can only take the value of 0 or 1 (i.e., the maze navigation problem is a deterministic and finite MDP). Thus, the Bellman's equation (2.3) can be written as

$$J^*(x, u) = \arg \max_u \left(r(x, u) + \gamma \sum_{j=1}^N J^*(x', u') \right) \quad (7.1)$$

where $J^*(x, u)$ is the maximum total reward at state x by taking the action u . The objective for this maze navigation is to employ learning algorithms to learn the value table of the maze online, so that the agent can move according to the

direction that maximizes the total reward (towards the goal location).

Figure 7.1. Simulation setup: flowchart of the GrHDP approach on maze navigation problem. Value table is updated at the end of each trial. The learning process will be terminated when the trial number reaches the maximum trial number.

In order to provide the learning steps of our proposed GrHDP on maze navigation benchmarks, the flowchart is provided for the entire simulation process in Fig.7.1. The simulation steps can be described as follows.

1. Load the predefined updating sequence. Each updating sequence is assumed to visit all the state enough times;
2. Obtain the output from action network. Apply WTA method to decide the

direction, and execute it;

3. Obtain the new state from the maze and update the inputs for the goal network, obtain internal goal s ;
4. Update the inputs for the critic network, and obtain the J value. Update the $J(x, u)$ table;
5. Check if the agent move out of bound? If yes, turn to another trial (punishment is assigned) and load the next initial state to start again; if no, move the agent to another step (same trial);
6. Check if the agent reach the goal? If yes, turn to another trial (reward is assigned) and load the next initial state to start again; if no, move the agent to another step (same trial);
7. Terminate the entire learning process if the trial number satisfies the maximum trial number;

7.2.3 Simulation Studies and Analysis

Assume that 1) every state in the maze has been visited enough times; 2) every action (up, down, left, right) has been taken enough times for each state; 3) for every initial state, the agent can go infinite steps forward unless it meets the goal or it hits the bound. The input for the action network is the current state vector $x_a = [x_1, x_2]$, where x_1 is the coordinate of horizontal axis and x_2 is the coordinate of vertical axis. The input for the goal network and the critic network are $x_g = [x_1, x_2, u_1, u_2, u_3, u_4]$ and $x_c = [x_1, x_2, u_1, u_2, u_3, u_4, s]$, respectively, where u_1 is for the direction of “up”, u_2 is for the direction of “down”, u_3 is for the direction of “left”, u_4 is for the direction of “right”. The external reward r is

defined as

$$r = \begin{cases} 1, & \text{reach the goal} \\ -0.2, & \text{out of bound} \\ 0, & \text{regular move} \end{cases} \quad (7.2)$$

In this simulation, U_c is assigned as 1 and the inputs for the action network are scaled to be in $[0, 2]$, and 10 independent updating sequences are used for 10 independent runs (i.e. the updating sequence in each runs is independent). Each run includes a given number of trials and each trial starts with the initial state loaded from the updating sequence. Each trial will be terminated when the agent meets the goal or hits the bound. Under this setting, the steps that the agent move in each trial are not necessarily the same. The $J(x, u)$ table is only updated after the agent finish each trial and is then normalized to $[0, 1]$ to show the difference with the reference value table.

2-D Maze Problem

The maze with size 16 by 16 is applied in this study, and the goal is set at the upper-right corner (i.e., $[16, 16]$). The trial number is 1000 for each run and the counter threshold is 30 here. The adaptive learning rates (ALR) are set as: the learning rates will be decreased by dividing 2 every 10 trials and will be kept to be $1e - 10$ thereafter if they are under $1e - 10$. The Q reference value table is still assigned according to the distance between the current location and the goal (i.e., the value difference between two consecutive states is $\frac{1}{32}$).

All the learning curves in Fig.7.2 are the average value from 10 independent runs. From Fig.7.2, one can see that GrHDP and HDP converge within a few hundred trials, while Sarsa(λ) and Q-learning require more than 1000 trials to learn. The learning curves of both GrHDP and HDP approach drop quickly at the very beginning, yet the GrHDP approach can achieve lower steady error (i.e., the value table learned by GrHDP approach can be more close to the reference value

table). In addition, the average of value table learned in 10 runs with GrHDP approach is presented color surface in Fig.7.3. It is clear that the values smoothly become higher as the agent approaches the goal location.

Figure 7.2. Learning curves with GrHDP, HDP, Sarsa(λ) and Q-learning approaches in 16 * 16 maze navigation. GrHDP approach shows the fastest learning speed and lowest final sum of square error than the other three approaches.

There are two important observations from Fig. 7.2. First, both HDP and GrHDP can converge faster to the optimal policy compared to the Q-learning and Sarsa(λ) approaches. This may indicate that for the 2-D maze navigation problem, ADP methods could be able to provide better learning performance. Further observation also suggests that with the help of the goal network, the GrHDP can converge faster than the regular HDP approach. Second, as far as the final sum squared error is concerned, the GrHDP approach can also achieve the best performance in this case. The reason that Q-learning and Sarsa(λ) approaches did not converge to a zero value is because the reference table is defined according to its distance to the goal. This type of reference table is reasonable for the maze-navigation benchmark as been discussed in [114, 115, 116]. Certainly, if one adopts

Figure 7.3. Surface plot of the value table learned by GrHDP approach on maze navigation problem (16*16). x and y axis refer to the coordinates of the agent while z axis refers to the J-value.

the Q-reference table as in the traditional RL literature [27, 113], it is expected the final errors of the Q-learning and Sarsa(λ) to approach zero. The key interests from this perspective in this work are the convergence speed and the optimal policy, in which GrHDP approach achieves much better performance compared to the regular HDP, Q learning, and the Sarsa(λ) method.

3-D Maze Problem

The structure of the 3-D maze is presented in Fig.7.4, where the goal is located in the upper-right corner (i.e. [5, 5, 5]) and the agent starts from a random position within this maze. This 3-D maze navigation benchmark is more difficult than the 2-D maze navigation benchmarks above, as the agent needs to learn from more directions (i.e., the agent need more trials to learn). The agent has to try 6 actions (i.e. forward, backward, up, down, left, and right) before it can make the right decision. The only setting difference here is that the trial number is 1500 for each run. The Q reference value table is still assigned according to the distance between

the current location and the goal (i.e., the value difference between two consecutive states is $\frac{1}{15}$).

Figure 7.4. Diagram of 3-D maze (5*5*5) navigation benchmark. The goal locates at the upper-right corner of the maze and the agent needs to try 6 directions before it can learn the policy.

The learning curves with the four algorithms are provided in Fig.7.5. One can see that the GrHDP approach can achieve the fastest convergence speed with respect to the sum of squared error and also the lowest final sum of square error. While for both Q-learning and Sarsa(λ) approaches, convergence tendency can easily be obtained yet they may need more trials to learn. Although these two examples clearly demonstrate the powerful learning performance of GrHDP approach, they are still relatively toy problems in the field. Next, a more interesting example in smart grid field is tested to show the potential applicable in the real-world complex engineering applications.

7.3 Smart Grid Application

Computational intelligence (CI) has been introduced into power system stability and control areas, and has also shown promising control performances on

Figure 7.5. Learning curves with GrHDP, HDP, Sarsa(λ) and Q-learning approaches in 5*5*5 maze navigation. GrHDP approach shows the fastest learning speed and lowest final sum of square error than the other three approaches.

various applications based on adaptive dynamic programming (ADP)¹ [117] [118] [119]. In ADP based controller design, the exact mathematic model function is not a prerequisite. The controller observes the input vector from the power system, and provides the supplementary control signal for the exciter. A reward signal will be provided based on the current system performance and a value function will be used to critique the performance of this control action. There are usually two neural networks in the ADP design: an action network is used to provide the control action while a critic network is used to evaluate the performance of this control action with a value function. In many cases, a model network is also adopted to identify the system dynamics. In literature, the researchers have implemented the dual heuristic dynamic programming (DHP) approach into the multi-machine turbogenerator control, and compared the performance with the conventional automatic voltage regulator (AVR) and power system stabilizer (PSS) in [120]. The

¹Both HDP and DHP designs mentioned are the different type of implementations in the ADP design family. Specifically, HDP design is used as a benchmark here.

model network was built to represent the dynamics of the turbogenerator based on the input and output data, and the action/critic networks were trained offline to achieve shorter rise time and faster convergence to synchronous speed than that of the conventional governor and PSS. In [121], the authors investigated the coordinated reactive power control of a large wind farm and a shunt static synchronous compensator (STATCOM). The similar power system modeling and offline training for the action and critic networks were conducted. In [68], the authors proposed the model-free heuristic dynamic programming and demonstrated the promising results on general nonlinear systems. Then, in [122], the authors demonstrated this online model-free HDP for the damping control on a four-machine two-area example, and further in the China Southern Power Grid. This model-free adaptive control scheme was also investigated for the reactive power control on the grid connected wind-farm [103]. The further comparison between the PSS and the HDP controller on the doubly-fed induction generator (DFIG) was also provided in [104]. The intelligent local area signals damping control in power system oscillations was investigated in comparison with existing intelligent controllers [123]. Many others also studied the ADP based adaptive control approach on smart grid frontier applications, including grid-connected converter [124], static compensator in multi-machine power system [125], wide area optimal control [126], and many others [127] [128] [95] [129] [130]. Among these research work, the reward signal is usually defined as the fixed (and derivable) formulas, such as the (weighted) linear quadric forms [120] [121] [68] [122]. I realize that the fixed or pre-defined reward function may not be a good choice when the system is under different operation conditions. In addition, the parameters in the reward function are significantly relying on the engineering knowledge for such system, which may not be a good thing if the system is unstructured or with uncertainties.

In this chapter, a multi-machine power system modeling and the simulation platform is introduced and the schematic diagram of the GrHDP controller is presented in Fig. 7.6. The ADP controller observes the measurements from each generator and provide supplementary control action u_1 , u_2 , and u_3 for the multi-machine accordingly.

Figure 7.6. The schematic diagram of the three-machine nine-bus power system. The dot lines show the observations from each generator. The ADP controller provides three supplementary control signals for three generators respectively based on these (delayed) measurements.

Fig.7.15 shows a schematic diagram for a single generator. The controller (i.e., the ADP controller) observes the difference among the rotor speeds and provides the supplementary control signals (with a limiter) for the excitation system. The synchronous machine is connected to the grid and the feedback observations include the rotor speed for all the machines. Note that in the simulation studies, the delayed signals of the differences of the rotor speed are included for better performance. During the regular operation, the generators are working synchronously and the parameters (e.g., the rotor speed ω and the output active power P_e) are

Figure 7.7. The schematic diagram for a single generator. The ADP and PSS controller will be connected into the closed-loop if S1 switches to 1 or 2. The dot arrows show the observed signals from the generator and the grid, while the solid arrows show the signal paths.

at their per unit values. According to our environment settings, the inputs for the ADP controller are zero under this case. The corresponding outputs (i.e., the supplementary control signal u_1 , u_2 , and u_3) are also zero. During the fault conditions, a proper (external) reward signal r will be assigned and the ADP controller will learn to provide the proper control strategy over time. The ultimate objective for the ADP controller is to increase the damping and help the oscillation of $\Delta\omega$ to converge to zero as soon as possible (i.e., help the generator recover synchronous operation again).

7.3.1 Multimachine Power System Environment Setup

There are two criteria for the learning of each network: one is the error threshold (corresponding to T_a , T_c and T_r in Table 7.2) and the other one is the maximum iteration number (corresponding to N_a , N_c and N_r in Table 7.2). If the squared error is minimized under the threshold or the iteration number exceeds the tolerance, the learning is terminated in this network. Once the weights updating of

Table 7.1. Summary of the parameters used in the GrHDP/HDP controller. The notations are kept the same as those in Chapter 3

Para. value	l_c 0.01	l_a 0.01	l_g 0.01	N_c 50	N_a 50
Para. value	N_g 20	T_c $1e-7$	T_a $1e-6$	T_g $1e-7$	α 0.95

l_* : Learning rate of the neural network;
 N_* : Max iteration number of the backpropagation;
 T_* : Error threshold of the objective function;
 α : Discount factor.

the goal, the critic and the action networks have been finished, it is regarded as completed of the learning process as presented in Fig.7.8. The signal flowing steps are described as following:

1. At t step, x_t is observed from the multi-machine power system. u_t can be obtained from action network and meanwhile the reward r_t is observed.
2. s_t and J_t will be calculated from the goal network and the critic network, respectively.
3. Retrieve the history data of s_{t-1} and J_{t-1} , and calculate the temporal difference for the objective functions in both networks.
4. Update the weights parameters in the sequence of goal network, critic network and action network. The weights are carried-on to the next time step.
5. Repeat from the first step when entering the $t + 1$ step.

7.3.2 Simulation Studies and Analysis

The simulation results of GrHDP, HDP, PSS and without PSS are provided in this section under the same environment (fault) settings. As mentioned in literature [120] [122], ADP learning controllers need to learn offline for the optimized parameters. Assume the GrHDP and HDP controllers learn offline for the first trial and then test their final control policy online in the second trial. There are

Figure 7.8. The data flowchart of GrHDP controller with neural network implementation.

two faults in the simulation studies: one is the three-phase-ground fault at line 4 and the other one is the $\pm 5\%$ step change for the output active power in generator 1. The learning ability is compared in the first case and the robustness is checked in the second case. The admittance matrices between each line and bus for the system and the per unit values for each generator are provided in Table 7.3 and Table 7.4, respectively. The parameters for the PSS is $K_s = 0.015$, $T_w = 1.5$, $T_1 = 0.3$ and $T_2 = 0.06$, and the output limiter is ± 0.035 pu [131].

Three-Phase Ground Fault

A single three-phase-ground fault is applied on the line 4 at $0.5s$, and last for $0.1s$ with tripping the line. At $1.1s$, the line is re-closed. Define that

- $\Delta\omega_{12} = \omega_1 - \omega_2$: the difference of the rotor speed between the generator 1 (G1) and the generator 2 (G2);
- $\Delta\omega_{13} = \omega_1 - \omega_3$: the difference of the rotor speed between the generator 1

(G1) and the generator 3 (G3);

- $\Delta\omega_{23} = \omega_2 - \omega_3$: the difference of the rotor speed between the generator 2 (G2) and the generator 3 (G3);

The input for the ADP controller is defined as

$$\begin{aligned} x(t) = & [\Delta\omega_{12}(t) \quad \Delta\omega_{13}(t) \quad \Delta\omega_{23}(t) \\ & \Delta\omega_{12}(t-1) \quad \Delta\omega_{13}(t-1) \quad \Delta\omega_{23}(t-1) \\ & \Delta\omega_{12}(t-2) \quad \Delta\omega_{13}(t-2) \quad \Delta\omega_{23}(t-2)]. \end{aligned} \quad (7.3)$$

The output of the ADP controller is defined as

$$u_{ADP}(t) = [u_1(t) \quad u_2(t) \quad u_3(t)]. \quad (7.4)$$

The control action $u_i(t)$ ($1 < i < 3$) is applied to each generator respectively. The weighted factors for (7.5) are set as

$$\begin{aligned} b_1 &= b_2 = b_3 = 0.4; \\ b_4 &= b_5 = b_6 = 0.3; \\ b_7 &= b_8 = b_9 = 0.3. \end{aligned} \quad (7.5)$$

In literature, the coefficients for most recent squared errors (i.e., b_1 , b_2 and b_3) will be defined with a relatively large value and the coefficients for the previous squared errors (i.e., b_4 , b_5 , b_6 , b_7 , b_8 and b_9) will be defined with a relatively small value. These coefficients can usually sum up to 1 (i.e., $b_1+b_4+b_7=1$). The damping results of $\Delta\omega_{12}$, $\Delta\omega_{13}$ and $\Delta\omega_{23}$ are provided in Fig. 7.9, Fig. 7.10 and Fig. 7.11, respectively. Among the four approaches in the simulation, one can see that the GrHDP and HDP controllers can achieve faster convergence to synchronous speed than that with the other two conventional control strategies. Between the GrHDP and HDP control approaches, the GrHDP controller shows less overshoot and faster convergence to synchronous speed. Note that the typical

Table 7.2. Summary of the parameters used in the GrHDP/HDP controller. The notations are kept the same as those in Chapter 3

Para. value	l_c 0.01	l_a 0.01	l_g 0.01	N_c 50	N_a 50
Para. value	N_g 20	T_c $1e-7$	T_a $1e-6$	T_g $1e-7$	α 0.95

l_* : Learning rate of the neural network;
 N_* : Max iteration number of the backpropagation;
 T_* : Error threshold of the objective function;
 α : Discount factor.

learning process of the ADP controller includes two trials described as that in [122] [104]. In trial 1, the neural networks were initialized with random weights and the supplementary control outputs might not be in accordance with the desired actions. The simulation was terminated when the system reached the failure criterion. In this process, the ADP controller hopefully leaned a considerable amount of useful information about the state-action pairs. In trial 2, the fully trained control policies could be achieved after the first trial process. Additional online learning in trail 2 can obtain better control performance over time. The performance of both the GrHDP and the HDP controllers are based on their trained weights in neural networks (i.e., trail 2). The parameters for both controllers are presented in the Table 7.2.

Output Active Power Step Changes

The parameters of GrHDP and HDP approaches, as well as PSS design, are kept the same parameters as those in Chapter 7.2. The -5% step change of output active power is added for the generator 1 at $0.5s$ and the $+5\%$ step change of the output active power is added for the generator 1 at $8s$. The performance of output active power P_{e1} , P_{e2} and P_{e3} are provided in Fig. 7.12, Fig. 7.13 and Fig. 7.14, respectively. An interesting observation from these three figures is that the conventional PSS controller does not help the generator go back to the original operation point. The reason is that the PSS is usually dedicated for specific fault.

Figure 7.9. Damping performance on $\Delta\omega_{12}$ under three-phase-ground fault. Four approaches are compared on the same environment settings.

If the fault changes, then the PSS controller may not be able to achieve expected performance. For instance, the PSS designed for three-phase ground fault is not a good choice for the step change of output active power. While for the HDP and GrHDP controllers, both of them can achieve promising control performance. The GrHDP controller achieves shorter rising time for the output active power and less overshoot after the step changes. In Fig.7.14, it seems that the step changes on generator 1 do not have a significant impact on generator 3, while the designed GrHDP and HDP controllers can still show better performance compared with the conventional PSS controllers.

Sequential Load Fluctuations

Additionally, the investigated GrHDP control approach for a sequential load fluctuation changes together with the other three approaches is tested here. The GrHDP and HDP approaches are tested based on the same weight parameters as those in section 7.3.2, and the PSS is also applied based on the same designed parameters in section 7.3.2. The load fluctuation is added for load A in a sequence

Figure 7.10. Damping performance on $\Delta\omega_{13}$ under three-phase-ground fault. Four approaches are compared on the same environment settings.

of random disturbances as described in Table 7.5. The disturbances are presented as the percentage of the load and the positive or negative sign indicate adding or subtracting on the load. The simulation length is set as 30s and assume that the output active power has been stabilized before the next disturbance happens.

The simulation results of output active power P_{e1} are provided in Fig.7.15. It is interesting to observe that the GrHDP can achieve very consistent control performance as those in section 7.3.2. In this case, the GrHDP approach can still demonstrate better performance in terms of overshoot and rise-time than any other approach. In Fig. 7.16, the supplementary control signals provided by GrHDP approach are also presented under this sequential load disturbances. As discussed in section 7.3.1, the convergence of u_1 , u_2 and u_3 further indicates that the investigated approach is stable and effective.

7.4 Summary

The GrHDP approach is demonstrated to be a faster learning algorithm than any other traditional reinforcement learning algorithms in the maze navigation ex-

Figure 7.11. Damping performance on $\Delta\omega_{23}$ under three-phase-ground fault. Four approaches are compared on the same environment settings.

ample. The surface plot of value table for the maze is also provided to show the learnt moving policy. This example clearly verifies the learning advantage of the goal representation technique in ADP design. Furthermore, a real-world complex application in smart grid field is tested with GrHDP approach against several other traditional approaches. Under various faults environment, the GrHDP approach successfully shows better adaptation and damping capacity over the others. The results clearly present the potential powerful applicable future for critical engineering applications. Note that it is also possible to apply the proposed approach on several many critical engineering applications and complex systems, such as flight/helicopter control [132, 133, 50], engine torque and air-fuel ratio control [80, 134, 135, 136], iron and steel company looper system control [77, 137, 138] and among others.

Figure 7.12. Comparison of the control performance on the output active power Pe_1 under both -5% and $+5\%$ step changes

Figure 7.13. Comparison of the control performance on the output active power Pe_2 under both -5% and $+5\%$ step changes.

Figure 7.14. Comparison of the control performance on the output active power Pe_3 under both -5% and $+5\%$ step changes.

Table 7.3. The admittance matrices between each line/bus in the three-machine nine-bus system.

$Z(1, 1) = 0 - 17.3611i$	$Z(4, 1) = 0 + 17.3611i$
$Z(2, 2) = 0 - 16.0000i$	$Z(7, 2) = 0 + 16.0000i$
$Z(3, 3) = 0 - 17.0648i$	$Z(9, 3) = 0 + 17.0648i$
$Z(1, 4) = 0 + 17.3611i$	$Z(4, 4) = 3.3074 - 39.3089i$
$Z(5, 4) = -1.3652 + 11.6041i$	$Z(6, 4) = -1.9422 + 10.5107i$
$Z(4, 5) = -1.3652 + 11.6041i$	$Z(5, 5) = 3.8138 - 17.8426i$
$Z(7, 5) = -1.1876 + 5.9751i$	$Z(4, 6) = -1.9422 + 10.5107i$
$Z(6, 6) = 4.1018 - 16.1335i$	$Z(9, 6) = -1.2820 + 5.5882i$
$Z(2, 7) = 0 + 16.0000i$	$Z(5, 7) = -1.1876 + 5.9751i$
$Z(7, 7) = 2.8047 - 35.4456i$	$Z(8, 7) = -1.6171 + 13.6980i$
$Z(7, 8) = -1.6171 + 13.6980i$	$Z(8, 8) = 3.7412 - 23.6424i$
$Z(9, 8) = -1.1551 + 9.7843i$	$Z(3, 9) = 0 + 17.0648i$
$Z(6, 9) = -1.2820 + 5.5882i$	$Z(8, 9) = -1.1551 + 9.7843i$
$Z(9, 9) = 2.4371 - 32.1539i$	

Table 7.4. The per unit value for the parameters of each generator/bus in the system.

	G1/#1	G2/#2	G3/#3
X_d	0.146	0.8958	1.3125
X'_d	0.0608	0.1198	0.1813
X_q	0.0969	0.8654	1.2578
X'_q	0.0969	0.1969	0.25
X_l	8.96s	6s	5.89s
T'_{d0}	0.0969	0.8654	1.2578
T'_{q0}	0s	0.535s	0.6s
H	23.64s	6.4s	3.01s
D	0	0	0

Table 7.5. The sequence of load fluctuation applied in load A

Time	0.5s	6s	12s	18s	24s
Disturbance	8.4%	2.8%	-7.0%	-2.8%	-1.4%

Figure 7.15. The output active power P_{e1} under the load fluctuations. The GrHDP approach is compared against the other three approaches under the same sequential load disturbances.

Figure 7.16. The supplementary control signals u_1 , u_2 and u_3 provided by GrHDP approach in this sequential load disturbances scenario.

CHAPTER 8

Conclusions and Future Research Directions

8.1 Conclusions

This dissertation research is motivated by brain-intelligence and animal intelligence that have multi-level internal goals to accomplish, in order to achieve the long-term goals. To this end, such intelligence is formulated into an engineering way that is mimic by multi-level goal networks based on ADP and RL. A new internal goal representation based ADP design is thus proposed based these inspirations. Promising statistical simulation results have been achieved based on several commonly used balancing benchmarks.

As this research moves forward, the proposed approach is named as goal representation adaptive dynamic programming (GrADP) design. The goal representation technique has been successfully applied to HDP and DHP as well as GDHP designs. Successfully learning control results have demonstrated the powerful learning control capacity of this GrADP designs. Further development of goal representation technique has also been integrated into the DHP as well as GDHP designs. Hierarchical goal representation principle (multiple goal networks) is also verified on HDP architecture. Not only the toy problems are tested with the proposed GrHDP designs, but also the real-world smart grid applications are also used. Better learning, adaptation and optimization capacity are justified through all these simulation studies. Theoretical analysis and stability assurance of this GrADP design are also included in the Appendix of this dissertation.

8.2 Original Contributions

This is the first time that adaptive internal goal representation has been proposed based on adaptive dynamic programming. This goal representation tech-

nique can be regarded as a general mapping function for the reward, and thus the reward signal is able to be adaptive adjusted over time online. In contrast with a fixed and predefined function in literature, the proposed GrADP design is more capable to mimic certain level of machine intelligence in terms of learning and optimization abilities. The original contributions of this dissertation is summarized as following:

- This is the first time that the reward feedback has been proposed as an general mapping function, rather than a fixed and predefined formula. This general mapping function is capable to be adaptive adjusted over time online based on the observation of the system variables. Such adaptation is the unique advantage in comparison with the fixed formula in literature.
- Goal representation design has been successfully integrated into HDP design, further on advanced ADP design, including DHP and GDHP designs. To this end, the GrADP design family has been first developed with explicit description. Simulation results verify the improved learning control and optimization performance. Further real-world applications and complex system design have also demonstrated the large-scale applicability of the proposed GrADP family.
- The theoretical analysis and stability assurance have also been provided to address the convergence and boundedness results. Existing literature works can not be directly applied for this new design, as the additional goal representation network and its contribution of the total value function. This is the first time to demonstrated in GrADP design that the error difference between the learning (weight) parameters and the expected values are uniformly ultimately bounded. The value function approximated by GrADP approach

is also close enough to the expected Q value (within an arbitrary small error). These theoretical results are very important for the new design, as they complete this new design family with a solid mathematical foundation.

8.3 Future Research Directions

The dissertation provides the first comprehensive study for the GrADP learning principle and its design family, including GrHDP, GrDHP and Gr-GDHP architectures. Promising results, including both simulation results and theoretical results, are provided to demonstrate the improved learning control performance. As this is a new design framework in the field, there are still many opportunities to conduct further research along this directions:

- As the GrADP approach demonstrates quite promising applicable in real-world complex applications, it is very interesting to see how much control and adaptation improvement it can achieve for several critical engineering problems. Such as the smart grid and renewable energy systems, hypersonic vehicle control and robot path planning problems, as well as operation research and logistic transportation optimizations. Preliminary results on several of these applications show that GrADP approach can provide better learning, optimization and prediction results from certain perspectives.
- Deep learning has become one of biggest topics in machine learning field, and deep reinforcement learning has also become one of frontier topics by taking the advantage of deep network learning principle. Convolutionary neural network, recurrent neural network, deep belief network and many other new techniques have showed powerful capacities in deep learning problems. It is very desirable to see if deep reinforcement learning methodologies can be applied in (goal representation) ADP field so that the “big data” learning

control issue can also be addressed based on deep learning ADP methods.

- Model-free technique has been presented in literature and demonstrated with simplified learning and adaptation algorithms. The computational cost and offline training are also significantly reduced. Literally, advanced ADP designs, such as DHP and GDHP designs, are still requiring a model network for precise prediction. I am very determined to develop model-free methodologies for DHP and GDHP designs in the field. Also, this will help to complete the model-free ADP design family and make it an alternative option for real-world applications. Furthermore, this model-free technique is also applicable for GrADP design family, including GrHDP, GrDHP and Gr-GDHP designs.
- In current literature, the ADP and GrADP designs are most focusing on the computer simulation. Many of the physical control systems needs dedicated and high-speed embedded systems to support. It is a nature movement if this research can also be applied in several high-speed embedded system, such as FPGA and GPU boards. One step ahead this direction could make the engineering intelligence more close to reality.

Machine intelligent system design is one of most exciting research topics in today's society. With the modern technologies, neuroscience, and fundamental research of artificial intelligence, our human being is very hopefully achieve the truly engineering intelligent systems. This dissertation provides a comprehensive study of machine intelligence based on goal representation adaptive dynamic programming, including design inspiration, new framework, new ADP design family, applications from toy problems to real-world examples and theoretical assurance as well as implementation-level pseudo code algorithm. Hopefully, the dissertation could contribute to the development of this most exciting and ambitious research topics in the field.

LIST OF REFERENCES

- [1] S. Grossberg, *Neural Networks and Natural Intelligence*. Cambridge, MA: MIT Press, 1988.
- [2] D. P. Bertsekas, *Dynamic programming and optimal control*. MA: Athena Scientific Belmont, 1995.
- [3] P. J. Werbos, *The Handbook of Brain Theory and Neural Netw.* Cambridge MA: MIT Press, 1995, ch. Backpropagation: Basics and New Developments, pp. 134–139.
- [4] H. He, *Self-Adaptive Systems for Machine Intelligence*. Wiley, 2011.
- [5] S. Singh, R. Lewis, and A. Barto, “Where do rewards come from?” in *Proc. Annual Conf. Cognitive Science Society*, 2009, pp. 2601–2606.
- [6] A. Barto, S. Singh, and N. Chentanez, “Intrinsically motivated learning of hierarchical collections of skills,” in *Proc. Int. Conf. Developmental Learning (ICDL’04)*, 2004.
- [7] S. Singh, A. Barto, and N. Chentanez, “Intrinsically motivated reinforcement learning,” in *Prof. Annual Conf. Neural Information Processing Systems (NIPS’04)*, Vancouver, Canada, 2004.
- [8] D. P. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.
- [9] J. Si, A. G. Barto, W. B. Powell, and D. C. Wunsch, Eds., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press and John Wiley & Sons, 2004.
- [10] F. Lewis and D. Liu, Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013.
- [11] F. Y. Wang, H. Zhang, and D. Liu, “Adaptive dynamic programming: An introduction,” *IEEE Comput. Intel. Mag.*, vol. 4, no. 2, pp. 39–47, 2009.
- [12] D. Prokhorov and D. Wunsch, “Adaptive critic designs,” *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 997–1007, 1997.
- [13] J. Lee and J. Lee, “Approximate dynamic programming strategies and their applicability for process control: A review and future directions,” *International Journal of Control Automation and Systems*, vol. 2, pp. 263–278, 2004.

- [14] S. Ferrari and R. F. Stengel, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004, ch. Model-based Adaptive Critic Designs, pp. 64–94.
- [15] P. J. Werbos, “Building and understanding adaptive systems: a statistical/numerical approach to factory automation and brain research,” *IEEE Trans. Syst. Man Cybern.*, vol. 17, no. 1, pp. 7–20, Jan. 1987.
- [16] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2007.
- [17] P. J. Werbos, “An overview of neural networks for control,” *IEEE Control Syst. Mag.*, vol. 11, no. 1, pp. 40–41, 1991.
- [18] P. J. Werbos, “Intelligence in the brain: A theory of how it works and how to build it,” *Neural Networks*, pp. 200–212, 2009.
- [19] P. J. Werbos, “Using ADP to understand and replicate brain intelligence: the next level design,” in *IEEE Int. Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 209–216.
- [20] P. J. Werbos, *Automatic Differentiation: Applications, Theory and Implementations, Lecture Notes in Computational Science and Engineering*. Springer, 2005, vol. 50, ch. Backwards Differentiation in AD and Neural Nets: Past Links and New Opportunities, pp. 15–34.
- [21] J. Graham, J. Starzyk, Z. Ni, and H. He, “Advancing motivated learning with goal creation,” in *Proc. IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI’14)*, Dec. 2014, Orlando, FL, pp. 1–8.
- [22] J. A. Starzyk, Y. Liu, and H. He, “Challenges of embodied intelligence,” in *Proc. Int. Conf. Signals and Electronic Syst.*, 2006, pp. 534–541.
- [23] J. A. Starzyk, *Frontiers in Robotics, Automation and Control*. InTech Education and Publishing, 2008, ch. Motivation in Embodied Intelligence, pp. 83–110.
- [24] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins, “Learning and sequential decision making,” in *Learning and computational neuroscience*. MIT Press, 1989, pp. 539–602.
- [25] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
- [26] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *arXiv preprint cs/9605103*, 1996.

- [27] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge Univ. Press, 1998.
- [28] D. E. Moriarty and R. Miikulainen, “Efficient reinforcement learning through symbiotic evolution,” *Machine Learning*, vol. 22, pp. 11–32, 1996.
- [29] P. J. Werbos, “Consistency of HDP applied to a simple reinforcement learning problem,” *Neural Networks*, vol. 3, no. 2, pp. 179 – 189, 1990.
- [30] Z. Ni, H. He, and X. Zhong, *Frontiers of Intelligent Control and Information Processing*. World Scientific Publishing, 2014, ch. Experimental Studies on Data-Driven Heuristic Dynamic Programming for POMDP, pp. 88–105.
- [31] K. Vamvoudakis and F. L. Lewis, “Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem,” in *Proc. IEEE Int. Conf. Neural Netw.*, 2009.
- [32] S. N. Balakrishnan, J. Ding, and F. L. Lewis, “Issues on stability of adp feedback controllers for dynamical systems,” *IEEE Trans. on Syst. Man, Cybern., Part B, special issue on ADP/RL, invited survey paper*, vol. 38, no. 4, pp. 913–917, 2008.
- [33] A. Al-Tamimi, M. Abu-Khalaf, and F. L. Lewis, “Adaptive critic designs for discrete-time zero-sum games with application to h-infinity control,” *IEEE Trans. on Syst. Man, Cybern., Part B*, vol. 37, no. 1, pp. 240–247, 2007.
- [34] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
- [35] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. NJ, USA: Wiley, 2012.
- [36] S. Singh, A. Barto, R. Grupen, and C. Connolly, “Robust reinforcement learning in motion planning,” *Advances in neural information processing systems*, pp. 655–655, 1994.
- [37] R. S. Sutton, “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” in *In Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann, 1990, pp. 216–224.
- [38] R. Sutton, “Learning to predict by the methods of temporal differences,” *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.
- [39] R. Sutton, “Temporal credit assignment in reinforcement learning,” Ph.D. dissertation, University of Massachusetts Amherst, 1984.

- [40] F. Yuan, L. Feldkamp, G. Puskorius, and L. Davis, "A Simple Solution to the Bioreactor Benchmark Problem by Application of Q-learning," in *Proc. World Congress on Neural Networks*, Washington, DC, July, 1995, pp. II-326–331.
- [41] M. Wiering and H. Van Hasselt, "Two novel on-policy reinforcement learning algorithms based on td (λ)-methods," in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2007, pp. 280–287.
- [42] P. Werbos, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Reinforcement Learning and Approximate Dynamic Programming (RLADP)–Foundations, Common Misconceptions and Challenges Ahead, pp. 3–30.
- [43] P. J. Werbos, "ADP: The key direction for future research in intelligent control and understanding brain intelligence," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 898–900, 2008.
- [44] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it," *Neural Networks*, vol. 22, no. 3, pp. 200–212, 2009.
- [45] G. G. Lendaris and J. C. Neidhoefer, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004, ch. Guidance in the Use of Adaptive Critics for Control, pp. 97–118.
- [46] S. N. Balakrishnan and V. Biega, "Adaptive Critic Based Neural Networks for Control," in *Proc. Am. Contr. Conf.*, Seattle, WA, June, 1995, pp. 335–339.
- [47] P. Dayan, "The Convergence of TD (λ) for General λ ," *Machine Learning*, vol. 8, no. 34, pp. 341–362, 1992.
- [48] P. J. Werbos, *Handbook of intelligent control*. New York: Van Nostrand Reinhold, 1992, ch. Approximate dynamic programming for real-time control and neural modeling.
- [49] J. Si and Y. T. Wang, "On-line learning control by association and reinforcement," *IEEE Trans. on Neural Netw.*, vol. 12, no. 2, pp. 264–276, 2001.
- [50] R. Enns and J. Si, "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE Transactions on Neural Networks*, vol. 14, no. 4, pp. 929–939, 2003.
- [51] L. Yang, J. Si, K. S. Tsakalis, and A. A. Rodriguez, "Direct heuristic dynamic programming for nonlinear tracking control with filtered tracking error," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 39, no. 6, pp. 1617–1622, 2009.

- [52] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Trans. on Cybernetics*, vol. 43, no. 2, pp. 779–789, 2013.
- [53] D. V. Prokhorov, "Adaptive critic designs and their applications, phd. dissertation, texas tech university, lubbock, tx," Ph.D. dissertation, Oct, 1997.
- [54] D. Liu, D. Wang, D. Zhao, Q. Wei, and N. Jin, "Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming," *IEEE Trans. on Automation Science and Engineering*, vol. 9, no. 3, pp. 628–634, 2012.
- [55] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, pp. 1825–1832, 2012.
- [56] D. Liu and D. Wang, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Optimal Control of Unknown Nonlinear Discrete-Time Systems Using the Iterative Globalized Dual Heuristic Programming Algorithm, pp. 52–74.
- [57] P. J. Werbos, *Handbook of Intelligent Control*. New York: Van Nostrand, 1992, ch. Approximate Dynamic Programming for Real Time Control and Neural Modeling, pp. 493–525.
- [58] J. Si, A. G. Barto, W. B. Powell, and D. C. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.
- [59] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. on Neural Netw.*, vol. 8, no. 5, pp. 997–1007, 1997.
- [60] F. Y. Wang, H. Zhang, and D. Liu, "Adaptive Dynamic Programming: An Introduction," *IEEE Comput. Intel. Mag.*, vol. 4, no. 2, pp. 39–47, 2009.
- [61] Z. Ni, H. He, X. Zhong, and D. Prokhorov, "Model-Free Dual Heuristic Dynamic Programming," *IEEE Trans. on Neural Networks and Learning Systems*, 2015, to be appear.
- [62] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012.
- [63] H. He, Z. Ni, and D. Zhao, "Data-driven learning and control with multiple critic networks," in *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, july 2012, pp. 523 –527.

- [64] H. G. Zhang, Q. L. Wei, and D. Liu, "An iterative approximate dynamic programming method to solve for a class of nonlinear zero-sum differential games," *Automatica*, vol. 47, no. 1, pp. 207–214, 2011.
- [65] H. G. Zhang, Q. L. Wei, and Y. H. Luo, "A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy hdp iteration algorithm," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 937–942, 2008.
- [66] F. L. Lewis, D. Liu, and G. G. Lendaris, "Special issue on adaptive dynamic programming and reinforcement learning in feedback control," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 896–897, 2008.
- [67] G. G. Lendaris, "Higher level application of adp: A next phase for the control field?" *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 901–912, 2008.
- [68] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 264–276, 2001.
- [69] D. Liu, Y. Zhang, and H. G. Zhang, "A self-learning call admission control scheme for cdma cellular networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1219–1228, 2005.
- [70] J. Si, L. Yang, and D. Liu, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004, ch. Direct Neural Dynamic Programming, pp. 125–151.
- [71] S. Haykin, *Neural networks and learning machines*. Upper Saddle River: Prentice Hall, 2009, vol. 3rd Edition.
- [72] C. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [73] Z. Ni, H. He, and J. Wen, "Adaptive learning in tracking control based on the dual critic network design," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 6, no. 24, pp. 913–928, 2013.
- [74] Z. Ni, X. Fang, H. He, D. Zhao, and X. Xu, "Real-time tracking control on adaptive critic design with uniformly ultimately bounded condition," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL'13), IEEE Symposium Series on Computational Intelligence (SSCI)*, Apr. 2013.
- [75] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Francis, London, UK, 1999, ch. Neural Network Robot Control, pp. 173–219.

- [76] P. He and S. Jagannathan, "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 37, no. 2, pp. 425–436, 2007.
- [77] J. Fu, H. He, and X. Zhou, "Adaptive learning and control for mimo system based on adaptive dynamic programming," *IEEE Trans. Neural Networks*, vol. 22, no. 7, pp. 1133–1148, 2011.
- [78] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *Computational Intelligence Magazine, IEEE*, vol. 4, no. 2, pp. 39–47, 2009.
- [79] H. Zhang, Q. Wei, and Y. Luo, "A novel infinte-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy hdp iteration algorithm," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 937–942, 2008.
- [80] D. Liu, H. Javaherian, O. Kovalenko, and T. Huang, "Adaptive critic learning techniques for engine torque and air-fuel ratio control," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 988–993, 2008.
- [81] T. L. Chien, C. C. Chen, Y. C. Huang, and W.-J. Lin, "Stability and almost disturbance decoupling analysis of nonlinear system subject to feedback linearization and feedforward neural network controller," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1220–1230, 2008.
- [82] P. H. Eaton, and D. V. Prokhorov, and D. C. Wunsch II., "Neurocontroller alternatives for fuzzy ball-and-beam systems with nonuniform nonlinear friction," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 423–435, 2000.
- [83] H. Zhang, L. Cui, X. Zhang, and Y. Luo, "Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method," *IEEE Trans. Neural Networks*, vol. 22, no. 12, pp. 2226–2236, Dec. 2011. [Online]. Available: <http://dx.doi.org/10.1109/TNN.2011.2172628>
- [84] H. Zhang, R. Song, Q. Wei, and T. Zhang, "Optimal tracking control for a class of nonlinear discrete-time systems with time delays based on heuristic dynamic programming," *IEEE Trans. Neural Networks*, vol. 22, no. 12, pp. 1851–1862, Dec. 2011. [Online]. Available: <http://dx.doi.org/10.1109/TNN.2011.2172628>
- [85] C. Burdea and P. Coiffe, *Virtual Reality Technology*. Piscataway, NJ: Wiley-IEEE Press, 2003.

- [86] F. P. Brooks, “What’s real about virtual reality?” *Computer Graphics and Applications*, vol. 19, no. 6, pp. 16–27, 1999.
- [87] R. Carey, “The virtual reality modeling language explained,” *MultiMedia*, vol. 5, no. 3, pp. 84–93, 1998.
- [88] J. Novak-Marcincin, “Application of the virtual reality modeling language for design of automated workplaces,” *World Academy of Science, Engineering and Technology*, vol. 31, pp. 160–163, 2007.
- [89] X. Fang, D. Zheng, H. He, and Z. Ni, “Data-driven heuristic dynamic programming with virtual reality,” *Neurocomputing*, 2015, in press.
- [90] X. Fang, H. He, Z. Ni, and Y. Tang, “Learning and control in virtual reality for machine intelligence,” in *Intelligent Control and Information Processing (ICICIP), 2012 Third International Conference on*, july 2012, pp. 63–67.
- [91] [Online]. <http://www.youtube.com/watch?v=OeZEDBz6ki0&feature=youtu.be>.
- [92] H. G. Zhang, Y. H. Luo, and D. Liu, “Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints,” *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1490–1503, 2009.
- [93] M. Fairbank, E. Alonso, and D. V. Prokhorov, “Simple and fast calculation of the second-order gradients for globalized dual heuristic dynamic programming in neural networks,” *IEEE Trans. Neural Netw. Learning Syst.*, vol. 23, no. 10, pp. 1671–1676, 2012.
- [94] Z. Ni, H. He, D. Zhao, X. Xu, and D. V. Prokhorov, “GrDHP: A General Utility Function Representation for Dual Heuristic Dynamic Programming,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 614–627, Mar. 2015.
- [95] Z. Ni, Y. Tang, H. He, and J. Wen, “Multi-machine power system control based on dual heuristic dynamic programming,” in *Proc. of 2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, Dec. 2014, Orlando, FL, pp. 1–7.
- [96] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, “Dual heuristic programming excitation neurocontrol for generators in a multimachine power system,” *IEEE Trans. on Industry Applications*, vol. 39, no. 2, pp. 382–394, 2003.
- [97] Z. Ni, H. He, J. Wen, and X. Xu, “Goal representation heuristic dynamic programming on maze navigation,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, no. 12, pp. 2038–2050, 2013.

- [98] Z. Ni and H. He, “Heuristic dynamic programming with internal goal representation,” *Soft Computing*, vol. 17, pp. 2101–2108, 2013.
- [99] S. Ferrari and R. F. Stengel, *Handbook of learning and approximate dynamic programming*. Wiley-IEEE Press, 2004, ch. Model-Based Adaptive Critic Designs, pp. 65–95.
- [100] D. V. Prokhorov, and R. A. Santiago, and D. C. Wunsch II., “Adaptive critic designs: A case study for neurocontrol,” *Neural Networks*., vol. 8, no. 9, pp. 1367–1372, 1995.
- [101] D. V. Prokhorov, *Adaptive Critic Designs and Their Applications, PhD. Dissertation*. Texas Tech University, Lubbock, Tx, Oct, 1997.
- [102] X. Xu, Z. Hou, C. Lian, and H. He, “Online learning control using adaptive critic designs with sparse kernel machines,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, pp. 762–775, 2013.
- [103] Y. Tang, H. He, Z. Ni, J. Wen, and X. Sui, “Reactive power control of grid-connected wind farm based on adaptive dynamic programming,” *Neurocomputing*, vol. 125, pp. 125–133, Feb. 2014.
- [104] Y. Tang, H. He, and J. Wen, “Comparative study between hdp and pss on dfig damping control,” in *Computational Intelligence Applications In Smart Grid (CIASG), 2013 IEEE Symposium on*. IEEE, 2013, pp. 59–65.
- [105] Z. Ni, H. He, D. Zhao, and D. Prokhorov, “Reinforcement learning control based on multi-goal representation using hierarchical heuristic dynamic programming,” in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012.
- [106] H. He, Z. Ni, and D. Zhao, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Learning and Optimization in Hierarchical Adaptive Critic Design, pp. 78–95.
- [107] S. Thrun, *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Boston, MA, Kluwer, 1996.
- [108] H. White and A. Gallant, “On learning the derivatives of an unknown mapping with multilayer feedforward networks,” *Neural Networks*, vol. 5, pp. 129–138, 1992.
- [109] D. Wunsch, “The cellular simultaneous recurrent network adaptive critic design for the generalized maze problem has a simple closed-form solution,” in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, vol. 3. IEEE, 2000, pp. 79–82.

- [110] P. Werbos and X. Pang, "Generalized maze navigation: SRN critics solve what feedforward or Hebbian nets cannot," in *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, vol. 3, oct 1996, pp. 1764–1769.
- [111] X. Pang and P. J. Werbos, "Neural network design for j function approximation in dynamic programming," in *Math. Modelling and Scientific Computing*, [Online] Available: <http://arxiv.org/pdf/adap-org/9806001.pdf>, 1996.
- [112] Z. Ni, Y. Tang, X. Sui, H. He, and J. Wen, "An adaptive neuro-control approach for multi-machine power systems," *International Journal of Electrical Power and Energy Systems*, 2015, under review.
- [113] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1997.
- [114] R. Ilin, R. Kozma, and P. Werbos, "Cellular SRN trained by extended Kalman filter shows promise for ADP," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2006, pp. 506–510.
- [115] R. Ilin, R. Kozma, and P. Werbos, "Efficient learning in cellular simultaneous recurrent neural networks-the case of maze navigation problem," in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2007, pp. 324–329.
- [116] R. Ilin, R. Kozma, and P. Werbos, "Beyond feedforward models trained by backpropagation: A practical training tool for a more efficient universal approximator," *Neural Networks, IEEE Transactions on*, vol. 19, no. 6, pp. 929–937, 2008.
- [117] P. Werbos, "Computational intelligence for the smart grid-history, challenges, and opportunities," *Computational Intelligence Magazine, IEEE*, vol. 6, no. 3, pp. 14–21, 2011.
- [118] R. G. Harley and J. Liang, "Computational intelligence in smart grids," in *Computational Intelligence Applications In Smart Grid (CIASG), 2011 IEEE Symposium on*.
- [119] G. K. Venayagamoorthy, K. Rohrig, and I. Erlich, "One step ahead: short-term wind power forecasting and intelligent predictive control based on data analytics," *Power and Energy Magazine, IEEE*, vol. 10, no. 5, pp. 70–78, 2012.
- [120] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 764–773, 2002.

- [121] W. Qiao, R. G. Harley, and G. K. Venayagamoorthy, "Coordinated reactive power control of a large wind farm and a statcom using heuristic dynamic programming," *Energy Conversion, IEEE Transactions on*, vol. 24, no. 2, pp. 493–503, 2009.
- [122] C. Lu, J. Si, and X. Xie, "Direct heuristic dynamic programming for damping oscillations in a large power system," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 1008–1013, 2008.
- [123] D. Molina, G. K. Venayagamoorthy, J. Liang, and R. G. Harley, "Intelligent local area signals based damping of power system oscillations using virtual generators and approximate dynamic programming," *Smart Grid, IEEE Transactions on*, vol. 4, no. 1, pp. 498–508, 2013.
- [124] X. Fu, S. Li, M. Fairbank, D. C. Wunsch, and E. Alonso, "Training recurrent neural networks with the levenberg–marquardt algorithm for optimal control of a grid-connected converter," *Neural Networks and Learning Systems, IEEE Transactions on*, 2015, in press.
- [125] S. Mohagheghi, Y. Del Valle, G. K. Venayagamoorthy, and R. G. Harley, "A proportional-integrator type adaptive critic design-based neurocontroller for a static compensator in a multimachine power system," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 1, pp. 86–96, 2007.
- [126] S. Mohagheghi, G. K. Venayagamoorthy, and R. G. Harley, "Optimal wide area controller and state predictor for a power system," *Power Systems, IEEE Transactions on*, vol. 22, no. 2, pp. 693–705, 2007.
- [127] C. Wei, Z. Zhang, W. Qiao, and L. Qu, "Reinforcement learning-based intelligent maximum power point tracking control for wind energy conversion systems," *Industrial Electronics, IEEE Transactions on*, 2015, in press.
- [128] Y. Tang, H. He, Z. Ni, and J. Wen, "Optimal operation for energy storage with wind power generation using adaptive dynamic programming," in *Proc. IEEE Power and Energy Society General Meeting (IEEE PES General Meeting'15)*, Jul., Denver, CO, pp. 1–6.
- [129] X. Sui, Y. Tang, H. He, and J. Wen, "Energy-Storage-Based Low-Frequency Oscillation Damping Control Using Particle Swarm Optimization and Heuristic Dynamic Programming," *IEEE Transactions on Power Systems*, 2015, in press.
- [130] Y. Tang, P. Ju, H. He, C. Qin, and F. Wu, "Optimized control of dfig-based wind generation using sensitivity analysis and particle swarm optimization," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 509–520, 2013.

- [131] W. Yao, J. Wen, S. Chen, and L. Jiang, "Development of matlab/simulink based power system transient stability analysis toolbox," *Power System Technology (Chinese)*, vol. 36, no. 6, pp. 95–101, 2012.
- [132] S. Ferrari and R. Stengel, "Online adaptive critic flight control," *Journal of Guidance Control and Dynamics*, vol. 27, no. 5, pp. 777–786, 2004.
- [133] S. Ferrari, J. Steck, and R. Chandramohan, "Adaptive feedback control by constrained approximate dynamic programming," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 982–987, 2008.
- [134] P. Shih, B. C. Kaul, S. Jagannathan, J. Drallmeier, *et al.*, "Reinforcement-learning-based dual-control methodology for complex nonlinear discrete-time systems with application to spark engine egr operation," *Neural Networks, IEEE Transactions on*, vol. 19, no. 8, pp. 1369–1388, 2008.
- [135] H. Bui, B. C. Kaul, J. Sarangapani, J. Drallmeier, and P. He, "Neuro emission controller for minimizing cyclic dispersion in spark ignition engines with egr levels," 2004.
- [136] J. B. Vance, B. C. Kaul, S. Jagannathan, J. Drallmeier, *et al.*, "Output feedback controller for operation of spark ignition engines at lean conditions using neural networks," *Control Systems Technology, IEEE Transactions on*, vol. 16, no. 2, pp. 214–228, 2008.
- [137] J. Fu, H. He, Q. Liu, and Z. Ni, "An adaptive dynamic programming approach for closely-coupled mimo system control," in *The 8th International Symposium on Neural Networks (ISNN)*, vol. 6677, 2011, pp. 1–10.
- [138] J. Fu, H. He, and Z. Ni, "Adaptive dynamic programming with balanced weights seeking strategy," in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, april 2011, pp. 210–217.
- [139] P. He and S. Jagannathan, "Reinforcement-based neuro-output feedback control of nonlinear discrete-time systems with input constraints," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 35, no. 1, pp. 150–154, 2005.
- [140] S. Jagannathan and P. He, "Neural network-based state feedback control of nonlinear discrete-time system in non-strict feedback form," *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 2073–2087, 2008.
- [141] J. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, New Jersey, 1991.

- [142] H. Khalil, *Nonlinear Systems (3rd edition)*. Prentice Hall, New Jersey, 2001.
- [143] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, “Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof,” *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 943–949, 2008.
- [144] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, “A boundedness result for the direct heuristic dynamic programming,” *Neural Networks*, vol. 32, pp. 229–235, 2012.
- [145] X. Zhong, Z. Ni, H. He, X. Xu, and D. Zhao, “Event-triggered reinforcement learning approach for unknown nonlinear continuous-time system,” in *Proc. Int. Joint Conf. Neural Networks (IJCNN’14)*, Jul. 2014, Beijing, China, pp. 1–8.
- [146] X. Zhong, H. He, H. Zhang, and Z. Wang, “Optimal control for unknown discrete-time nonlinear markov jump systems using adaptive dynamic programming,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2141–2155, 2014.
- [147] Y. Jiang and Z.-P. Jiang, “Robust adaptive dynamic programming and feedback stabilization of nonlinear systems,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 5, pp. 882–893, May 2014.
- [148] S. Ray, G. K. Venayagamoorthy, B. Chaudhuri, and R. Majumder, “Comparison of adaptive critics and classical approaches based wide area controllers for a power system,” *IEEE Trans. on Syst. Man, Cybern., Part B*, vol. 38, no. 4, pp. 1002–1007, 2008.
- [149] X. Zhong, H. He, H. Zhang, and Z. Wang, “A neural network based online learning and control approach for markov jump systems,” *Neurocomputing*, vol. 149, pp. 116–123, 2015.
- [150] A. Bernstein, “Adaptive state aggregation for reinforcement learning,” Master’s thesis, Israel Institute of Technology, 2007. [Online]. Available: <http://eprints.pascal-network.org/archive/00005853/>
- [151] A. L. Strehl and M. L. Littman, “A theoretical analysis of model-based interval estimation,” in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 856–863.
- [152] A. L. Strehl, L. Li, and M. L. Littman, “Reinforcement learning in finite MDPs: PAC analysis,” *The Journal of Machine Learning Research*, vol. 10, pp. 2413–2444, 2009.
- [153] A. Nouri and M. L. Littman, “Multi-resolution exploration in continuous spaces.” in *NIPS*, 2008, pp. 1209–1216.

- [154] A. Bernstein and N. Shimkin, “Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains,” *Machine learning*, vol. 81, no. 3, pp. 359–397, 2010.
- [155] D. Zhao and Y. Zhu, “MEC—A Near-Optimal Online Reinforcement Learning Algorithm for Continuous Deterministic Systems,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 2, pp. 346–356, Feb. 2015.

APPENDIX A

Pseudo Code of GrHDP based Tracking Control Algorithm

*Algorithm level of implementation of GrHDP based tracking control
(Chapter 4)*

$/* \hat{f}(\mathbf{x}) \Leftarrow \text{ActNet}(\mathbf{x}, \mathbf{w}_a)$, nonlinear function approximation with the action network;

ActNet: the action network;

$\mathbf{x}$: state vector;

$\mathbf{w}_a$: weights of *ActNet*;

$\hat{f}(\mathbf{x})$: nonlinear system function approximation, the output of *ActNet*;

$\mathbf{u} \Leftarrow \text{Filter}(\mathbf{x}_d, \mathbf{x}, \hat{f}(\mathbf{x}))$, control action calculation;

Filter: the tracking filter;

$\mathbf{x}_d$: the desired reference signal;

$\mathbf{u}$: control action;

$\mathbf{s} \Leftarrow \text{RefNet}(\mathbf{x}, \mathbf{u}, \mathbf{w}_r)$, internal goal representation with the reference network;

RefNet: the reference network;

$\mathbf{w}_r$: weights of the *RefNet*;

$\mathbf{s}$: internal goal signal;

$\mathbf{J} \Leftarrow \text{CritNet}(\mathbf{x}, \mathbf{u}, \mathbf{s}, \mathbf{w}_c)$, total cost-to-go signal approximated by the critic network;

CritNet: the critic network;

$\mathbf{w}_c$: weights of the *CritNet*;

$\mathbf{J}$: total cost-to-go signal, the output of the critic network; */

/*Note: the parameters N_r , T_r , η_r , N_c , T_c , η_c , N_a , T_a , η_a are all defined in Table 4.1; */

- 1) Initiate $\mathbf{x}(0)$
- 2) Uniformly randomize $\mathbf{w}_a(0)$, $\mathbf{w}_r(0)$, $\mathbf{w}_c(0)$ in $[-1, 1]$
- 3) $\hat{f}(\mathbf{x}(0)) \Leftarrow ActNet(\mathbf{x}(0), \mathbf{w}_a(0))$
- 4) $\mathbf{u}(0) \Leftarrow Filter(\mathbf{x}_d, \mathbf{x}(0), \hat{f}(\mathbf{x}(0)))$
- 5) $\mathbf{s}(0) \Leftarrow RefNet(\mathbf{x}(0), \mathbf{u}(0), \mathbf{w}_r(0))$
- 6) $J(0) \Leftarrow CritNet(\mathbf{x}(0), \mathbf{u}(0), \mathbf{s}(0), \mathbf{w}_c(0))$
- 7) $J_{prev} = J(0)$
- 8) **for** 1 to MaxStep **do**;
- 9) //weights are carried on through the whole learning process;
- 10) $CurrentState \Leftarrow (\mathbf{x}(\mathbf{k} - 1), \mathbf{u}(\mathbf{k} - 1));$ //obtain current state vectors from the external environment
- 11) $\mathbf{w}_a(k) = \mathbf{w}_a(k - 1);$
- 12) $\mathbf{w}_c(k) = \mathbf{w}_c(k - 1);$
- 13) $\mathbf{w}_r(k) = \mathbf{w}_r(k - 1);$
- 14) $\hat{f}(\mathbf{x}(k)) \Leftarrow ActNet(\mathbf{x}(k), \mathbf{w}_a(k));$
- 15) $\mathbf{u}(k) \Leftarrow Filter(\mathbf{x}_d, \mathbf{x}(k), \hat{f}(\mathbf{x}(k)));$
- 16) $\mathbf{s}(k) \Leftarrow RefNet(\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}_r(k));$
- 17) $J(k) \Leftarrow CritNet(\mathbf{x}(k), \mathbf{u}(k), \mathbf{s}(k), \mathbf{w}_c(k));$
- 18) Obtain the tracking error $\bar{e}(k)$ via (4.3)

```

19)      if  $\|\bar{e}(k)\| < c$  then
20)           $r(k) = 0$ ; // reward
21)      else
22)           $r(k) = -1$ ; // punishment
23)      end if //corresponding to step 19
24)      if  $Step \geq 100$  then
25)          calculate  $MSE$  via equation (4.46);
26)          if  $MSE < threshold$  then
27)               $\eta_r(k), \eta_c(k), \eta_a(k)$  are divided by 6, respectively
28)               $threshold = threshold/2$ ; //update
29)          elseif  $MSE \geq 4 * threshold$  then
30)               $\eta_r(k), \eta_c(k), \eta_a(k)$  and  $threshold$  are reset to the initial
values, respectively;
31)          end if; //corresponding to step 26
32)      end if; //corresponding to step 24
33)       $E_r(k) = \frac{1}{2}(\alpha J(k) - (J(k-1) - r(k)))^2$ ;
34)       $cyc = 0$ ;
35)      while ( $E_r(k) > T_r \& cyc > N_r$ ) do
// update the weights recursively;
36)           $\mathbf{w}_r(k) = \mathbf{w}_r(k) + \Delta \mathbf{w}_r(k)$  via (4.20) and (4.23);
// update the  $\mathbf{s}(k)$ ,  $J(k)$ ,  $E_r(k)$ ,  $cyc$  correspondingly
37)           $\mathbf{s}(k) \Leftarrow RefNet(\mathbf{x}(k), \mathbf{u}(\mathbf{k}), \mathbf{w}_r(k))$ ;
38)           $J(k) \Leftarrow CritNet(\mathbf{x}(k), \mathbf{u}(\mathbf{k}), \mathbf{s}(\mathbf{k}), \mathbf{w}_c(k))$ ;
39)           $E_r(k) = \frac{1}{2}(\alpha J(k) - (J(k-1) - r(k)))^2$ ;
40)           $cyc = cyc + 1$ ;
41)      end while // online learning of the reference network

```

```

42)       $E_c(k) = \frac{1}{2}(\alpha J(k) - (J(k-1) - s(k)))^2;$ 
43)       $cyc = 0;$ 
44)      while ( $E_c(k) > T_c \& cyc > N_c$ ) do
45)           $\mathbf{w}_c(k) = \mathbf{w}_c(k) + \Delta \mathbf{w}_c(k)$  via (4.27) and (4.30);
46)           $J(k) \Leftarrow CritNet(\mathbf{x}(k), \mathbf{u}(\mathbf{k}), \mathbf{s}(\mathbf{k}), \mathbf{w}_c(k));$ 
47)           $E_c(k) = \frac{1}{2}(\alpha J(k) - (J(k-1) - s(k)))^2;$ 
48)           $cyc = cyc + 1;$ 
49)      end while // online learning of the critic network
50)       $E_a(k) = \frac{1}{2}(\alpha J(k) + \hat{f}(x(k)) - U_c)^2;$ 
51)       $cyc = 0;$ 
52)      while ( $E_a(k) > T_a \& cyc > N_a$ ) do
53)           $\mathbf{w}_a(k) = \mathbf{w}_a(k) + \Delta \mathbf{w}_a(k)$  via (4.36) and (4.40);
          // update the  $\hat{f}(x(k))$ ,  $u(k)$ ,  $\mathbf{s}(k)$ ,  $J(k)$ ,  $E_r(k)$ ,  $cyc$  correspondingly
54)           $\hat{f}(\mathbf{x}(k)) \Leftarrow ActNet(\mathbf{x}(k), \mathbf{w}_a(k));$ 
55)           $\mathbf{u}(\mathbf{k}) \Leftarrow Filter(\mathbf{x}_d, \mathbf{x}(\mathbf{k}), \hat{f}(\mathbf{x}(\mathbf{k})));$ 
56)           $\mathbf{s}(k) \Leftarrow RefNet(\mathbf{x}(k), \mathbf{u}(\mathbf{k}), \mathbf{w}_r(k));$ 
57)           $J(k) \Leftarrow CritNet(\mathbf{x}(k), \mathbf{u}(\mathbf{k}), \mathbf{s}(\mathbf{k}), \mathbf{w}_c(k));$ 
58)           $E_a(k) = \frac{1}{2}(\alpha J(k) + \hat{f}(x(k)) - U_c)^2;$ 
59)           $cyc = cyc + 1;$ 
60)      end while // online learning of the action network
61)  end for //corresponding to step 8

```

APPENDIX B

Stability Analysis for GrHDP-based Tracking Control

It is assumed ϕ_a for the output of the hidden neurons in the action network s.t. $\phi_a(k) = [\phi_{a,1}(k) \ \phi_{a,2}(k) \ \dots \ \phi_{a,N_{ah}}(k)]^T$ and then $\|\phi_a(k)\|^2 = \phi_a^T(k)\phi_a(k)$. For the critic network and reference network, it is defined with the similar notations s.t. ϕ_c and ϕ_r for the output of the hidden neurons in the critic network, and reference network, respectively. Let the components in ϕ_a , ϕ_c and ϕ_r are bounded in a certain range [76, 139, 140, 75]. The estimated weights in networks are denoted as $\hat{\omega}_a$, $\hat{\omega}_c$, and $\hat{\omega}_r$, while the expected weights are denoted as ω_a , ω_c and ω_r . Therefore, the differences are defined as $\tilde{\omega}_a = \hat{\omega}_a - \omega_a$, $\tilde{\omega}_c = \hat{\omega}_c - \omega_c$ and $\tilde{\omega}_r = \hat{\omega}_r - \omega_r$. Although $\hat{\omega}_a$, $\hat{\omega}_c$, and $\hat{\omega}_r$ are unknown parameters, they have upper bounds in this analysis because if they exceed the preset upper bounds, they will be normalized according to (4.41)-(4.43).

Define the Lyapunov function candidate as follows

$$V = V_1 + V_2 + V_3 + V_4 + V_5 \quad (\text{B.1})$$

where

$$V_1 = \frac{1}{\gamma_1} \bar{e}^T(k) \bar{e}(k) \quad (\text{B.2})$$

$$V_2 = \frac{1}{\eta_c} \text{tr}(\tilde{\omega}_c^T(k) \tilde{\omega}_c(k)) \quad (\text{B.3})$$

$$V_3 = \frac{1}{2} \|\zeta_c(k-1)\|^2 \quad (\text{B.4})$$

$$V_4 = \frac{1}{\gamma_2 \eta_a} \text{tr}(\tilde{\omega}_a^T(k) \tilde{\omega}_a(k)) \quad (\text{B.5})$$

$$V_5 = \frac{1}{\gamma_3 \eta_r} \text{tr}(\tilde{\omega}_r^T(k) \tilde{\omega}_r(k)) \quad (\text{B.6})$$

In (B.4), $\zeta_c(k-1) = (\hat{\omega}_c(k-1) - \omega_c)^T \phi_1(k-1) = \tilde{\omega}_c^T(k-1)\phi_1(k-1)$ and $\gamma_i > 0$ for $i = 1, 2, 3$.

The first difference of the Lyapunov function candidate can be written as

$$\Delta V = \Delta V_1 + \Delta V_2 + \Delta V_3 + \Delta V_4 + \Delta V_5 \quad (\text{B.7})$$

For ΔV_1 , it is derived as

$$\bar{e}(k+1) = K_v \bar{e}(k) - \tilde{f}(x(k)) + d(k). \quad (\text{B.8})$$

where

$$\begin{aligned} \tilde{f}(x(k)) &= f(x(k)) - \hat{f}(x(k)) \\ f(x(k)) &= \omega_a^T(k)\phi_a(k) + \varepsilon_a(x(k)) \\ \tilde{f}(x(k)) &= (\hat{\omega}_a(k) - \omega_a(k))^T \phi_a(k) + \varepsilon_a(x(k)) \\ &= \tilde{\omega}_a^T(k)\phi_a(k) + \varepsilon_a(x(k)) \end{aligned} \quad (\text{B.9})$$

Therefore

$$\begin{aligned} \Delta V_1 &= \frac{1}{\gamma_1} (\bar{e}^T(k+1)\bar{e}(k+1) - \bar{e}^T(k)\bar{e}(k)) \\ &= \frac{1}{\gamma_1} [(K_v \bar{e}(k) + \zeta_a(k) + \varepsilon_a(k) + d(k))^T \\ &\quad \cdot (K_v \bar{e}(k) + \zeta_a(k) + \varepsilon_a(k) + d(k)) - \bar{e}^T(k)\bar{e}(k)] \\ &\leq \frac{1}{\gamma_1} [3(K_v^2 \|\bar{e}(k)\|^2 + \|\zeta_a(k)\|^2 + \|\varepsilon_a(k) + d(k)\|^2) - \|\bar{e}(k)\|^2] \\ &\leq \frac{3}{\gamma_1} ((K_{vmax}^2 - \frac{1}{3}) \|\bar{e}(k)\|^2 + \|\zeta_a(k)\|^2 + \|\varepsilon_a(k) + d(k)\|^2) \end{aligned} \quad (\text{B.10})$$

where K_{vmax} is the maximum eigenvalue of K_v .

For ΔV_2 , it is derived as

$$\begin{aligned} \hat{\omega}_c(k+1) &= \hat{\omega}_c(k) - \eta_c \alpha \phi_c(k) (\alpha \hat{\omega}_c^T(k) \phi_c(k) + s(k) \\ &\quad - \hat{\omega}_c^T(k-1) \phi_c(k-1))^T \end{aligned} \quad (\text{B.11})$$

and the corresponding $\tilde{\omega}_c(k+1)$ can be expressed as

$$\begin{aligned}
\tilde{\omega}_c(k+1) &= \tilde{\omega}_c(k) - \eta_c \alpha \phi_c(k) (\alpha \hat{\omega}_c^T(k) \phi_c(k) + s(k) \\
&\quad - \hat{\omega}_c^T(k-1) \phi_c(k-1))^T \\
&= (I - \eta_c \alpha^2 \phi_c(k) \phi_c^T(k)) \tilde{\omega}_c(k) - \eta_c \alpha \phi_c(k) \\
&\quad (\alpha \omega_c^T \phi_c(k) + s(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1))^T
\end{aligned} \tag{B.12}$$

Then

$$\begin{aligned}
\Delta V_2 &= \frac{1}{\eta_c} \text{tr}(\tilde{\omega}_c^T(k+1) \tilde{\omega}_c(k+1) - \tilde{\omega}_c^T(k) \tilde{\omega}_c(k)) \\
&= \frac{1}{\eta_c} \text{tr}(\tilde{\omega}_c^T(k) A^T A \tilde{\omega}_c(k) - \tilde{\omega}_c^T(k) \tilde{\omega}_c(k) \\
&\quad + B \alpha^2 \eta_c^2 \phi_c^T(k) \phi_c(k) B^T - \tilde{\omega}_c^T(k) A^T \eta_c \alpha \phi_c(k) B^T \\
&\quad - B \eta_c \alpha \phi_c^T(k) A \tilde{\omega}_c(k))
\end{aligned} \tag{B.13}$$

where $A = I - \eta_c \alpha^2 \phi_c(k) \phi_c^T(k)$ and $B = \alpha \omega_c^T \phi_c(k) + s(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1)$

Note:

$$\begin{aligned}
&\tilde{\omega}_c^T(k) A^T A \tilde{\omega}_c(k) - \tilde{\omega}_c^T(k) \tilde{\omega}_c(k) \\
&= \tilde{\omega}_c^T(k) [(I - \eta_c \alpha^2 \phi_c \phi_c^T)^T (I - \eta_c \alpha^2 \phi_c \phi_c^T)] \tilde{\omega}_c(k) - \tilde{\omega}_c^T(k) \tilde{\omega}_c(k) \\
&= -\eta_c \alpha^2 \|\zeta_c(k)\|^2 - \eta_c \alpha^2 \tilde{\omega}_c^T(k) \phi_c \phi_c^T (I - \eta_c \alpha^2 \phi_c \phi_c^T) \tilde{\omega}_c(k)
\end{aligned} \tag{B.14}$$

where $\zeta_c(k) = \tilde{\omega}_c^T(k) \phi_c(k)$.

Therefore

$$\begin{aligned}
\Delta V_2(k) &= -\alpha^2 \|\zeta_c(k)\|^2 - \alpha^2 (1 - \eta_c \alpha^2 \|\phi_c(k)\|^2) \|\zeta_c(k)\|^2 \\
&\quad + \eta_c \alpha^2 \|\phi_c(k)\|^2 \cdot \|\alpha \omega_c^T \phi_c(k) + s(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1)\|^2 \\
&\quad - 2 \text{tr}[\alpha (I - \eta_c \alpha^2 \|\phi_c(k)\|^2) \zeta_c(k) \\
&\quad \cdot (\alpha \omega_c^T \phi_c(k) + s(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1))^T]
\end{aligned} \tag{B.15}$$

We would like to seek the upper bound of (B.15) by applying Cauchy-Schwarz inequality for the fourth term. Therefore, we have

$$\begin{aligned}
& \Delta V_2(k) \\
& \leq -\alpha^2 \|\zeta_c(k)\|^2 - \alpha^2(1 - \eta_c \alpha^2 \|\phi_c(k)\|^2) \cdot \|\zeta_c(k)\|^2 \\
& + \left\| \alpha \omega_c^T \phi_c(k) + s(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1) \right\|^2 \\
& - \alpha^2(1 - \eta_c \alpha^2 \|\phi_c(k)\|^2) \\
& \cdot \left\| \zeta_c(k) + \omega_c^T \phi_c(k) + \alpha^{-1} s(k) - \alpha^{-1} \hat{\omega}_c^T(k-1) \phi_c(k-1) \right\|^2
\end{aligned} \tag{B.16}$$

From (B.16), we can further get

$$\begin{aligned}
& \Delta V_2(k) \\
& \leq -\alpha^2 \|\zeta_c(k)\|^2 + \frac{1}{2} \|\zeta_c(k-1)\|^2 \\
& + 2 \left\| \alpha \omega_c^T \phi_c(k) + s(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1) \right\|^2 \\
& - \alpha^2(1 - \eta_c \alpha^2 \|\phi_c(k)\|^2) \\
& \cdot \left\| \zeta_c(k) + \omega_c^T \phi_c(k) + \alpha^{-1} s(k) - \alpha^{-1} \hat{\omega}_c^T(k-1) \phi_c(k-1) \right\|^2
\end{aligned} \tag{B.17}$$

For $\Delta V_3(k)$, we have

$$\Delta V_3(k) = \frac{1}{2} (\|\zeta_c(k)\|^2 - \|\zeta_c(k-1)\|^2) \tag{B.18}$$

For $\Delta V_4(k)$, we have

$$\begin{aligned}
& \hat{\omega}_a(k+1) = \hat{\omega}_a(k) - \eta_a \phi_a(k) \hat{\omega}_c^T(k) C_a(k) \\
& \cdot (\hat{\omega}_c^T(k) \phi_c(k) + K_v \bar{e}(k) - \bar{e}(k+1))^T
\end{aligned} \tag{B.19}$$

where $C_a(k)$ is a $N_{ch} \times 1$ vector and its elements can be defined as $C_a(k) = \frac{1}{2}(1 - \phi_c(k))\omega_{c,n+1}(k)$. And then

$$\begin{aligned}
& \tilde{\omega}_a(k+1) = \tilde{\omega}_a(k) - \eta_a \phi_a(k) \hat{\omega}_c^T(k) C_a(k) \\
& \cdot (\hat{\omega}_c^T(k) \phi_c(k) + K_v \bar{e}(k) - \bar{e}(k+1))^T
\end{aligned} \tag{B.20}$$

With tracking error dynamics, we can rewrite (B.20) as

$$\begin{aligned}\tilde{\omega}_a(k+1) &= \tilde{\omega}_a(k) - \eta_a \phi_a(k) \hat{\omega}_c^T(k) C_a(k) \\ &\quad \cdot (\hat{\omega}_c^T(k) \phi_c(k) + \zeta_a(k) - \varepsilon_a(k) - d(k))^T\end{aligned}\tag{B.21}$$

Therefore, we have

$$\begin{aligned}\Delta V_4(k) &= \frac{1}{\gamma_2 \eta_a} \text{tr} [\tilde{\omega}_a^T(k+1) \tilde{\omega}_a(k+1) - \tilde{\omega}_a^T(k) \tilde{\omega}_a(k)] \\ &= \frac{1}{\gamma_2} \text{tr} \left[-2 \hat{\omega}_c^T(k) C_a(k) \zeta_a(k) \right. \\ &\quad \cdot (\hat{\omega}_c^T(k) \phi_c^T(k) + \zeta_a(k) - \varepsilon_a(k) - d(k))^T \\ &\quad + \eta_a \phi_a^2(k) \cdot \|\hat{\omega}_c^T(k) C_a(k)\|^2 \\ &\quad \left. \cdot \|\hat{\omega}_c^T(k) \phi_c(k) + \zeta_a(k) - \varepsilon_a(k) - d(k)\|^2 \right]\end{aligned}\tag{B.22}$$

In order to further simplify the formula, we can rewrite (B.22) as follows

$$\begin{aligned}\Delta V_4(k) &= \frac{1}{\gamma_2} \text{tr} [-(1 - \eta_a \phi_a^2(k)) \|D\|^2 \|E\|^2 \\ &\quad + \|D \cdot E - \zeta_a(k)\|^2 - \|\zeta_a(k)\|^2]\end{aligned}\tag{B.23}$$

where

$$D = \hat{\omega}_c^T(k) C_a(k)\tag{B.24}$$

$$E = \hat{\omega}_c^T(k) \phi_c(k) + \zeta_a(k) - \varepsilon_a(k) - d(k)\tag{B.25}$$

Applying the Cauchy-Schwarz inequality for (B.23), we have

$$\Delta V_4(k) \leq \frac{1}{\gamma_2} \text{tr} [F + 2 \cdot \|D \cdot E\|^2 + \|\zeta_a(k)\|^2]\tag{B.26}$$

where

$$F = -(1 - \eta_a \|\phi_a(k)\|^2) \|D\|^2 \|E\|^2.\tag{B.27}$$

For $\Delta V_5(k)$, we have

$$\begin{aligned} \tilde{\omega}_r(k+1) = & \tilde{\omega}_r(k) - \eta_r \phi_r(k) \hat{\omega}_c^T(k) C_r(k) \\ & \cdot (\alpha \hat{\omega}_c^T(k) \phi_c(k) + r(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1))^T \end{aligned} \quad (\text{B.28})$$

where $C_r(k)$ is a $N_{ch} \times 1$ vector and its elements can be defined as $C_r(k) = \frac{1}{2}(1 - \phi_c(k))\omega_{c,n+2}(k)$.

Therefore

$$\begin{aligned} \Delta V_5(k) = & \frac{1}{\gamma_3 \eta_r} \text{tr} [\tilde{\omega}_r^T(k+1) \tilde{\omega}_r(k+1) - \tilde{\omega}_r^T(k) \tilde{\omega}_r(k)] \\ = & \frac{1}{\gamma_3} \text{tr} \left[-2 \hat{\omega}_r^T(k) C_r(k) \zeta_r(k) \right. \\ & \cdot (\alpha \hat{\omega}_c^T(k) \phi_c(k) + r(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1))^T \\ & + \eta_r \phi_r(k) \|\hat{\omega}_r^T(k) C_r(k)\|^2 \\ & \left. \cdot \|\alpha \hat{\omega}_c^T(k) \phi_c(k) + r(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1)\|^2 \right] \end{aligned} \quad (\text{B.29})$$

We can also simplify (B.29) by rewriting as

$$\Delta V_5(k) = \frac{1}{\gamma_3} \text{tr} [H + \|G \cdot I - \zeta_r(k)\|^2 - \|\zeta_r(k)\|^2] \quad (\text{B.30})$$

where

$$G = (\alpha \hat{\omega}_c^T(k) \phi_c(k) + r(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1))^T. \quad (\text{B.31})$$

$$H = -(1 - \eta_r \|\phi_r(k)\|^2) \|G\|^2 \|I\|^2. \quad (\text{B.32})$$

$$I = \hat{\omega}_r^T(k) C_r(k). \quad (\text{B.33})$$

Applying the Cauchy-Schwarz inequality for (B.30), we have

$$\Delta V_5(k) \leq \frac{1}{\gamma_3} \text{tr} [H + 2 \cdot \|G \cdot I\|^2 + \|\zeta_r(k)\|^2] \quad (\text{B.34})$$

Substituting (B.10), (B.17), (B.18), (B.26), (B.34) and into (B.1), we can get the first difference of the Lyapunov function candidate as (B.35)

$$\begin{aligned}
& \Delta V(k) \\
& \leq \frac{3}{\gamma_1} \left((K_{vmax}^2 - \frac{1}{3}) \|\bar{e}(k)\|^2 + \|\zeta_a(k)\|^2 + \|\varepsilon_a(k) + d(k)\|^2 \right) \\
& \quad - \alpha^2 \|\zeta_c(k)\|^2 + 2 \left\| \alpha \omega_c^T \phi_c(k) + s(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1) \right\|^2 \\
& \quad - \alpha^2 (1 - \eta_c \alpha^2 \|\phi_c(k)\|^2) \\
& \quad \cdot \left\| \zeta_c(k) + \omega_c^T \phi_c(k) + \alpha^{-1} s(k) - \alpha^{-1} \hat{\omega}_c^T(k-1) \phi_c(k-1) \right\|^2 \\
& \quad + \frac{1}{2} \|\zeta_c(k-1)\|^2 + \frac{1}{2} (\|\zeta_c(k)\|^2 - \|\zeta_c(k-1)\|^2) \\
& \quad + \frac{1}{\gamma_2} \text{tr} \left[F + (2 \cdot \|D \cdot E\|^2) + \|\zeta_a(k)\|^2 \right] \\
& \quad + \frac{1}{\gamma_3} \text{tr} \left[H + (2 \cdot \|G \cdot I\|^2) + \|\zeta_r(k)\|^2 \right]
\end{aligned} \tag{B.35}$$

In order to express (B.35) in a clear way, we rewrite it as

$$\begin{aligned}
& \Delta V(k) \\
& \leq \frac{3}{\gamma_1} \left((K_{vmax}^2 - \frac{1}{3}) \|\bar{e}(k)\|^2 + \|\zeta_a(k)\|^2 + \|\varepsilon_a(k) + d(k)\|^2 \right) \\
& \quad - (\alpha^2 - \frac{1}{2}) \|\zeta_c(k)\|^2 - \alpha^2 \left(I - \eta_c \alpha^2 \|\phi_c(k)\|^2 \right) \\
& \quad \cdot \left\| \zeta_c(k) + \omega_c^T \phi_c(k) + \alpha^{-1} s(k) - \alpha^{-1} \hat{\omega}_c^T(k-1) \phi_c(k-1) \right\|^2 \\
& \quad - \frac{1}{\gamma_2} (1 - \eta_a \|\phi_a(k)\|^2) \|D\|^2 \|E\|^2 \\
& \quad - \frac{1}{\gamma_3} (1 - \eta_r \|\phi_r(k)\|^2) \|G\|^2 \|I\|^2 \\
& \quad + Z^2.
\end{aligned} \tag{B.36}$$

where Z^2 is defined as

$$\begin{aligned}
Z^2 = & \frac{1}{\gamma_2} \|\zeta_a(k)\|^2 + \frac{1}{\gamma_3} \|\zeta_r(k)\|^2 \\
& + 2 \left\| \alpha \omega_c^T \phi_c(k) + s(k) - \hat{\omega}_c^T(k-1) \phi_c(k-1) \right\|^2 \\
& + \frac{1}{\gamma_2} (2 \cdot \|D \cdot E\|^2) + \frac{1}{\gamma_3} (2 \cdot \|G \cdot I\|^2)
\end{aligned} \tag{B.37}$$

And the upper bound for Z^2 is

$$\begin{aligned}
Z_2 \leq & \frac{2}{\gamma_2} \|\omega_{am}^T \phi_{am}\|^2 + \frac{2}{\gamma_3} \|\omega_{rm}^T \phi_{rm}\|^2 \\
& + 6(\alpha^2 + 1) \cdot \|\omega_{cm}^T \phi_{cm}\|^2 + 6s_m^2 + \frac{6}{\gamma_2} \|\omega_{cm}^T C_{am}\|^2 \\
& \cdot (\|\omega_{cm}^T \phi_{cm}\|^2 + 2 \|\omega_{am}^T \phi_{am}\|^2 + \|\varepsilon_{am} + d_m\|^2) \\
& + \frac{6}{\gamma_3} \|\omega_{rm}^T C_{rm}\|^2 \cdot ((\alpha^2 + 1) \cdot \|\omega_{cm}^T \phi_{cm}\|^2 + \|r_m\|^2) \\
= & 6 \cdot \left(\alpha^2 + 1 + \frac{1}{\gamma_2} \|\omega_{cm}^T C_{am}\|^2 + \frac{1}{\gamma_3} (\alpha^2 + 1) \cdot \|\omega_{rm}^T C_{rm}\|^2 \right) \\
& \cdot \|\omega_{cm}^T \phi_{cm}\|^2 + \frac{2}{\gamma_2} \left(1 + 6 \|\omega_{cm}^T C_{am}\|^2 \right) \cdot \|\omega_{am}^T \phi_{am}\|^2 \\
& + \frac{2}{\gamma_3} \|\omega_{rm}^T \phi_{rm}\|^2 + 6s_m^2 + \frac{6}{\gamma_2} \|\omega_{cm}^T C_{am}\|^2 \cdot \|\varepsilon_{am} + d_m\|^2 \\
& + \frac{6}{\gamma_3} \|\omega_{rm}^T C_{rm}\|^2 \cdot \|r_m\|^2 \\
= & Z_m^2
\end{aligned} \tag{B.38}$$

where $\omega_{cm}, \omega_{am}, \omega_{rm}, \phi_{cm}, \phi_{am}, \phi_{rm}, C_{am}, C_{rm}, s_m$ and r_m are the upper bounds of $\omega_c, \omega_a, \omega_r, \phi_c, \phi_a, \phi_r, C_a, C_r, s$ and r , respectively.

Equation (B.38) further implies that $\Delta V(k) \leq 0$ if the following conditions hold

$$0 < K_{vmax} < \frac{\sqrt{3}}{3} \tag{B.39}$$

$$\frac{\sqrt{2}}{2} < \alpha < 1 \tag{B.40}$$

$$\eta_c \alpha^2 \|\phi_c(k)\|^2 < 1, \quad \eta_a \|\phi_a(k)\|^2 < 1, \quad \eta_r \|\phi_r(k)\|^2 < 1 \tag{B.41}$$

and

$$\|\bar{e}(k)\| > \sqrt{\frac{\gamma_1}{1 - 3K_{v\max}}} \|Z_m\| \quad (\text{B.42})$$

or

$$\|\zeta_c(k)\| > \sqrt{\frac{1}{(\alpha^2 - \frac{1}{2})}} \|Z_m\| \quad (\text{B.43})$$

According to a standard Lyapunov extension theorem [141, 142], this demonstrates that the auxiliary error and the error in the weights estimates are uniformly ultimately bounded (UUB). Also this further implies that the weights estimates are bounded correspondingly [75].

APPENDIX C

Boundedness Theoretical Results of Maze Navigation Example

Unlike the proofs of convergence for adaptive dynamic programming (ADP) in literature [143, 144, 145, 146], a new insight for the error bound between the estimated value function and the expected value function is provided. The critic network in GrADP approach is used to approximate the Q value function, and the action network is used to provide the control policy. The goal network is adopted to provide the internal reinforcement signal for the critic network over time. Finally, the estimated Q value function is illustrated to be close enough to the expected value function.

C.1 Problem Formulation

Define the instant reward between any two state x and x' as $s(x, x')$, and assume that there are N possible states in the maze. The probability between any two state x and x' can only take the value of 0 or 1 (i.e., this maze navigation problem is a deterministic and finite MDP). Thus, the Bellman's equation can be rewritten as

$$V(x) = s + \gamma \max V(x') \quad (\text{C.1})$$

and

$$s = f_g(x, u) \quad (\text{C.2})$$

where $V(x)$ is the value function at state x , and s is the internal reward function that is in a relationship with the system variables and control action. γ is a discounted factor with the range of $0 < \gamma < 1$. It is worth pointing out that such s function could represent the similar formulas (e.g., binary values, quadratic

forms, sum of weighted factors, etc) with those in [68, 147, 92, 148, 149], or some other proper formulas over time if necessary. In addition, the representation of the internal reward function could be automatically adjusted and updated during the learning process online, which has been claimed as the main contribution and foundation of the GrADP design in recent years. As this paper is dedicated for the boundedness results for the problem described in [97, 98], the following remarks and other mathematical derivations have been customized to exactly fit the approach adopted in the same papers.

Remarks:

1. The action space $\mathbb{U}$ is a finite set. Usually, there are four directions: left, right, forward and backward for the agent. Thus the action space is within a finite space set.
2. The state space $\mathbb{X}$ is a boundedness subset of R^n . Thus, there exists a constant $\Delta_{max} < \infty$, such that for all $x, x' \in \mathbb{X}$, the distance is defined as $d(x, x') \leq \Delta_{max}$. Generally, we can discretize the state space, and take the coordinates of the agent as the input for the controller. The state space can be arbitrarily large, but definitely it is within a boundedness set.

Given any control policy π , it is considered to be optimal if the generated value function can achieve the optimal value, such that $V^\pi(x) = V^*(x)$ holds for all $x \in \mathbb{X}$, where $V^*(x)$ is the optimal value function. For any $\epsilon > 0$, a policy π is considered as ϵ -optimal if the generated value function has been close enough towards the optimal value, such that $V^\pi(x) \geq V^*(x) - \epsilon$ holds valid for all $x \in \mathbb{X}$ [150, 151, 152].

Assumption 1: The instant reward signal has both upper and lower bounds s.t. $s \in [s_{min}, s_{max}]$, and then it results to

$$V_b = \frac{1}{1 - \gamma}(s_{max} - s_{min}). \quad (\text{C.3})$$

where V_b denotes the maximum difference between the returns of any two policies. γ is a discounted factor, which is smaller than 1.

Assumption 2: There exists certain constants $\alpha > 0$ and $\beta > 0$, s.t. $x, x' \in \mathbb{X}$ and $u \in \mathbb{A}$ that

$$|s(x, u) - s(x', u)| \leq \alpha \cdot d(x, x') \quad (\text{C.4})$$

and

$$d(f(x, u) - f(x', u)) \leq \beta \cdot d(x, x') \quad (\text{C.5})$$

where f refers to the system function and $x' = f(x, u)$ and $x'' = f(x', u)$. Note x' and x'' are denoted as the corresponding future state for the agent. d can be considered as the distance between the two states and can also be regarded as the upper difference value between two rewards.

Definition 1 [153]: The modulus of the continuity of the optimal value function V is defined as

$$\bar{\omega} = \sup_{x, x': d(x, x') \leq Z} |V(x) - V(x')| \quad (\text{C.6})$$

The modulus of continuity bound (MCB) is obtained as

1. $\omega(z) \leq \bar{\omega}(z), \forall z > 0;$
2. $\bar{\omega}(z) \longrightarrow 0$, as $z \longrightarrow 0$.

The MCB proof is similar with that in [153] and here we simply treat it as a definition for our analysis. To evaluate the performance of a learning algorithm, the definition of Policy-Mistake Count (PMC) is introduced. This counted number specifies the total steps that the algorithm spent on the non-optimal actions

during the learning process. That is to say, the algorithm can be able to learn the optimal/near optimal control policy after a finite number of steps and satisfies the PAC principle.

Definition 2 [153]: In the learning algorithm, define that $A_t = \pi_k|_{k=t}^\infty$ is the control policy that the algorithm implements at time t . This discounted return by executing A_t is denoted by $J^{A_t} = \sum_{k=t}^\infty \gamma^{(t-k)} s(k) \mid u_k = \pi_k(x_k)$. Therefore the PMC is defined as

$$PMC(\varepsilon) = \sum_{t=0}^\infty \Pi\left(J^{A_t}(x_t) < V(x_t) - \varepsilon\right) \quad (\text{C.7})$$

where Π is a signal function. It will output 1 if the event in the brace comes, otherwise output 0. From the definition, we can see that the algorithm can achieve the optimal/near optimal performance after the PMC steps. If the learning algorithm's PMC is finite and bounded, the algorithm is regarded as PAC algorithm [150, 152].

C.2 Analysis of GrADP based Learning Approach

The boundedness results for the GrADP based learning control approach is presented in this section. The difference between the value function learnt by the GrADP approach and the expected value function is under an arbitrary small bound. The algorithm will learn the near-optimal policy within a finite number of steps using PMC and PAC evaluation. The lemmas and the theorem are provided as below:

Lemma 1: For any given $x, x' \in \mathbb{X}$, it has

$$|V(x) - V(x')| \leq \bar{\omega}(d(x, x')). \quad (\text{C.8})$$

Proof: For any given $x, x' \in \mathbb{X}$, substituting the formula of V into equation (C.8), it has that

$$\begin{aligned}
|V(x) - V(x')| &= \\
|s(x, u) + \gamma \cdot \max_{u \in A} V(f(x, u)) - s(x', u) - \\
&\quad \gamma \cdot \max_{u \in A} V(f(x', u))| \\
&\leq |s(x, u) - s(x', u)| + \gamma \cdot \max_{u \in A} |V(f(x, u)) - V(f(x', u))|
\end{aligned} \tag{C.9}$$

The last item of the above inequality can be further rewritten as

$$\begin{aligned}
&|V(f(x, u)) - V(f(x', u))| \\
&= |s(f(x, u)) - s(f(x', u))| \\
&\quad + \gamma \cdot \max_{u \in A} |V(f(x', u')) - V(f(x'', u'))| \\
&\leq \alpha d(f(x, u), f(x', u)) + \gamma \cdot \max_{u \in A} |(V(f(x', u')) - V(f(x'', u')))| \\
&\leq \alpha \beta d(x, x') + \gamma \cdot \max_{u \in A} |(V(f(x', u')) - V(f(x'', u')))|.
\end{aligned} \tag{C.10}$$

Substituting the formula (C.10) into (C.9), it results to

$$\begin{aligned}
|V(x) - V(x')| &= \\
&\leq \gamma [\alpha \beta d(x, x') + \gamma \cdot \max_{u \in A} |V(f(x', u')) - V(f(x'', u'))|] \\
&\quad + \alpha d(x, x') \\
&\leq \alpha d(x, x') \cdot [1 + \alpha \beta + \dots] + \gamma^H \cdot V_b.
\end{aligned} \tag{C.11}$$

where V_b is the maximum difference of the value function between the infinite-time iterations. Let $H \rightarrow \infty$, it thus has

$$|V(x) - V(x')| \leq \alpha d(x, x') \cdot \sum_{k=0}^{H-1} (\gamma \beta)^k + \gamma^H V_b. \tag{C.12}$$

From equation (C.12), we can see that the analysis can be addressed in terms

of the range of $\gamma\beta$. (i) If $\gamma\beta < 1$, $H \rightarrow \infty$, then

$$|V(x) - V(x')| \leq \frac{\alpha}{1 - \gamma\beta} d(x, x'). \quad (\text{C.13})$$

Note γ is a discount factor in as presented in (C.8), and therefore $\gamma^H \cdot V_b$ is approaching zero when H goes to infinity. In this case, $\bar{\omega}(d) = \frac{\alpha}{1 - \gamma\beta} \cdot d$. The equation (C.13) is then in the same format as that in Lemma 1.

(ii) Consider $\gamma\beta > 1$, as $\gamma < 1$, assume there exists a minimal $H = H_0$ s. t.

$$\gamma^{H_0} V_b \leq \alpha d(x, x') \frac{(\gamma\beta)^{H_0}}{\gamma\beta - 1}. \quad (\text{C.14})$$

This indicates that

$$|V(x) - V(x')| \leq 2\alpha d(x, x') \frac{(\gamma\beta)^{H_0}}{\gamma\beta - 1}. \quad (\text{C.15})$$

Next, I am going to derive the upper value of the $(\gamma\beta)^{H_0}$, and thus find the upper value for the left side of inequity (C.15). Since H_0 is the minimal H satisfying the inequity above, it holds that

$$\alpha d(x, x') \frac{(\beta)^{H_0-1}}{\gamma\beta - 1} < V_b \leq \alpha d(x, x') \frac{\beta^{H_0}}{\gamma\beta - 1}. \quad (\text{C.16})$$

The left inequity is obtained from the following analysis: in (C.14), the right side value is almost equal or slight larger than the left side formula (as H_0 is the minimal value to satisfy inequity (C.14)). Then, V_b is expected to be larger than $\alpha d(x, x') \frac{(\beta)^{H_0-1}}{\gamma\beta - 1}$, as β is larger than 1 in this case.

It can further be derived the range for β_0^H that

$$\frac{V_b(\gamma\beta - 1)}{\alpha d(x, x')} \leq \beta^{H_0} < \frac{V_b\beta(\gamma\beta - 1)}{\alpha d(x, x')}. \quad (\text{C.17})$$

Then

$$H_0 \geq \log_\beta \left(\frac{V_b(\gamma\beta - 1)}{\alpha d(x, x')} \right). \quad (\text{C.18})$$

and

$$\begin{aligned}\gamma^{H_0} &\leq \gamma^{\log_\beta(\frac{V_b(\gamma\beta-1)}{\alpha d(x,x')})} \\ &= \left(\frac{V_b(\gamma\beta-1)}{\alpha d(x,x')}\right)^{\log_\beta \gamma}.\end{aligned}\tag{C.19}$$

and,

$$\begin{aligned}(\gamma\beta)^{H_0} &\leq \beta^{H_0} \left(\frac{V_b(\gamma\beta-1)}{\alpha d(x,x')}\right)^{\log_\beta \gamma} \\ &< \left(\frac{V_b\beta(\gamma\beta-1)}{\alpha d(x,x')}\right) \cdot \left(\frac{V_b(\gamma\beta-1)}{\alpha d(x,x')}\right)^{\log_\beta \gamma} \\ &= \beta \cdot \left(\frac{V_b(\gamma\beta-1)}{\alpha d(x,x')}\right)^{\log_\beta \gamma\beta}.\end{aligned}\tag{C.20}$$

Substituting the results from inequity (C.20) into inequity (C.15), we have

$$\begin{aligned}|V(x) - V(x')| &\leq 2\alpha d(x,x') \cdot \frac{(\gamma\beta)^{H_0}}{\gamma\beta-1} \\ &\leq 2\beta \cdot V_b^{\log_\beta \gamma\beta} \cdot \left(\frac{\alpha}{\gamma\beta-1}\right)^{\log_\beta \frac{1}{\gamma}} \\ &\quad \cdot d(x,x')^{(\log_\beta \frac{1}{\gamma})}.\end{aligned}\tag{C.21}$$

In this case, $\bar{\omega}(d) = P \cdot d^{(\log_\beta \frac{1}{\gamma})}$, where $P = 2\beta \cdot V_b^{\log_\beta \gamma\beta} \cdot \left(\frac{\alpha}{\gamma\beta-1}\right)^{\log_\beta \frac{1}{\gamma}}$. The inequality (C.21) is then in the same format as that in Lemma 1.

If $\alpha\beta = 1$, the above lemma will not satisfy MCB in this case according to the literature [154, 150]. This complete the proof of Lemma 1. The conclusion of this lemma will be called in the final proof of the Theorem. Interested readers may also refer to [154, 152, 155] for a more generalized theoretical proof. $\square$

The lemma provides the MCB property for the difference between two arbitrary given value functions (these two arbitrary states x and x' belong to $\mathbb{X}$). Next, I am going to derive the error bound for the difference between estimate Q value function and expected Q value function for any specific state. Therefore, we define that the Q value function is provided as

$$Q(x, u) = s(x, u) + \gamma \max_u Q(x', u')\tag{C.22}$$

In addition, an operator is defined as: given a function g s.t. $\mathbb{X} \times \mathbb{A} \longrightarrow \mathbb{R}$ and for arbitrary x , it satisfies $x \in \Omega$. The operator formula is presented as:

$$T_g(x) = s(x, u) + \gamma \max_{u \in A} g(f(x, u), u'). \quad (\text{C.23})$$

Lemma 2: The operator T is a contraction mapping in the l_∞ norm with contraction factor γ . Given any g_1 and g_2 , it has

$$\begin{aligned} |T_{g_1}(x) - T_{g_2}(x)| &= \gamma \max_{u \in A} \left(g_1(f(x, u), u') - g_2(f(x, u), u') \right) \\ &\leq \gamma \max_{u \in A} |g_1(f(x, u), u') - g_2(f(x, u), u')| \\ &= \gamma \cdot \|g_1 - g_2\|_\infty. \end{aligned} \quad (\text{C.24})$$

As T is a contraction, it will have a fixed solution. According to the definition in (C.22) and (C.23), Q value function is the fixed solution of T in $\mathbb{R}$. Also, from the formula in (C.24), one can see the operator T is a contraction with factor γ in an infinity form [155].

Theorem: For every time k , estimated value function $\hat{Q}$ has the following relationship w.r.t the optimal Q function:

$$\hat{Q}(x, u) \geq Q(x, u) - \varepsilon. \quad (\text{C.25})$$

where

$$\varepsilon = \frac{\alpha d + \gamma \bar{\omega}(d)}{1 - \gamma}. \quad (\text{C.26})$$

Proof: From Lemma 2, $\hat{Q}$ is the fixed solution of the operator T . The proof of this theorem can start with

$$\begin{aligned} &\hat{Q}(x, u) - Q(x, u) \\ &= \hat{s}(x, u) - s(x, u) + \gamma \cdot \max_{u \in A} (\hat{Q}(x, u) - Q(x, u)) \\ &\geq -\alpha d + \gamma \max_{u \in A} [\hat{Q}(x, u) - Q(x, u)]. \end{aligned} \quad (\text{C.27})$$

With the results in Lemma 1, we have

$$| \hat{Q}(x, u) - Q(x, u) | \leq \bar{\omega}(d) \quad (\text{C.28})$$

Substitute equation (C.28) into equation (C.27), it has

$$\begin{aligned} \hat{Q}(x, u) - Q(x, u) &\geq \sum_{k=0}^{\infty} \gamma^k (-\alpha d - \gamma \bar{\omega}(d)) \\ &= -\frac{\alpha d + \gamma \bar{\omega}(d)}{1 - \gamma}. \end{aligned} \quad (\text{C.29})$$

From (C.29), one can see that the difference between the estimated Q value function and the expected Q value function is within a certain bound after enough iterations. It demonstrates the PAC technique on the GrADP approach on approximating the value function. Also, the policy-mistake count will also be a finite number according to equation (C.7). The derived theorem has concluded the boundedness results of the paper. In the following section, we will provide the numerical results to verify the convergence of the GrADP approach through the maze navigation example. $\square$

APPENDIX D

Abbreviations

AD: Action Dependent

AI: Artificial Intelligence

ADP: Adaptive Dynamic Programming

CI: Computational Intelligence

DHP : Dual Heuristic Dynamic Programming

GDHP : Globalized Dual Heuristic Dynamic Programming

GrADP : Goal Representation Adaptive Dynamic Programming

GrDHP : Goal Representation Dual Heuristic Dynamic Programming

Gr-GDHP : Goal Representation Globalized Dual Heuristic Dynamic Programming

GrHDP : Goal Representation Heuristic Dynamic Programming

HDP : Heuristic Dynamic Programming

MDP: Markov Decision Processing

ML: Machine Learning

RL: Reinforcement Learning

TD: Temporal Difference

BIBLIOGRAPHY

- Al-Tamimi, A., Abu-Khalaf, M., and Lewis, F. L., “Adaptive critic designs for discrete-time zero-sum games with application to h-infinity control,” *IEEE Trans. on Syst. Man, Cybern., Part B*, vol. 37, no. 1, pp. 240–247, 2007.
- Al-Tamimi, A., Lewis, F. L., and Abu-Khalaf, M., “Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof,” *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 943–949, 2008.
- Balakrishnan, S. N. and Biega, V., “Adaptive Critic Based Neural Networks for Control,” in *Proc. Am. Contr. Conf.*, Seattle, WA, June, 1995, pp. 335–339.
- Balakrishnan, S. N., Ding, J., and Lewis, F. L., “Issues on stability of adp feedback controllers for dynamical systems,” *IEEE Trans. on Syst. Man, Cybern., Part B, special issue on ADP/RL, invited survey paper*, vol. 38, no. 4, pp. 913–917, 2008.
- Barto, A., Singh, S., and Chentanez, N., “Intrinsically motivated learning of hierarchical collections of skills,” in *Proc. Int. Conf. Developmental Learning (ICDL’04)*, 2004.
- Barto, A. G., Sutton, R. S., and Watkins, C. J. C. H., “Learning and sequential decision making,” in *Learning and computational neuroscience*. MIT Press, 1989, pp. 539–602.
- Bellman, R., *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
- Bernstein, A., “Adaptive state aggregation for reinforcement learning,” Master’s thesis, Israel Institute of Technology, 2007. [Online]. Available: <http://eprints.pascal-network.org/archive/00005853/>
- Bernstein, A. and Shimkin, N., “Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains,” *Machine learning*, vol. 81, no. 3, pp. 359–397, 2010.
- Bertsekas, D. P. and Tsitsiklis, J., *Neuro-Dynamic Programming*. Athena Scientific, 1996.
- Bertsekas, D. P., *Dynamic programming and optimal control*. MA: Athena Scientific Belmont, 1995.
- Bishop, C., *Pattern recognition and machine learning*. springer, 2006.

- Brooks, F. P., “What’s real about virtual reality?” *Computer Graphics and Applications*, vol. 19, no. 6, pp. 16–27, 1999.
- Bui, H., Kaul, B. C., Sarangapani, J., Drallmeier, J., and He, P., “Neuro emission controller for minimizing cyclic dispersion in spark ignition engines with egr levels,” 2004.
- Burdea, C. and Coiffe, P., *Virtual Reality Technology*. Piscataway, NJ: Wiley-IEEE Press, 2003.
- Carey, R., “The virtual reality modeling language explained,” *MultiMedia*, vol. 5, no. 3, pp. 84–93, 1998.
- Chien, T. L., Chen, C. C., Huang, Y. C., and Lin, W.-J., “Stability and almost disturbance decoupling analysis of nonlinear system subject to feedback linearization and feedforward neural network controller,” *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1220–1230, 2008.
- D. V. Prokhorov, and R. A. Santiago, and D. C. Wunsch II., “Adaptive critic designs: A case study for neurocontrol,” *Neural Networks*, vol. 8, no. 9, pp. 1367–1372, 1995.
- Dayan, P., “The Convergence of TD (λ) for General λ ,” *Machine Learning*, vol. 8, no. 34, pp. 341–362, 1992.
- Enns, R. and Si, J., “Helicopter trimming and tracking control using direct neural dynamic programming,” *IEEE Transactions on Neural Networks*, vol. 14, no. 4, pp. 929–939, 2003.
- Fairbank, M., Alonso, E., and Prokhorov, D. V., “Simple and fast calculation of the second-order gradients for globalized dual heuristic dynamic programming in neural networks.” *IEEE Trans. Neural Netw. Learning Syst.*, vol. 23, no. 10, pp. 1671–1676, 2012.
- Fang, X., He, H., Ni, Z., and Tang, Y., “Learning and control in virtual reality for machine intelligence,” in *Intelligent Control and Information Processing (ICICIP), 2012 Third International Conference on*, july 2012, pp. 63–67.
- Fang, X., Zheng, D., He, H., and Ni, Z., “Data-driven heuristic dynamic programming with virtual reality,” *Neurocomputing*, 2015, in press.
- Ferrari, S., Steck, J., and Chandramohan, R., “Adaptive feedback control by constrained approximate dynamic programming,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 982–987, 2008.

- Ferrari, S. and Stengel, R. F., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004, ch. Model-based Adaptive Critic Designs, pp. 64–94.
- Ferrari, S. and Stengel, R. F., *Handbook of learning and approximate dynamic programming*. Wiley-IEEE Press, 2004, ch. Model-Based Adaptive Critic Designs, pp. 65–95.
- Ferrari, S. and Stengel, R., “Online adaptive critic flight control,” *Journal of Guidance Control and Dynamics*, vol. 27, no. 5, pp. 777–786, 2004.
- Fu, J., He, H., Liu, Q., and Ni, Z., “An adaptive dynamic programming approach for closely-coupled mimo system control,” in *The 8th International Symposium on Neural Networks (ISNN)*, vol. 6677, 2011, pp. 1–10.
- Fu, J., He, H., and Ni, Z., “Adaptive dynamic programming with balanced weights seeking strategy,” in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, april 2011, pp. 210 –217.
- Fu, J., He, H., and Zhou, X., “Adaptive learning and control for mimo system based on adaptive dynamic programming,” *IEEE Trans. Neural Networks*, vol. 22, no. 7, pp. 1133–1148, 2011.
- Fu, X., Li, S., Fairbank, M., Wunsch, D. C., and Alonso, E., “Training recurrent neural networks with the levenberg–marquardt algorithm for optimal control of a grid-connected converter,” *Neural Networks and Learning Systems, IEEE Transactions on*, 2015, in press.
- Graham, J., Starzyk, J., Ni, Z., and He, H., “Advancing motivated learningn with goal creation,” in *Proc. IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI’14)*, Dec. 2014, Orlando, FL, pp. 1–8.
- Grossberg, S., *Neural Networks and Natural Intelligence*. Cambridge, MA: MIT Press, 1988.
- Harley, R. G. and Liang, J., “Computational intelligence in smart grids,” in *Computational Intelligence Applications In Smart Grid (CIASG), 2011 IEEE Symposium on*.
- Haykin, S., *Neural networks and learning machines*. Upper Saddle River: Prentice Hall, 2009, vol. 3rd Edition.
- He, H., Ni, Z., and Fu, J., “A three-network architecture for on-line learning and optimization based on adaptive dynamic programming,” *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012.
- He, H., *Self-Adaptive Systems for Machine Intelligence*. Wiley, 2011.

- He, H., Ni, Z., and Zhao, D., “Data-driven learning and control with multiple critic networks,” in *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, july 2012, pp. 523–527.
- He, H., Ni, Z., and Zhao, D., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Learning and Optimization in Hierarchical Adaptive Critic Design, pp. 78–95.
- He, P. and Jagannathan, S., “Reinforcement-based neuro-output feedback control of nonlinear discrete-time systems with input constraints,” *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 35, no. 1, pp. 150–154, 2005.
- He, P. and Jagannathan, S., “Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints,” *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 37, no. 2, pp. 425–436, 2007.
- Ilin, R., Kozma, R., and Werbos, P., “Cellular SRN trained by extended Kalman filter shows promise for ADP,” in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2006, pp. 506–510.
- Ilin, R., Kozma, R., and Werbos, P., “Efficient learning in cellular simultaneous recurrent neural networks-the case of maze navigation problem,” in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2007, pp. 324–329.
- Ilin, R., Kozma, R., and Werbos, P., “Beyond feedforward models trained by back-propagation: A practical training tool for a more efficient universal approximator,” *Neural Networks, IEEE Transactions on*, vol. 19, no. 6, pp. 929–937, 2008.
- Jagannathan, S. and He, P., “Neural network-based state feedback control of nonlinear discrete-time system in non-strict feedback form,” *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 2073–2087, 2008.
- Jiang, Y. and Jiang, Z.-P., “Robust adaptive dynamic programming and feedback stabilization of nonlinear systems,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 5, pp. 882–893, May 2014.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W., “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W., “Reinforcement learning: A survey,” *arXiv preprint cs/9605103*, 1996.
- Khalil, H., *Nonlinear Systems (3rd edition)*. Prentice Hall, New Jersey, 2001.

- Lee, J. and Lee, J., “Approximate dynamic programming strategies and their applicability for process control: A review and future directions,” *International Journal of Control Automation and Systems*, vol. 2, pp. 263–278, 2004.
- Lendaris, G. G., “Higher level application of adp: A next phase for the control field?” *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 901–912, 2008.
- Lendaris, G. G. and Neidhoefer, J. C., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004, ch. Guidance in the Use of Adaptive Critics for Control, pp. 97–118.
- Lewis, F. and Liu, D., Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013.
- Lewis, F. L., Jagannathan, S., and Yesildirek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Francis, London, UK, 1999, ch. Neural Network Robot Control, pp. 173–219.
- Lewis, F. L., Liu, D., and Lendaris, G. G., “Special issue on adaptive dynamic programming and reinforcement learning in feedback control,” *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 896–897, 2008.
- Lewis, F. L., Vrabie, D., and Syrmos, V. L., *Optimal control*. NJ, USA: Wiley, 2012.
- Liu, D., Javaherian, H., Kovalenko, O., and Huang, T., “Adaptive critic learning techniques for engine torque and air-fuel ratio control,” *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 988–993, 2008.
- Liu, D. and Wang, D., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Optimal Control of Unknown Nonlinear Discrete-Time Systems Using the Iterative Globalized Dual Heuristic Programming Algorithm, pp. 52–74.
- Liu, D., Wang, D., Zhao, D., Wei, Q., and Jin, N., “Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming,” *IEEE Trans. on Automation Science and Engineering*, vol. 9, no. 3, pp. 628–634, 2012.
- Liu, D. and Wei, Q., “Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems,” *IEEE Trans. on Cybernetics*, vol. 43, no. 2, pp. 779–789, 2013.

- Liu, D., Zhang, Y., and Zhang, H. G., “A self-learning call admission control scheme for cdma cellular networks,” *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1219–1228, 2005.
- Liu, F., Sun, J., Si, J., Guo, W., and Mei, S., “A boundedness result for the direct heuristic dynamic programming,” *Neural Networks*, vol. 32, pp. 229–235, 2012.
- Lu, C., Si, J., and Xie, X., “Direct heuristic dynamic programming for damping oscillations in a large power system,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 1008–1013, 2008.
- Mitchell, T. M., *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1997.
- Mohagheghi, S., Del Valle, Y., Venayagamoorthy, G. K., and Harley, R. G., “A proportional-integrator type adaptive critic design-based neurocontroller for a static compensator in a multimachine power system,” *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 1, pp. 86–96, 2007.
- Mohagheghi, S., Venayagamoorthy, G. K., and Harley, R. G., “Optimal wide area controller and state predictor for a power system,” *Power Systems, IEEE Transactions on*, vol. 22, no. 2, pp. 693–705, 2007.
- Molina, D., Venayagamoorthy, G. K., Liang, J., and Harley, R. G., “Intelligent local area signals based damping of power system oscillations using virtual generators and approximate dynamic programming,” *Smart Grid, IEEE Transactions on*, vol. 4, no. 1, pp. 498–508, 2013.
- Moriarty, D. E. and Miikulainen, R., “Efficient reinforcement learning through symbiotic evolution,” *Machine Learning*, vol. 22, pp. 11–32, 1996.
- Ni, Z., He, H., Zhao, D., and Prokhorov, D., “Reinforcement learning control based on multi-goal representation using hierarchical heuristic dynamic programming,” in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012.
- Ni, Z., Fang, X., He, H., Zhao, D., and Xu, X., “Real-time tracking control on adaptive critic design with uniformly ultimately bounded condition,” in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL’13), IEEE Symposium Series on Computational Intelligence (SSCI)*, Apr. 2013.
- Ni, Z. and He, H., “Heuristic dynamic programming with internal goal representation,” *Soft Computing*, vol. 17, pp. 2101–2108, 2013.
- Ni, Z., He, H., and Wen, J., “Adaptive learning in tracking control based on the dual critic network design,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 6, no. 24, pp. 913–928, 2013.

- Ni, Z., He, H., Wen, J., and Xu, X., “Goal representation heuristic dynamic programming on maze navigation,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, no. 12, pp. 2038–2050, 2013.
- Ni, Z., He, H., Zhao, D., Xu, X., and Prokhorov, D. V., “GrDHP: A General Utility Function Representation for Dual Heuristic Dynamic Programming,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 614–627, Mar. 2015.
- Ni, Z., He, H., and Zhong, X., *Frontiers of Intelligent Control and Information Processing*. World Scientific Publishing, 2014, ch. Experimental Studies on Data-Driven Heuristic Dynamic Programming for POMDP, pp. 88–105.
- Ni, Z., He, H., Zhong, X., and Prokhorov, D., “Model-Free Dual Heuristic Dynamic Programming,” *IEEE Trans. on Neural Networks and Learning Systems*, 2015, to be appear.
- Ni, Z., Tang, Y., He, H., and Wen, J., “Multi-machine power system control based on dual heuristic dynamic programming,” in *Proc. of 2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, Dec. 2014, Orlando, FL, pp. 1–7.
- Ni, Z., Tang, Y., Sui, X., He, H., and Wen, J., “An adaptive neuro-control approach for multi-machine power systems,” *International Journal of Electrical Power and Energy Systems*, 2015, under review.
- Nouri, A. and Littman, M. L., “Multi-resolution exploration in continuous spaces.” in *NIPS*, 2008, pp. 1209–1216.
- Novak-Marcincin, J., “Application of the virtual reality modeling language for design of automated workplaces,” *World Academy of Science, Engineering and Technology*, vol. 31, pp. 160–163, 2007.
- [Online]. <http://www.youtube.com/watch?v=OeZEDBz6ki0&feature=youtu.be>.
- P. H. Eaton, and D. V. Prokhorov, and D. C. Wunsch II., “Neurocontroller alternatives for fuzzy ball-and-beam systems with nonuniform nonlinear friction,” *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 423–435, 2000.
- Pang, X. and Werbos, P. J., “Neural network design for j function approximation in dynamic programming,” in *Math. Modelling and Scientific Computing*, [Online] Available: <http://arxiv.org/pdf/adap-org/9806001.pdf>, 1996.
- Powell, W. B., *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2007.
- Prokhorov, D. V., *Adaptive Critic Designs and Their Applications*, *PhD. Dissertation*. Texas Tech University, Lubbock, Tx, Oct, 1997.

- Prokhorov, D. V., “Adaptive critic designs and their applications, phd. dissertation, texas tech university, lubbock, tx,” Ph.D. dissertation, Oct, 1997.
- Prokhorov, D. V. and Wunsch, D. C., “Adaptive critic designs,” *IEEE Trans. on Neural Netw.*, vol. 8, no. 5, pp. 997–1007, 1997.
- Prokhorov, D. and Wunsch, D., “Adaptive critic designs,” *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 997–1007, 1997.
- Qiao, W., Harley, R. G., and Venayagamoorthy, G. K., “Coordinated reactive power control of a large wind farm and a statcom using heuristic dynamic programming,” *Energy Conversion, IEEE Transactions on*, vol. 24, no. 2, pp. 493–503, 2009.
- Ray, S., Venayagamoorthy, G. K., Chaudhuri, B., and Majumder, R., “Comparison of adaptive critics and classical approaches based wide area controllers for a power system,” *IEEE Trans. on Syst. Man, Cybern., Part B*, vol. 38, no. 4, pp. 1002–1007, 2008.
- Shih, P., Kaul, B. C., Jagannathan, S., Drallmeier, J., *et al.*, “Reinforcement-learning-based dual-control methodology for complex nonlinear discrete-time systems with application to spark engine egr operation,” *Neural Networks, IEEE Transactions on*, vol. 19, no. 8, pp. 1369–1388, 2008.
- Si, J., Barto, A. G., Powell, W. B., and Wunsch, D. C., Eds., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press and John Wiley & Sons, 2004.
- Si, J., Barto, A. G., Powell, W. B., and Wunsch, D. C., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.
- Si, J. and Wang, Y. T., “On-line learning control by association and reinforcement,” *IEEE Trans. on Neural Netw.*, vol. 12, no. 2, pp. 264–276, 2001.
- Si, J. and Wang, Y.-T., “Online learning control by association and reinforcement,” *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 264–276, 2001.
- Si, J., Yang, L., and Liu, D., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004, ch. Direct Neural Dynamic Programming, pp. 125–151.
- Singh, S., Barto, A., and Chentanez, N., “Intrinsically motivated reinforcement learning,” in *Prof. Annual Conf. Neural Information Processing Systems (NIPS’04)*, Vancouver, Canada, 2004.
- Singh, S., Lewis, R., and Barto, A., “Where do rewards come from?” in *Proc. Annual Conf. Cognitive Science Society*, 2009, pp. 2601–2606.

- Singh, S., Barto, A., Grupen, R., and Connolly, C., “Robust reinforcement learning in motion planning,” *Advances in neural information processing systems*, pp. 655–655, 1994.
- Slotine, J. and Li, W., *Applied Nonlinear Control*. Prentice Hall, New Jersey, 1991.
- Starzyk, J. A., *Frontiers in Robotics, Automation and Control*. InTech Education and Publishing, 2008, ch. Motivation in Embodied Intelligence, pp. 83–110.
- Starzyk, J. A., Liu, Y., and He, H., “Challenges of embodied intelligence,” in *Proc. Int. Conf. Signals and Electronic Syst.*, 2006, pp. 534–541.
- Strehl, A. L., Li, L., and Littman, M. L., “Reinforcement learning in finite MDPs: PAC analysis,” *The Journal of Machine Learning Research*, vol. 10, pp. 2413–2444, 2009.
- Strehl, A. L. and Littman, M. L., “A theoretical analysis of model-based interval estimation,” in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 856–863.
- Sui, X., Tang, Y., He, H., and Wen, J., “Energy-Storage-Based Low-Frequency Oscillation Damping Control Using Particle Swarm Optimization and Heuristic Dynamic Programming,” *IEEE Transactions on Power Systems*, 2015, in press.
- Sutton, R. S., “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” in *In Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann, 1990, pp. 216–224.
- Sutton, R., “Learning to predict by the methods of temporal differences,” *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.
- Sutton, R., “Temporal credit assignment in reinforcement learning,” Ph.D. dissertation, University of Massachusetts Amherst, 1984.
- Sutton, R. and Barto, A., *Reinforcement learning: An introduction*. Cambridge Univ. Press, 1998.
- Tang, Y., He, H., Ni, Z., and Wen, J., “Optimal operation for energy storage with wind power generation using adaptive dynamic programming,” in *Proc. IEEE Power and Energy Society General Meeting (IEEE PES General Meeting’15)*, Jul., Denver, CO, pp. 1–6.
- Tang, Y., He, H., Ni, Z., Wen, J., and Sui, X., “Reactive power control of grid-connected wind farm based on adaptive dynamic programming,” *Neurocomputing*, vol. 125, pp. 125–133, Feb. 2014.

- Tang, Y., He, H., and Wen, J., “Comparative study between hdp and pss on dfig damping control,” in *Computational Intelligence Applications In Smart Grid (CIASG), 2013 IEEE Symposium on*. IEEE, 2013, pp. 59–65.
- Tang, Y., Ju, P., He, H., Qin, C., and Wu, F., “Optimized control of dfig-based wind generation using sensitivity analysis and particle swarm optimization,” *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 509–520, 2013.
- Thrun, S., *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Boston, MA, Kluwer, 1996.
- Vamvoudakis, K. and Lewis, F. L., “Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem,” in *Proc. IEEE Int. Conf. Neural Netw.*, 2009.
- Vance, J. B., Kaul, B. C., Jagannathan, S., Drallmeier, J., *et al.*, “Output feedback controller for operation of spark ignition engines at lean conditions using neural networks,” *Control Systems Technology, IEEE Transactions on*, vol. 16, no. 2, pp. 214–228, 2008.
- Venayagamoorthy, G. K., Harley, R. G., and Wunsch, D. C., “Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator,” *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 764–773, 2002.
- Venayagamoorthy, G. K., Harley, R. G., and Wunsch, D. C., “Dual heuristic programming excitation neurocontrol for generators in a multimachine power system,” *IEEE Trans. on Industry Applications*, vol. 39, no. 2, pp. 382–394, 2003.
- Venayagamoorthy, G. K., Rohrig, K., and Erlich, I., “One step ahead: short-term wind power forecasting and intelligent predictive control based on data analytics,” *Power and Energy Magazine, IEEE*, vol. 10, no. 5, pp. 70–78, 2012.
- Wang, D., Liu, D., Wei, Q., Zhao, D., and Jin, N., “Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming,” *Automatica*, vol. 48, pp. 1825–1832, 2012.
- Wang, F. Y., Zhang, H., and Liu, D., “Adaptive dynamic programming: An introduction,” *IEEE Comput. Intel. Mag.*, vol. 4, no. 2, pp. 39–47, 2009.
- Wang, F. Y., Zhang, H., and Liu, D., “Adaptive Dynamic Programming: An Introduction,” *IEEE Comput. Intel. Mag.*, vol. 4, no. 2, pp. 39–47, 2009.
- Wang, F.-Y., Zhang, H., and Liu, D., “Adaptive dynamic programming: An introduction,” *Computational Intelligence Magazine, IEEE*, vol. 4, no. 2, pp. 39–47, 2009.

- Wei, C., Zhang, Z., Qiao, W., and Qu, L., “Reinforcement learning-based intelligent maximum power point tracking control for wind energy conversion systems,” *Industrial Electronics, IEEE Transactions on*, 2015, in press.
- Werbos, P., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Reinforcement Learning and Approximate Dynamic Programming (RLADP)–Foundations, Common Misconceptions and Challenges Ahead, pp. 3–30.
- Werbos, P. J., “Building and understanding adaptive systems: a statistical/numerical approach to factory automation and brain research,” *IEEE Trans. Syst. Man Cybern.*, vol. 17, no. 1, pp. 7–20, Jan. 1987.
- Werbos, P. J., “Consistency of HDP applied to a simple reinforcement learning problem,” *Neural Networks*, vol. 3, no. 2, pp. 179 – 189, 1990.
- Werbos, P. J., “An overview of neural networks for control,” *IEEE Control Syst. Mag.*, vol. 11, no. 1, pp. 40–41, 1991.
- Werbos, P. J., *Handbook of Intelligent Control*. New York: Van Nostrand, 1992, ch. Approximate Dynamic Programming for Real Time Control and Neural Modeling, pp. 493–525.
- Werbos, P. J., *Handbook of intelligent control*. New York: Van Nostrand Reinhold, 1992, ch. Approximate dynamic programming for real-time control and neural modeling.
- Werbos, P. J., *The Handbook of Brain Theory and Neural Netw.* Cambridge MA: MIT Press, 1995, ch. Backpropagation: Basics and New Developments, pp. 134–139.
- Werbos, P. J., *Automatic Differentiation: Applications, Theory and Implementations, Lecture Notes in Computational Science and Engineering*. Springer, 2005, vol. 50, ch. Backwards Differentiation in AD and Neural Nets: Past Links and New Opportunities, pp. 15–34.
- Werbos, P. J., “Using ADP to understand and replicate brain intelligence: the next level design,” in *IEEE Int. Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 209–216.
- Werbos, P. J., “ADP: The key direction for future research in intelligent control and understanding brain intelligence,” *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 898–900, 2008.
- Werbos, P. J., “Intelligence in the brain: A theory of how it works and how to build it,” *Neural Networks*, pp. 200–212, 2009.

- Werbos, P. J., “Intelligence in the brain: A theory of how it works and how to build it,” *Neural Networks*, vol. 22, no. 3, pp. 200–212, 2009.
- Werbos, P., “Computational intelligence for the smart grid-history, challenges, and opportunities,” *Computational Intelligence Magazine, IEEE*, vol. 6, no. 3, pp. 14–21, 2011.
- Werbos, P. and Pang, X., “Generalized maze navigation: SRN critics solve what feedforward or Hebbian nets cannot,” in *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, vol. 3, oct 1996, pp. 1764–1769.
- White, H. and Gallant, A., “On learning the derivatives of an unknown mapping with multilayer feedforward networks,” *Neural Networks*, vol. 5, pp. 129–138, 1992.
- Wiering, M. and Van Hasselt, H., “Two novel on-policy reinforcement learning algorithms based on td (λ)-methods,” in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2007, pp. 280–287.
- Wunsch, D., “The cellular simultaneous recurrent network adaptive critic design for the generalized maze problem has a simple closed-form solution,” in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, vol. 3. IEEE, 2000, pp. 79–82.
- Xu, X., Hou, Z., Lian, C., and He, H., “Online learning control using adaptive critic designs with sparse kernel machines,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, pp. 762–775, 2013.
- Yang, L., Si, J., Tsakalis, K. S., and Rodriguez, A. A., “Direct heuristic dynamic programming for nonlinear tracking control with filtered tracking error,” *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 39, no. 6, pp. 1617–1622, 2009.
- Yao, W., Wen, J., Chen, S., and Jiang, L., “Development of matlab/simulink based power system transient stability analysis toolbox,” *Power System Technology (Chinese)*, vol. 36, no. 6, pp. 95–101, 2012.
- Yuan, F., Feldkamp, L., Puskorius, G., and Davis, L., “A Simple Solution to the Bioreactor Benchmark Problem by Application of Q-learning,” in *Proc. World Congress on Neural Networks*, Washington, DC, July, 1995, pp. II–326–331.
- Zhang, H. G., Luo, Y. H., and Liu, D., “Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints,” *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1490–1503, 2009.

- Zhang, H. G., Wei, Q. L., and Liu, D., “An iterative approximate dynamic programming method to solve for a class of nonlinear zero-sum differential games,” *Automatica*, vol. 47, no. 1, pp. 207–214, 2011.
- Zhang, H. G., Wei, Q. L., and Luo, Y. H., “A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy hdp iteration algorithm,” *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 937–942, 2008.
- Zhang, H., Cui, L., Zhang, X., and Luo, Y., “Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method,” *IEEE Trans. Neural Networks*, vol. 22, no. 12, pp. 2226–2236, Dec. 2011. [Online]. Available: <http://dx.doi.org/10.1109/TNN.2011.2172628>
- Zhang, H., Song, R., Wei, Q., and Zhang, T., “Optimal tracking control for a class of nonlinear discrete-time systems with time delays based on heuristic dynamic programming,” *IEEE Trans. Neural Networks*, vol. 22, no. 12, pp. 1851–1862, Dec. 2011. [Online]. Available: <http://dx.doi.org/10.1109/TNN.2011.2172628>
- Zhang, H., Wei, Q., and Luo, Y., “A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy hdp iteration algorithm,” *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 937–942, 2008.
- Zhao, D. and Zhu, Y., “MEC—A Near-Optimal Online Reinforcement Learning Algorithm for Continuous Deterministic Systems,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 2, pp. 346–356, Feb. 2015.
- Zhong, X., He, H., Zhang, H., and Wang, Z., “Optimal control for unknown discrete-time nonlinear markov jump systems using adaptive dynamic programming,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2141–2155, 2014.
- Zhong, X., He, H., Zhang, H., and Wang, Z., “A neural network based online learning and control approach for markov jump systems,” *Neurocomputing*, vol. 149, pp. 116–123, 2015.
- Zhong, X., Ni, Z., He, H., Xu, X., and Zhao, D., “Event-triggered reinforcement learning approach for unknown nonlinear continuous-time system,” in *Proc. Int. Joint Conf. Neural Networks (IJCNN’14)*, Jul. 2014, Beijing, China, pp. 1–8.