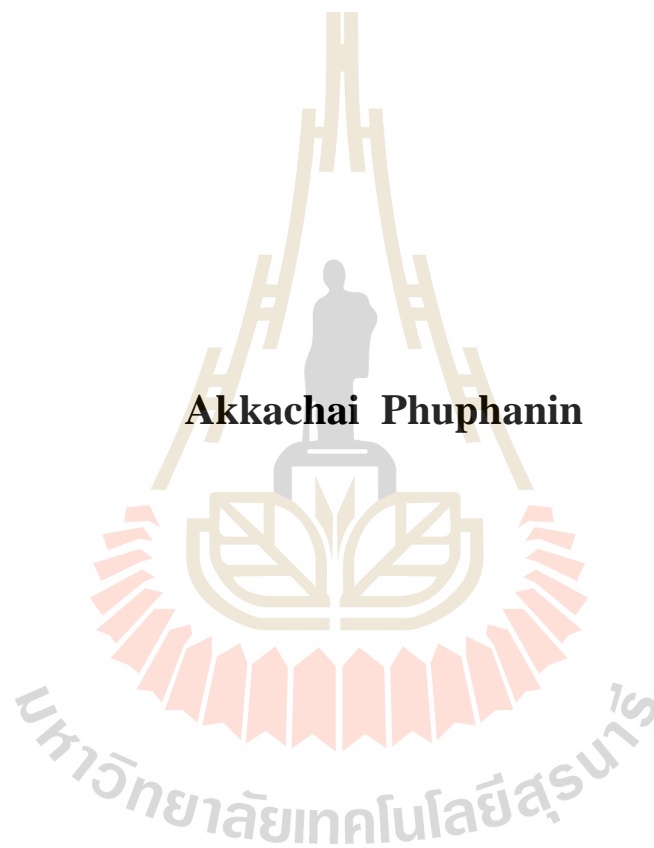


**COVERAGE CONTROL IN WIRELESS SENSOR
NETWORKS USING MULTI-AGENT SYSTEMS**



**A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy in Telecommunication Engineering**

Suranaree University of Technology

Academic Year 2017

การควบคุมพื้นที่ครอบคลุมในเครือข่ายเซ็นเซอร์ไร้สาย โดยใช้
ระบบมัลติเอเจนต์



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2560

COVERAGE CONTROL IN WIRELESS SENSOR NETWORKS USING MULTI-AGENT SYSTEMS

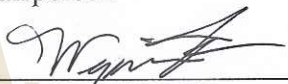
Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

Thesis Examining Committee



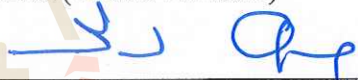
(Assoc. Prof. Dr. Monthippa Uthansakul)

Chairperson




(Asst. Prof. Dr. Wipawee Hattagam)

Member (Thesis Advisor)



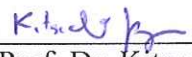
(Assoc. Prof. Dr. Peerapong Uthansakul)

Member



(Assoc. Prof. Dr. Chaodit Aswakul)

Member



(Asst. Prof. Dr. Kitsuchart Pasupa)

Member



(Assoc. Prof. Flt. Lt. Dr. Kontorn Chamniprasart)



(Prof. Dr. Santi Maensiri)

Vice Rector for Academic Affairs
and Internationalization

Dean of Institute of Engineering

อรรถชัย ภูพานิล : การควบคุมพื้นที่ครอบคลุมในเครือข่ายเซ็นเซอร์ไร้สาย โดยใช้ระบบมัลติเอเจนต์ (COVERAGE CONTROL IN WIRELESS SENSOR NETWORKS USING MULTI-AGENT SYSTEMS) อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.วิภาวี หัตถกรรม, 152 หน้า

จุดประสงค์หลักของวิทยานิพนธ์นี้คือพัฒนารูปแบบมัลติ-เอเจนต์ควบคุมพื้นที่ครอบคลุมแบบกระจายตัวสำหรับความร่วมมือเครือข่ายเซ็นเซอร์ไร้สาย โดยอาศัยการหาค่าที่เหมาะสมที่สุดแบบหลายวัตถุประสงค์ โดยเฉพาะอย่างยิ่งแต่ละเซ็นเซอร์โหนดรับบทบาทเป็นเอเจนต์ซึ่งทำการตัดสินใจเปิดใช้งานหรือปิดการใช้งานเพื่อให้บรรลุวัตถุประสงค์ในการรักษาพื้นที่ครอบคลุมและประสิทธิภาพการใช้พลังงาน โดยนำเสนอระบบมัลติ-เอเจนต์อัลกอริทึมที่มีการกระจายตัว, ต้องการข้อมูลเฉพาะแห่งและแสดงให้เห็นถึงความยืดหยุ่น นอกจากนี้วิธีการระบบมัลติ-เอเจนต์มีความสามารถในการหาค่าที่เหมาะสมที่สุดของฟังก์ชันวัตถุประสงค์ที่ขัดแย้งกันเพื่อกำหนดนโยบายที่เหมาะสมที่สุดในการควบคุมพื้นที่ครอบคลุมในเครือข่ายเซ็นเซอร์ไร้สาย โครงร่างประยุกต์ใช้กับการควบคุมพื้นที่ครอบคลุมแสงภายในบ้านหรืออาคารอัจฉริยะ

องค์ความรู้หลักในงานวิจัยนี้มีสี่ประการ องค์ความรู้ประการแรกคือนำเสนอรูปแบบมัลติเอเจนต์ควบคุมพื้นที่ครอบคลุมโดยอาศัยการปรับปรุงฟังก์ชันต้นทุนในวิธีการฟังก์ชันค่าการกระจายตัว (DVF) เพื่อเป็นการปรับตัวแบบกระจายตัวและมีความยืดหยุ่นในการควบคุมพื้นที่ครอบคลุมซึ่งจะต้องรักษาพื้นที่ครอบคลุมและลดพื้นที่ครอบคลุมที่ซ้ำซ้อนเพื่อลดการใช้พลังงาน องค์ความรู้ประการที่สองคือรูปแบบการหาค่าที่เหมาะสมที่สุดแบบหลายวัตถุประสงค์ (MOO) ที่เรียกว่า วิธีการสเกลาร์ไลซ์คิวมัลติออฟเจกทีฟรีอินฟอร์สเมนต์เลิร์นนิ่ง (SQMORL) สำหรับควบคุมพื้นที่ครอบคลุมและการใช้พลังงานอย่างมีประสิทธิภาพในการควบคุมแสงแบบอัตโนมัติ องค์ความรู้ประการที่สามคือการติดตั้งอุปกรณ์ต้นแบบเพื่อประเมินวิธีการ SQMORL ที่นำเสนอและวิธีการ DVF องค์ความรู้ประการสุดท้ายคือ นำเสนอการปรับปรุงวิธีการสเกลาร์ไลซ์คิวมัลติออฟเจกทีฟรีอินฟอร์สเมนต์เลิร์นนิ่งสำหรับควบคุมแสงสว่างแบบอัตโนมัติโดยอาศัยการประมาณค่าคุณลักษณะสำหรับการควบคุมแสงโดยอาศัยปริภูมิสถานะแบบต่อเนื่อง

ผลการทดลองชี้ให้เห็นว่า รูปแบบ เอ็มเอเอส (MAS) ควบคุมพื้นที่ครอบคลุมโดยอาศัยวิธีการหลายวัตถุประสงค์สามารถได้รับพื้นที่ครอบคลุมอย่างมีประสิทธิภาพและเป็นการปรับตัวด้วยตัวเอง เพราะฉะนั้นจึงเหมาะสำหรับการใช้งานควบคุมพื้นที่ครอบคลุมในเครือข่ายเซ็นเซอร์ไร้สาย นอกจากนี้ดีวีเอฟที่มีการปรับปรุงและสเกลาร์ไลซ์คิวมัลติออฟเจกทีฟรีอินฟอร์สเมนต์เลิร์นนิ่งสามารถเพิ่มพื้นที่ครอบคลุมมากที่สุดและลดพื้นที่ครอบคลุมที่ซ้อนทับและการใช้พลังงานสำหรับ

การประยุกต์ใช้งานเช่น การควบคุมแสงในสำนักงานอัจฉริยะ นอกจากนี้โดยอาศัยการประมาณค่าฟังก์ชันของปริภูมิสถานะแบบต่อเนื่อง สามารถค้นหา นโยบายที่เหมาะสมที่สุดและตัดสินใจปรับเปลี่ยนการควบคุมแสงได้อย่างรวดเร็วเมื่ออยู่ภายใต้แสงภายนอก, การมีโหนดผิดพลาด



สาขาวิชาวิศวกรรมโทรคมนาคม

ปีการศึกษา 2560

ลายมือชื่อนักศึกษา

Quirk.

ลายมือชื่ออาจารย์ที่ปรึกษา

Wepi...

AKKACHAI PHUPHANIN : COVERAGE CONTROL IN WIRELESS
SENSOR NETWORKS USING MULTI-AGENT SYSTEMS. THESIS
ADVISOR : ASST. PROF. WIPAWEE HATTAGAM, Ph.D., 152 PP.

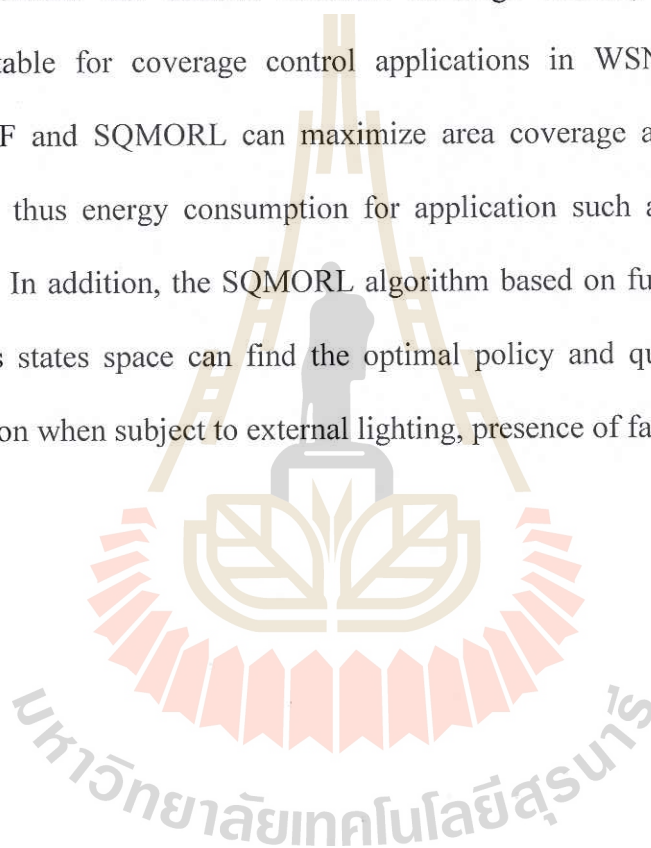
WIRELESS SENSOR NETWORKS/MULTI-AGENT/MULTI OBJECTIVE/
CONTINUOUS STATE Q-LEARNING/COVERAGE CONTROL

The underlying aim of this thesis is to develop a distributed multi-agent coverage control scheme for cooperative wireless sensor networks (WSNs) based on multiple objectives optimization. In a particular, each sensor node takes role as an agent which makes a decision to activate or remain deactivated in order to achieve the objective of maintaining coverage and energy efficiency. The proposed multi-agent system (MAS) algorithms are distributed, requires localized information and exhibits scalability. In addition, the MAS algorithms have the ability to optimize conflicting objective functions to find the optimal policy in coverage control in WSNs. The framework is applied to lighting coverage control in smart homes or buildings.

The main contribution of this research proposal is four-fold. The first contribution is the MAS coverage control scheme based on the modified cost function in a distributed value function (DVF) algorithm for a distribution adaptive and scalable area coverage control algorithm which maintains the required coverage control and reduces redundant coverage area to reduce energy consumption. The second contribution is the multiple objective optimization (MOO) framework called, Scalarized Q Multi-Objective Reinforcement Learning (SQMORL) algorithm, to address coverage control and energy efficient automatic lighting control. The third

contribution is the hardware implementation and test bed evaluation the proposed SQMORL and DVF algorithms. The final contribution is an adaptive SQMORL algorithm for automatic lighting control using feature approximation for lighting coverage control based on continuous state space.

Results suggest that the MAS coverage control scheme based on multi objective algorithm can achieve efficient coverage control, is self-adaptive and therefore suitable for coverage control applications in WSNs. Furthermore, the modified DVF and SQMORL can maximize area coverage and reduce redundant coverage and thus energy consumption for application such as lighting control in smart offices. In addition, the SQMORL algorithm based on function approximation of continuous states space can find the optimal policy and quickly adjust its light control decision when subject to external lighting, presence of faulty nodes.



School of Telecommunication Engineering

Academic Year 2017

Student's Signature

Advisor's Signature

ACKNOWLEDGEMENT

I am grateful to all those, who by their direct or indirect involvement have helped in the completion of this thesis.

First and foremost, I would like to express my sincere thanks to my thesis advisors, Asst. Prof. Dr. Wipawee Hattagam for her invaluable help and constant encouragement throughout the course of this research. I am most grateful for her teaching and advice, not only the research methodologies but also many other methodologies in life. I would not have achieved this far and this thesis would not have been completed without all the support that I have always received from her.

I would also like to thank Assoc. Prof. Dr. Monthipa Uthansakul, Assoc. Prof. Dr. Peerapong Uthansakul, Assoc. Prof. Dr. Chaodit Aswakul and Asst. Prof. Dr. Kitsuchart Pasupa for accepting to serve in my committee.

My sincere appreciation goes to Ms. Pranitta Arthan for their valuable administrative support during the course of my dissertation.

I also grateful to the funding One Research One Graduate (OROG) support from Suranaree University of Technology (SUT).

Finally I am most grateful to my parents and my friends both in both masters and doctoral degree courses for all their support throughout the period of this research

Akkachai Phuphanin

TABLE OF CONTENTS

	Page
ABSTRACT (THAI).....	I
ABSTRACT (ENGLISH).....	III
ACKNOWLEDGMENTS.....	V
TABLE OF CONTENTS.....	VI
LIST OF TABLE.....	XI
LIST OF FIGURES.....	XII
SYMBOLS AND ABBREVIATIONS.....	XVII
CHAPTER	
I INTRODUCTION.....	1
1.1 Significance of problem.....	3
1.2 Automatic lighting control protocol in our focus.....	4
1.3 Algorithm design requirement.....	6
1.4 Multi-Agent Systems (MAS) and Multi-Objective Reinforcement Learning (MORL).....	7
1.5 Research objectives.....	9
1.6 Research hypothesis.....	9
1.7 Basic agreements.....	10
1.8 Scope of limitation.....	10
1.9 Research procedures.....	11
1.10 Expected benefit.....	13

TABLE OF CONTENTS (Continued)

	Page
II BACKGROUND THEORY	
2.1 Characteristic of WSNs.....	15
2.2 Challenging issues in WSNs.....	16
2.3 Multi-agent systems in WSNs.....	19
2.4 Single-agent and multi-agent systems.....	21
2.4.1 Single-agent systems.....	21
2.4.2 Multi-agent systems.....	22
2.5 Markov decision process theory.....	23
2.5.1 Markov decision process.....	24
2.6 Reinforcement learning.....	27
2.7 Multiple agent Q-learning algorithm.....	30
2.8 Distributed reinforcement learning.....	30
2.8.1 Distributed value functions.....	32
2.8.2 Distributed value function (DVF) algorithm.....	32
2.9 Summary.....	33
III A MULTI-AGENT SCHEME FOR ENERGY-EFFICIENT COVERAGE CONTROL IN WIRELESS SENSOR NETWORKS	
3.1 Introduction.....	34
3.2 Multi-agent coverage control.....	36

TABLE OF CONTENTS (Continued)

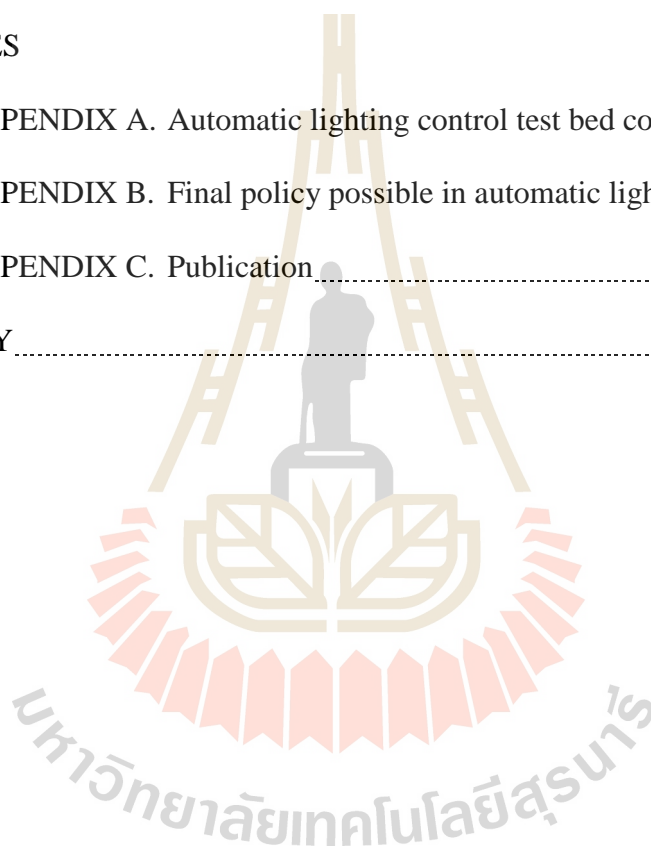
	Page
3.2.1 Distributed Value Function Scheme.....	36
3.2.2 Modified DVF Framework for Coverage Control.....	37
3.3 Performance Evaluation.....	38
3.4 Summary.....	43
IV SCALALIZED Q MULTI OBJECTIVE REINFORCEMENT	
LEARNING FOR AREA COVERAGE CONTROL AND IN	
LIGHT CONTROL DEMONSTRATION	
4.1 Introduction.....	44
4.2 Multi objective reinforcement learning for coverage control.....	46
4.2.1 Multi objective optimization.....	49
4.2.2 Scalarized Q multi objective reinforcement learning for area coverage control.....	49
4.3 SQMORL for area coverage control performance evaluation: simulation part.....	53
4.3.1 Simulation result: random position.....	54
4.3.2 Simulation result: grid position.....	57
4.3.3 Discussion on simulation results.....	60
4.4 SQMORL for automatic light control test bed.....	61
4.4.1 SQMORL automatic lighting control result.....	62
4.4.2 Discussion on test bed results.....	71
4.5 Summary.....	71

TABLE OF CONTENTS (Continued)

	Page
V CONTINUOUS STATE MULTI-AGENT REINFORCEMENT LEARNING FOR LIGHT CONTROL	
5.1 Introduction	72
5.2 Continuous state reinforcement learning for automatic lighting Control	74
5.3 Result of simulation	78
5.4 Light control test bed	94
5.5 Summary	101
VI CONCLUSIONS AND FUTURE WORK	
6.1 Conclusions	105
6.1.1 Chapter 3	105
6.1.2 Chapter 4	106
6.1.3 Chapter 5	109
6.2 Future work	112
6.2.1 Extend SQMORL algorithm to densely deployed network	112
6.2.2 Extension SQMORL algorithm to weight learning approach	113
6.2.3 Extension SQMORL algorithm to continuous action space	113

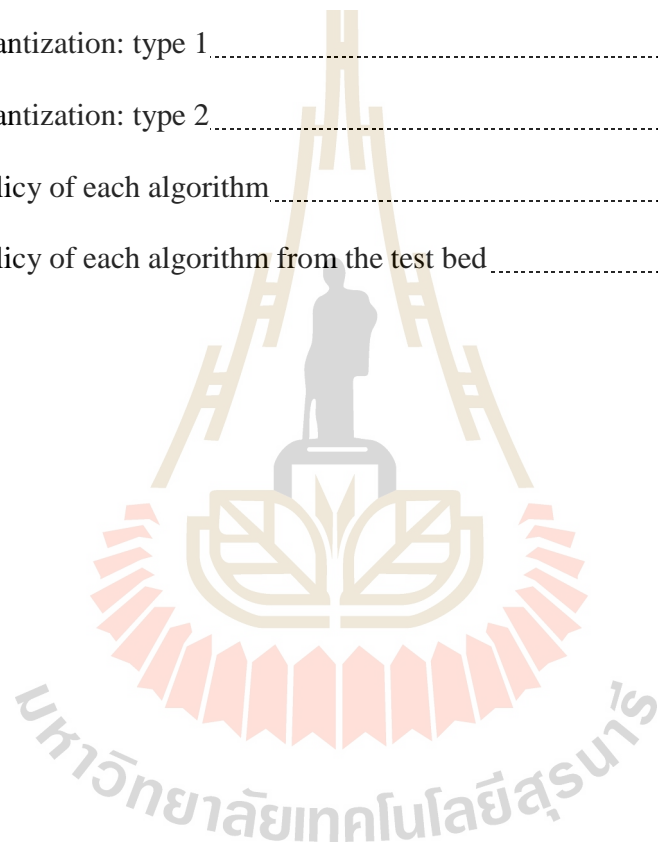
TABLE OF CONTENTS (Continued)

	Page
6.2.4 Extend MOO framework to other systems in smart home.....	113
REFERENCES.....	114
APPENDICES	
APPENDIX A. Automatic lighting control test bed code.....	124
APPENDIX B. Final policy possible in automatic lighting control.....	141
APPENDIX C. Publication.....	144
BIOGRAPHY.....	152



LIST OF TABLES

Table	Page
5.1 State quantization: type 1.....	84
5.2 State quantization: type 2.....	84
5.3 Final policy of each algorithm.....	86
5.4 Final policy of each algorithm from the test bed.....	98



LIST OF FIGURES

Figure	Page
2.1 Structure of Sensor node.....	17
2.2 A general-agent framework. The agent models itself, the environment, and their interaction. If other agents exist, they are considered part of the environment.....	22
2.4 A multi-agent scenario. Each agent models each other's goals, actions, and domain knowledge. Direct interaction (communication) are indicated by the arrows between the agent.....	23
2.4 A MDP model.....	25
2.5 Diagram of agent-environment interaction in reinforcement learning.....	28
2.6 Distributed RL diagram representing logical node in the distributed RL formulation. Each node senses its own state of the environment, takes its own action, and receives its own reward signal.....	31
3.1 Average number of working nodes against the number of deployed nodes.....	40
3.2 Percentage of coverage against the number of deployed nodes.....	41
3.3 Average coverage area per working node against number of deployed Nodes.....	41
3.4 Coverage lifetime for DVF.....	42
3.5 Coverage lifetime for OGDC.....	42

LIST OF FIGURES (Continued)

Figure	Page
3.6 Coverage lifetime for PEAS8.....	42
4.1 Pseudo code of continuous state reinforcement learning for lighting control.....	53
4.2 Number of active nodes against number of nodes placed in the network.....	56
4.3 Percentage area coverage against number of nodes placed in the network.....	56
4.4 Ratio between the coverage area and working sensor nodes.....	57
4.5 Number of active nodes against number of nodes placed in the network.....	59
4.6 Percentage area coverage against number of nodes placed in the network.....	59
4.7 The coverage area and active sensor nodes ratio.....	60
4.8 Process of SQMORL automatic lighting control.....	63
4.9 Automatic lighting control prototype.....	64
4.10 Distribution of the final policy obtained by each algorithm.....	67
4.11 Light intensity of each sensor nodes of DVF algorithm.....	67
4.12 Average reward of all 5 sensor nodes from the DVF algorithm.....	68
4.13 Light intensity of each sensor node from the SQMORL algorithm.....	68
4.14 Average reward of 1 st objective of 5 sensor nodes of the SQMORL algorithm.....	69
4.15 Average reward of 2 st objective of 5 sensor nodes of the SQMORL algorithm.....	69

LIST OF FIGURES (Continued)

Figure	Page
5.1 Pseudo code of continuous state reinforcement learning for lighting control.....	77
5.2 The location of the five node sensors.....	78
5.3 Effect of discount factor against average reward.....	79
5.4 Effect of discount factor on policy accuracy.....	80
5.5 Effect of learning rate on average reward.....	81
5.6 Effect of learning rate on policy accuracy.....	83
5.7 Average reward of each sensor node at each time step.....	88
5.8 Action selection of each sensor of Discrete state RL algorithm.....	89
5.9 Average reward of each sensor node of Discrete state RL algorithm.....	90
5.10 Action selection of each sensor of Threshold algorithm.....	91
5.11 Average reward of each sensor node of Threshold algorithm.....	93
5.12 Action selection of each sensor of Continuous state RL algorithm.....	93
5.13 Average reward of each sensor node of Continuous state RL algorithm.....	93
5.14 Diagram of Continuous state RL for automatic lighting control testbed.....	96
5.15 Automatic lighting control test bed.....	97
5.16 Average reward of all 5 sensor nodes from the Discrete state RL algorithm.....	100
5.17 Average reward of all 5 sensor nodes from the Continuous state RL algorithm.....	101

SYMBOLS AND ABBREVIATIONS

WSNs	Wireless sensor networks
PEAS	Probing Environment and Adaptive Sleeping
OGDC	Optimal Geographical Density Control
LACS	Lighting Automatic Control System
WSN-ILS	Wireless Sensor Network-driven Intelligent Lighting System
MAS	Multi-agent system
MORL	Multi-Objective Reinforcement Learning
RL	Reinforcement learning
MARL	Multi-agent reinforcement learning
TMP	Topology maintenance protocols
FMQ	Frequency maximum Q-learning
DVF	Distributed value function
MDP	Markov decision process
MOO	Multi Objective Optimization
PS	Pareto-optimal Set
PF	Pareto-optimal Front
SQMORL	Scalalized Q Multi Objective Reinforcement Learning
IoT	Internet of Thing
AAEC	Activity-Appliance-Energy Consumption
TD	Temporal Difference

SYMBOLS AND ABBREVIATIONS (Continued)

t	Time step index
s_t	State of the process at time t
S	State space
s	Current state
s'	Next state
A	Action space
a	Action
$E[\cdot]$	Expectation operator
γ	Discount factor
$R(s, a, s')$	Expected reward given any current state s and an action a with any next state s'
r	Reward
P	State transition probability matrix
f	Policy
f^*	Optimal policy
$P[A]$	Distribution over the action space
$Q_t^f(s, a)$	The action-value function of a given policy f associated to each state-action pair (s, a)
R_t	Expected discounted return of the agent

SYMBOLS AND ABBREVIATIONS (Continued)

$E^f [\cdot]$	Expectation operator under the policy f
$V^f (s)$	Value function of a state (s) under policy f
$V^*(s)$	Value function of a state (s) under optimal policy f^*
Γ	Learning rate
$Q^*(s, a)$	The action-value function of a given optimal policy f^* associated to each state-action pair (s, a)
i	Agent index
X	the base search space
q	Constraint index
x	a solution of the feasible region
$\Omega \subset R^n$	The decision space
R^m	Objective space
$x^* \in \Omega$	Pareto-optimal
MQ^f	The vectored state – action value function
$w(s, a)$	The value representing the continuous state
D	The distance between the light bulb to the sensor node
S	The angle between the normal lines of sensor nodes with distance
F	The luminous flux of 1030 lumen
n	Number of sensor node are neighbors

CHAPTER I

INTRODUCTION

1.1 Significance of the problem

A wireless sensor networks (WSNs) is a wireless network consisting of spatially distributed autonomous sensory devices that can communicate with each other to perform sensing and data processing cooperatively (Stankovic, 2006). The overall objective of a WSNs is to provide a low-cost solution to gather physical data from the environment, such as humidity, sound, pressure, noise, light, vibration or temperature, at different locations, observation and transmit it to a base station. The most common energy storage device used in a sensor node is a battery which is suitable for a micro-sensor with very low power consumption (Yick et al., 2008). Therefore, WSNs promises unlimited potential for numerous application areas including environmental (Chitnis et al., 2009), medical, military, transportation, entertainment, crisis management, homeland defense, and smart space (Han et al., 2007), (Yu et al., 2007), (Li et al., 2006).

Recently, the production of low-cost, low-power, multifunction, and tiny sensors (Akyildiz et al., 2002), allow numerous of sensors to be deployed to monitor an area. The large number of sensors prolongs the network lifetime and increases the quality of the monitored area. As one main function of a WSN is gathering physical data, coverage control is of significant importance. To accomplish this function, it must schedule, and organize the network in such a way that it can effectively observe

the environment of interest, and then collect desired information. This problem is the main focus in this thesis.

Various types of coverage problems and coverage control protocols has been presented in the literature. In coverage control problems, the objective of coverage control problem is to cover some areas such as a forest. In some other cases, its objective is to cover a set of subjects such as collection of paintings in a museum. Other research works investigate coverage problem where sensor nodes have adjustable sensing range. In most of the work, sensors could not change their sensing radii, i.e., they have fixed sensing ranges. To give the sensors flexible choices to accomplish their optimization job, sensors are assumed to adjust their sensing ranges in some of the recent works (Vu et al., 2009), (Chen et al., 2009).

There are some research works focusing on the k -coverage problem. Due to the uncertainty of sensors, 1-coverage may not be enough for applications that require highly accurate and reliable data transmission. For such applications, each point in the area may need to be covered by at least k sensors at the same time where $k > 1$ and k is a user-specific parameter (Wang et al., 2003).

Some research works place emphasis on maximization of coverage areas. Ye et al. (2002), (2003) present the design of PEAS, a simple protocol that can build a long-lived sensor network and maintain robust operations using large quantities of economical, short-lived sensor nodes. PEAS extends system functioning time by keeping only a necessary set of sensors working and putting the rest into sleep mode. The ones in sleep mode wake up now and then, probing the local environment and replacing failed ones. The sleeping periods are self-adjusted dynamically, so as to keep the sensors wakeup rate roughly constant, thus adapting to high node densities.

On the other hand, the optimal geographical density control algorithm (OGDC) is a recently developed coverage control scheme for wireless sensor networks (Shang et al., 2005). The algorithm captures several of the aforementioned new aspects introduced by wireless sensor networks. The optimal geographical density control algorithm (OGDC) is a power-aware algorithm, whose goal is to maintain complete sensing coverage and connectivity for as long as possible.

From an operation point of view, OGDC and PEAS algorithms control the area coverage in wireless sensor network in a distributed manner. Each sensor makes its own decision based on its own objective (i.e. coverage area). Each sensor can be considered as an agent with a single objective of maintaining its own coverage area. However, apart from the coverage area, in most deployment situations, energy efficiency is also a critical factor. This is because maintaining coverage with a minimal number of active nodes can reduce energy consumption, prolong network lifetime as well as minimize costs. Both PEAS and OGDC, sensor nodes do not consider energy efficiency in making their coverage decision. Furthermore, both OGDC and PEAS are non-learning algorithms. This is because in deciding every action of the sensor, the decision of the sensor node is only based on the information sent from the neighboring sensor nodes. There will be no adjustment to improve the choice of action for the subsequent decisions. Therefore, sensor nodes may not attain the global optimal of the system, particularly upon changing environment conditions.

This thesis therefore is focused on developing a learning scheme for distributed coverage control in WSNs based on multiple objectives i.e. coverage and energy efficiency. The framework can be applied to coverage control problems in smart home or buildings as described next.

1.2 Automatic lighting control protocol in our focus

Smart home requires use of technology to control home appliances which is based on three key technologies: *electronic and embedded systems*, *automatic control systems* and *information and communications*. Smart homes aim to facilitate the residents, energy management system, and automatic security system inside and around the house. A widespread application is the automatic lighting control which has a purpose to adjust the light bulbs to suit every situation for energy saving and user convenience. The key part of the system is the control algorithm embedded in the sensor node. There are many researches on this subject.

We can classify the automatic lighting control algorithms according to their ability to recognize environmental changes and quickly adapt to different situations. Ref. (Mohamaddoust et al., 2011) designed a lighting automatic control system (LACS) with the use of finite state machine method. LACS divided the local control unit into zones, which consist of a light sensor and a sensor that detects the user's location for system information. The system is capable of calculating and communicating to enable illumination that best suits the needs and activities of the user by taking account effect of external brightness as well. However, LACS is based on a single agent with a single objective framework. In particular, the system requires the intensity of light produced by the effect of the neighboring light bulbs. Therefore, if the bulb location is changed or the surrounding is changed, it is necessary to measure the effect of the light intensity every time before use. In other lighting control works, agents can make their own decisions based on the information they need from their neighbors. Kumar et al. (2010) has implemented a wireless sensor network system which the sensor node can measure the light intensity and transmit

the data to the master node. Then, the master node makes a decision, based on the light intensity measured by the sensor node and compares it to a reference value, to turn the light bulb on or off, or select the appropriate action to save energy. This system is shown to reduce energy consumption due to external daylight. However, in this research the only purpose is to save energy but not optimizing light intensity. Huynh et al. (2011) propose an algorithm that manages light intensity in a smart lighting system using the principle of PID closed-loop control. Meng-Shiuan et al. (2008) presents an intelligent lighting control system for indoor use based on usage patterns in each user activity. The systems divides illumination into two types, i.e. *whole lighting device* and *local lighting device*, which can adjust light intensity more suitably for the user. Meng-Shiuan et al. (2008) also demonstrates the installation of equipment and the actual use within the building. Okada et al. (2015) proposes wireless sensor network-driven intelligent lighting system (WSN-ILS) for dealing with illumination for each user. WSN-ILS system manages the light from each sensor node and provides the most efficient power and environment for the user. In research works (Huynh et al., 2011), (Meng-Shiuan et al., 2008), (Okada et al., 2015), information from many agents is required to decide the appropriate lighting control to achieve the optimal light intensity, energy saving, and adjust to user behavior. Singhiv et al. (2005) proposes an intelligent method to control the energy saving light bulb, based on the optimization problem to find a tradeoff between the lux intensity and minimal energy consumption. Their work demonstrates the use of the wireless sensor network to study the optimal between user, brightness, and power consumption.

In most aforementioned researches related to automatic lighting control, most systems are non-learning and use the threshold value of light intensity, or use other

principles to determine the optimal decision based on a particular tradeoff to make decisions to control the light bulbs depending on the environment condition. In addition, most lighting control schemes are centralized, as it is necessary to send data to sensor nodes which act as a base stations or personal computer to determine the optimal choice of action. Thus, such a centralized scheme is prone to communication overhead, decision-making delay.

1.3 Algorithm design requirement

In this section, we list out desired features of coverage control algorithm in WSNs which include:

- **Energy-efficiency:** A sensor is a battery-driven device and in most cases, the battery is irreplaceable. However, every operation of a sensor consumes a certain amount of energy. Thus, to extend network lifetime, a sensor network algorithm must be energy-efficient. The best way to maximize network lifetime is to balance energy consumption among all the sensors in the network. Load balancing should be in the sense that sensors with more energy should have more chance to be active, and the more exhausted ones should have more chance to go to remain inactive.
- **Distributed algorithms:** Sensors have limited computational ability and small memory size. Therefore, they are not able to perform complex operations. The task of running any algorithm should be shared among sensors or alternate nodes work in the network. To cater network scalability and possible frequent topology changes, the convergence

time of any algorithm designed for deployment in WSN needs to be short enough to keep up with the changes in the networks. For these reasons, in most cases, only localized, distributed algorithms are suitable for WSNs.

- **Area coverage:** Coverage is usually referred as how well a sensor network can monitor a field of interest. Coverage can be a measure of quality-of-service, of which can be measured in different ways depending on the application. In this proposal, the problem under consideration is how to maintain required coverage for area monitoring with low energy consumption.
- **Multi-objective optimization:** In wireless sensor networks, there may be several objectives which need to be optimized simultaneously. If we were to address only single objective optimization problem separately, one objective may contradict with another objective. One direct approach to address this issue is to combine such objective functions into one single objection function. The optimization problem may have a feasible solution in less complicated problems or with small number of sensor nodes. However, with conflicting objective or with a larger number of sensor nodes. The incorporation of the problem into a single objective function may not be sufficient. A multi objective optimization framework is more suitable to find the right policy.

1.4 Multi-Agent Systems (MAS) and Multi-Objective Reinforcement Learning (MORL)

From the previous section, a localized, distributed adaptive coverage control method which is not computationally or resource demanding is therefore needed. Reinforcement learning (RL) (Sutton et al., 1998), (Kaelbling et al., 1996) has been a common approach to coordinately and cooperatively improve the performance in WSNs. RL is usually defined as the problem faced by a learner of how to take actions, or make optimal decisions, through trial and error interactions with a dynamic environment. A common RL method called Q-learning is an algorithm which directly approximates the optimal action-value function (a function that describes how good an action was, given that the agent is at a particular state). Each learning agent takes an action, receives a reward, updates local information with an input from the environment, and repeats the process by learning its own optimal strategy. Guestrin et al. (2002) proposed the Frequency Maximum Q-learning (FMQ) to encourage cooperative coverage control in WSNs. FMQ is based on Q-learning, which enables autonomous self learning/adaptive applications with inherent support for efficient resource/task management. In (Tham et al., 2005), the authors considered a multi-agent system controlling coverage in a lighting control experiment. Each agent controls a light bulb illuminating a room represented by a 10x10 grid. Therefore, the coverage of each agent was considered in terms of the number of grids illuminated. Although MAS has conceptually been shown to perform well under this setting, factors such as actual overlapping sensing coverage area has not been considered. Phuphanin et al. (2016) use multi-agent system combined with reinforcement learning for area coverage control in wireless sensor networks. Results show that the use of RL in conjunction with the MAS can work well compared to non-learning algorithms. The advantages of RL do not only include online learning, but also a high degree of

scalability, low algorithm complexity and fast convergence. It is suitable for use in systems that are distributed and limited in computational resources, such as wireless sensor networks. Although (Phuphanin et al., 2016) perform online learning coverage control, it is a single objective optimal algorithm.

Since energy efficiency is of significant importance, the algorithms should also have the ability to solve complex and conflicting objective functions to find the optimal policy for dealing with area coverage control in wireless sensor network. Therefore, the contribution of this research proposal is four-fold:

- 1) A distribution adaptive and scalable area coverage control algorithm which maintains the required coverage control with low energy consumption to extend network lifetime,
- 2) Algorithm for energy efficient automatic lighting control,
- 3) Hardware implementation and evaluation the proposed algorithm,
- 4) An adaptive algorithm for automatic lighting control based on feature approximation of continuous state space.

1.5 Research objective

- 1) To study cooperative coverage control schemes in wireless sensor networks using multi-agent reinforcement learning.
- 2) To apply MAS to a sensing coverage control problem in a WSN which considers coverage area and overlapping area.
- 3) To develop a distributed adaptive and scalable coverage control algorithm which maintains the required coverage control with reduced overlapping area.

- 4) To develop and evaluate hardware implementation of the proposed approach as prototype for smart home illumination control.
- 5) To develop algorithm a self-adaptive for automatic lighting control scheme based on continuous state space.

1.6 Research hypothesis

- 1) Cooperative coverage control is beneficial when the network is sparse or when the environment is hostile.
- 2) Determining the overlapping area coverage together the MAS can provide energy efficiency and extend network lifetime.
- 3) A multi objective optimization method integrated with the reinforcement learning algorithm can deal with conflicting objectives.
- 4) The multi objective optimization reinforcement leaning can be applied to automatic lighting control in smart homes.
- 5) The continuous state space using feature approximation can reduce training time and quickly to responded to the changing environment.

1.7 Basic agreements

- 1) Visual C++ and Matlab is used to simulation the cooperative coverage control in wireless sensor networks.

1.8 Scope and limitation

- 1) The multi-agent system includes sensor nodes randomly placed in a 35x35 m² region. Each sensor node has to make coverage decisions based on the status of neighboring sensor node.
- 2) Three algorithms are compared, namely, the Optimal Geographical Density Control or OGDC (Shang et al., 2005), A Robust Energy Conserving Protocol for Long-lived Sensor Network or PEAS (Ye et al. 2003) and the proposed MAS algorithm to maximize coverage and minimize energy consumption which takes into account the overlapping coverage area of the sensor node.

1.9 Research procedures

1) Progressions

1.1 Review of literatures and related theories.

1.2 Study the methodologies which motivate cooperation in coverage wireless sensor networks and their effects.

1.3 Design and test the proposed algorithm by simulation using Visual C++ and Matlab.

1.4 Analyze and conclude results.

1.5 Prepare publication.

1.6 Write thesis.

2) Research methodology

Objective 1: To study cooperative coverage control schemes in wireless sensor networks using multi-agent reinforcement learning.

1. Review literatures and related works in cooperative coverage control and topology control in WSNs.

Objective 2: To apply MAS to a sensing coverage control problem in a WSN which considers coverage area and overlapping area.

1. Review literatures and related works in sensing coverage control by using MAS approach.
2. Evaluate performance of sensing coverage control scheme.

Objective 3: To develop a distributed adaptive and scalable coverage control algorithm which maintains the required coverage control with reduced overlapping area.

1. Review literatures and related works in distributed coverage control algorithm.
2. Develop an algorithm can maintains the required coverage control with low overlapping area.
3. Evaluate performance of distribution coverage control algorithm.

Objective 4: To develop and evaluate hardware implementation of the proposed approach as prototype for smart home illumination control.

1. Review literatures and related works in illumination coverage control application.
2. Develop and evaluate hardware implementation of the proposed approach.
3. Evaluate performance of hardware implementation.

Objective 5: To develop algorithm a self-adaptive for automatic lighting control scheme based on continuous state space.

1. Review literatures and RL related works in continuous state space.
2. Develop and evaluate proposed approach algorithm.
3. Evaluate performance of algorithm.

3) Research location

Wireless Communication Research and Laboratory, F4

111 University Avenue, Muang District, Nakhon Ratchasima 30000,

Thailand

4) Research equipments

4.1 Personal Computer

4.2 Visual C++ software

4.3 Matlab software.

5) Data collection

5.1 Information collected by reviewing literatures and related works.

5.2 Data collected from Visual C++ simulations.

5.3 Data collected from Matlab simulations.

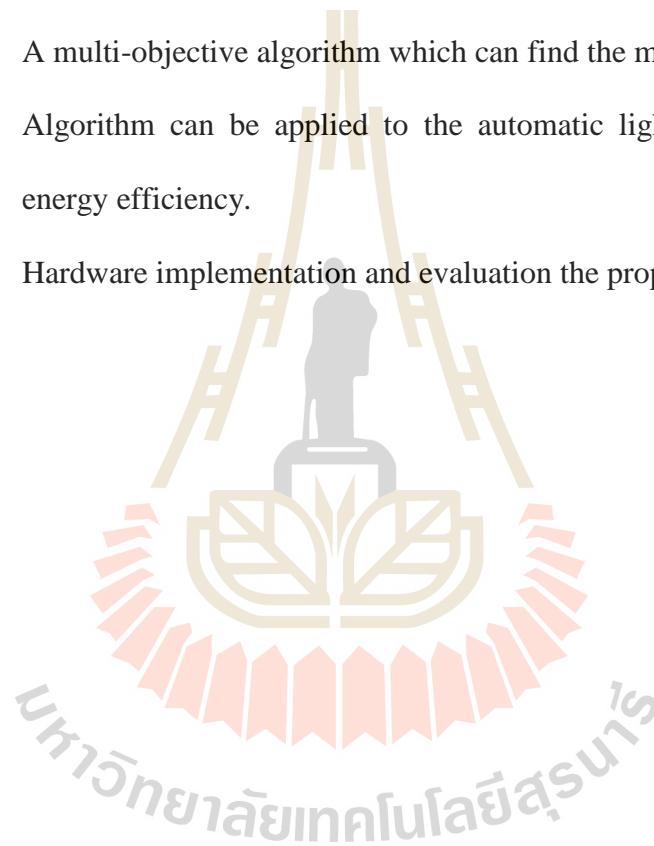
6) Data analysis

Information collected from the node cooperation experiment will be analyzed, compared and concluded in terms of graphs and tables.

1.10 Expected benefit

The expected benefit of this research proposal is five-fold:

- 1) A distributed adaptive and scalable area coverage control algorithm which maintains the required coverage control with low overlapping area which is adaptive to perturbation such as external lighting or node faults.
- 2) A modified multi-agent coverage control scheme based on a redundancy coverage area cost function.
- 3) A multi-objective algorithm which can find the multiple or conflict.
- 4) Algorithm can be applied to the automatic lighting control for and energy efficiency.
- 5) Hardware implementation and evaluation the proposal algorithm.



CHAPTER II

BACKGROUND THEORY

2.1 Characteristics of WSNs

Many protocols have been applied to wired and traditional wireless networks (such as wireless LAN). However, those protocols cannot be directly employed to sensor networks. This is mainly because sensor networks possess some special characteristics and constraints that distinguish it from the other types of networks. Those constraints of sensor network may include:

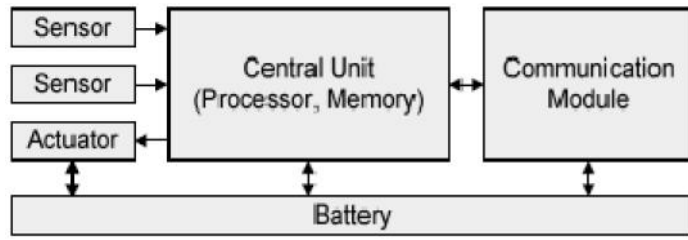
- Limited support for networking. Sensor nodes are likely to communicate with very low quality, high latency, limited bandwidth, and high failure-rate links. A sensor's transmission range is short and greatly affected by energy. In a WSN, the communication mainly depends on broadcasting. Moreover, the network is peer-to-peer, with a mesh topology and dynamic, possibly mobile, and unreliable connectivity.
- Energy constraint. In most cases, the sensor's battery of a sensor cannot be reused. Therefore, energy conservation is always the most critical requirement on designing a sensor network protocol. A sensor consumes significantly more energy on communication than on computation, thus the designed protocols for WSN usually try to attain efficient computation to compensate for communication cost.

- Dynamic topology. The topology of a WSN may change very frequently due to the movement of sensors, the sensors' temporary or permanent failure and the battery depletion of sensors.
- Scalability and heterogeneity. A WSN may consist of a large number of sensors with different sensing units, communicational ability, computational power, and memory size. Many sensors may be deployed in hostile environments, thus it can be difficult to maintain and manage the network.
- Failure of sensor node. A sensor node may fail to operate due to numerous reasons such as depletion of energy and environmental interference making it vulnerable to the environment, i.e. node can be physically damaged easily. Sensing data is prone to error under the environment effects such as noise and obstacles.

2.2 Challenging issues in WSNs

Before a WSN can be deployed, many issues need to be resolved. In this section, some issues ones that draw the much attention from researchers are presented.

- Hardware for WSNs: A mote (sensor node) consists of processor, memory, battery, analog-to-digital (A/D) converter for connecting a sensor to a radio transmitter for forming an ad hoc network. A mote and sensor together form a sensor node. The structure of the sensor node is as shown in Figure 2.1 there can be different sensors for different purposes mounted on a mote. A sensor node forms a basic unit of the sensor network (Vieira et al., 2003).



enable long operating lifetime by facilitating low duty cycle operation, local signal processing. The research issues that can be considered are different strategies to improve signal reception, design of low power, less cost sensors and processing units. Various schemes to conserve node power consumption and node optimization and simple modulation schemes may also be considered for sensor nodes.

- **Algorithm efficiency:** An important resource of a sensor node is energy because every operation requires energy. Though most sensors are battery-driven, battery is not always replaceable. Thus, energy-efficiency has been a critical aspect for any protocol designed for a WSN. Another limitation is the limited memory size, communication and computation capability. Thus, algorithms designed for WSNs need to be simple but robust and fault-tolerant. That is also the reason why decentralized algorithm is always preferable in WSNs. Other requirements for a “good” protocol are simplicity, energy-efficiency, localized, distributed, scalability and flexibility, robustness, fault-tolerance, and low communication overhead.
- **Topology control:** For a prone-to-failure network such as WSN, the sensors may fail at any time or any place. In addition, the topology may change due to the status selection of the sensors in the network. It follows that the topology of a WSN may be highly dynamic and unpredictable.

- **Routing:** After sensors collect information, streams of data need to be sent to the base station. The challenge is thus of how to efficiently, reliably, and securely route the data through the network.

2.3 Multi-agent systems in WSNs

Due to scarce battery supply, maintaining and maximizing coverage control has become a challenging issue in WSNs. Distributed self-adaptive coverage control schemes are of particular interest as WSNs are typically deployed in dynamically changing environments which may be difficult to access and manually configure. Such autonomous coverage control can be achieved by multi-agent systems (MAS). The implementation of MAS in a WSN requires sensor-actuator nodes with processing capability which enable these nodes to perform tasks in a coordinated manner to achieve some desired system-wide objective.

Multi-agent systems (MAS) differ from single-agent systems in that there are many different agents that learn a task. Furthermore, all of the agents' actions affect the state of the environment. Thus, the optimal policy not only relies on only one agent, but on all agents. There are works which directly applied a commonly used RL method called Q-learning to multi-agent systems whereby each agent disregards other agents in the system and takes action to maximize its own benefit. By neglecting the presence of other agents, suboptimal decisions are likely to be achieved. Therefore, an individual agent should consider the effect of joint actions from other agents as well to achieve better decisions in MAS.

To promote cooperation between sensor nodes, multi-agent systems (MAS) have been applied in WSNs. MAS has the potential to tackle the resource constraints

inherent in these networks by efficiently coordinating the activities among the nodes. MAS is made up of a number of agents, each with its own set of states and actions. Each agent must coordinate with one another in order to maximize the overall benefit for all agents. The work in (Seah et al., 2007) examined how coordination between the wireless sensor nodes could lead to maximization of coverage of the sensing field as well as minimization of the total energy consumption, thereby increasing the lifetime of network. They tested three algorithms, i.e. the Fully Distributed Q-learning, the Distributed Value Function (DVF) and the Coordinated algorithm (COORD). The work in (Tham et al., 2005) presented an algorithm for multi-agent reinforcement learning called coordinated reinforcement learning. In this algorithm, agents coordinate both their action selection and their parameter update. Within the limits of their parametric representation, the agent determines a joint action without explicitly considering every possible action in their exponentially large joint action space. In (Tham et al., 2005), the authors implemented a multi-agent system on a wireless sensor network comprising sensor-actuator nodes with processing capability. Their approach enabled these nodes to perform tasks in a coordinate manner to achieve maximum coverage. In (Tham et al., 2005), authors considered the implementation of several algorithms including the Ind learners algorithm, the DVF algorithm and the OptDRL algorithm. The optimal algorithm for coverage control in multi-agent system was found to be the DVF algorithm, in terms of trade-off between achieved area coverage and energy consumption. The DVF algorithm extends a commonly used RL method called, Q-learning, to encourage cooperative behavior between agents in multi-agent systems to achieve maximum coverage area in the network. In this thesis, this algorithm was used as a benchmark for coverage control comparison in WSN.

This chapter presents the fundamental theory of reinforcement learning which is the basis of the contribution of this thesis is. The next section explains the concept of single-agent and multi-agent RL. The next section provides a theoretical background on Markov decision process (MDP). A description of reinforcement learning is given in section 2.4. Section 2.5 presents the multi-agent Q-learning. Section 2.6 presents the distributed reinforcement learning and a summary is presented in the final section.

2.4 Single-agent and multi-agent systems

2.4.1 Single-agent systems

Before studying and categorizing MAS, we first consider the most obvious centralized, single-agent systems. Centralized systems have a single agent which makes all the decisions. A single-agent system may have multiple entities, several actuators, or several physically separated components. However, if each entity sends its perceptions to and receives its actions from a single central process, then there is only a single agent in the central process. The central agent models all of the entities as a single “self”.

In general, the agent in a single-agent system models itself, the environment, and the interactions between the agent and environment. The agent is an independent entity with its own goal, action, and domain knowledge. In a single-agent system, other agents are not recognized by the agent. Thus, even if there are other agents in the system, they are not modeled as having a goal. That is, they are just considered part of the environment. The point being emphasized is that although

agents are also a part of the environment, they are explicitly modeled as having their own goals, actions, and domain knowledge can be shown in Figure 2.3.

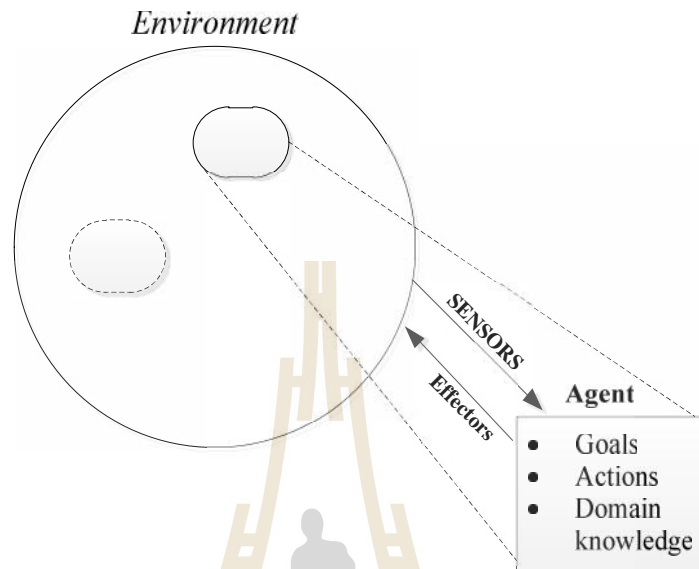


Figure 2.2 A general-agent framework. The agent models itself, the environment, and their interaction. If other agents exist, they are considered part of the environment.

2.4.2 Multi-agent systems

Multi-agent systems differ from single-agent systems in that several agents co-exist, each with their own goals and actions. From an individual agent's point of view, multi-agent systems differ from single-agent systems in that the environment's dynamics can be affected by other agents. Thus, all multi-agent systems can be viewed as having dynamic environments. Figure 2.4 depicts a multi-agent system where each agent is both part of the environment and modeled as a separate entity. There may be any number of agents, with different degrees of heterogeneity and with or without the ability to communicate directly.

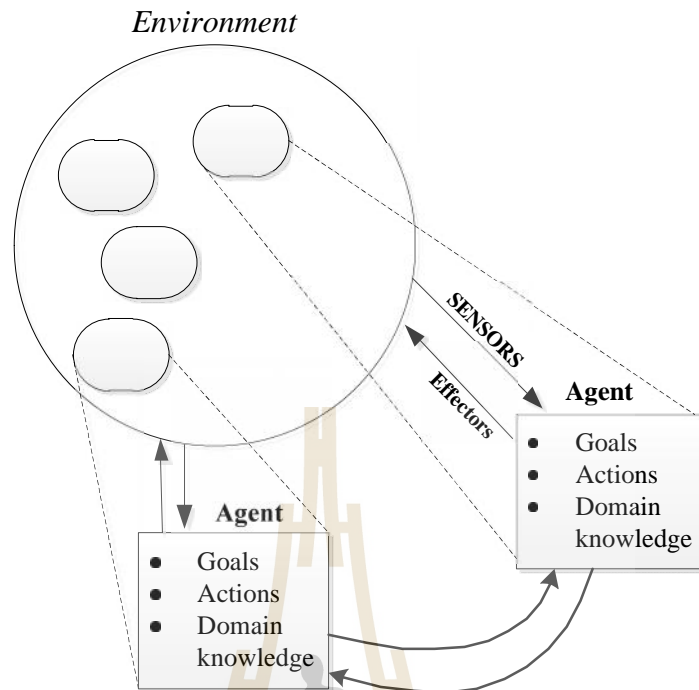


Figure 2.3 A multi-agent scenario. Each agent models each other's goals, actions, and domain knowledge. Direct interaction (communication) are indicated by the arrows between the agent.

2.5 Markov decision process theory

Markov decision process (MDP) has been used for system modeling and has many potential applications over a wide range of topics such as inventory control, computer science, maintenance, resource allocation, etc.

MDP is based on the concept of Markov process (Stroock, 2005). Markov processes (also called Markov chains), in turn, are based on two fundamental concepts: states and state transitions. A state is treated as a random variable which describes some properties of the system. A state transition describes a change in the system state at a given time instance. One can classify Markov processes into discrete-time and continuous-time Markov processes. In many cases, the system can

be modeled in either of these categories. If one is interested only in the state sequence, it is more convenient to use the discrete-time description. In this thesis, it is assumed that the environment the multi-agent systems is a discrete-time. The foundation of Markov decision process is presented as follows.

2.5.1 Markov Decision Process

In the preceding description of Markov processes, we assumed that the system model is known so that the transition probabilities and the transition rates are given. Nevertheless, in many cases the state transitions can be controlled by the system itself or the system user. In such situations, the goal is thus to find the optimal control decisions. Markov decision theory provides a framework for analysis of the probabilistic sequential decision processes. In this section, we concentrate on discrete-time stationary processes with infinite planning horizon. In general, a Markov decision model can be formulated from the system cost or the system reward perspective. These two perspectives are basically equivalent since the maximization of the system reward under given potential maximum reward corresponds to the minimization of the system cost. In addition, the corresponding decision models are essentially the same. In this thesis, we use the reward maximization formulation. The objective of our Markov decision is thus to find an optimal control policy which maximizes the long-time average reward per unit time.

A Markov decision process (MDP) is a discrete-time random decision process contains a set of possible states (S), a set of possible actions (A), a real-valued reward function $R(s, a, s')$, and the one-step state transition of the environment. The

goal for a agent is to maximize the expected sum of (discounted) rewards for a state $E[\sum_t \gamma^t R(s_t)]$, where $0 \leq \gamma \leq 1$ is the discount factor.

In MDP, it is assumed that the probability of transiting to another state only depends on the current state. Since the ultimate goal is to maximize the expected utility, we will need to learn a policy function f that maps states to actions. Given any state s and action a , the probability of occurrence of each possible next state s' is

$$P(s' | s, a) = P(S_{t+1} = s' | S_t = s, a_t = a). \quad (2.1)$$

This equation is called the state transition probability. At time step t , at a current state $s_t = s$ and action $a_t = a$, a reward r_{t+1} is obtained. A MDP model can be shown in Figure 2.5. The expected value of the incurred reward for any given state, action, and next state is

$$R(s, a, s') = E[r_{t+1} | S_{t+1} = s', S_t = s, a_t = a], \quad (2.2)$$

where $E[\cdot]$ is the expectation operator. Equation (2.1) and (2.2), completely specify the most important aspects of the dynamics of the MDP.

Figure 2.4 A MDP model.

In order to find an optimal policy, we need to define the optimal value function of a state. In other words, the optimal value function is the expected utility we will get at state s if the optimal action is selected, i.e.,

$$V^*(s) = \max_a \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V^*(s')] \quad (2.3)$$

This equation can be called the Bellman Equations which characterize the optimal values. It is also common to define a new quantity called a Q-value with respect to state-action pairs

$$Q^*(s, a) = \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V^*(s')] \quad (2.4)$$

In other words, $Q(s, a)$ is the expected utility for starting at state s , taking action a , and taking optimal actions thereafter. Thus, $V(s)$ and $Q(s)$ are related according to the following equation

$$V^*(s) = \max_a Q^*(s, a). \quad (2.5)$$

To determine the optimal policy, policy iteration and value iteration algorithms are used (Puterman, 1994). However, policy iteration is generally an improvement over value iteration because policies often converge long before the value functions do. Policy iteration, we alternate between policy evaluation and policy improvement steps. In policy iteration starts with an initial policy f . Then the value functions of each state when following policy f is determined by

$$V^f(s) = \sum_{s'} P(s, f(s), s') [R(s, f(s), s') + \gamma V^f(s')]. \quad (2.6)$$

Note that the maximum operator no longer needed become policy $f(s)$ gives us the action for state s . A policy improvement is obtained by finding an action at state s such that

$$f^*(s) = \arg_a \max \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V^*(s')] \quad (2.7)$$

After policy improvement, policy evaluation following (2.6) under the improved policy is performed. Then policy according to (2.7) is performed. The iterated until convergence is achieved.

2.6 Reinforcement learning

“Machine Learning is the study of computer algorithms that improve automatically through experience” (Mitchell, 1997). There exists three major learning methods in Machine Learning, i.e., supervised learning, unsupervised learning and reinforcement learning (RL). In supervised learning, the learning system is provided with training data in the form of pairs of input objects (often vectors) and correct outputs. The task of the supervised learner is to learn from these samples the function that maps the input to outputs and to predict the value of this function for any valid input object and to generalize from the presented data to unseen situations. On the other hand, in unsupervised learning, the system is given no a priori output and the learner has to learn a model that fits to the observations. RL is located between supervised and unsupervised learning. In particular, RL is “learning what to do –how to map situations to actions– so as to maximize a numerical reward signal” (Sutton et al., 1998). The learner is not told which the correct actions are but it has to determine (learn) them through continuous trial-and-error interactions with a dynamic environment in order to achieve a particular goal.

In the standard RL model, the learner and decision-maker is called an agent and is connected to its environment via perception or sensing, and actions. Figure 2.6 shows the agent can detect changes in the environment from the reward signal and

respond to the changes by taking an action. More specifically, the agent and environment interact at each of a sequence of discrete time steps t . At each step of the interaction, the agent senses some information about its environment (input), determines the world state and then chooses and takes an action (output). The action changes the state of the environment and this of the agent. One time step later, the value of the state transition following that action is given to the agent by the environment as a scalar called reward. The agent should behave so as to maximize the received rewards, or more particularly, a long-term sum of rewards.

Let s_t be the state of the system at time t and assume that the learning agent chooses action a_t , leading to two consequences. First, the agent receives a reward r_{t+1} from the environment at the next time step $t+1$. Second, the system state changes to a new state s_{t+1} .

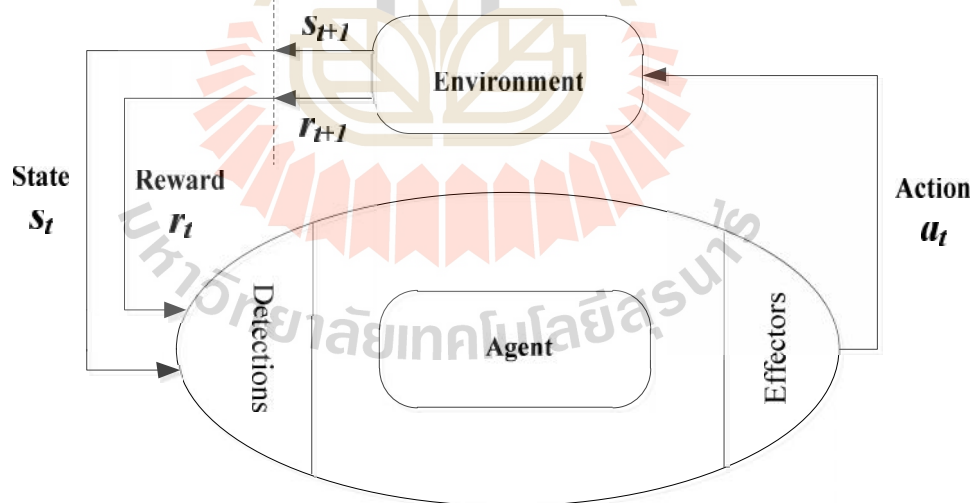


Figure 2.5 Diagram of agent-environment interaction in reinforcement learning.

As oppose to policy iteration or value iteration which require models of the system, RL is a model-free learning schemes. Model-free learning schemes does not

require computation of the state transition model but instead, but only compute the actual value function directly by estimating from interacting with the environment. Given that certain conditions are satisfied, the value function do coverage albeit slowly, since direct value function estimation does not take into account the Bellman Equations.

In particular, in model-free learning, the value functions are updated according to the following equation after each state transition (s, a, s', r) ,

$$V^f(s) = (1 - \gamma)V^f(s) + \gamma[R(s, f(s), s') + \gamma V^f(s')]. \quad (2.8)$$

Such method is called temporal difference (TD) learning (Sutton et al., 1998). TD learning works by adjusting the value function estimates towards the ideal equilibrium as stated by the Bellman equations. Because TD learning does not require state transition probabilities, Q-values are learned instead, since it is easier to extract actions from Q-values. The Q-values are updated according to,

$$Q(s, a) = (1 - \gamma)Q(s, a) + \gamma[R(s, a, s') + \gamma \max_a Q(s', a')]. \quad (2.9)$$

Because the Q-value makes the action explicit, we can estimate the Q values on-line using a method essentially the same as TD. These Q values are also used to define the policy, because an action can be chosen just by taking the one with the maximum Q value for the current state. The Q-learning rule is

$$Q(s, a) = Q(s, a) + \gamma[r + \gamma \max_a Q(s', a') - Q(s, a)]. \quad (2.10)$$

where (s, a, s', r) is an experience tuple as described earlier. If each action is executed in each state an infinite number of times on an infinite run and γ, α is decayed

appropriately, the Q-values will converge with probability 1 to $Q^*(s, a)$. However, the above mentioned principle is based on a single agent Q-learning, which is appropriate for the small size and low complexity problem. However, in this thesis, Q-learning is applied to the area coverage control, which requires a large number of sensor nodes in the same area. Therefore, Single agent Q-learning is not suitable for our framework. The multi-agent Q-learning scheme is more suitable to cater decision-making of several sensor nodes so that the agent (each sensor node) can find the optimal policy for the entire coverage area.

2.7 Multiple agent Q-learning algorithm

Multi-agent systems differ from single-agent systems in that there are many different agents that learn a task and that all of the agents' actions affect the environment. Thus, the optimal policy does not rely on only one agent, but conditions on all agents. There are works which directly applied single agent Q-learning to multi-agent systems where an individual agent maximizes its own benefit. By doing so, their works neglect the presence of the other agents. As a result, suboptimal decisions may be obtained. Therefore, an individual agent should take account of the effect of joint actions as a more suitable strategy for multi-agent systems.

2.8 Distributed reinforcement learning

In recent years, several extensions to RL and Q-learning for distributed systems have been proposed. Many interesting problems which require solving with reinforcement learning (RL) also have properties that make distributed solutions desirable. In scenarios where the state and/or action space are large, a distributed

approach to perform the computation is desirable because it speeds up computation. In many systems such as WSNs, access to sensors and actuators is inherently distributed, thus making a distributed solution method an attractive alternative to implementing a global high bandwidth communication network. Potential applications include control of power grids (or any other distribution of a resource such as water, gas, etc.), automobile traffic control, electronic network routing, control of robot teams, and communication networks. Figure 2.7 illustrates a distributed RL framework.

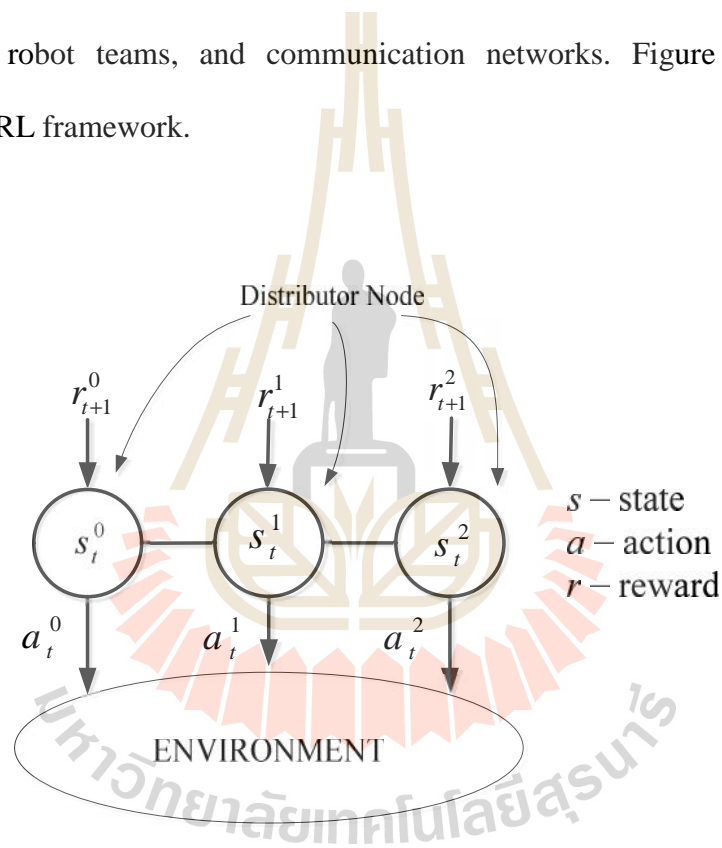


Figure 2.6 Distributed RL diagram representing logical nodes in the distributed RL formulation. Each node senses its own state of the environment, takes its own action, and receives its own reward signal.

2.8.1 Distributed value functions

In this section, we present an algorithm for distributed reinforcement learning based on distributing the representation of the value function between nodes. Each node in the system only has the ability to sense state locally, choose actions locally, and receive rewards locally. The goal of the system is to maximize the sum of the discounted rewards over all nodes and over time. However, each node is allowed to give its neighbors the current estimate of its value function for the states it transits through. A value function learning rule (described in the next section) uses information that allows each node to learn a value function that is an estimate of a weighted sum of future rewards for all the nodes in the network. With this representation, each node can choose actions to improve the performance of the overall system.

2.8.2 Distributed value function (DVF) algorithm

Usually, in MAS, agents only have local state information since the global state of the system is not fully observable from each agent's point of view. Hence, (Schneider et al., 1999) proposed a Q-learning based algorithm for the distributed value function (DVF) algorithm. This approach allows each node to compute its local value function based only on available local information. Hence, agents only need to transmit the estimated value of the current state they land in, i.e. $V^i(s_t^i)$ for agent i at time t at each iteration. The update rule at time step t for agent i is given by

$$Q_{t+1}^i(s_t^i, a_t^i) = (1-r)Q_t^i(s_t^i, a_t^i) + r(r_{t+1}^i(s_{t+1}^i) + \alpha \sum_{j \in \text{Neigh}(i)} f^i(j)V_t^j(s_{t+1}^j)) \quad (2.11)$$

$$V_{t+1}^i(s_t^i) = \max_{a \in A^i} Q_{t+1}^i(s_t^i, a), \quad (2.12)$$

where τ is the learning rate, $f^i(j)$ are factors that weigh the value functions of the neighbors of agent i such that

$$f^i(j) = \begin{cases} \frac{1}{|Neigh(i)|} & , \text{ if } Neigh(i) \neq 0 \\ 1 & , \text{ otherwise,} \end{cases} \quad (2.13)$$

where $j \in Neigh(i)$ is the set of neighbors of node i .

2.9 Summary

In this chapter, an overview of the Markov Process, Markov Decision process, reinforcement learning and a multi-agent Q-learning algorithm called Distributed Value Function (DVF) has been given. This algorithm was used to determine maximum coverage control in WSNs. The main point for selecting this method was that the algorithm allows the agent to rationally determine the near-optimal policy and receive maximum coverage and maximum trade-off between achieved coverage with energy consumption in WSNs. In the next chapter, the DVF algorithm is applied to the coverage control problem in WSNs.

CHAPTER III

A MULTI-AGENT SCHEME FOR ENERGY-EFFICIENT COVERAGE CONTROL IN WIRELESS SENSOR NETWORKS

3.1 Introduction

Wireless sensor networks (WSNs) are a collection of numerous cheap sensory devices installed within a particular environment to gather the physical parameters of interest. Measurements of these sensor devices are then acquired and relayed through the network to be processed or collected at the base station. Such data acquisition gives the ability to continuously monitor the particular surroundings of interest and respond quickly to any changes that may incur. WSNs have emerged in biomedical, military, agricultural monitoring and control applications (Alaiad et al., 2015), (Hussain et al., 2009), (Santoshkumar et al., 2015). In smart homes or buildings, lighting control have been a particular application which coverage control is needed to reduce energy consumption while maintaining a required level of light intensity.

Coverage control problems have been a significant issues arising in wireless sensor networks with the aim to extend the longevity of network lifetime and efficient energy consumption in the network due to the limited on-board battery power of a sensor node (Wang et al., 2011). Vu et al. (2009) and (Chen et al., 2009) proposed a distributed wireless sensor network with adjustable sensing radii enabling a flexible and efficient coverage. In (Ye et al., 2006), the authors proposed a Probing Environment and Adaptive Sleeping (PEAS) scheme which is a coverage

maintenance scheme that increases the network lifetime by maintaining a necessary number of working nodes and shutting down the rest as reserve. By querying neighboring nodes, a particular working node can determine the status of neighboring working and sleeping nodes prior to deciding on its own status. In (Zhang et al., 2005), the optimal geographical density control (OGDC) scheme was proposed as a guaranteed full coverage control scheme based on grid redundancy check and sequential node activation. The grid redundancy requires that each sensor node maintain a list of the grid points it covers. The sequential node activation requires that each active node sends out activation messages to neighboring nodes to reset their timers.

The aforementioned coverage control methods are non-learning schemes which require reconfiguration should the environment change. Due to node deployment in potentially wide areas, direct access to reconfigure the nodes may not be feasible. On the other hand, multi-agent system (MAS) technologies have shown to be promising due to their flexibility and self- adaptability which caters autonomous self-awareness at sensor nodes (Phuphanin et al., 2016). In a multi-agent system (MAS), nodes act as agents which have the ability to learn and adjust their coverage in a distributed manner thereby enabling a light weight self-adaptive coverage control. The nodes in a MAS decide their actions in a cooperative manner to achieve a mutual goal of maximizing the network coverage by using the minimal amount of energy. To do so, a cost function based on a function of redundant coverage areas of a sensor node is introduced.

The contribution of this chapter is thus twofold: 1) a modified multi-agent coverage control scheme based on a redundancy coverage area cost function; 2)

comparison of the scheme with non-learning coverage control schemes, i.e., OGDC and PEAS. The objective of the proposed algorithm is to maximize the coverage control efficiency by maximizing the obtained coverage control per unit of energy consumed. Our results suggests the suitability of applying MAS in coverage control in WSNs.

3.2 Multi-agent coverage control

3.2.1 Distributed Value Function Scheme

A multi-agent coverage control scheme called the Distributed Value Function (DVF) has been proposed to co-ordinately and cooperatively improve the coverage control performance in wireless sensor networks (Tham et al., 2005). In this method, each node communicates and exchanges information about its value function. A value function is a function that quantifies how well the agent (sensor node) performs at a given state $s \in S$ where S is a discrete set of all possible states of the sensor network. Let $a \in A$ be the action selected by an agent, where A is the discrete set of all possible actions available at each state. The decision rule of an agent, so called policy f , is defined as a rule which the agent selects an action as a function of its state. In other words, it is the mapping from a state $s \in S$ and $a \in A$ action to the probability of selecting action a at state s . The *value function* of state s under a given policy f is formally defined by $V^f(s) = E^f \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}$, where r_{t+1} is the reward of taking a particular action in a given state s at time t , γ is the discount factor and $E^f \{ \}$ is the expectation operator. Similarly, we define the *action value function* of taking action a at a given state under policy f by

$$Q^f(s, a) = E^f \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\} \quad (3.1)$$

The objective is to find a policy f^* such that $f^* = \underset{\forall f}{\operatorname{argmax}} Q^f(s, a)$. To achieve this objective, each agent i (node) in the DVF algorithm performs an update of its own action value function. The update rule at time step t for agent i is given by (Tham et al., 2005):

$$Q_{t+1}^i(s_t^i, a_t^i) = (1-\gamma)Q_t^i(s_t^i, a_t^i) + \gamma(r_{t+1}^i(s_{t+1}^i) + \gamma \sum_{j \in \operatorname{Neigh}(i)} f^i(j)V_t^j(s_{t+1}^j)) \quad (3.2)$$

$$V_{t+1}^i(s_t^i) = \max_{a \in A^i} Q_{t+1}^i(s_t^i, a) \quad (3.3)$$

where γ is the learning rate, $f^i(j)$ are factors that weigh the value functions of the neighbors of agent i such that

$$f^i(j) = \begin{cases} \frac{1}{|\operatorname{Neigh}(i)|} & , \text{if } \operatorname{Neigh}(i) \neq 0 \\ 1 & , \text{otherwise} \end{cases} \quad (3.4)$$

where $j \in \operatorname{Neigh}(i)$ is the set of neighbors of node i (Tham et al., 2005).

3.2.2 Modified DVF Framework for Coverage Control

Consider a wireless sensor network comprising multiple light sensor nodes. For a particular sensor node i , the local state and actions taken are defined as follows.

Local agent state: Each sensor node i can sense the level of coverage in its area. Its local state s^i is state of each agent based on its mode and coverage area.

Local agent actions: Each sensor node i has the ability to take one of the following actions in any state it lands in. The action space A^i is the set of all possible actions for each state $A^i = \{Action_0, Action_1\}$ where $Action_0$ ($Action_1$) refers to sensor node i turning off (on).

Each action decided by sensor node i results in a reward, denoted as $r^i(s_t^i)$ which is a function of sensor node i 's state s^i at time t defined by

$$r^i(s_t^i) = G^i(s_t^i) - C^i \quad (3.5)$$

where $G^i(s_t^i)$ is a function of the number of cells within the coverage area of sensor node i such that

$$G^i(s_t^i) = Area_coverage(a^i) \times GAIN_CELL_BRIGHT, \quad (3.6)$$

and C^i is the area overlapped as a result from the action taken by sensor node i at time t .

3.3 Performance Evaluation

To compare the coverage control performance of a WSN, we considered a gridded area of 1000 x 1000 sq.m. containing a number of sensor nodes ranging from 100,200,300,400,500 sensor nodes placed randomly in the area.

The objective is for the sensor node to learn to cooperate with one another in order to completely coverage area in an energy-efficient way, i.e. minimize the number of sensor node turned on. The coverage area of node (agent) i was given within a transmission range of 100m. The initial energy of each sensor node was 10 Joule. Comparison was based on the number of working nodes, coverage percentage

and coverage lifetime. The DVF was compared with OGDC, PEAS8 and PEAS9, where the latter two are PEAS with probing ranges of 80 and 90m, respectively.

According to equations (3.2) and (3.6), the value of the learning rate $r = 0.4$, the discount factor $\gamma = 0.7$ and the $GAIN_CELL_BRIGHT = 0.5$. The values of the learning rate and discount factor were obtained from experimenting a range of values and selecting the parameters which received the best performance in terms of average accumulated reward in (3.5). The simulation results were averaged over 10 runs to achieve the desired accuracy.

Figure 3.1 depicts the number of working nodes against the number of deployed nodes in each algorithm. Note that OGDC, PEAS8 and PEAS9 consistently use a gradually increasing number of working nodes which is higher than DVF. The reason is due to DVF determines the working nodes which achieves the best coverage while saving energy consumption.

Figure 3.2 shows the percentage of coverage achieved by all algorithms against the number of deployed nodes. Note that above 200 deployed nodes, all algorithms can achieve full coverage with DVF attaining 99.2% coverage at 200 nodes and 99.8% at 500 deployed nodes. The reason is because DVF must conservatively select working nodes so as to reduce the amount of energy consumption. Even so, the modified DVF can achieve a nearly full coverage with only 13-64% of active sensor nodes whereas the OGDC and PEAS required 14-68% and 16-76% of active sensor nodes, respectively for high to low node densities.

Figure 3.3 illustrates the efficiency in terms of coverage area per working node for each algorithm. The rational is from the fact that coverage area attained is a

trade-off with the energy dissipated by the working node. Results show that DVF achieved the highest coverage per working node, followed by OGDC and PEAS.

Figure 3.4 - 3.6 depict the area coverage lifetime for each algorithm for various numbers of deployed nodes. The area coverage lifetime is defined by the duration from the start of network operation until the coverage requirement is no longer satisfied. It is evident that the coverage lifetime for DVF is prolonged the most in terms of number of time steps due to the least number of working nodes. PEAS8 attained the least coverage lifetime. This is because PEAS requires acknowledgement messages in addition to the higher number of working nodes than OGDC and DVF.

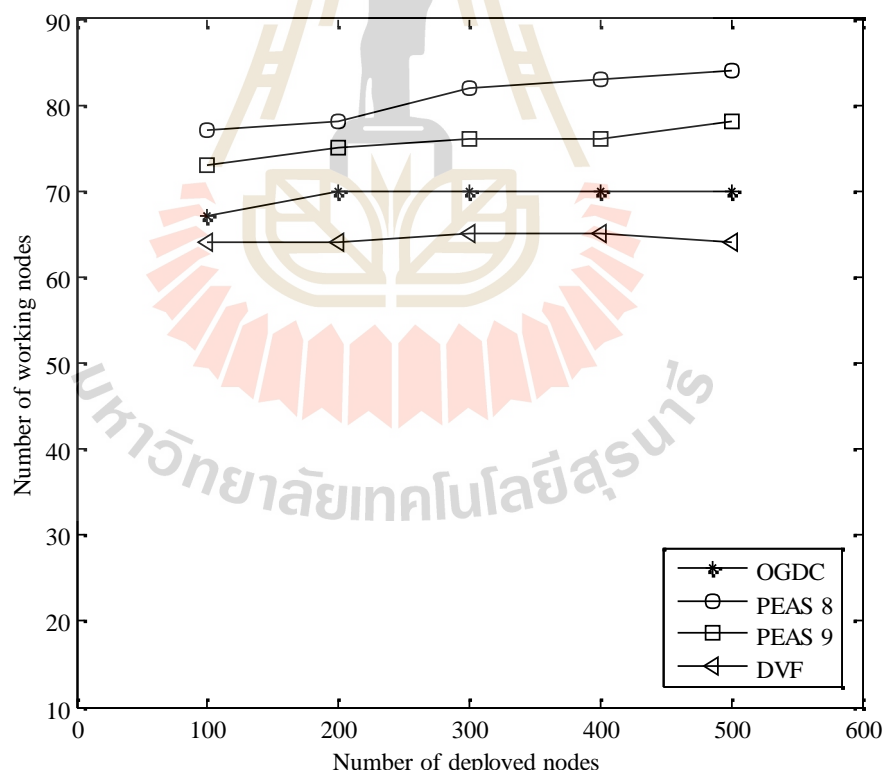


Figure 3.1 Average number of working nodes against the number of deployed nodes.

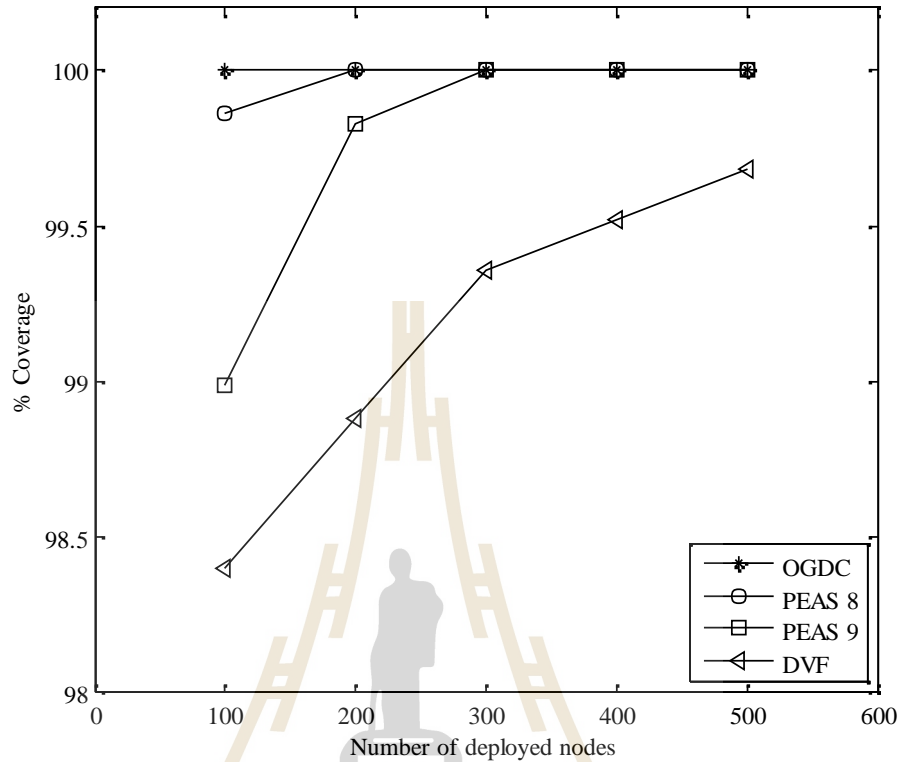


Figure 3.2 Percentage of coverage against the number of deployed nodes.

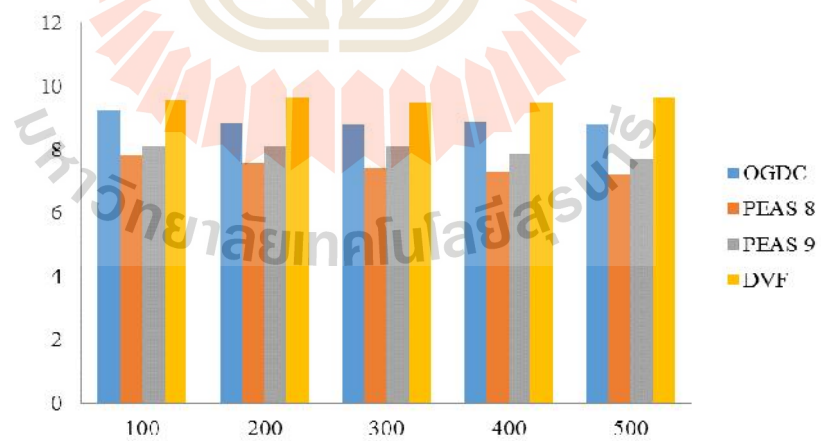


Figure 3.3 Average coverage area per working node against number of deployed nodes.

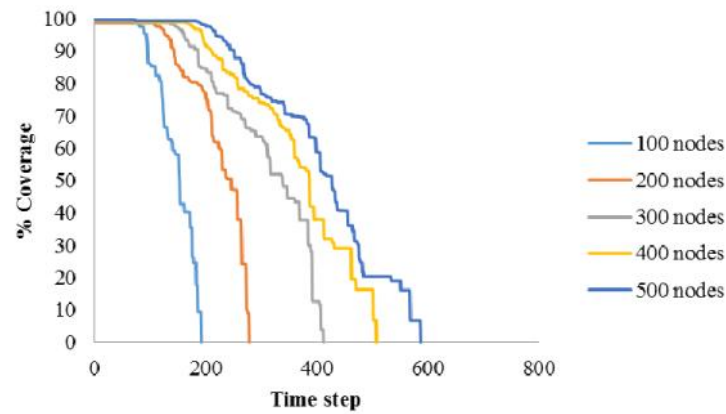


Figure 3.4 Coverage lifetime for DVF.

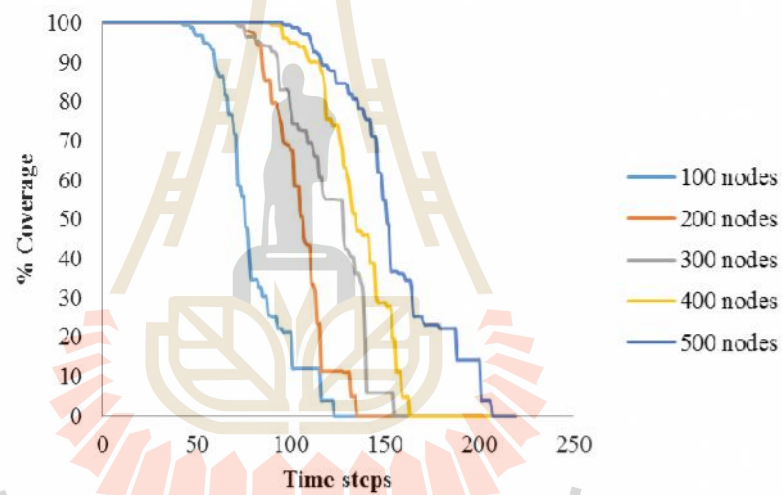


Figure 3.5 Coverage lifetime for OGDC.

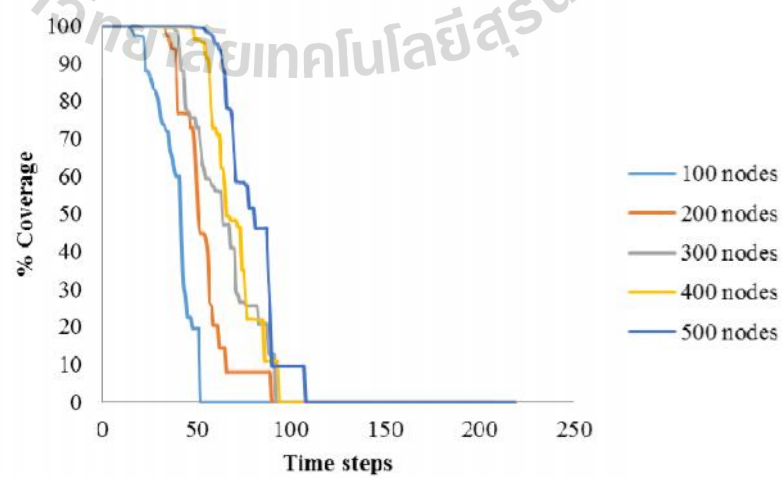


Figure 3.6 Coverage lifetime for PEAS8.

3.4 Summary

In this chapter, we have proposed a cost function-modified distributed value function (DVF) scheme which is a multi-agent scheme aimed at energy-efficient coverage control in wireless sensor networks. Results were compared with two non-learning coverage control schemes i.e., PEAS which is a partial coverage control scheme and OGDC which is a guaranteed coverage control scheme. Results showed that the proposed modified DVF attained the least working nodes of all while still achieving nearly complete coverage. Therefore, DVF outperformed PEAS and OGDC in terms of area coverage energy efficiency and area coverage lifetime. Results suggest the suitability of employing MAS for coverage control problems in WSNs. Note that the proposed objective function in (3.5) give are to a single objective optimization framework. However, to cater conflicting objectives or multiple objectives, a multiple objective optimization framework is need. This is propose in the following chapter.

CHAPTER IV

SCALARIZED Q MULTI OBJECTIVE REINFORCEMENT LEARNING FOR AREA COVERAGE CONTROL AND LIGHT CONTROL IMPLEMENTATION

4.1 Introduction

Wireless sensor networks (WSNs) consist of small computing devices with limited computational capabilities and energy supply. Various systems have developed and implemented such devices for a wide range of applications, such as automatic systems, environmental monitoring systems, elderly people monitoring systems and smart homes. These systems rely on the interaction and cooperation between the sensor nodes to carry out the operation. There are several key performance indicators to measure the performance of a WSN, such as coverage area, energy consumption, the number of working nodes, the coverage area per working node, or coverage area per unit energy consumed, etc. These metrics are vital in measuring the quality-of-service (QoS) of the network.

Coverage control has gained much research interest in wireless sensor networks. Typically, coverage control has the aim to maintain or maximize coverage while preserving network lifetime and energy consumption of the WSN. Due to the multiple parameters which can affect coverage, coverage control problems are typically formulated as optimization problems with single or multiple objectives. In this work (Zhang et al., 2005) proposed a single objective coverage control, the

optimal geographical density control (OGDC) scheme was proposed for guaranteed full coverage control. The scheme is based on grid redundancy check and sequential node activation. The grid redundancy requires that each sensor node maintains a list of the grid points it covers. Each active node sends out activation messages to neighboring nodes to reset their timers. Thus, the OGDC scheme is a deterministic optimization scheme which aims to maximize a single objective (i.e. maximize the coverage area).

Single objective coverage control has also been applied to enable energy efficient usage and convenience in smart homes or smart buildings. Ref. (Kumar et al., 2010), (Huny et al., 2011), (Mohamaddoust et al., 2015), (Meng-Shiuan et al., 2008), (Okada et al., 2015) apply coverage control for lighting control applications. Similar to (Zhang et al., 2005), (Kumar et al., 2010), (Huny et al., 2011), (Mohamaddoust et al., 2015), (Meng-Shiuan et al., 2008), (Okada et al., 2015) are also based on a single objective which is to maintain coverage (light intensity) required by user. These works do not consider energy consumption which is an important parameter in WSNs.

Most aforementioned literature focus on applying wireless sensor networks to manage a single objective of maximizing coverage (Zhang et al., 2005) or satisfying light intensity requirements (Kumar et al., 2010), (Huny et al., 2011), (Mohamaddoust et al., 2015), (Meng-Shiuan et al., 2008), (Okada et al., 2015). However, such single objective optimization schemes may well conflict with other objectives such as minimizing energy consumption, or wasteful overlapping areas. In certain applications such as in visible light communication (VLC), overlapping coverage areas is undesirable as the identification data cannot be read in the light

overlapping areas. On the other hand, the principles of multi-objective optimization (MOO) can support multiple objectives and be used to determine solutions (Iqbal et al., 2016). Multiple objectives have been considered simultaneously in many wireless sensor network applications. In (Ratnasingham et al., 2009), a multiple target tracking sensor management algorithm was investigated. The problem was to select subsets of sensors and assign frequency and minimize transmission power to working sensors in order to maximize the tracking performance of multiple targets. As for coverage control application, MOO is generally used for optimizing contradicting objectives, for example, coverage maximization, minimization of working sensor nodes, minimization the unbalanced energy consumption and minimization of energy consumption to prolong the network lifetime (Iqbal et al., 2015), (Fei et al., 2016). Singhiv et al. (2005) presented an intelligent lighting control which considered two objectives of maintaining light intensity for users and minimizing energy expenditure. Each round of decision is dependent on light intensity information gathered from the immediate indoor and outdoor environment of the occupants. It can be seen that these aforementioned works in (Iqbal et al., 2016), (Ratnasingham et al., 2009), (Iqbal et al., 2015), (Fei et al., 2016), (Singhiv et al., 2005) rely on rule-based or threshold-based decisions. Such works may not adapt well in constantly changing surroundings such as lighting control with changing external lights. Furthermore, these algorithms have centralized operations which may be suitable for small scale coverage. However, such schemes may not be suitable for implementation in individual sensor nodes for distributed coverage control.

On the other hand, there are several researches in the existing literature which applied adaptive learning methods to the MOO framework. Such learning methods

can predict the optimal policy from learning experience in presence of a constantly changing environment. (Carlos et al., 2004) presents an approach in which Pareto dominance is incorporated into particle swarm optimization (PSO) in order to allow this heuristic to handle problems with several objective functions. Jia et al. (2009) applied the multi-objective genetic algorithm (GA), called Energy-efficient Coverage Control Algorithm (ECCA), to the coverage control problem in WSNs. The objective of ECCA is to minimize the number of working sensor nodes while maximizing the coverage area. Another common algorithm to solve MOO problems is the Artificial Neural Network (ANN). Barbancho et al. (2007) used ANN to find the optimal transmission path with the objective to minimize the global energy consumption. Tafa et al. (2016) also applied ANN a problem of selecting randomly placed sensors to maximize a barrier coverage and minimize energy consumption. Although PSO, GA and ANN approaches can solve MOO problems, such algorithms are typically complex and slow in finding the optimal policy (Fei et al., 2016). Another method used to solve MOO problems is reinforcement learning (RL). RL is a learning scheme which is based on the actual interaction between an agent and the environment. Upon each action decided by the agent when the environment is in a particular state, a reward is returned and the agent uses the reward to iteratively improve its action. One common RL tool is Q-learning which is an algorithm that an agent updates iteratively to improve its actions based the goodness of state-action pair function. RL approach can be easily implemented in a distributed architecture like in WSNs. Rovcanin et al. (2015) propose a service-wise protocol optimization technique for multi-objective, co-located and complex heterogeneous network. The proposed solution efficiently combines MOO with the reinforcement learning (RL) method. Phuphanin et al.

(2016) applied a multi-agent Q-learning scheme to an energy efficient coverage control problem in WSNs. Their results show that the multi-agent learning scheme is scalable and can outperform non-learning coverage control schemes despite its low complexity. However, their multi-agent RL scheme is based on a single cost function of multiple conflicting objectives. Such a single combined objective function may not attain the best policy particularly if each objective is contradicting.

Therefore, this chapter is focused on the application of a MOO framework to an online learning scheme for coverage control in WSNs. In particular, the Scalarized Q Multi-Objective Reinforcement Learning (SQMORL) method, which uses a MOO framework is applied to the coverage control problem in WSNs. Such online learning scheme is also adaptive to changes and perturbations. The algorithm has low complexity and is distributed. Therefore, it provides a promising implementation in sensor nodes which are resource constrained. Furthermore, this work also implements a hardware testbed to evaluate the performance of SQMORL in a multi-agent lighting control experiment.

The contribution of this chapter is therefore three-fold: 1) The SQMORL coverage control and performance evaluation by means of simulation in uniform random and grid sensor layout; 2) Comparison of SQMORL with both learning and non-learning coverage control schemes in WSNs; 3) Development of a testbed and hardware performance evaluation of an automatic lighting control using SQMORL algorithm.

4.2 Multi-objective reinforcement learning for coverage control

Coverage control is a critical issue in WSNs as it is one of the parameters which affects the QoS of the network. However, an increase in area coverage may have influence on other resources such as more energy consumption as well. Furthermore, if the network has a large number of sensor nodes placed in the network, excessive number of working sensor nodes in nearby areas may result in overlapping areas which is a waste of energy. Therefore, multi-objective reinforcement learning can be applied to find the optimum policy to select sensor nodes to maintain its coverage while simultaneously reducing overlapping area and hence energy consumption.

4.2.1 Multi-objective optimization

In general, multi-objective optimization (MOO) problems include a number of objectives required for optimization simultaneously. Each objective, may be related or conflicting. Therefore, the main function of MOO is to find the equilibrium points of different objectives. A multi-objective optimization problem (MOP) with n variables and m objectives ($m > 1$) can be formulated as (Jameii et al., 2014):

$$\min \text{ or } \max g(x) = [g_1(x), g_2(x), \dots, g_m(x)] \quad (4.1)$$

subject to $x \in \Omega$, where $\Omega \subset R^n$ is the decision space, $g \subset G$ and $G: \Omega \rightarrow R^m$ consist of m real valued objective functions, and R^m is the objective space.

4.2.2 Scalarized Q multi objective reinforcement learning for area coverage control

Multi-objective reinforcement learning (MORL) problems differ from general RL problem in that there are multiple objectives to be achieved by the

learning agent. Each objective has its own reward or penalty signals. This is a basic architecture which a single agent is simultaneously faced with a set of different objectives. For each objective m and a stationary policy f , there is a corresponding action value function, $Q^f(s,a)$, which satisfies the Bellman equation. Let the vector MQ be defined by

$$MQ^f(s,a) = [Q_1^f \quad Q_2^f \quad \dots \quad Q_m^f]^T \quad (4.2)$$

where MQ is the vectored action value function, which also satisfies the Bellman equation. The optimal vector state-action value function is defined as

$$MQ^*(s,a) = \max_f MQ^f(s,a). \quad (4.3)$$

Thus, the optimal policy f^* can be obtained by

$$f^*(s) = \arg \max_a MQ^*(s,a) \quad (4.4)$$

In this basic architecture, the optimization problems of $\max_f MQ^f(s,a)$ and $\arg \max_a MQ^*(s,a)$ are both MOO problems.

In the design of the area coverage control Scalarized Q Multi-Objective Reinforcement Learning (SQMORL), a weighted-sum approach (Moffaert et al., 2013) is used. For the SQMORL algorithm, the state-action value function of each objective determined by $Q(s,a,o)$ which is a function of a state-action pair and an objective function. By applying the weighted-sum approach, we obtain the weighted sum of the objective functions

$$SQ(s, a) = \sum_{o=1}^m w_o \cdot Q(s, a, o) \quad (4.5)$$

whereby the weights are such that $w_o \in [0,1]$ and $\sum_{o=1}^m w_o = 1$. In the SQMORL scheme, each agent i (i.e., sensor node i) in the SQMORL algorithm performs an update of its own action-value function. The update rule at time step t for agent i is given by

$$Q_{t+1}^i(s_t^i, a_t^i, o) \leftarrow (1-\gamma)Q_t^i(s_t^i, a_t^i, o) + \gamma(r_{o,t+1}^i(s_{t+1}^i) + \chi \sum_{j \in Neigh(i)} f^i(j)V_t^j(s_{t+1}^j)) \text{ , for } o=1,2 \quad (4.6)$$

$$V_{t+1}^i(s_t^i) = \max_{a \in A^i} Q_{t+1}^i(s_t^i, a, o), \quad (4.7)$$

where γ is the learning rate where $\gamma \rightarrow 1$ refers to a rapid learning rate as the old estimate ($Q_t^i(s_t^i, a_t^i, o)$) is forgotten rapidly, $f^i(j)$ is a factor that weighs the value function of neighbour j of agent i such that

$$f^i(j) = \begin{cases} \frac{1}{|Neigh(i)|} & , \text{ if } Neigh(i) \neq \emptyset \\ 1 & , \text{ otherwise} \end{cases} \quad (4.8)$$

where $j \in Neigh(i)$ is in the set of neighbors of node i . Thus, the optimal policy f^* is a policy that satisfies

$$f^*(s) = \arg \max_a SQ^*(s, a). \quad (4.9)$$

In this chapter, the SQMORL framework is applied to the coverage control problem. The following assumptions are used.

Local agent state: Assume that each agent can sense the coverage area. Let i be the index of a sensor node and S_t^i be the local state of sensor node i (agent i) at time t . Its local state S_t^i is the state of each agent i which is based on its coverage area.

Local agent actions: Let A^i be the set of all possible actions for each agent. Each agent i has the ability to take one of the following two actions in any state it lands in, i.e., $a^i \in A^i$ where $A^i = \{Action\ 0\ (Turn\ off\ sensor),\ Action\ 1\ (Turn\ on\ the\ sensor)\}$.

Objective functions: There are two objective functions. The first objective function is to achieve maximized coverage area. The reward function of objective function is given by:

$$r_1^i(s_t^i, a^i) = Area_coverage(a^i) \times GAIN_CELL_BRIGHT, \quad (4.10)$$

where $Area_coverage(a^i) \times GAIN_CELL_BRIGHT$ is a function of the number of cells within the coverage area of sensor node i . The second objective is to achieve minimized overlapping area that occurs between sensor nodes located nearby in order to reduce the energy consumption.

$$r_2^i(s_t^i, a^i) = C^i, \quad (4.11)$$

where C^i is the overlapping area in terms of the number of cells in state s_t^i as a result from action $a^i \in A^i$ taken by sensor node i . Thus, the objective function can be defined separately for each of the objective function as $g_1(s, a) = max[areacoverage]$ and $g_2(s, a) = min[area_overlapping]$. Figure 4.1 depicts the pseudo code of SQMORL. The algorithm converges provided that all state-action pairs are visited infinitely often (Moffaert et al., 2013).

 Scalarized Q Multi Objective Reinforcement Learning for area coverage control

BEGIN

```

1  Random topology
2  for time step 1: end time step
3    Each agent chooses action  $a^i \in A^i$ , reward  $r_1^i, r_2^i$ , and next state
4     $s^i =$  number cell coverage
5    Update  $Q^i$ 
6       $Q_{t+1}^i(s_t^i, a_t^i, o) \leftarrow (1-\gamma)Q_t^i(s_t^i, a_t^i, o) + \gamma(r_{o,t+1}^i(s_{t+1}^i)$ 
           $+ \alpha \sum_{j \in \text{Neigh}(i)} f^i(j)V_t^j(s_{t+1}^j))$  , for  $o=1,2$ 
7    where  $V_{t+1}^i(s_t^i) = \max_{a \in A^i} Q_{o,t+1}^i(s_t^i, a, o)$ ,
           $f^i(j) = \begin{cases} \frac{1}{|\text{Neigh}(i)|} & , \text{ if } \text{Neigh}(i) \neq 0 \\ 1 & , \text{ otherwise} \end{cases}$ 
8       $SQ(s, a) \leftarrow \sum_{o=1}^m w_o \cdot Q(s, a, o)$  ,  $m=2$ 
9  endfor
END

```

Figure 4.1 Pseudo code of SQMORL for lighting coverage control.

4.3 SQMORL for area coverage control performance evaluation: simulation part

This section evaluates the performance of the SQMORL algorithm for area coverage control in WSNs. We consider an area of 35 x 35 sq.m. space. Each sensor node has an area coverage of radius 5 m which covers an area of 81 cells per sensor node. Therefore, there are 82 possible states for each agent in the system. The sensors are laid out in 1) a uniform randomly placement with varying number of 25, 50, 75, 100 sensor nodes; and 2) a grid placement with varying number of 25, 81, 121, 169 sensors nodes. From Figure 4.1, the learning rate is 0.4, the discount factor is 0.7 and GAIN_CELL_BRIGHT is 0.5. These are the values of the learning rate and discount factor which allows the system to perform best and have been evaluated

from the experiment. In the MOO framework, the weights in line 8 of Figure 4.1 represent a trade-off which is parameterized by $w_1, w_2 = 0.5$. This represents a scenario which both objectives are equally important. The simulation results are averaged over 10 repeated runs.

When each sensor node is placed in the area, each sensor node must learn to adjust its decision in order to discover the optimal action. The optimal action is one which satisfies the purpose of covering the maximum possible area and attaining the least number of active nodes. For performance comparison the following metrics are measured: the number of working sensor nodes selected, the percent of coverage area and the ratio of coverage area per working sensor node. The SQMORL algorithm is compared to the Distributed Value Function (DVF) (Phuphanin et al., 2016) which is a multi-agent reinforcement learning scheme and the non-learning Optimal Geographical Density Control (OGDC) scheme which has guaranteed full coverage (Zhang et al., 2005).

4.3.1 Simulation results: uniform random layout

In this scenario, the sensor nodes are placed in the area following a uniform random placement. We first consider the number of working sensor nodes which each algorithm decides to cover the area. Figure 4.2 shows the number of working sensors nodes at node densities of 25 to 100 nodes per area. The DVF and SQMORL algorithms use 8 to 16 and 8 to 15 working nodes, respectively. As for the OGDC algorithm, it uses 18 to 26 working nodes to cover the area. From the figure, results indicate that SQMORL algorithm uses a comparable number of working nodes to the DVF algorithm while the OGDC algorithm uses the most number of working nodes.

Figure 4.3 depicts the percentage of coverage area obtained by each algorithm given the number of working nodes, for node densities of 25 to 100 nodes per area. Note that the DVF, SQMORL and OGDC algorithm can attain 75 to 95, 79 to 97 and 88 to 100% of coverage, respectively. At node density of 25 sensor nodes, no algorithm can attain full coverage due to the insufficient number of nodes. In terms of coverage area, the OGDC algorithm has more coverage than the other algorithms. At 100 nodes, SQMORL algorithm attains 96.5% coverage which is 1% more than DVF algorithm and 3.5% less than OGDC algorithm. However, SQMORL uses 1 and 11 fewer nodes than DVF and OGDC, respectively (see Figure 4.2).

To show the energy efficiency, in terms of the coverage area size (in cells) per working node, we compare the ratio of the number of cells in the coverage area over the number of working sensor nodes in Figure 4.4. The SQMORL algorithm can outperform DVF and OGDC algorithm in all cases. This is because SQMORL uses fewer number of working nodes, while the coverage area is comparable to the two other algorithms. Results show that though OGDC can attain the maximum coverage, it is at the expense of high number of working nodes.

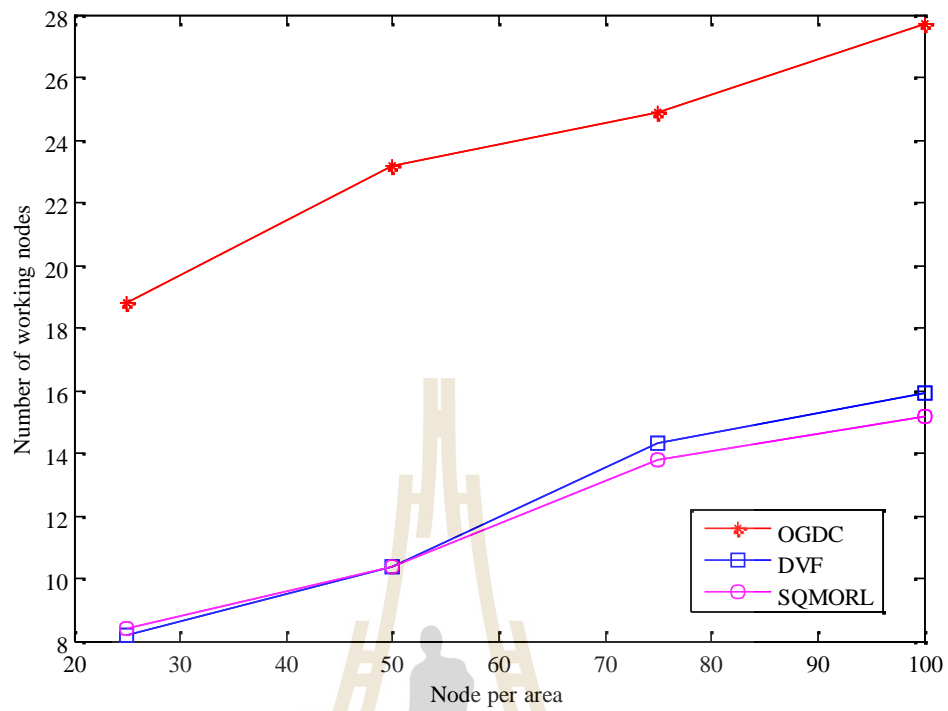


Figure 4.2 Number of active nodes against number of nodes placed in the network.

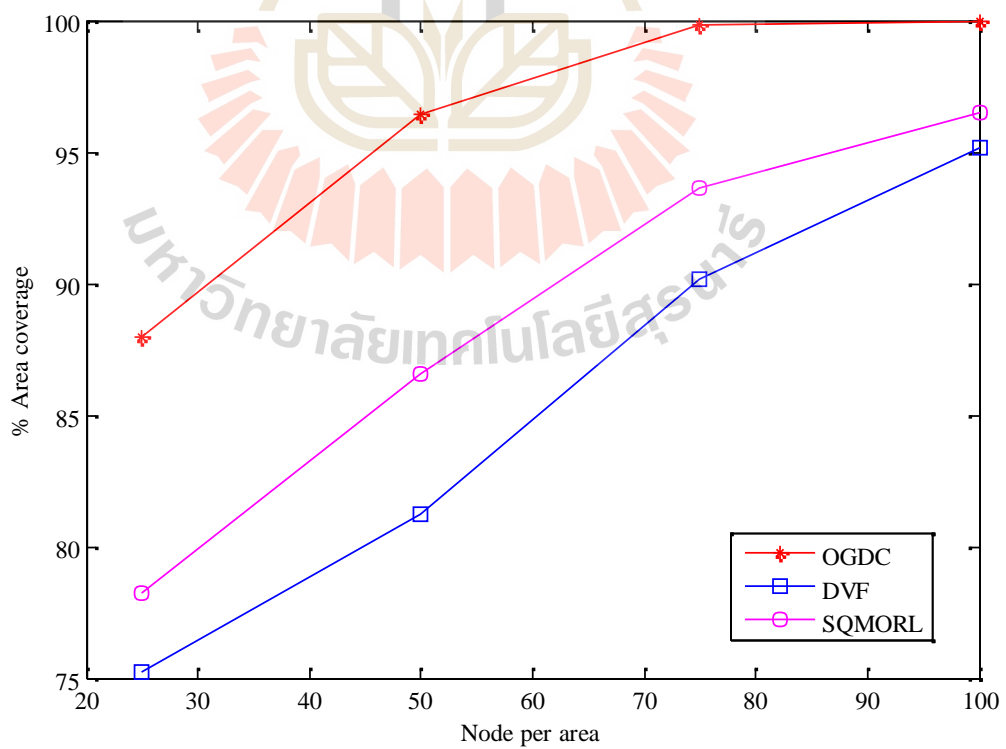


Figure 4.3 Percentage area coverage against number of nodes placed in the network.

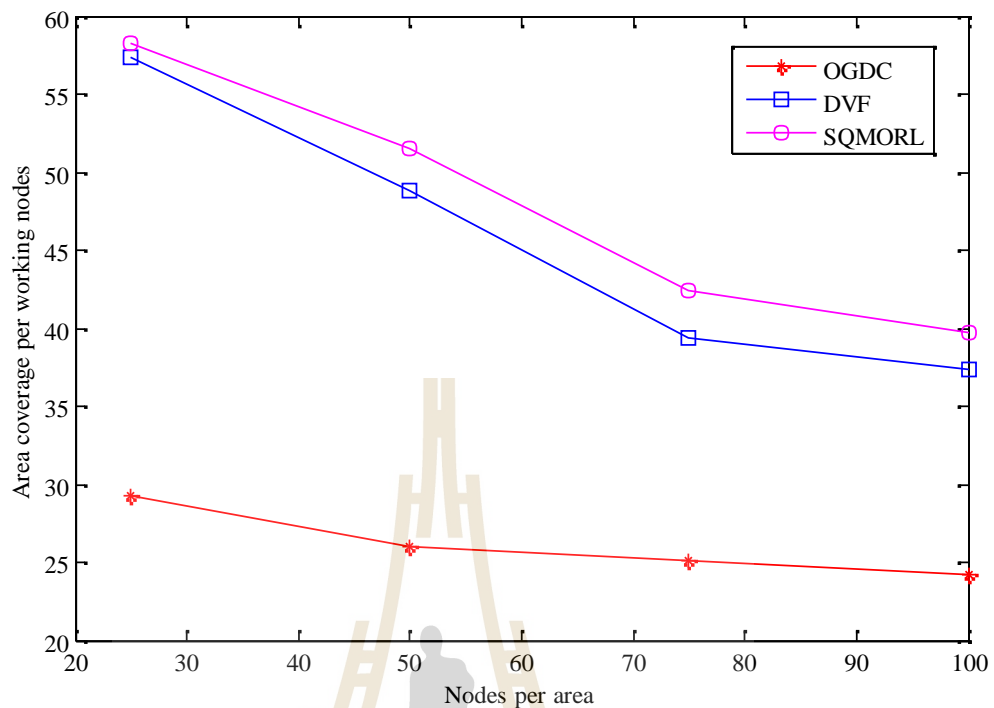


Figure 4.4 Ratio between the coverage area and working sensor nodes

4.3.2 Simulation result: grid position

In order to evaluate SQMORL in a scenario similar to the light bulb placement as the automatic lighting control testbed, simulation was also conducted in a grid layout of sensors nodes. The number of sensor nodes is varied to 25, 81, 121 and 169 nodes. The sensors are placed in a regular grid spaced 5m apart in an area of 30 x 30, 50 x 50, 60 x 60 sq.m. space, respectively. The purpose of this experiment is to evaluate each algorithm in a setting of light bulbs placed indoors of a building. The SQMORL, DVF and OGDC algorithms have been compared.

Figure 4.5 shows the number of working nodes selected to cover the area. The SQMORL algorithm used 9 to 49 node working nodes, the DVF algorithm used 9 to 54 working nodes, whereas OGDC used the most working nodes with its selection of 22 to 166 nodes. This is because the SQMORL algorithm can select non-

overlapping working nodes which cover the most area, whereas the DVF algorithm missed some positions. On the other hand, OGDC is only focused on uncovered area and thereby activated almost every node in the uncovered area, thus used the most number of working nodes.

Figure 4.6 depicts the percentage of area coverage that each algorithm attained given their selected number of working nodes. It is found that the coverage remains relatively constant as the sensor nodes are placed at regular grid positions throughout the area. The DVF and SQMORL algorithm attained approximately 80% coverage. Note that the OGDC algorithm achieved 98% coverage area for all cases as it selects working node based on cells which are not yet covered by other nodes.

Figure 4.7 shows the energy efficiency of the working nodes selected. SQMORL algorithm achieved the highest number of cell coverage per working node at 80 cells/working node. This is because SQMORL algorithm uses fewer working nodes while attaining 80% coverage of the area (as seen in Figure 4.6). On the other hand, the DVF algorithm obtained 73 to 77 cells per working node and the OGDC algorithm obtained the least energy efficiency. Even though OGDC can achieve the most percentage of coverage area (Figure 4.6), a trade-off exists as OGDC uses the highest number of working nodes (see Figure 4.7). This is because OGDC considers uncovered areas and selects working nodes which overlap. Consequently, as the number of nodes placed in the area increases, the energy efficiency of OGDC algorithm decreases.

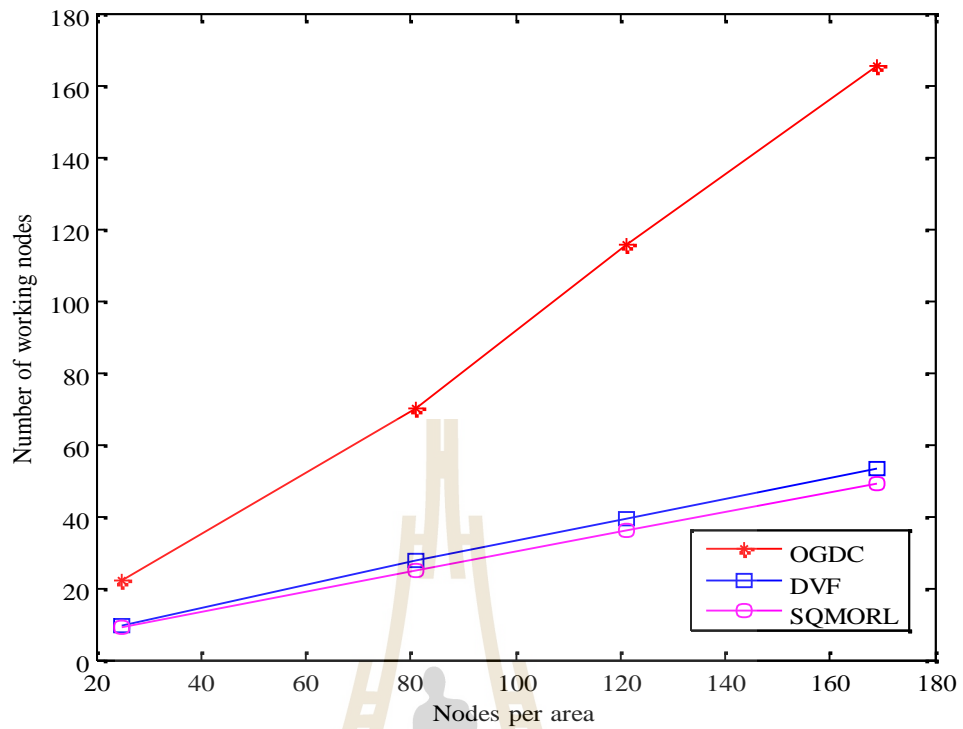


Figure 4.5 Number of active nodes against number of nodes placed in the network.

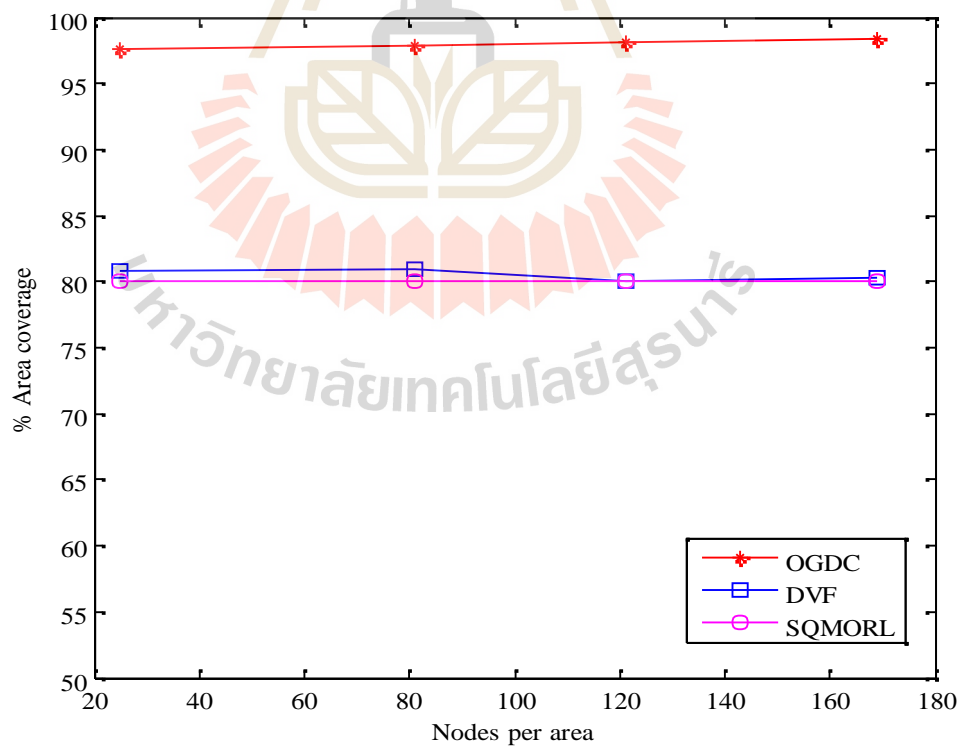


Figure 4.6 Percentage area coverage against number of nodes placed in the network.

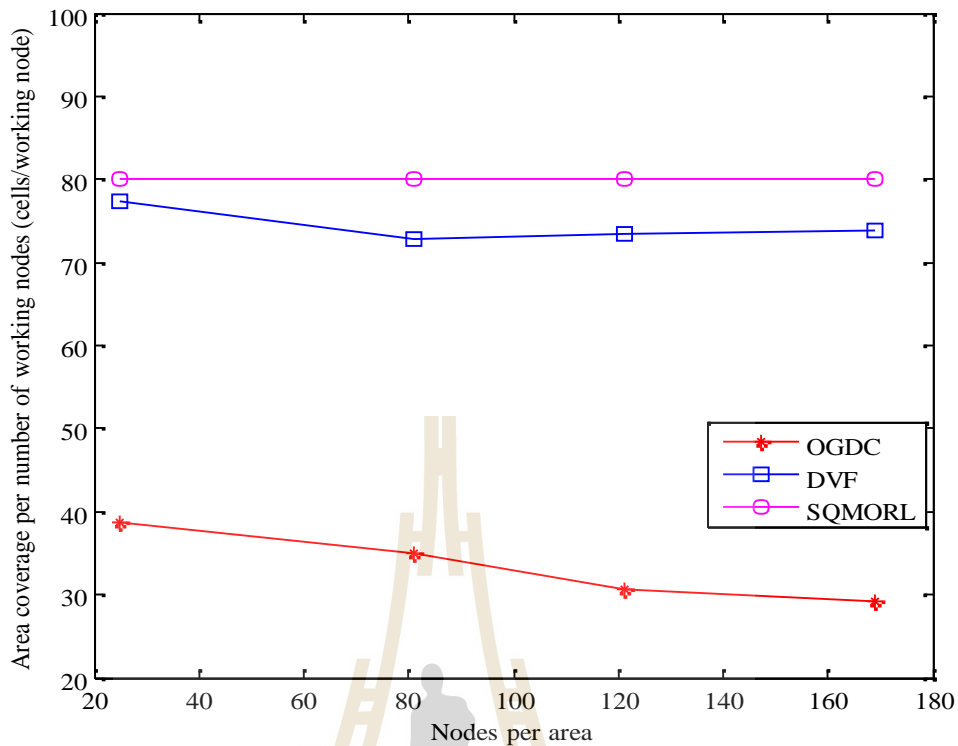


Figure 4.7 Ratio between the coverage area and active sensor nodes.

4.3.3 Discussion on simulation results

In the simulation part, we present the MOO framework for area coverage control in WSNs. There are two objective functions, i.e. to maximize area coverage objective and to reduce the area overlapping that occurs between neighboring nodes. A multi-objective reinforcement learning method in conjunction with the weighted-sum approach, called Scalarized Q Multi Objective Reinforcement Learning (SQMORL), is then applied to find the optimal policy in area coverage control. To evaluate the performance of the SQMORL algorithm, the simulation is divided into two parts, i.e., uniform random and the grid sensor layout. The performance of SQMORL is compared with DVF algorithm and OGDC algorithm. In

terms of the number of cells covered per working node ratio, the SQMORL algorithm outperforms DVF and OGDC algorithm in all cases as it requires the fewest number of working nodes. Similar results were obtained from both the uniform random and grid layout.

In the next section, SQMORL and DVF algorithms have been selected for performance evaluation in an implemented automatic lighting control testbed. These two algorithms have been selected because they both have self-learning characteristics, good efficient energy consumption with respect to area coverage and outperform OGDC.

4.4 Performance Evaluation: testbed

In this section, SQMORL and DVF are evaluated in an automatic lighting control testbed. In the testbed, each sensor is initialized to the initial default value setting i.e., Q-value, state, action and reward of each agent equal zero. There are two actions, i.e., “action 0” refers to turning off a light bulb, and “action 1” refers to turning on a light bulb. As the SQMORL convergence condition requires that every state-action pair be visited infinitely often, an “explore and exploit” scheme is implemented at each agent. In particular, each agent is set to randomly select (explore) actions for 100 times step in the training phase. This enables each agent to explore all possible state-action pairs and update the action value functions. After the training phase, these values are then used to select (exploit) optimal action according to (4.4) with some probability and explore other actions randomly with the remaining probability. This is referred to as the ϵ -greedy action selection scheme. When an action is selected, every node waits for 5 seconds for the light intensity to

become stable as a certain amount of delay is required in the hardware to turn on or off each light bulb. The nodes then measure the resulting light intensity and obtain the reward values from equations (4.10), (4.11). Each node then updates their state-action value functions according to equation (4.6). Then agent sends its own value function in (4.7) to its neighboring sensor nodes. The neighbors can then have the up-to-date value functions for their own state-action value updates. The process is repeated as shown in Figure 4.8 until convergence is achieved, i.e., agent can find the optimal action.

4.4.1 SQMORL automatic lighting control result

In order to evaluate the performance of the DVF and SQMORL algorithm, an automatic lighting control test bed was developed in the Wireless Communication Laboratory F4, Suranaree University of Technology (SUT), Thailand.

The automatic lighting control system consists of sensor nodes, each of which is equipped with a wireless communication module with XBee Series 2, a microcontroller part with Arduino Uno R3 and an additional external memory unit for recording measurements for control purposes, a light dependent resistor (LDR) to measure the light intensity. Each sensor node has the ability to measure the intensity of light within its own area, exchange information between the neighboring node sensors and collect the data at the memory unit. Five sensors nodes are placed at the positions as shown in Figure 4.9, in the experiment room of which external light is blocked. The results are averaged over 20 repeated runs.

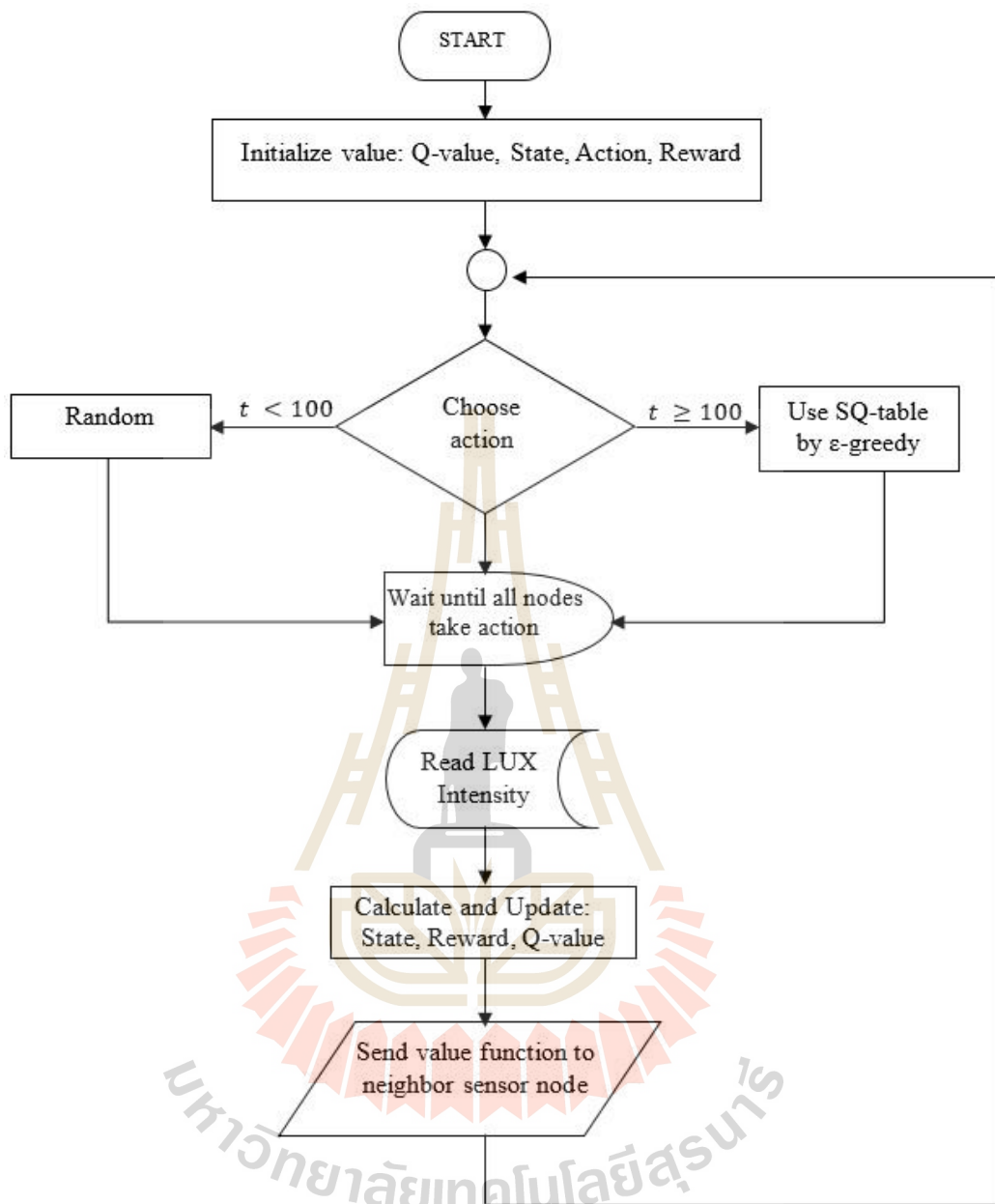


Figure 4.8 Diagram of SQMORL for automatic lighting control testbed.



Figure 4.9 Automatic lighting control testbed.

Figure 4.10 shows the final policy obtained by SQMORL and DVF algorithms. Since there are 5 sensor nodes and each sensor node has 2 actions, there are 32 possible policies. Sensor node 0 is placed in the middle of the room (see Figure 4.9). If sensor node 0 turns on, its light coverage would overlap with that from all the other sensor nodes. As the light intensity of the 4 sensor nodes in the corner provides sufficient coverage area, the sensor node placed at the center can be turned off to save energy. Thus, the optimal policy for this setting is where the 4 sensors in the corner of the room are turned on and sensor node in the middle of the room is turned off (i.e., the 16th policy). Note that both algorithms were programmed to select random actions for exploration during training in the first 100 time steps. Then the algorithms learned

on their own according to the ϵ -greedy action selection scheme. However, it was initially found that a training phase of 100 time steps was insufficient as both algorithms could not find the optimal policy. This is shown by the Direct Learning DVF graph in

Figure 4.10, which the optimal policy (the 16th policy) was not found at all in the 20 repeated runs. Therefore, to enhance the learning rate, we trained the sensor nodes off-line through simulation to obtain the state-action value tables prior to the testbed implementation. Then, we saved the trained value tables in the memory unit of each sensor node. With this off-line training method, the sensor nodes are able to quickly adjust their actions to the testbed environment. Results are shown by the graphs labeled Trained DVF and Trained SQMORL which are the algorithms that learned policies from off-line training, i.e. with initialization from the trained the value tables. Figure 4.10 shows that the Trained DVF and Trained SQMORL algorithms are able to find the optimal policy at 55% and 80% of the 20 repeated runs, respectively.

Figure 4.11 shows the measured light intensity of each sensor node running the DVF algorithm (sensor node 0 is placed at the center and the rest of the sensor nodes are placed in the corners). It should be noted that these measurements are made only after the DVF algorithm has learned the optimal policy. During training in the first 100 time steps, each node explores all actions by randomly selecting its actions. After training, each sensor node makes its decision based on exploitation of its trained value table. Sensor node 0 achieves the least light intensity of 116 lux as its own light bulb is turned off. Its light intensity obtained is from the 4 neighboring sensor nodes.

Figure 4.12 shows the average reward of all sensor nodes running the DVF algorithm. Note that during the training phase with the random selection of actions, the average reward of all sensor nodes does not increase. However, once the training period is over at the 100th time step, each sensor node can choose the optimal action at the 117th time step, which is seen from the increase in average reward consistently.

Figure 4.13 shows the measured light intensity of each sensor node under the SQMORL algorithm. The SQMORL algorithm can learn the optimal policy. This result is similar to that of the DVF algorithm as the same the optimal policy (i.e., the 16th policy) is expected.

Figure 4.14 shows the average reward of all 5 sensor nodes obtained from the 1st objective function which aims to maximize the light intensity. Results show that during training period in the first 100 time steps, the average reward does not increase due to the exploration from randomly selected actions. But after the training period, each sensor node can choose its optimal action consistently from time step 130 onwards, as seen from an increase of the average reward of the sensor nodes. Figure 4.15 shows the average cost of all 5 sensor nodes obtained from the 2nd objective function, which aims to reduce the overlapping areas. Results show that the consistently decreasing average cost at all sensor nodes occurs after time step 130, implies that the light intensity which occur from overlapping area decreases after this time step.

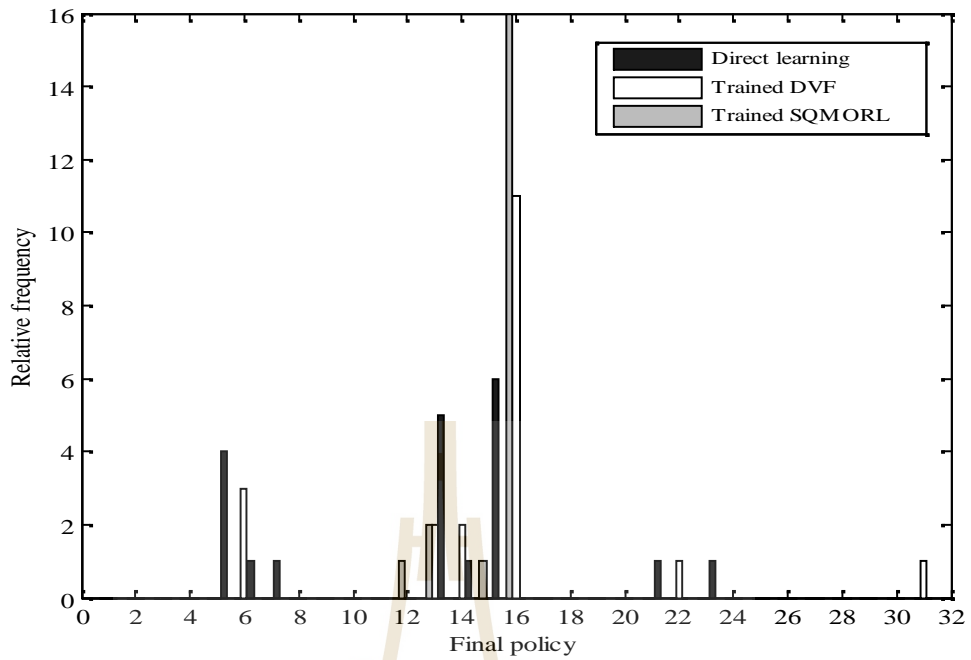


Figure 4.10 Distribution of the final policy obtained by each algorithm.

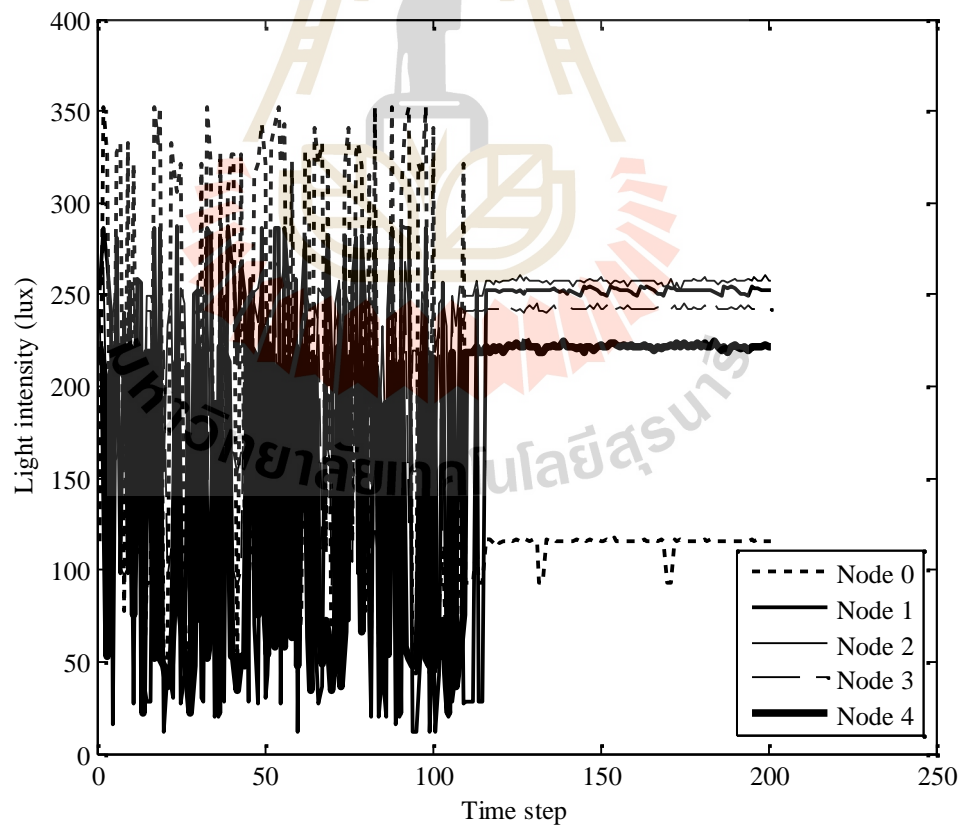


Figure 4.11 Light intensity of each sensor node from the DVF algorithm.

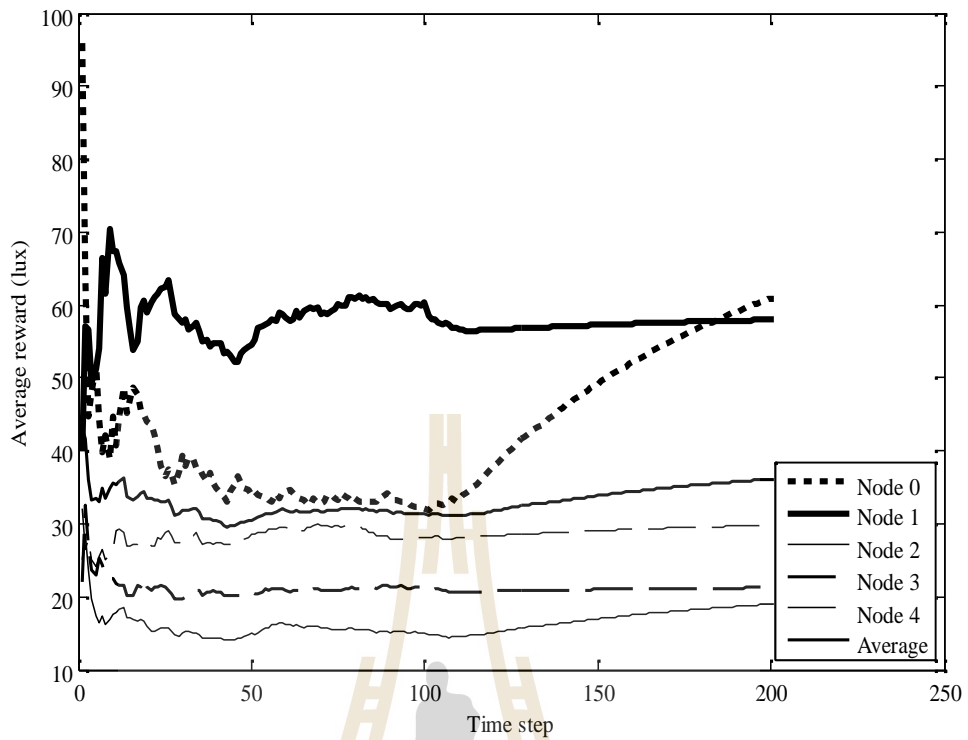


Figure 4.12 Average reward of all 5 sensor nodes from the DVF algorithm.

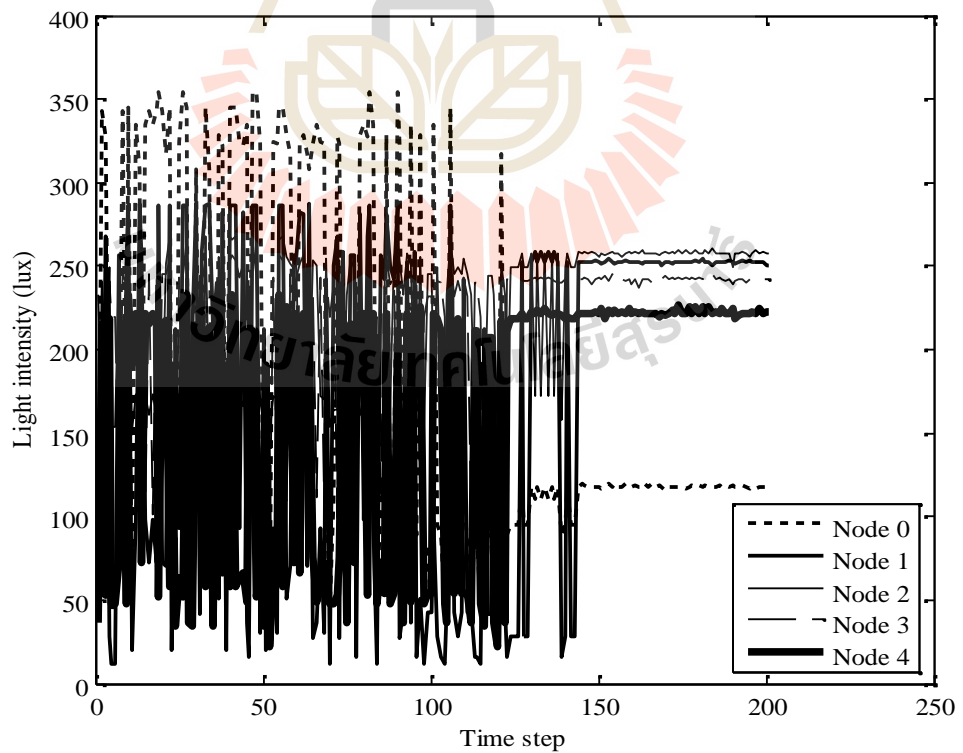


Figure 4.13 Light intensity of each sensor node from the SQMORL algorithm.

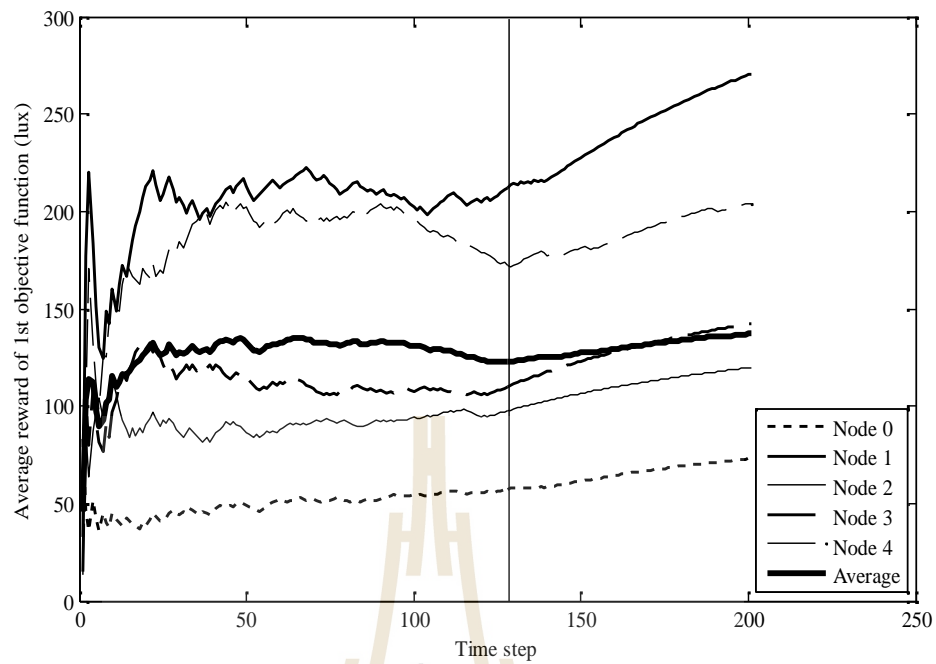


Figure 4.14 Average reward of 1st objective of all 5 sensor nodes from the SQMORL algorithm.

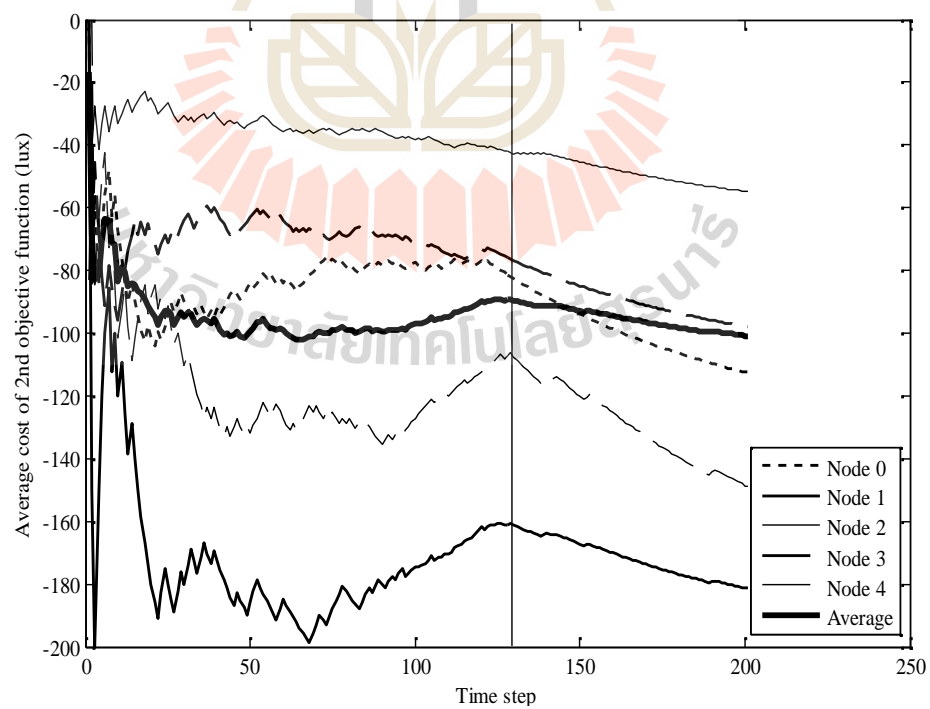


Figure 4.15 Average reward of 2st objective of all 5 sensor nodes from the SQMORL algorithm.

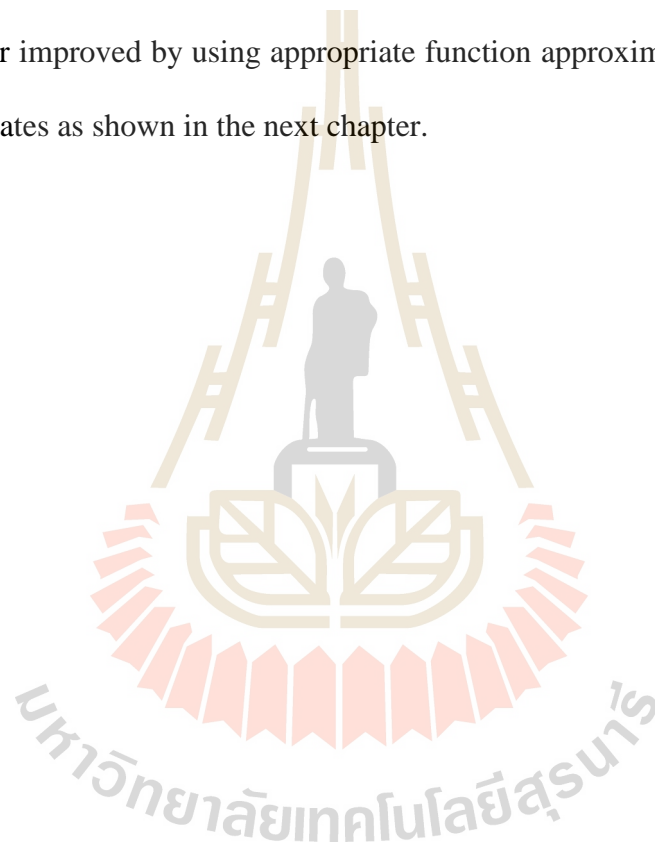
4.4.2 Discussion on test bed results

To test the performance of both the SQMORL and DVF algorithms, an automatic lighting control testbed has been implemented to evaluate the performance of SQMORL and DVF algorithms. From the experiment results, the DVF and SQMORL algorithms can obtain the optimal policy at 55% and 80% from the 20 repeated runs, respectively. In terms of the convergence speed, the DVF and SQMORL algorithm reached the optimal policy at time step 117 and 130, respectively. Such results suggest that the SQMORL algorithm can find the optimal policy more frequently than the DVF algorithm at a comparable convergence rate. Therefore, results suggest that MOO framework based on the SQMORL algorithm may be suitable for coverage control in WSNs, particularly for applications which require maximum (or maintaining coverage) and minimum overlapping coverage.

4.5 Summary

The objective of this chapter is to extend coverage control using the single combined objective reinforcement learning algorithm to a multi-objective optimization (MOO) reinforcement learning framework. In particular, this chapter proposes the Scalarized Q Multi-Objective Reinforcement Learning (SQMORL) which uses a MOO based on a weighted-sum approach, for the coverage control problem in WSNs. The algorithm has advantages of low complexity and scalability. Thus, the MOO framework also allows the optimal policy particularly for contradicting objectives to be found more effectively than the combined single objective function. This is evident from the simulation results in the uniform random node placement and grid node placements. In addition, this chapter has also developed

a hardware testbed to evaluate the performance of SQMORL and the DVF in a multi-agent lighting control experiment which SQMORL algorithm can efficiently find the optimal policy more accurately over the DVF algorithm. However, despite its advantages, the SQMORL is based on a discretized state space. The current SQMORL is appropriate for control problems with inherent discrete states. However, light intensity is of continuous value. It is expected that SQMORL for lighting control can be further improved by using appropriate function approximation to represent the continuous states as shown in the next chapter.



CHAPTER V

CONTINUOUS STATE MULTI-AGENT REINFORCEMENT LEARNING FOR LIGHT CONTROL

5.1 Introduction

Automatic lighting control is among the applications for smart homes which typically use a computer or smartphone to control light bulbs via wireless sensor network embedded in a house. Wang et al. (2014) proposes an adaptive algorithm design for smart buildings, which controls constant illumination when external lighting varies. In particular, this system measures the light intensity at each position and then decides whether to switch light on or off for the targeted area based on non-linear programming. However, for decision-making purposes, it is necessary to gather the light intensity on each lamp, which could be inconvenient if the location or position of the lamp is changed. Sung et al. (2013) uses a Self-Adaptive Weighted Data Fusion algorithm in a Smart LED lighting system which controls LEDs through applications on tablets or smartphones to achieve suitable brightness for each activity. Jabbar et al., (2016) propose low cost devices for smart home applications such as the Arduino. Based on this system, residents can enjoy the convenience of controlling LEDs via the Internet by using Android applications. These application can also monitor the temperature and status of the LEDs of each room through the mobile screen as well. The work also demonstrates the convenience of power management of home electrical appliances via the Internet. However decision systems are systems

that do not learn and optimization because they need to be setting suitable value for light intensity, which may be inconvenient when the environment changes, such as changing the position of the lamp or changing the location.

Khalili et al. (2010) propose a system for optimized light control in smart home by considering energy efficiency and user preference that is called Hierarchical Reinforcement Learning (HRL). The method is based on learning the user preferences online and different status such as time, location and activity. The HRL algorithm learns user preferences when the given feedback to the system through changing the offered light control. Gokul et al. (2016) presents deep reinforcement learning model for home automation systems, where the system learns the patterns and behaviors of the user automatically from experience and take actions accordingly. The system was tested based on its ability to predict actions for lights. Result show that the system can choose to switch on and off the light bulbs were 98% accurate. Based on both researches, the strengths of reinforcement learning system include precise action selection and requirement of only a single feedback value (state value) such on user preference, location and time, etc. Such RL based algorithm enable easy and simple implementation. However, the drawback of learning systems is that the training time requires for the system to find the optimal policy if the state space is large may take a long time.

Typically, Reinforcement Learning (RL) uses the Markov decision process (MDPs) framework which defines action and state sets as discrete finite sets (Hasselt et al., 2007). which (Khalili et al., 2010) and (Gokul et al., 2016) used discrete finite sets for light intensity setting. Therefore, the system requires a Q-table to decide choose actions are appropriate for each state, which causes of learning systems

problems as mentioned above. In addition in the real world, certain parameters are continuous, such as the light intensity. By quantizing the continuous state into discrete states, certain features of the state may be lost. As a result, the RL may not be able to find the optimal policy. Hasselt et al. (2007) propose a function approximation to handle the RL problem regarding the continuous state space. However, to select a policy that is optimal and suitable for use in real world, the optimal policy should be based on actual light intensity that is appropriate for the user. Furthermore, the continuous reinforcement learning may reduce system memory requirement. This is because it does not require a complete table of action-value function for every state-action pair. Instead, it uses function approximation to represent the state of light intensity for finding the optimal policy.

Therefore, the contribution of this chapter is four-fold: 1) Applying function approximation to the automatic lighting framework where light intensity is considered a continuous state. 2) Study of parameters that effect to continuous state reinforcement learning. 3) Comparison of the Continuous state RL algorithm with Discrete state RL and Threshold algorithm. 4) Development of a testbed and hardware performance evaluation of an automatic lighting control using SQMORL algorithm.

5.2 Continuous state reinforcement learning for automatic lighting control

In reinforcement learning agents have explicit goals. Agents can sense aspects of their environments, and can choose actions to influence their environments (Sutton et al., 1998). The goal of an agent is to achieve the highest long term average reward or lowest long term average cost. However, if reinforcement learning method is

applied to problems with large states a significant amount of memory is required to store the value functions. Moreover, if we want to apply the RL method for automatic lighting control, the continuous state of lighting intensity should be considered in order to allow a finer state transition. In addition, if we consider the external light that may change the environment of the system, we can see that there is a continuity in state space. Therefore, continuous state reinforcement learning a promising tool for the lighting coverage control.

In this dissertation, the definition of continuous state reinforcement learning for lighting control which use in Figure 5.1 is following:

Definition 1. Q-value: Q-value function in continuous state reinforcement learning defined by:

$$Q_{t+1}^i(s^i, a^i) = W_t^i(s^i, a^i) \cdot r_t \quad (5.1)$$

where $W^i(s^i, a^i)$ call feature define as (4), $W^i(s^i, a^i)$ denote the value of feature for state-action pair (s^i, a^i) . Note that $Q^i(s, a)$ which has $|r| \times |a_0| + |r| \times |a_1| = (1) \times (1) + (1) \times (1) = 2$ element which used in comparison to select the maximum Q value for the appropriate action. When a_0, a_1 is action off and on the Light bulb and r are member of real number which is the weight vector specifying the contribution of each feature across all state-action pairs.

Then, agent needs to calculate u , the temporal difference (TD) error at each is the error in the estimate made at that time. Because the TD error at time step t depends on the next state and next reward, it is not actually available until time step

$t+1$. So updating the value function with the TD error is called a backup, which is the difference of Q-value in the previous time step and the current time step

$$u_{t+1}^i = Q_{t+1}^i(s, a^i) - Q_t^i(s, a^i). \quad (5.2)$$

There is always at least one policy that is better than or equal to all other policy. This is an optimal policy which can achieve maximize average reward in long term. We denote optimal policies by f^* . Thus, the optimal policy f^* can be obtained by

$$Q^f(s, a^i) = \arg \max Q(s, a^i). \quad (5.3)$$

Definition 2. Action: Each sensor node can take one of the following two actions in any state. The action space A^i is the set of all possible actions for each state $\{a_0^i = \text{Action 0 (Turn off light)}, a_1^i = \text{Action 1 (Turn on light)}\}$.

Definition 3. State: In order to represent a continuous state and to represent the light intensity, let $w(s, a)$ call the feature be the light intensity that occurs at any point. The feature function $w^i(s, a)$ has $|S| \times |A| = (1) \times (2) = 2$ entries at each agent. The intensity of light in spherical coordinates is defined by (Rea, 2000):

$$w(s^i, a_i) = \sum_{i=1}^{i=n} \frac{a_i \cos S_i}{D_i^2} \cdot \frac{F}{4f} \quad (5.4)$$

where D is the distance between the light bulb to the sensor node,

s is the angle between the normal lines of sensor nodes with distance D ,

F is the luminous flux of 1030 lumen (Panasonic fluorescent 36 watt),

n is the number of neighboring sensor node.

Definition 4. Reward: Sensor node can detect the lux intensity which may from its own light bulb or its neighbors. If a sensor node attains lux intensity by itself and not from neighboring sensor node, the lux intensity is considered as that sensor's reward. In other words, if a particular sensor node i and neighbors turn the light bulb on at the same time, we consider this as a lux overlap calculated by equation (4) and that sensor i 's reward is deducted. Therefore, the reward function for sensor node i is given by

$$r^i(s^i, a^i) = \text{Lux received} - \text{Lux overlapping} \quad (5.5)$$

CONTINUOUS STATE REINFORCEMENT LEARNING FOR LIGHTING CONTROL

BEGIN

1 Topology setting

2 $u_0^i \leftarrow$ Initialize arbitrariness

3 for time step 1: end time step*

4 Each agent choose action $A^i = [a_0^i, a_1^i]$, and next

state $s^i = w_{t+1}^i(s^i, a^i)$ for $i=1-5$ (five node sensor in testbed)

5 Receive reward $r^i = \text{Lux received} - \text{Lux overlapping}$

6 Update Q^i, u^i and u^i

7 $Q_{t+1}^i(s, a^i) \leftarrow W_t^i(s^i, a^i) \cdot u_t^i$

8 when

$$W_t(s, a^i) = \sum_{i=1}^{i=n} \frac{a_i \cos S_i \cdot F}{D_i^2 \cdot 4f}, \quad n = \text{number of sensor node}$$

9 $u_{t+1}^i = Q_{t+1}^i(s, a^i) - Q_t^i(s, a^i)$

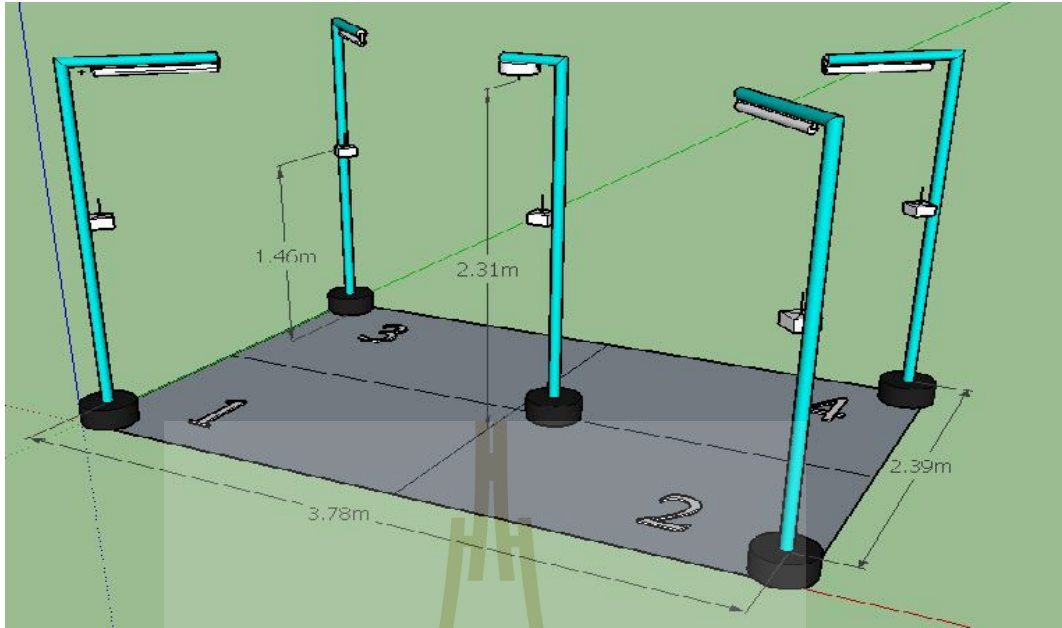
10 $u_{t+1}^i = u_t^i + \gamma u_{t+1}^i W_{t+1}^i(s^i, a^i)$

11 endfor

END

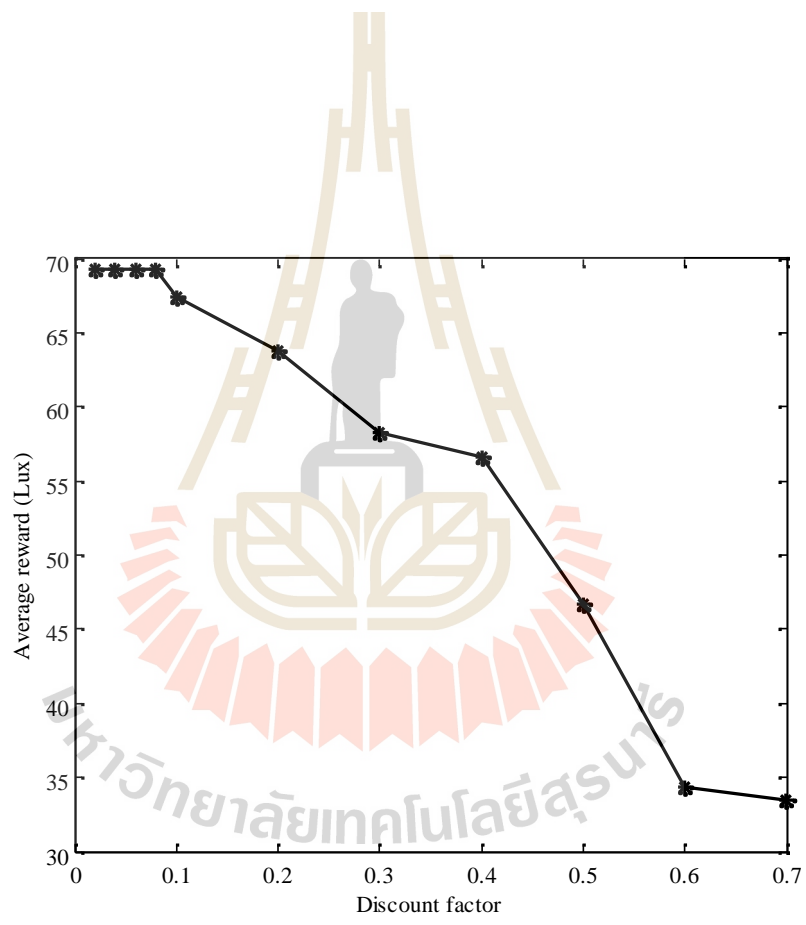
* Algorithm can find optimal policy

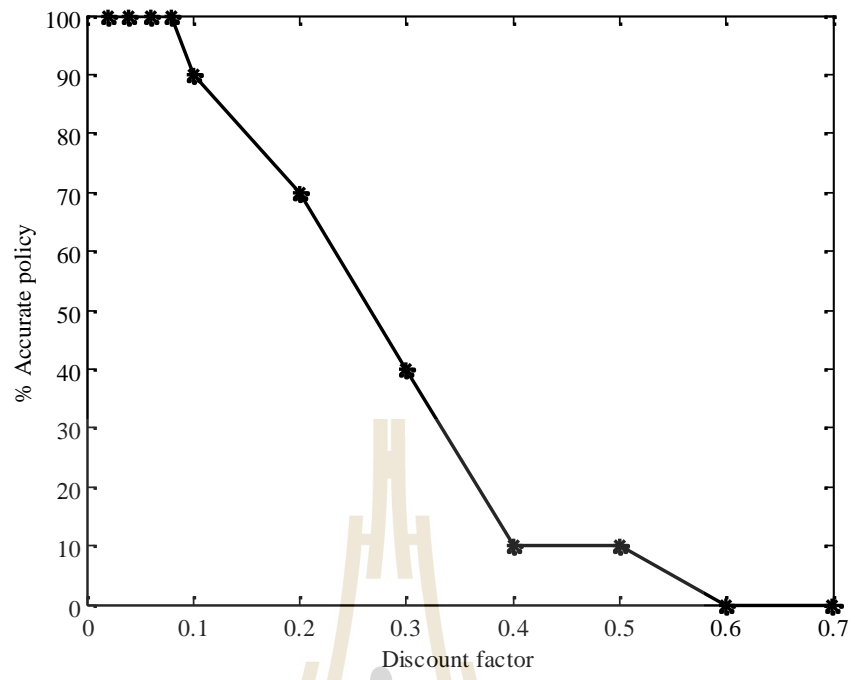
Figure 5.1 Pseudo code of continuous state reinforcement learning for lighting control.



(x)

(r)





มหาวิทยาลัยเทคโนโลยีสุรนารี

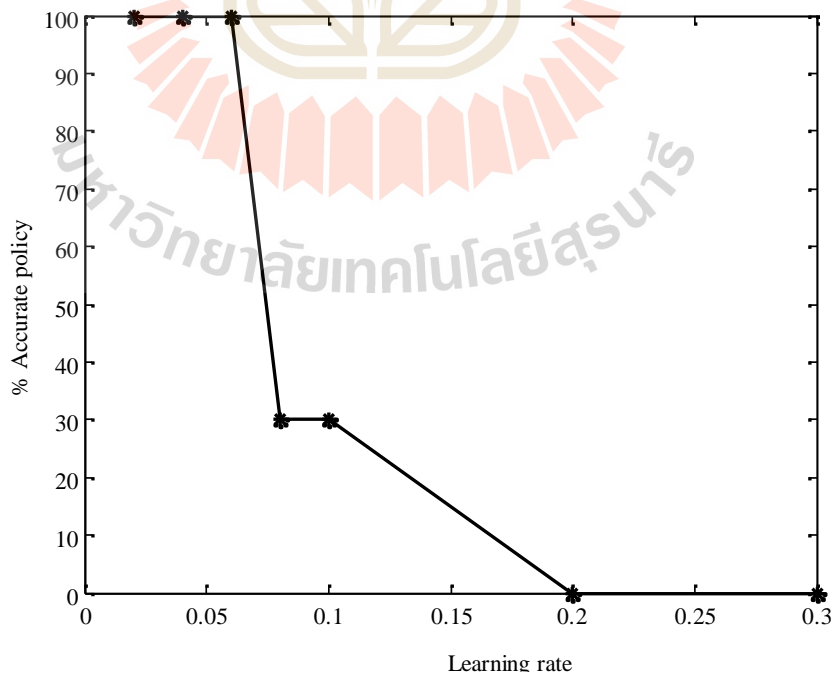
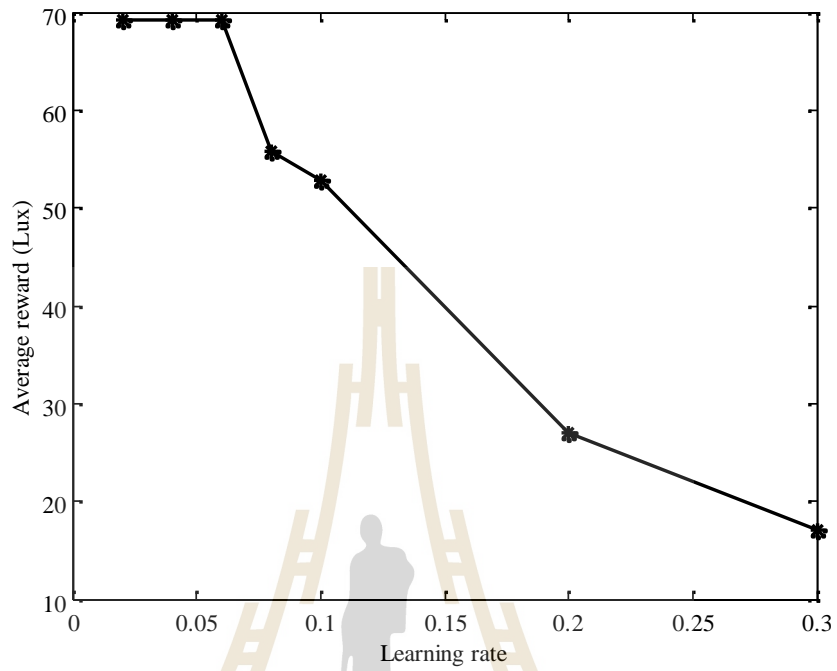
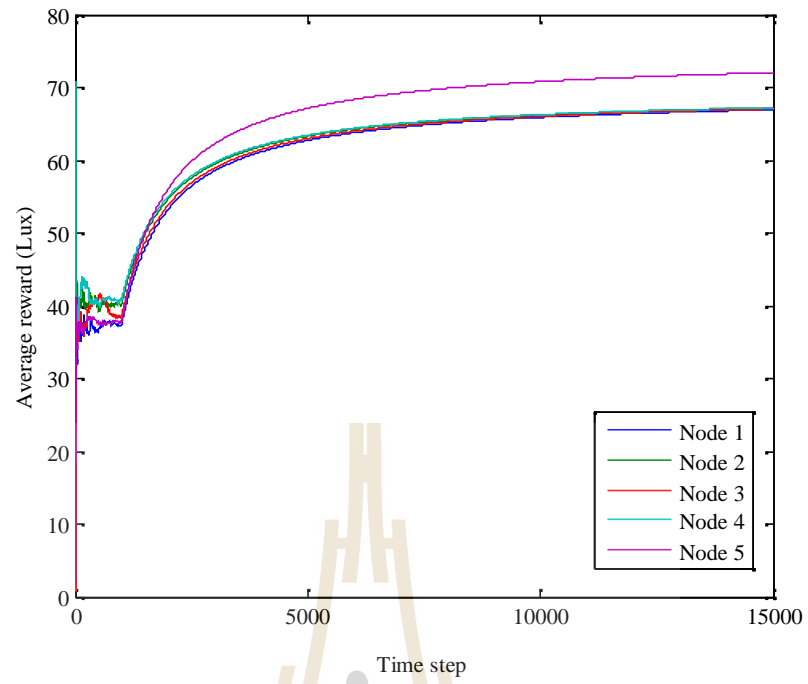


Figure 5.5 shows the effect of learning rate on the average reward. The higher the learning rate, the lower the average reward. This is at high learning rate, the step size determines to what extent the newly acquired information will override the old information. In the case where the learning rate between 0.02-0.06, the average reward is the highest at 69.27 lux. After this, the average reward gradually decreases. In Figure 5.6, it is evident that when the learning rate in the range of 0.02 - 0.06, the algorithm can accurately find the optimal policy. Note that when the learning rate is 0.2 - 0.3, the algorithm cannot find the optimal policy, thus the learning rate of 0.02 is the chosen value which the algorithm performed best.

Given the selected values of discount factors and learning rate, Figure 5.7 shows the average reward received from the selected action at each time step of each node sensor. The first 1000 time steps is the learning duration. After that, each sensor node learns and selects its own action in a distributive manner. Based on Figure 5.7, sensors node 1-4 (placed at four corners) choose the same action which is to turn on their bulbs, so they achieve comparable average reward, while the sensor node 5 (placed at the center) chooses to shut itself down to achieve reward from its neighbouring sensor nodes. Therefore, the average reward is slightly higher than the other sensor nodes. The ascending average reward over time shows that every node sensor can learn the optimal policy over time.



In discrete state RL algorithm, since there are 5 sensor nodes. Each sensor node has 2 actions, so there are 32 possible policies. Thus, we calculate the lux intensity using equation (5.4) to represent state in the algorithm. The lux intensity for each sensor node is quantized equally into 4 states shown in Table 5.1. We also consider another method of discretizing state space to evaluate how the method of state discretization affects the performance. In particular, the lux intensity calculated by equation (5.4) is rearranged in increasing order, then grouped into quartiles to create 4 states. The resulting discretization of state space is shown in Table 5.2.

Table 5.1 State quantization: type 1.

Sensor nodes	State discretization: type 1			
	0	1	2	3
Node 1, 2, 3, 4	0 - 23	24 - 47	48 - 71	72 - 95
Node 5	0 - 38	39 - 77	78 - 116	117 - 155

Table 5.2 State quantization: type 2.

Sensor nodes	State discretization: type 2			
	0	1	2	3
Node 1, 2, 3, 4	0 - 7.05	7.06-14.76	14.77-84.88	84.89-92.59
Node 5	0 - 37.14	37.15-55.71	55.72-114.97	114.98-152.12

Table 5.3. shows the final optimal policy of each algorithm. The first 1000 time steps is the training period for the two discrete state RL algorithm and the continuous state RL algorithm. After this period, all the algorithms learned independently until the end of the run. After that, we consider the final policy each algorithm. Results show that continuous state RL algorithm achieves 100% of policy

accuracy. For both discrete state RL algorithms of type 1 and 2 achieve only 50, 60 % of policy accuracy, respectively. This result suggested that the training duration of 1000 time steps may not be enough to learn for discrete state RL algorithms possibly due to the number of state. In addition, how the state space is discretized also affected the learning and the ability to find optimal policy. This is evident from type 2 state discretization which every state actually occurs, thereby achieving higher policy accuracy than type 1. As for the Threshold algorithm, we define a threshold for decisions to switch the bulb at 77.77 lux. If sensor node detects lux intensity less than this threshold, the bulb is turned on. Otherwise, the sensor node turns off the bulb. From Table 5.3 the Threshold algorithm chooses to turn on all light bulbs in all the runs. The reason for this may be caused by unsuitable value of threshold. However, it is difficult to know the appropriate threshold values for each situation if the system does not learn.

Apart from policy accuracy, the convergence rate to the optimal policy of each algorithm is also investigated. The algorithms are considered to converge once the standard deviation of the average reward is less than 2%. Results show that the continuous state RL algorithm can converge to the optimal policy at time step 95 whereas the discrete state RL algorithms of type 1, 2 can converge to the optimal policy at time step 1796 and 1815, respectively. This is because the discrete state RL algorithms require that all state-action pairs in the Q-value table need to be visited infinitely. However, the continuous state RL algorithm uses parametric feature approximation to represent the continuous state in eq. (5.4) and is updated at every time step, regardless of the state-action pair. Hence, the parametric function approximation is updated every time step thereby increasing the convergence speed.

Furthermore, such representation allows for a reduced dimension of learned parameters thereby contributing to the convergence speed.

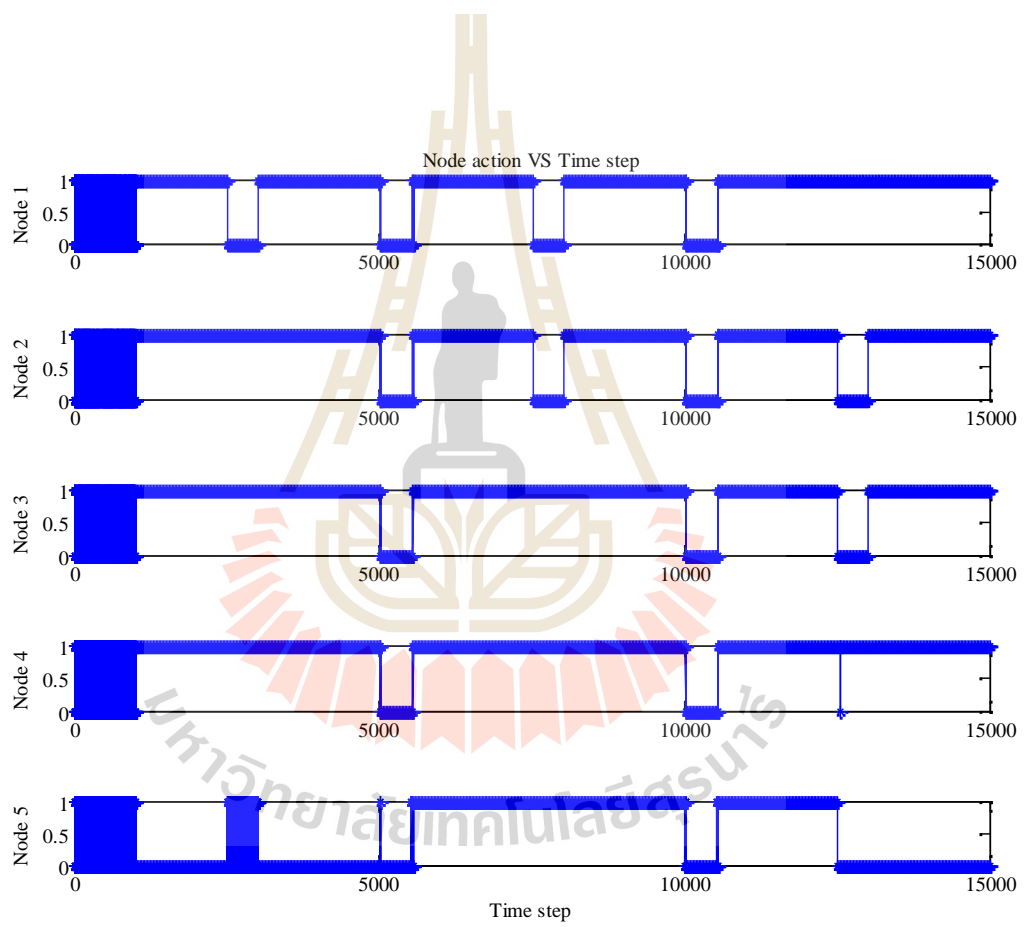
To demonstrate the adaptability to perturbation, we introduced certain perturbations to simulate scenarios that the environment can change at any time. For example, some external light is introduced to the system, or a sudden shutdown of nodes. Results are compared with the performance of continuous state RL algorithm with Threshold algorithm and discrete state RL algorithm of type 2. Note that type 1 has not been selected due to its poor performance.

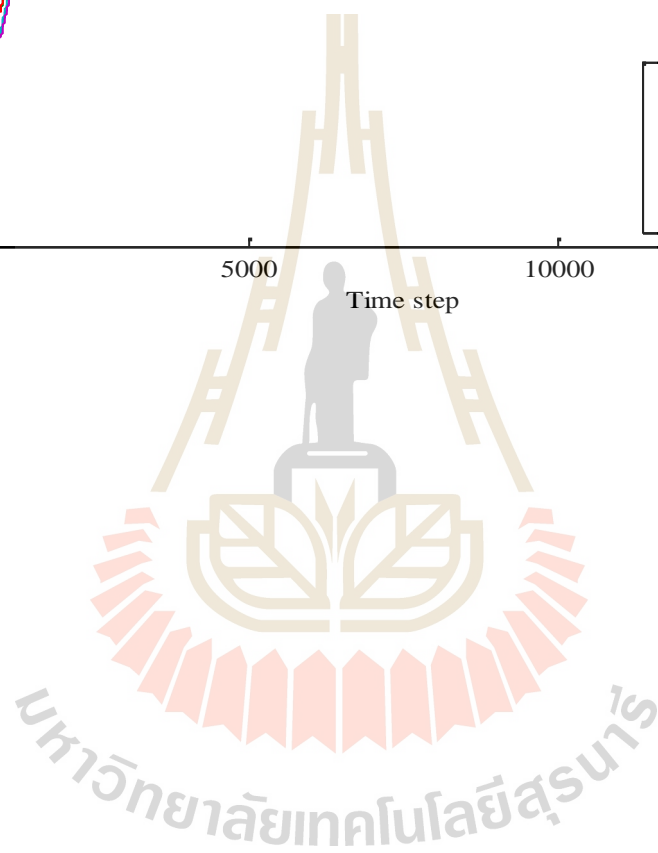
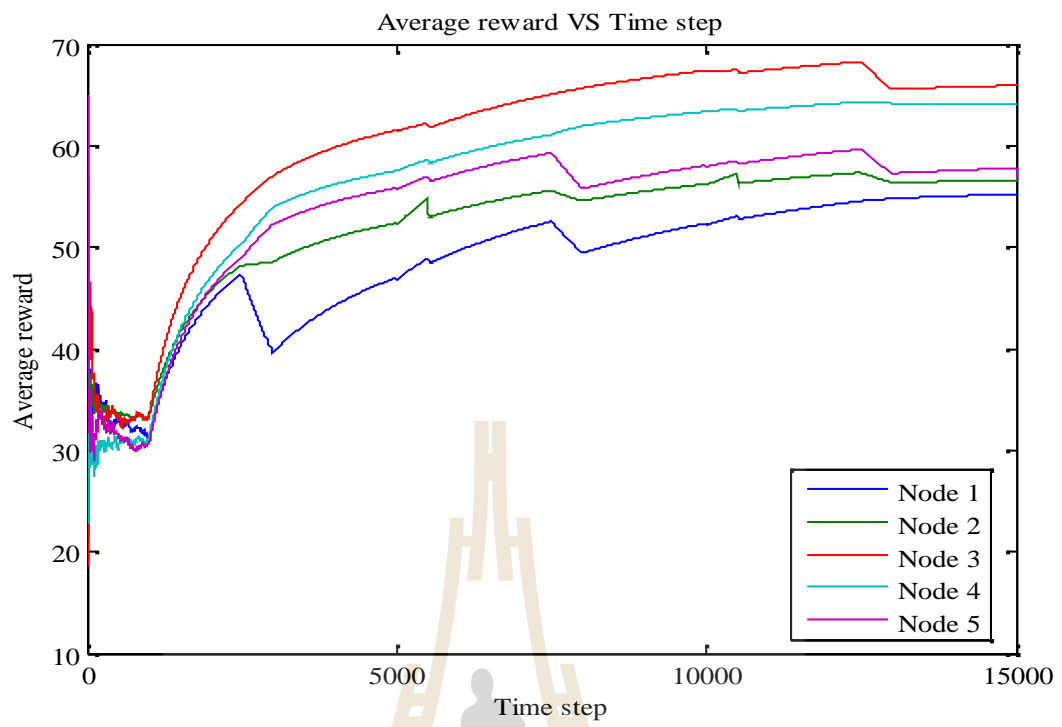
Table 5.3 Final policy of each algorithm.

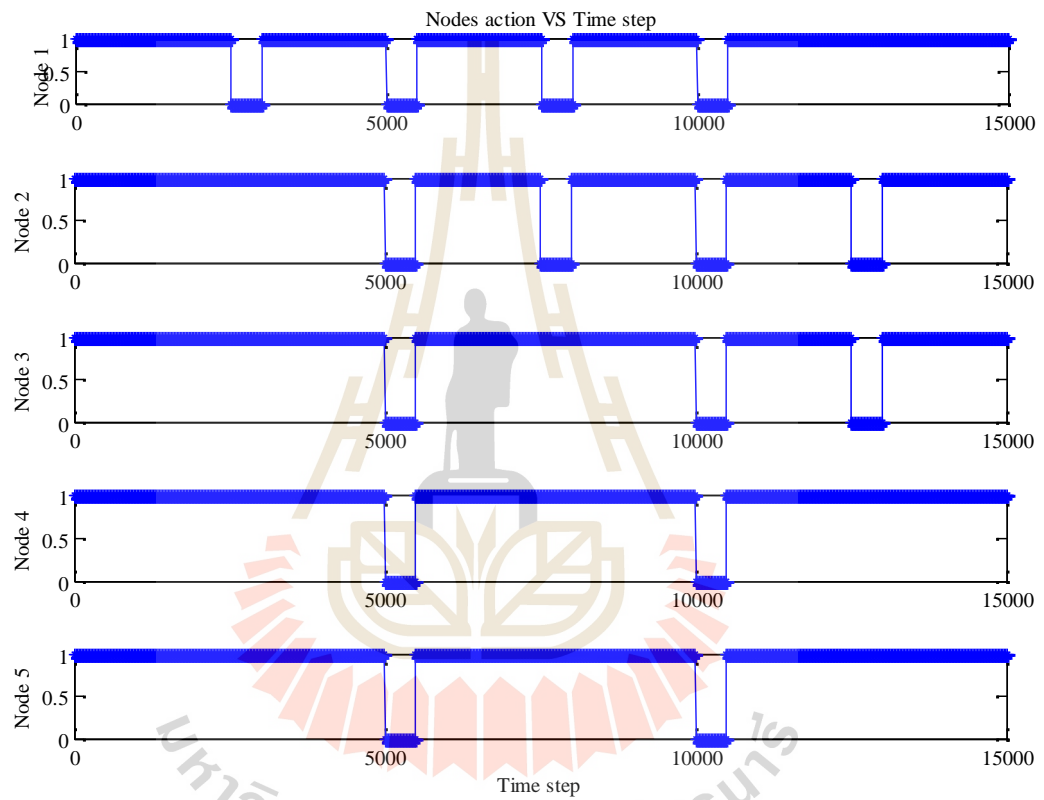
Final policy				
Number of runs	Discrete state RL algorithm (type 1)	Discrete state RL algorithm (type 2)	Threshold algorithm	Continuous state RL algorithm
1	01111	11101	11111	11110
2	11110	11110	11111	11110
3	11110	10111	11111	11110
4	01101	11110	11111	11110
5	11110	01111	11111	11110
6	11011	11110	11111	11110
7	10101	11110	11111	11110
8	10111	11110	11111	11110
9	11110	10011	11111	11110
10	11110	11110	11111	11110
% accuracy	50%	60%	0%	100%
Convergence (time steps)	1796	1815	-	95

Note: The optimal policy is 11110.

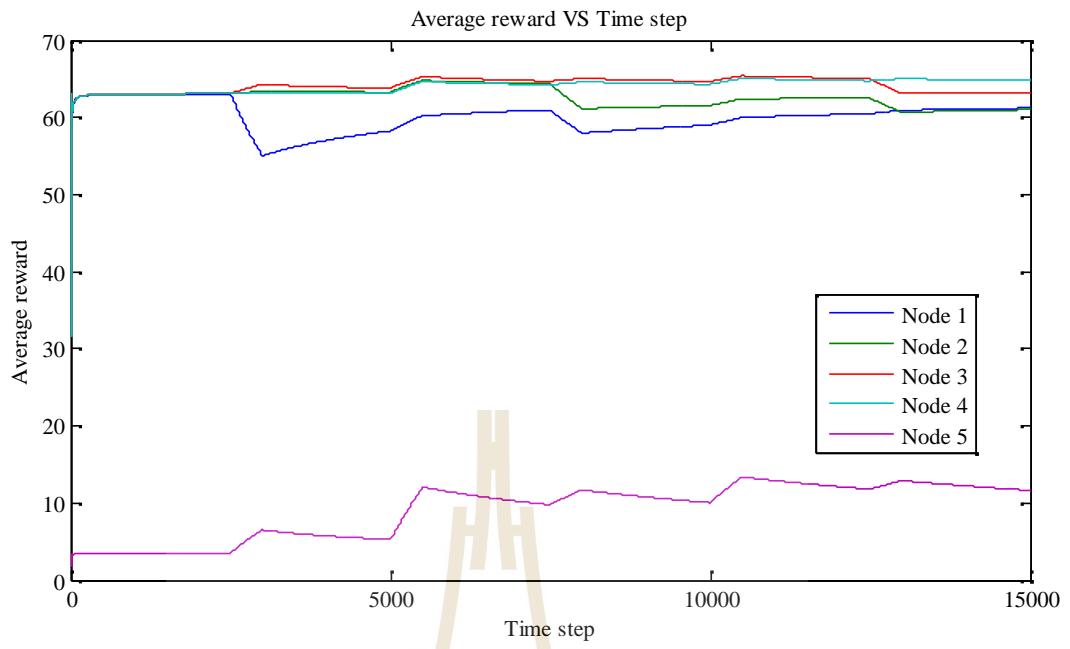
In the experiment, certain light bulbs are switched off and external lighting is introduced during certain periods as shown in Figure 5.8 and 5.9 for discrete state RL algorithm of type 2. From Figure 5.8, it can be seen that during the first 1000 time steps, the algorithms select actions randomly so that all state-action pairs can be visited. After this phase, the algorithms can find the optimal policy. Recall that the optimal policy is the one which the sensor node located at the center of the room (sensor node 5) chooses to shut down its own light bulb to reduce the overlapping of light intensity and reduce energy consumption, while the remaining four sensor nodes (sensor node 1, 2, 3 and 4) located at the corners of the room turn on their light bulbs. The average reward shown in Figure 5.9 steadily increases after the first 1000 time steps because each sensor node can choose suitable action and achieved maximize average reward. To test the response to perturbations, during time step 2500 to 3000, the light bulb at sensor node 1 is deliberately turned off resulting in a reduced average reward for sensor node 1. Sensor nodes 2, 3, 4 keep their light bulbs switched on. These the average reward of sensor node 2, 3, 4 is not affected the turning off of sensor node 1. However, it affects sensor node 5 located at the center of the room, because without light intensity from sensor node 1, sensor node 5 must be turned on. However, when sensor node 5 turns on, as the intensity of light overlaps and Q-values of sensor node 5 alternates causing switching between on and off. In time step 5000 – 5500, additional perturbation is introduced by adding external light into the system. As a result, every node sensor switches off its own light bulb and receives the light from the outside. As it is the correct decision, the average reward of every node sensor increases. However, after the external light is removed after time step 5500, sensor node 5 turns on its light bulb which correct decision. In the time step 7500 -



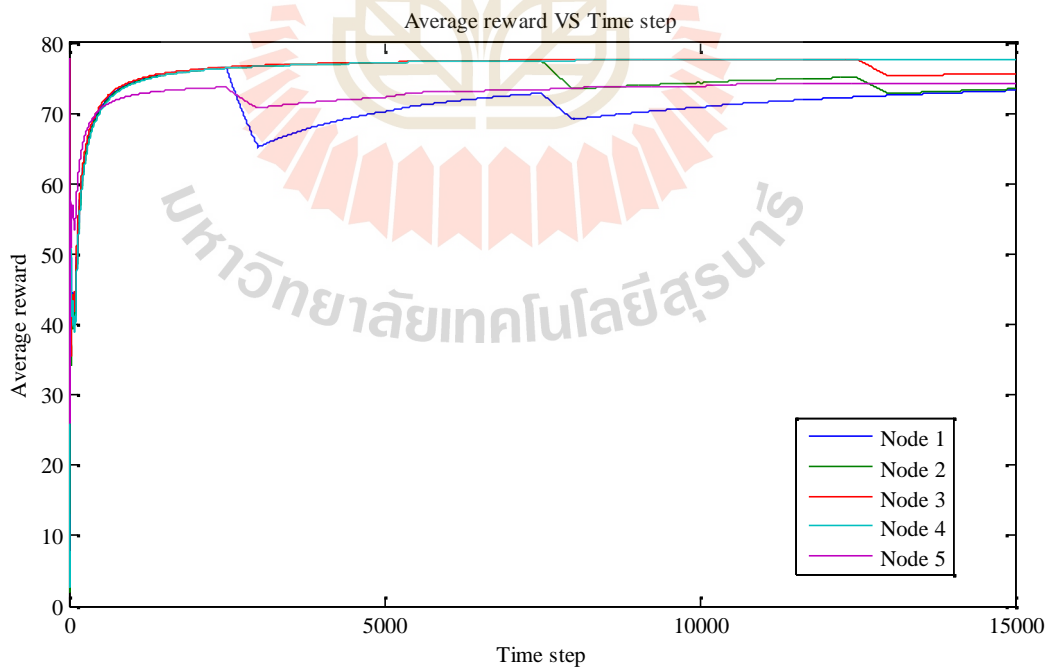
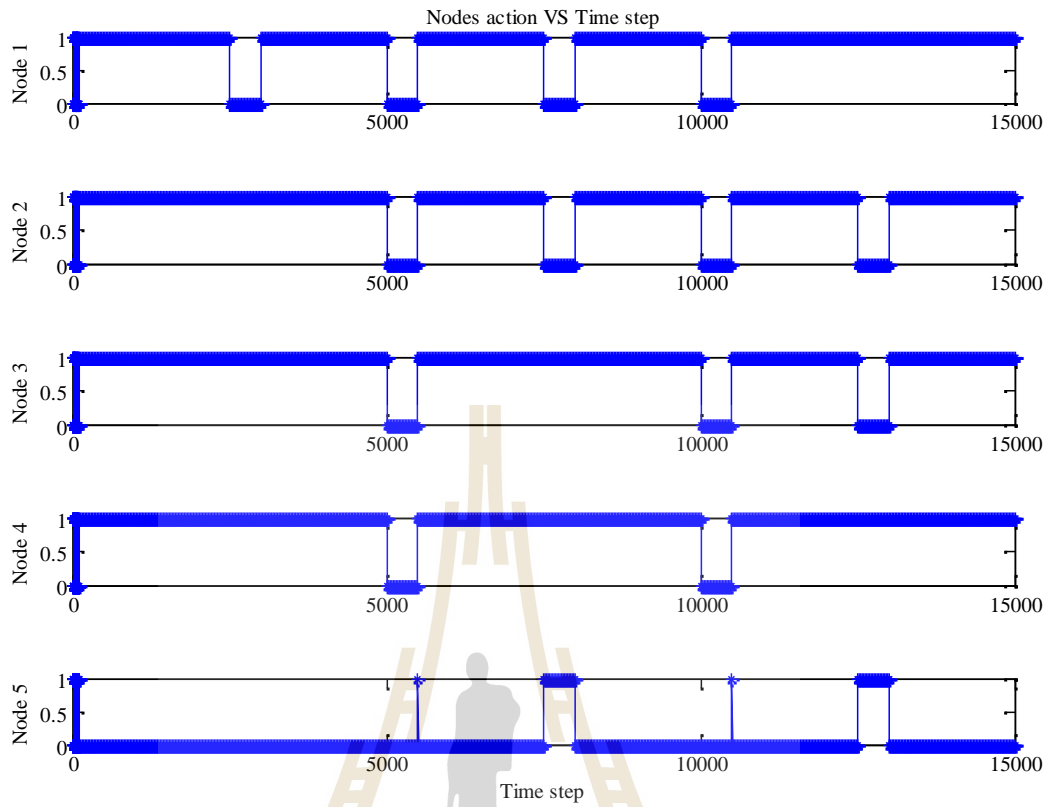




มหาวิทยาลัยเทคโนโลยีสุรนารี



5500, the external light is removed, and every sensor goes back to the optimal action which is to switch on sensor nodes at all corners and switched off the center sensor node. Then, during the time step 7500 to 8000, we tested the system again by deliberately switching off sensor node 1 and 2. Results show that sensor node 3 and 4 still turn on their light bulbs but sensor node 5 changes from switching off to on. This indicates that the system has decided that with the two light bulbs turned on has insufficient light intensity. Thus, sensor node 5 is turned on to maintain the brightness of the room. At time step 10000 to 10500, external light is introduced to the room. Once again, all five sensor nodes are switched off again to save energy because there is sufficient external light. Finally, at time step 12500, we deliberately switch off light bulbs at sensor node 2 and 3. The algorithm chooses to turn on sensor node 5 (similar to time step 7500) to maintain the brightness level. The Continuous state RL algorithm can thus correctly respond to all the perturbation in the experiment.



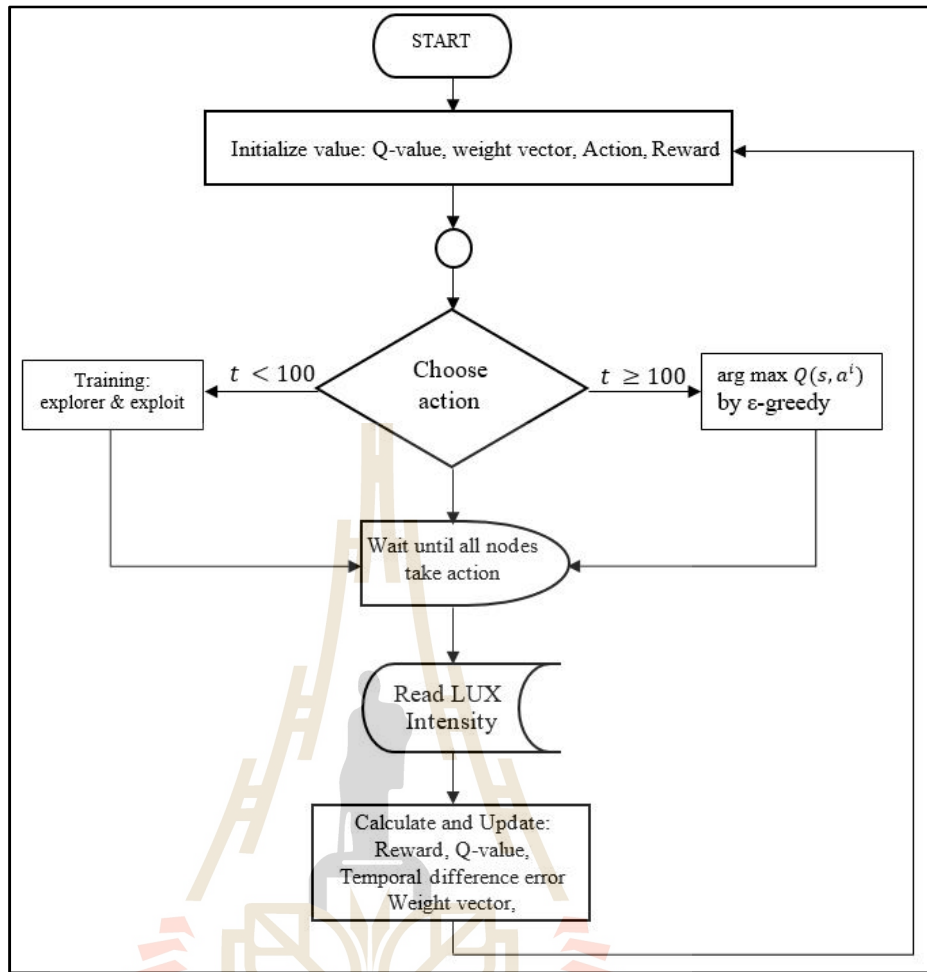
The memory requirement for storing entries for each algorithm is also considered. The Continuous state RL algorithm does not need to store Q-values for all state-action pairs. It only requires memory storage for storing the parametric values of $Q^i(s, a)$ which has $|_n| \times |a_0| + |_n| \times |a_1| = (1) \times (1) + (1) \times (1) = 2$ entries, and for storing the feature function of the current state $w^i(s, a)$ which has $|_n| \times |A| = (1) \times (2) = 2$ entries at each agent. Suppose that each entry requires 8 Bytes, a reasonable amount of memory of 48 Bytes ((4 x 8) + (2 x 8)) Bytes is required. As only the parametric value, θ , is learned, the training time is reduced and convergence speed increases. On the other hand, the Discrete state RL algorithm requires memory storage for storing Q-values for state-action pairs, which has $|S| \times |A| = (4)(2) = 8$ entries at each node. Suppose that each entry requires 8 Bytes, a reasonable amount of memory of 64 Bytes (8 x 8 Bytes) is required. However, the continuous state is coarsely discretized into 4 state and performed poorly in terms of policy accuracy percentage. Thus, a finer discretization maybe needed. For example, a finer discretized state of 81 states, requires memory storage for storing all Q-values of $|S| \times |A| = (81)(2) = 162$ entries, resulting in 1296 Bytes (162 x 8 bytes) for memory storage at each agent.

5.4 Light control test bed

In this section, Continuous state RL and Discrete state RL algorithm type 2 are evaluated in an automatic lighting control testbed. The Discrete state RL algorithm of type 1 is not studied due to its poor performance in the simulation part in section 3. In the testbed, each sensor is initialized to the initial default value setting (i.e., the weight vector specifying the contribution of each feature across all state-action pairs each

agent equal zero). There are two actions, i.e., “action 0” refers to turning off a light bulb, and “action 1” refers to turning on a light bulb. As the Discrete state RL algorithm convergence condition requires that every state-action pair be visited infinitely often, an “explore and exploit” scheme is implemented at each agent. In particular, each agent is set to randomly select (explore) actions for 100 time steps in the training phase. This enables each agent to explore all possible state-action pairs and update the action value (Q) functions. On the other hand, the continuous state RL does not require state quantization. Instead, the Q function is a function of a parametric value which is updated at every time step, regardless the state-action pair visited. After the training phase, the parametric Q value are then used to select (exploit) optimal action according to (5.3) with some probability and explore other actions randomly with the remaining probability. This is referred to as the ϵ -greedy action selection scheme. When an action is selected, each node waits for 5 seconds for the light intensity to become stable as a certain delay is required in the hardware to turn on or off each light bulb. The nodes then measure the resulting light intensity and obtain the reward values from equations (5.5). Each node then updates their Q value functions according to equation (5.1). The process is repeated as shown the flow chart in Figure 14 until convergence is achieved, i.e., agent can find the optimal policy.

In order to evaluate the performance of the Continuous state RL and Discrete state RL algorithms, an automatic lighting control testbed as shown in Figure 15 was developed in the Wireless Communication Laboratory, Suranaree University of Technology (SUT), Thailand.



positions as shown in Figure 5.2 in the experiment room of which external light is blocked. The results were averaged over 20 repeated runs.



Figure 5.15. Automatic lighting control test bed.

Table 5.4. Final policy of each algorithm from the test bed.

Final policy		
Number of runs	Discrete state RL algorithm (Type 2)	Continuous state RL algorithm
1	11110	11110
2	11100	11110
3	11110	11110
4	01111	11110
5	01110	11110
6	11110	11110
7	11110	11110
8	11110	11110
9	11100	11110
10	11110	11110
11	11110	11110
12	11110	11110
13	01110	11110
14	11110	11110
15	01110	11110
16	11110	11110
17	11110	11110
18	11110	11110
19	10110	11110
20	11110	11110
% accuracy	65%	100%
Convergence rate time (time steps)	497	395

Table 5.4 shows the final optimal policy of each algorithm test bed result. The first 100 time steps is the training period for the Discrete state RL algorithm and the Continuous state RL algorithm. After this period, all the algorithm learned independently until the end of the run. In Discrete state RL algorithm, we trained the sensor nodes off-line through simulation to obtain the Q value tables prior to the test bed implementation. Then, we saved the trained Q value tables in the memory unit of

each sensor node. However, the Continuous state RL algorithm does not require training Q values because this is parameterized by a feature function based on a parametric weight. At the end of each run, we consider the final policy each algorithm. Results show that Continuous state RL algorithm achieves 100% of policy accuracy. This algorithm does not quantization status, but uses the parametric feature function to determine the Q value and thus the appropriate action and subsequent state to help the algorithm learn better. Furthermore the parametric value is updated at every time step regardless the state or action visited. Therefore, Continuous state RL algorithm can learn the optimal policy faster than the discrete state RL algorithm. Note that Discrete state RL algorithm which is a state discretization of (Type 2) can achieve only 65 % of policy accuracy.

Apart from policy accuracy, the convergence rate to the optimal policy of each algorithm is also investigated. The algorithms are considered to converge once the standard deviation of the average reward (see Figure 5.15, 5.16) varies less than 2%. Results show that the Continuous state RL algorithm can converge to the optimal policy at time step 395 whereas the Discrete state RL algorithm (Type 2) can converge to the optimal policy at time step 497. This is because the Discrete state RL algorithms requires that all state-action pairs in the Q-value table to be visited infinitely. However, the Continuous state RL algorithm use feature approximation to represent the continuous state. Such approximation is governed by a parametric value and is updated at every time step, regardless the state-action pair. Hence, the parametric representation allows for a reduced dimension of learning parameters which is updated at every time step thereby increasing the convergence speed.

Figure 5.16 and 5.17 show the average reward of all 5 sensor nodes obtained from the selected action of the Discrete state and Continuous state RL algorithms, respectively. The first 100 time steps is the learning duration, which the average reward of all sensor nodes does not increase. After that, each sensor node learns and selects its own action in a distributed manner. The result shows that when both algorithms have can achieve the optimal policy (11110) which is seen from the increase in average reward consistently. It can be seen that sensor nodes 1-4 (in the corner) receive comparable average reward from choosing action 1 (turn on the light bulb) while sensor node 5 will converge to a light intensity of 178 lux since it switched off its light bulb. However, the Continuous state RL algorithm is able to learn faster Discrete state RL algorithm due to its reduced dimension of learned parameters.

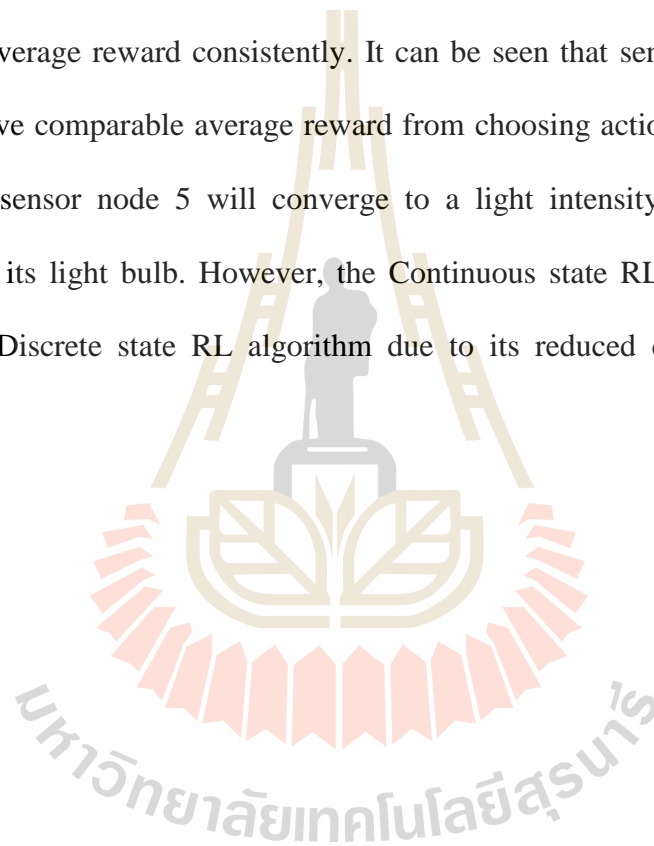


Figure 5.16. Average reward of all 5 sensor nodes from the Discrete state RL algorithm



Figure 5.17. Average reward of all 5 sensor nodes from the Continuous state RL algorithm.

5.5 Summary

This chapter proposed a continuous state multi-agent reinforcement for maintaining light coverage and, save energy consumption. The algorithm for automatic lighting control application. The proposed approach is an adaptive and distributed multi-agent scheme. We proposed to use function approximation method to deal with continuous light intensity as well as, reduce memory storage and training time from the Discrete state RL algorithm. In particular, to represent continuous lighting intensity state, we applied the intensity of diffused light in spherical coordinates equation for create characteristic function to represent lighting intensity state. In this chapter first we study the effect of parameters related to learning in the system on continuous state reinforcement learning. Based on the simulation result, the

suitable learning rate for the system is 0.02-0.04, the discount factor between 0.02-0.08 which achieved the maximum average reward. With respect to the ability to find the optimal policy, the learning rate value and discount factor in this range can consistently achieve the optimal policy.

In the second part to demonstrate the increased efficiency of the function approximation approach for Continuous state reinforcement learning. We compare the Continuous state RL algorithm with Discrete state RL algorithm and Threshold algorithm. In terms of final policy accuracy, the Continuous state RL can find optimal policy 100%, while the Discrete state RL algorithm of state quantization type 1, 2 achieve only 50%, 60% respectively. This is due to the fact that state-action pairs have not been visited. Thus, the Q-values have not been updated frequently enough to learn in the Discrete state RL algorithm. For the Threshold algorithm, we define threshold for deciding to switch the bulb at 77.77 lux. Although the Threshold algorithm is simple, but choosing the optimal threshold value is difficult when light intensity, sensor node position and energy consumption must be considered.

In terms of convergence rate, Continuous state RL algorithm can converge much faster to the optimal policy than Discrete state RL algorithm type 1, 2. Since, this is because Discrete state RL algorithms require that all state-action pairs be visited during training in order to learn the Q-values correctly. On the other hand, Continuous state RL algorithm uses features and function approximation to represent the continuous state. The function approximation is a parametric form which the parametric value is updated at each time step, regardless of the state-action pair visited. This the reason to that the training is reduced significantly.

Finally, to test algorithm in response to the environment perturbation that may change over time, the Continuous state RL algorithm, Discrete state RL algorithm and Threshold algorithm have been evaluated. Results have shown that the Continuous state RL algorithm can consistently make correct decisions while the other algorithms do not. Therefore, the Continuous state multi-agent reinforcement learning is a promising tool for dealing with changing environments such as automatic lighting control application in smart homes.

In terms of memory requirement for storing entries in the Q-value, continuous state multi-agent reinforcement algorithm not need used Q-value table only requires memory storage for storing all values of $Q^i(s, a)$ which has $|s| \times |a_0| + |s| \times |a_1| = (1) \times (1) + (1) \times (1) = 2$ and feature of state value $w^i(s, a)$ which has $|s| \times |A| = (1) \times (2) = 2$ entries at each agent. Thus, Continuous state RL algorithm require 48 Bytes for memory storage. On the other hand, the Discrete state RL algorithm requires 64 Byte for storing all values of Q-values which has $|S| \times |A| = (4)(2) = 8$ entries at each agent. However, the state is a coarse discretization of 4 states which results in poor performance. Thus, to improve the performance of the Discrete state RL algorithm, a finer discretization may be needed. However, this results in an increase of memory storage for storing all values in the Q-value table which has $|S| \times |A| = (81)(2) = 162$ entries at each agent, which is a total of 1296 Bytes for memory storage.

In the final section to test the efficiency of each algorithm we have create a testbed. The results of testbed correspond with the simulation results. In terms of accuracy, the Continuous state RL algorithm is more accurate in finding an optimal

policy of 100% than a discrete state RL algorithm that finds an optimal policy of 65%. When considered convergence speed, Continuous state, Discrete state RL algorithm converges to the optimal policy at time step 395, 497, respectively. The better performance of the Continuous state RL algorithm due to the use of the suitable equation as a continuous-state representation. The Continuous state RL algorithm has the ability to learn more precisely and quickly.



CHAPTER VI

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In multi-agent system applications in wireless sensor networks, information exchange and cooperation between the agents are required to achieve a desired objective. In this thesis, we have studied the multi agent system for coverage control, which exchanges information between agents in their own area, until they can learn to adapt to changes in the environment. Therefore, the work this thesis is divided in to three parts: 1) Chapter 3 proposes a distributed, adaptive and scalable area coverage control algorithm based on a single combined objective function called the modified distributed value function (DVF) algorithm; 2) Chapter 4 focuses on improving the efficiency of the algorithm with *multi-objective optimization* (MOO) framework whereby the Scalarized Q Multi-Objective Reinforcement Learning (SQMORL) algorithm was applied to the coverage control problem. The algorithm is applied in a test bed for automatic lighting control in smart home applications for the purpose of maintaining lighting suitably and energy efficiency; 3) Chapter 5 centers on incorporating function approximation to cater *continuous state* multi-agent reinforcement learning systems. The original contributions and findings in this thesis can be summarized as follows.

6.1.1 Chapter 3

The objective of this chapter is

- *To show that reinforcement learning (RL) can be applied to multi-agent system (MAS) coverage control in wireless sensor networks.*

In this chapter, we present an algorithm for control the coverage area using the Distributed Value Function (DVF) algorithm. In order to increase effectiveness of the algorithm we propose the overlapping area of the neighboring node is the cost function (3.5), which controlled the sensor node to maintain the coverage itself and reduced the overlapping area for energy efficiency.

The modified DVF algorithm presented in this chapter is compared with guaranteed complete coverage algorithms, namely, the optimal geographical density control (OGDC) and a partial area coverage algorithm, called Probing Environment and Adaptive Sleep (PEAS). Results show that the modified DVF can achieve nearly full coverage with only 13-64% of active sensor nodes, whereas the OGDC and PEAS required 14-68% and 16-76% of active sensor nodes, respectively, for high to low node densities. Results suggest that the MAS coverage control scheme based on the modified DVF can achieve efficient coverage control, is self-adaptive and therefore suitable for coverage control applications in WSNs.

The contribution of this chapter is thus twofold: 1) a modified multi-agent coverage control scheme based on a redundancy coverage area cost function; 2) comparison of the scheme with non-learning coverage control schemes, i.e., OGDC and PEAS.

6.1.2 Chapter 4

In Chapter 3, the multi-agent system which is based on RL is found suitable for controlled the area coverage in wireless sensor networks. RL has the advantage of an online learning algorithm which can adapt to the environment. The

tool can find the optimal policy to maintain coverage based on energy efficiency by using the proposed cost function (3.5) to reduce the energy consumption of sensor nodes. However, the modified DVF algorithm developed in Chapter 3 can be further enhanced to cater coverage area control in denser WSNs. In the particular, when the number of sensor nodes density increases, the number of active nodes selected by the algorithm increases too. To alleviate this problem, a multi-objective optimization (MOO) is introduced in Chapter 4. Furthermore, we want to evaluate the performance of the algorithm when this approach is applied to automatic illumination control with grid placement of light bulb in the Wireless Communication Laboratory F4, SUT. The purpose of the test bed is to construct and evaluate the performance for lighting control based on the MOO framework in a smart home application. This leads to the following objectives in Chapter 4.

The objectives of this chapter are

- *To study multi objective optimization for coverage control simulation in uniform randomly placed sensor nodes.*

The modified DVF algorithm in Chapter 3 is an algorithm that combines multiple objectives into a single combined objective function. This is suitable for uncomplicated solutions and fewer number of sensor nodes. However, when the problem is complex and the number of nodes increases or when the objective function is contradictory, the single objective optimization algorithm may not be able to find the most appropriate policy. This leads us to study a multi objective optimization based on reinforcement learning that targets multiple agents with different functionalities and objectives in the online learning system. In

particular, chapter 4 applies a multi objective reinforcement learning algorithm, called the SQMORL algorithm to the area coverage control in the wireless sensor network.

When compared the SQMORL algorithm is compared against the OGDC and DVF algorithms. The results show that when the sensor nodes are placed *randomly*, the SQMORL algorithm selects fewer working nodes than the DVF algorithm. At the highest node density, SQMORL algorithm has a coverage area close to that of OGDC algorithm. In the case of 100 nodes, SQMORL algorithm attained a 96.5% coverage which is more than DVF algorithm by 1% but used 1 node less than DVF. Similarly, SQMORL can attain 3.5% less coverage but used 11 nodes less than OGDC.

- *To study multi objective optimization for coverage control simulation in a grid layout WSN.*

The purpose of this experiment is to evaluate SQMORL performance in a simulation model for automatic lighting control for smart home application with light bulbs placed in a grid layout. Based on the results of the grid layout simulation, the results agree with the uniform randomly placed sensor nodes. In a grid of 25, 81, 121 sensor nodes, SQMORL DVF and OGDC algorithm selects 9 - 36, 10 - 39, 23 - 115 active nodes and attained coverage of 80%, 80%, 96-98%, respectively. The results show that SQMORL algorithm is the most efficient algorithm based on the metrics considered.

- *To construct a prototype for automatic lighting control using SQMORL algorithm and evaluate its performance.*

The performance evaluation is based on DVF and SQMORL algorithms. The results are averaged over 20 runs to achieve the desired accuracy. The DVF

algorithm attained the optimal policy rate of 55% from the 20 repeated trials, while the SQMORL algorithm attained an optimal policy rate of 80% from the 20 repeated trials. However, in terms of the convergence speed, the DVF algorithm reached the optimal policy at time step 127 whereas the SQMORL algorithm reached the optimal policy at time step 135 which is slightly slower than the DVF algorithm. Such results show that the SQMORL algorithm is suitable for controlled indoor lighting automation in order to save energy and obtain good illumination in smart home applications.

The contribution of this chapter is therefore three-fold: 1) The multi-objective optimization based on SQMORL for coverage control by simulation of a uniform randomly placed sensor nodes and grid layout; 2) Comparison of SQMORL with both learning and non-learning coverage control wireless sensor networks; 3) Development of prototype and performance evaluation of an automatic lighting control using SQMORL algorithm.

6.1.3 Chapter 5

While Chapter 4 applied a multi objective optimization based on a reinforcement learning approach called SQMORL for automatic lighting control in smart home, significant memory storage and training time is required due to the discrete state space. In particular, in Chapter 4, the state of the light intensity detected at each sensor node is quantized to discrete levels of light intensity. However, in the real world, such light intensity is a continuous value. This leads us to the objectives in Chapter 5 as follows.

- *To propose a function approximation for continuous state multi-agent reinforcement learning and apply it to the automatic lighting control application.*

Continuous state reinforcement learning method uses function approximation to map continuous state to a particular function, in order to minimize the loss of data attributes when data is discretized by quantization. However, it is difficult to implement a large number of discretized continuous states for wireless sensor networks with limited onboard power and memory storage. Therefore, it is necessary to introduce the concept of function approximation to create a feature which can represent the continuous state. Therefore, in order to cater lighting automation control, this chapter proposes a function approximation for light intensity to be used in continuous state RL. The proposed function approximation is calculated by the light intensity generated from the light source. In terms of memory requirement for storing entries of the Q-values in the SQMORL algorithm, continuous state SQMORL needs memory storage for storing all values of $Q^i(s,a)$ which has $|n_{old}| \times |A| + |n_{new}| \times |A|$ entries and a feature of state $w^i(s,a)$ which has $|n| \times |A|$ entries at each agent. By reducing the Q-value table, the number of state visits and training time also reduces, so convergence rate is increased.

- *To study the effects of parameters related to learning based on continuous state multi-agent reinforcement learning.*

Chapter 5 also investigates parameters that affect learning and the ability to find the optimal policy in continuous state multi-agent reinforcement learning, which are the learning rate and the discount factor. Based on the simulation

results, the suitable learning rate for the system is 0.02-0.04, the discount factor between 0.02-0.08 which achieves the best performance. With respect to the ability to find the optimal policy, the learning rate value and discount factor in this range can consistently determine the optimal policy.

- *To compare of the Continuous state RL algorithm with Discrete state RL and Threshold algorithm in terms of final policy accuracy, convergence rate and adaptability to certain perturbations.*

The Continuous state RL can find the final optimal policy consistently, while the Discrete state RL algorithm with its state discretization type 1 and 2 achieve only 50% and 60% , respectively. This is because the discrete state algorithms require visits to all state-action pairs, to correctly learn their Q-values. For the Threshold algorithm, we assign the decision threshold for switching each bulb at 77.77 lux. The resulting final policy is that all sensor nodes switch on the light bulbs in this experiment. Although Threshold algorithm is simple, but choosing the optimal threshold value is tedious as light intensity, position of the sensor nodes and energy consumption must be considered.

In terms of convergence rate, Continuous state RL algorithm can converge to the final optimal policy at time step 97, whereas Discrete state RL algorithms type 1, 2 can convergence into optimal policy at time step 1796, 1815 respectively. Since the Discrete state RL algorithms requires visits to all state-action pairs of the Q-value table. However, the Continuous state RL algorithm uses parametric feature approximation to represent the continuous state, thus visits to all state action pairs it is unnecessary. The parameter is tuned at every state-action visit.

Therefore, the algorithm updates the parametric weight vector across all state-action pair simultaneously resulting in an increased convergence speed.

Finally, to demonstrate the efficiency of algorithm in response to environment perturbations that may change over time as well as presence of faulty nodes, the continuous state SQMORL has been evaluate in situations which sudden shut down of some nodes and some external lighting is introduced into the system at some period. Based on simulation results, the SQMORL algorithm is able to adjust its light control decisions when subject to external lighting, by shutting off some light bulbs to reduce power consumption. In presence of faulty nodes when some of the nodes close the light bulbs, the system chooses to turn on the remaining light bulbs to maintain the desired brightness level.

The contribution of this chapter is three-fold: 1) The application of function approximation to the automatic lighting framework where light intensity is considered a continuous state. 2) The study parameters that affect Continuous state RL algorithm. 3) Performance comparison of the Continuous state RL algorithm with Discrete state RL algorithms and Threshold algorithm.

6.2 Future work

6.2.1 Extend SQMORL algorithm to densely deployed network

In thesis, to study the coverage control performance of a WSN, we assumed the agent functions as a lighting controller which illuminates an area size of 1000 x 1000 sq.m. The area contains a number of sensor nodes ranging from 100,200,300,400,500 sensor nodes placed randomly in the area. However, in the

practice, the number of sensor nodes can extend to even more nodes may be considered and how it effects the coverage control problem may be investigated.

6.2.2 Extension SQMORL algorithm to weight learning approach

The MOO framework may be extended to handle multiple objectives in coverage control simultaneously such as maximizing covered area, balanced energy consumption, minimizing the transmission range of nodes, minimizing the number of active nodes, and maintaining the connectivity of active nodes, etc. As the importance of each problem may differ depending on the intended application, the SQMORL algorithm with a fixed weight factor may not be appropriate. Therefore, the weight factors may require different values and may need adjustment to suit a particular application. Therefore, the weight learning approach is worthwhile investigating.

6.2.3 Extension SQMORL algorithm to continuous action space

In this thesis, the SQMORL algorithm has been applied to an automatic lighting control problem, which is designed for continuous state of light intensity. However, a continuous action change of system may be required in order to minimize the effect on the users' perceptions. Thus, continuous action SQMORL may warrant further investigation.

6.2.4 Extend MOO framework to other systems in smart home

So far in this thesis, SQMORL algorithm is used for automatic lighting control in smart homes. However, the multi-objective optimization framework can be applied to control other appliances in smart homes. The framework may also serve as a guideline to intelligent homes that brings convenience to everyday life and attains efficient energy usage.

REFERENCES

- Agarkhed, J. and Ankalgi, R. (2016). Energy efficient smart home monitoring system in wireless sensor network. **IEEE International Conference on Circuit, Power and Computing Technologies (ICCPCT)**.
- Akyildiz, F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002). Wireless sensor networks: a survey. **Computer Networks Journal**, vol. 38, pp. 393-422.
- Alaiad, A. and Zhou, L. (2015). Patients' Behavioral Intentions toward Using WSN based Smart Home Healthcare Systems: An Empirical Investigation. **Proceedings of the 2015 Hawaii International Conference on System Sciences (HICSS)**, p.824-833.
- Atzori, L., Iera, A. and Morabito, G. (2010). The Internet of Things: A survey. **Journal of Computer Networks**, Vol. 54, pp. 2787-2805.
- Barbancho, J., Leon, C., Molina, F.J. and Barbancho, A. (2007). Using artificial intelligence in routing schemes for wireless networks. **Journal of computer Communications**, Vol. 30, pp. 2802-2811.
- Brito, J., Gomes, T., Miranda, J., Monteiro, J., Cabral, J., Mendes, J. and Monteiro, J. (2014). An intelligent home automation control system based on a novel heat pump and Wireless Sensor Networks. **IEEE International Symposium on Industrial Electronic (ISIE)**.

- Carlos, A.C.C., Pulido, G.T. and Lechuga, M.S. (2004). Handling Multiple Objectives with Particle Swarm Optimization. **IEEE Transactions on Evolutionary Computation**, Vol. 8, pp. 256-279.
- Chandrakasan, A.P. et.al. (2000). An Architecture for a Power-Aware Distributed Microsensor Nodes. **In IEEE workshop on Signal Processing Systems (SiPS'00)**, Lafayette, LA.
- Chen, G., Vu, C. T., Zhao, Y. and Li, Y. (2009). A Universal Framework for Coverage Problem in Wireless Sensor Network. **Proceedings of the 2009 IEEE International Conference on Network Protocols (ICNP)**.
- Chitnis, L., Dobra A. and Ranka, S. (2009). Fault tolerant aggregation in heterogeneous sensor network. **Journal Parallel Distrib. Comput.**
- Cho-hoang, T. and Duy, C.N. (2017). Environment monitoring system for agricultural application based on wireless sensor network. **IEEE International Conference on Information Science and Technology (ICIST)**.
- D&R International**, Ltd. (2011). Buildings Energy Data Book.
- Gokul, V., Kannan, P. and Kumar, S. (2016). Deep Q-Learning for Home Automation. **International Journal of Computer Applications**, Vol. 152-No.6.
- Guestrin, C., Lagoudakis, M.G., and Parr, R. (2002). Coordinated reinforcement learning. **In ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

- Han, X., Cao, X., Lloyd, E.L. and Shen, C.C. (2007). Fault-tolerant Relay Node Placement in Heterogeneous Wireless Sensor Networks. **Proceedings of IEEE Communications Society subject matter experts for publication in the IEEE (INFOCOM)**.
- Haobijam, B., Huang, Y.P. and Lee, T.T. (2016). Intuitive IoT-based H2U healthcare system for elderly people. **IEEE International Conference on Networking, Sensing, and Control (ICNSC)**.
- Hasselt, H.V. and Wiering M.A. (2007). Reinforcement Learning in Continuous Action Spaces. **Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)**.
- Hill, J., Horton, M., Kling, R. and Krishnamurthy, L. (2004). The Platforms Enabling Wireless Sensor Networks. **Communications of the ACM Journal**, Vol. 47, No.6.
- Hussain, Md.A., Khan, P. and Sup K.K. (2009). WSN Research Activities for Military Application. **Proceedings of 2009 International Conference on Advanced Communication Technology**, p.271-274.
- Huynh, T.P., Tan, Y.K. and Tseng, K.J. (2011). Energy-Aware Wireless Sensor Network with Ambient Intelligence for Smart LED Lighting System Control. **Annual Conference on IEEE Industrial Electronics Society (IECON)**.
- Iqbal, M., Naeem, M., Anpalagan, A., Ahmed, A. and Azam, M. (2015). Wireless Sensor Network Optimization: Multi-Objective Paradigm. **Journal of sensors**, Vol. 15, pp.17572-17620.

- Iqbal, M., Naeem, M., Anpalagan, A., Qadri, N.N. and Imran, M. (2016). Multi-objective optimization in sensor networks: Optimization classification, applications and solution approaches. **Journal of Computer Networks**, Vol. 99, pp. 134-161.
- Jabbar, Z.A. and Kawitkar, R.S. (2016). Implementation of Smart Home Control by Using Low Cost Arduino & Android Design. **International Journal of Advanced Research in Computer and Communication Engineering**, Vol. 5, pp. 248-256.
- Jameii, S.M., Faez, K. and Dehghan, M. (2014). Multi-objective Optimization for Topology and Coverage Control in Wireless Sensor Networks. **International Journal of Distributed Sensor Networks**, vol. 2015, 11 page.
- Jia, J., Chen, J., Chang, G. and Tan, Z. (2009). Energy efficient coverage control in wireless sensor networks based on multi-objective genetic algorithm. **Journal of Computers and Mathematics with Application**, Vol. 57, pp. 1756-1766.
- Fei, Z., Li B., Yang S., Xing C., Chen H. and Hanzo L., (2016) A Survey of Multi-Objective Optimization in Wireless Sensor Networks: Metrics, Algorithms, and Open Problems, **IEEE Communications Surveys & Tutorials**, Vol. 19, pp. 550-586.
- Kaelbling, L., Littman, M., and Moore, A. (1996). Reinforcement learning: A survey. **Journal of Artificial Intelligence Research**, vol. 4.
- Khalili, A.H., Wu, C. and Aghajan, H. (2010). Hierarchical preference learning for light control from user feedback. **Workshops on IEEE Computer Society Conference on Computer Vision and Pattern Recognition**.

- Kuh, A., Zhu, C. and Mandic, D. (2006). Sensor Network Localization Using Least Squares Kernel Regression. **International Conference on Knowledge-Based and Intelligent Information and Engineering Systems**, pp. 1280-1287.
- Kul, B., Sen, M. and Ksa, K. (2016). IP Based Smart Energy Metering with Energy Saving. **Proceedings of International Scientific Electronic (ET2016)**.
- Kumaar, A.A., Kiran, G., and Sudarshan TSB. (2010). Intelligent Lighting System Using Wireless Sensor Networks. **International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC)**, Vol.1, No.4, pp. 17-27.
- Li, M., Lu1, Y. and Wee, L. (2006). Target Detection and Identification a Heterogeneous Sensor Network by Strategic Resource Allocation and Coordination. **Proceedings of IEEE 6th International Conference on ITS Telecommunications**.
- Meng-Shiuan, P., Lun-Wu, Y., Yen-Ann, C., Yu-Hsuan, L. and Yu-Chee, T. (2008). A WSN-Based Intelligent Light Control System Considering User Activities and Profiles. **IEEE Sensor Journal**, vol. 8, pp. 1710-1721.
- Ministry of energy**, Thailand 20-Year Energy Efficiency Development Plan (2011-2030)
- Moffaert, K.V., Drugan, M.M. and Nowe, A. (2013). Scalarized Multi-Objective Reinforcement Learning: Novel Design Techniques. **IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)**.
- Mohamaddoust, R., Haghightat, A.T., Sharif, M.J.M. and Capanni, N. (2011). A Novel Design of an Automatic Lighting Control System for a Wireless Sensor

Network with Increased Sensor Lifetime and Reduced Sensor Numbers. **Journal of sensors**, ISSN 1424-8220, pp. 8933-8952.

Mitchell, T. (1997). *Machine Learning*, Cambridge, MA, USA: McGraw-Hill Press.

Nangtin, P., Kumhom, P. and Chamnongthai, K. (2016). Adaptive actual load for energy saving in split type air conditioning. **International Symposium on Communications and Information Technologies (ISCIT)**.

Okada, M., Aida, H. and Ichikawa, H. (2015). Design and Implementation of an Energy-Efficient Lighting System Driven by Wireless Sensor Networks. **International Conference on Mobile Computing and Ubiquitous Networking (ICMU)**.

Phuphanin, A. and Usaha, W. (2011). Secure Coverage Control in Wireless Sensor Networks with Malicious Nodes using Multi-Agents. **Proceedings of the 2011 IFIP International Conference on Embedded and Ubiquitous Computing (EUC)**.

Phuphanin, A. and Usaha, W. (2016). A Multi-Agent Scheme for Energy-Efficient Coverage Control in Wireless Sensor Networks. **Proceedings of International Conference on Information Technology and Science (ICITS)**.

Puccinelli, D. and Haenggi, M. (2005). Wireless Sensor Networks: Applications and Challenges of Ubiquitous Sensing. **IEEE Circuits and Systems Magazine**, Volume 5, Issue 3, pp. 19-31.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. **Wiley-Interscience**.

- Ratnasingham, T. and Thiagalingam, K. (2009). Optimization-Based Dynamic Sensor Management for Distributed Multitarget Tracking. **IEEE Transaction on Systems, Man, and Cybernetics**, Vol. 39, pp. 534-546.
- Rea, M.S. (2000). The IESNA lighting handbook. **Illuminating Engineering Society of North America**, New York.
- Rovcanin, M., Poorter, E.D., Akker, D.V.D., Moerman, I., Demeester, P. and Blondia, C. (2015). Experimental validation of a reinforcement learning based approach for a service-wise optimisation of heterogeneous wireless sensor networks. **Journal of Wireless Networks**, Vol. 21, pp. 931-948.
- Santoshkumar and Udaykumar, R.Y. (2015). Development of WSN System for Precision Agriculture. **Innovations in Information Embedded and Communication Systems (ICIECS), 2015 International Conference**, pp.1-5.
- Seah, M. W. M., Tham, C. K., Srinivasan, V. and Xin. A. (2007). Achieving Coverage through Distributed Reinforcement Learning in Wireless Sensor Networks. **Proceedings of IEEE International Conference on Intelligent Sensors, Sensor Networks and Information**, 2007.
- Schneider, J., Wong, W. K., Moore, A. and Riedmiller, M. (1999). Distributed value functions. **Proceedings of 16th International Conference on Machine Learning**.
- Sinha, A. and Chandrakasan, A. (2001). Dynamic Voltage Scheduling using adaptive filtering of workload traces. **In proceedings of the 11th International Conference on VLSI Design**.

- Singhiv, V., Krause, A., Guestrin, C., Jame, Jr. and Matthews, H. (2005). Intelligent Light Control using Sensor Networks. **Proceedings of the 3rd international conference on Embedded networked sensor systems**, pp. 218-229.
- Souto, W., Pazzi, R.W. and Pramudianto, F. (2015). User Activity Recognition for Energy Saving in Smart Home Environment. **IEEE Symposium on Computers and Communication (ISCC)**.
- Stankovic, A.J. (2006). Wireless Sensor Networks. **Department of Computer Science University of Virginia**.
- Sung, W.T. and Lin, J.S. (2013). Design and Implementation of a Smart LED Lighting System Using a Self Adaptive Weighted Data Fusion Algorithm. **Journal of Sensors**, vol. 13, pp. 16915-16939.
- Sutton, R.S. and Barto, A.G. (1998). Introduction to Reinforcement Learning, Cambridge, MA, USA: MIT Press.
- Stroock, D.W. (2005). An Introduction to Markov Processes, Cambridge, MA, USA: Springer Press.
- Tafa, Z. (2016). Artificial Neural Networks in WSNs Design: Mobility Prediction for Barrier Coverage. **IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)**.
- Tham, C.K. and Renaud, J.C. (2005). Multi-Agent Systems in Sensor Networks: A Distributed Reinforcement Learning Approach. **Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing**.

- Vieira, M.et.al. (2003). Survey on Wireless sensor Network Devices. **In proceedings of Emerging Technologies and Factory Automation, 2003 IEEE Conference**, Vol. 1, 16-19, pp. 537-544.
- Vinyals, M., Rodriguez-Aguilar, J.A. and Cerquides, J. (2011). A Survey on Sensor Networks from a Multi-agent Perspective. **The Computer Journal**, 54(3), pp.455-470.
- Vu, C. T., Cai, Z. and Li, Y. (2009). Distributed Energy-Efficient algorithms to maximize network lifetime for coverage problem in adjustable sensing radii Wireless Sensor Networks. **Submitted to Discrete Mathematics, Algorithms and Applications (DMAA)**.
- Vu, C. T., Cai, Z. and Li, Y. (2009). A universal framework for -coverage problem in Wireless Sensor Network. **Submitted to The 17th IEEE International Conference on Network Protocols (ICNP)**.
- Ye, F., Zhang, H., Lu, S., Zhang, L. and Hou, J. (2006). A Randomized Energy-Conservation Protocol for Resilient Sensor Networks. **Wireless Networks**, 12(5), pp.637-652.
- Yick, J., Mukherjee, B. and Ghosal, D. (2008). Wireless sensor network survey. **Journal of Computer Networks**, 52.
- Yu, L., Wang, N., Zhang, W. and Zheng, C. (2007). Deploying a Heterogeneous Wireless Sensor Network. **This work is supported in part by Shanghai Municipal Science and Technology Commission under Grants, No. 05dz15004 IEEE**.

- Wang, B. (2011) Coverage Problems in Sensor Networks: A Survey. **ACM Computing Surveys**, 43(4), Article 32, 53 pages.
- Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R. and Gill, C. (2003). Integrated coverage and connectivity configuration in wireless sensor networks. **SenSys'03**, Los Angeles, California, USA.
- Wang, Y. and Dasgupta, P. (2014). Designing Adaptive Lighting Control Algorithms for Smart Buildings and Homes. **IEEE 11th International Conference on Networking, Sensing and Control (ICNSC)**.
- Zhang, H. and Hou, J. C. (2005). Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. **Ad Hoc & Sensor Wireless Networks**, 1(1), pp.89-124.
- Zhang, P., Sadler, M., Lyon, A. and Martonosi, M. (2004). Hardware Design Experiences in ZebraNet. **In proceedings of SenSys'04**, Baltimore, USA.



APPENDIX A

AUTOMATIC LIGHTING CONTROL TEST BED CODE

This appendix show distributed value function (DVF) algorithm and scalarized Q multi objective reinforcement learning (SQMROL) employed in chapter 4. To prove the efficiency of proposed algorithms, we setup automatic lighting control application consisted of Arduino Uno for test bed. And used code following.

Distributed Value Function algorithm:

```

#include <XBee.h>
#include <EEPROM.h>
#include <SPI.h>
#include <SD.h>
// create the XBee object
XBee xbee = XBee();
uint8_t payload[7];
// SH + SL Address of receiving XBee
XBeeAddress64 addr64 = XBeeAddress64(0x0, 0xFFFF);
ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
ZBTxStatusResponse txStatus = ZBTxStatusResponse();
//For receive command packet
XBeeResponse response = XBeeResponse();
// create reusable response objects for responses we expect to handle
ZBRxResponse rx = ZBRxResponse();
ModemStatusResponse msr = ModemStatusResponse();
File myFile;
const int chipSelect = 4;
int LUX_Voltage = 0;
int statusLed = 13;
int errorLed = 13;
int dataLed = 13;
int RELAY = 8;
int addr = 0;
int value = 0;
int Status = 0;
int State = 0;
long Time_step = 0;
int LDR_itself = 0;
int Reward = 0;
double Sum_reward = 0;
double Ave_reward = 0;
int Value_func = 0;
int Neigh_fac = 0;
const float Alpha = 0.1;
const float Gamma = 0.2;
int full_value = 0;
int Value_neighbor = 0;
int Value_neighbor_all = 0;

```

```

int A=0,B=0,C=0,D=0,E=0,F=0,G=0;
//Creat Q table
int Q00 = 0,Q01 = 0;
int Q10 = 0,Q11 = 0;
int Q20 = 0,Q21 = 0;
int Q30 = 0,Q31 = 0;
int Q[4][2] = {13,29,15,32,12,27,12,32};
byte BH_Q[6][2];
byte BL_Q[6][2];
int addrBH_Q[6][2] = {0,2,4,6,8,10,12,14,16,18,20,22};
int addrBL_Q[6][2] = {1,3,5,7,9,11,13,15,17,19,21,23};

void flashLed(int pin, int times, int wait)
{
    for (int i = 0; i < times; i++)
    {
        digitalWrite(pin, HIGH);
        delay(wait);
        digitalWrite(pin, LOW);
        if (i + 1 < times) { delay(wait); }
    }
}

void setup()
{
    pinMode(statusLed, OUTPUT);
    pinMode(errorLed, OUTPUT);
    pinMode(dataLed, OUTPUT);
    pinMode(RELAY, OUTPUT);
    Serial.begin(9600);
    xbee.setSerial(Serial);
    randomSeed(analogRead(0));

    Serial.print("Initializing SD card..");
    pinMode(4, OUTPUT);
    if (!SD.begin(chipSelect))
    {
        Serial.println("initialization failed!");
        return;
    }

    Serial.println("initialization done.");
    flashLed(statusLed, 3,25);

    //Setting Q table
    for(int i=0;i<=3;i++)
    {
        for(int j=0;j<=1;j++)
        {
            BH_Q[i][j] = Q[i][j] >> 8 & 0xff;
            BL_Q[i][j] = Q[i][j] & 0xff;
        }
    }
}

```

```

addr = 0;
for(int i=0;i<=3;i++)
    {
        for(int j=0;j<=1;j++)
            {
                EEPROM.write(addr,BH_Q[i][j]);
                addr++;
                EEPROM.write(addr,BL_Q[i][j]);
                addr++;
            }
    }

for(int i=0;i<=3;i++)
    {
        for(int j=0;j<=1;j++)
            {
                Q[i][j] = ((BL_Q[i][j] << 0) & 0xFF) + ((BH_Q[i][j] << 8) & 0xFF00);
            }
    }
}

void loop()
{

Serial.println("");
Serial.print("Time_step = ");
Serial.print(Time_step);
Serial.println("");
Serial.println(" Q table ");

for(int i=0;i<=3;i++)
    {
        for(int j=0;j<=1;j++)
            {
                Serial.print(Q[i][j]);Serial.print('\t');
            }
        Serial.println();
    }
if(Time_step < 100)
    {
        Serial.println("Choose action by random(e-greedy)");
        Status = random(2);
        if (Status == 0)
            {
                Serial.println("Turned off the lamp");

                digitalWrite(RELAY, LOW);
            }
        else
            {
                Serial.println("Turned on the lamp");
                digitalWrite(RELAY, HIGH);
            }
    }
}

if (Time_step >= 100)
    {

```

```

        Serial.println("Choose action by greedy");
        if(Q[State][0]>Q[State][1])
        {
            Status = 0;
            Serial.println("Turned off the lamp");
            digitalWrite(RELAY, LOW);
        }
        else
            {
                Status = 1;
                Serial.println("Turned on the lamp");
                digitalWrite(RELAY, HIGH);
            }
    }

    Serial.print("Status = ");
    Serial.println(Status);
    delay(5000);
    LUX_Voltage = analogRead(0);
    payload[0] = LUX_Voltage >> 8 & 0xff;
    payload[1] = LUX_Voltage & 0xff;
    EEPROM.write(addr,payload[0]);
    EEPROM.write(addr +1 ,payload[1]);

    if(LUX_Voltage >= 0 && LUX_Voltage <= 32){State = 0; goto out_quanti;}
    if(LUX_Voltage > 32 && LUX_Voltage <= 55){State = 1; goto out_quanti;}
    if(LUX_Voltage > 55 && LUX_Voltage <= 194){State = 2; goto out_quanti;}
    else{State = 3; goto out_quanti;}
    out_quanti:

    //Calculate reward
    if(Status == 0){LDR_itself = 0;}
    else{LDR_itself = 168;}
    Reward = LUX_Voltage - LDR_itself;
    Sum_reward = Sum_reward + Reward;
    Ave_reward = Sum_reward/Time_step;

    myFile = SD.open("datalog.txt",FILE_WRITE);

    if (myFile)
    {
        myFile.print(Time_step); // สั่งให้เขียนข้อมูล
        myFile.print("\t");
        myFile.print(Status);
        myFile.print("\t");
        myFile.print(LUX_Voltage);
        myFile.print("\t");
        myFile.print(State);
        myFile.print("\t");
        myFile.println( Ave_reward);
    }
}

```

```

myFile.close(); // ปิดไฟล์
//Serial.println("done.");
}
else
{
    // ถ้าเปิดไฟล์ไม่สำเร็จ ให้แสดง error
    Serial.println("error opening datalog.txt");
}

//Find value function from Neighbor
if(Q[State][0]>Q[State][1]){Value_func = Q[State][0];}
else{ Value_func = Q[State][1];}
payload[5] = Value_func >> 8 & 0xff;
payload[6] = Value_func & 0xff;

//Calculate neighbor of factor
Neigh_fac = 1/4;
Value_neighbor_all = Neigh_fac*(A+B+C+D);

//Calculate Q value
Q[State][Status] = ((1-Alpha)*Q[State][Status])+((Alpha)*(Reward+(Gamma*Value_neighbor_all)));
BH_Q[State][Status] = Q[State][Status] >> 8 & 0xff;
BL_Q[State][Status] = Q[State][Status] & 0xff;
EEPROM.write(addrBH_Q[State][Status],BH_Q[State][Status]);
EEPROM.write(addrBL_Q[State][Status],BL_Q[State][Status]);

if(digitalRead(RELAY)==0){payload[2] = 144 & 0xff;}
if(digitalRead(RELAY)==1){payload[2] = 145 & 0xff;}

byte Byte1 = EEPROM.read(addr);
byte Byte2 = EEPROM.read(addr + 1);
value = ((Byte2 << 0) & 0xFF) + ((Byte1 << 8) & 0xFF00);

Serial.print(" analogRead A0 = ");
Serial.println(value);
Serial.print("State = ");

Serial.println(State);
Serial.print("Reward = ");
Serial.println(Reward);
Serial.print(" Value function = ");
Serial.println(Value_func);
Serial.print("Q value = ");
Serial.println(Q[State][Status]);

Time_step = Time_step + 1;
payload[3] = Time_step >> 8 & 0xff;

```

```

payload[4] = Time_step & 0xff;
EEPROM.write(260,payload[3]);
EEPROM.write(261,payload[4]);
byte Time_byte1 = EEPROM.read(260);
byte Time_byte2 = EEPROM.read(261);
Time_step = ((Time_byte2 << 0) & 0xFF) + ((Time_byte1 << 8) & 0xFF00);

addr = addr + 1;
if (addr == 512) addr = 0;

for(int i=0;i<=9;i++)
{
    xbee.send(zbTx);
    // flash TX indicator
    flashLed(statusLed, 1, 100);
    // after sending a tx request, we expect a status response
    // wait up to half second for the status response
    if (xbee.readPacket(500))
    {
        // got a response!
        // should be a znet tx status
        if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE)
        {
            xbee.getResponse().getZBTxStatusResponse(txStatus);
            // get the delivery status, the fifth byte
            if (txStatus.getDeliveryStatus() == SUCCESS)
            {
                // success. time to celebrate
                flashLed(statusLed, 5, 50);
            }
            else
            {
                // the remote XBee did not receive our packet. is it powered on?
                flashLed(errorLed, 3, 500);
            }
        }
        else if (xbee.getResponse().isError())
        {
            //nss.print("Error reading packet. Error code: ");
            //nss.println(xbee.getResponse().getErrorCode());
        } else {
            // local XBee did not provide a timely TX Status Response -- should not happen

            flashLed(errorLed, 2, 50);
        }
    }

    delay(1000);

    //Recive value function from neighbor

    xbee.readPacket();

    if (xbee.getResponse().isAvailable())

```

```

    {
        // got something
        if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
            {
                // got a zb rx packet
                Serial.println(" ");
                Serial.println("got a zb rx packet");
                // now fill our zb rx class
                xbee.getResponse().getZBRxResponse(rx);
            }
        if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED) {
            // the sender got an ACK
            flashLed(statusLed, 10, 10);
        } else {
            // we got it (obviously) but sender didn't get an ACK
            flashLed(errorLed, 2, 20);
        }
    }

XBeeAddress64 senderLongAddress = rx.getRemoteAddress64();
uint32_t DH_sensor = senderLongAddress.getMsb();
Serial.print(DH_sensor,HEX);
Serial.print(" ");
uint32_t DL_sensor = senderLongAddress.getLsb();
Serial.println(DL_sensor);

// READ LUX Voltage
uint8_t V_High = rx.getData(5);
uint8_t V_Low = rx.getData(6);
Value_neighbor = ((V_Low << 0) & 0xFF) + ((V_High << 8) & 0xFF00);
if(DL_sensor == 1086054566)
    {
        A = Value_neighbor;
        //Serial.print(" Value function neighbor from EOA6 = ");
        //Serial.println(A);
        goto out_Value;
    }
if(DL_sensor == 1085895355)
    {
        B = Value_neighbor;
        //Serial.print(" Value function neighbor from 72BB = ");
        //Serial.println(B);
        goto out_Value;
    }

if(DL_sensor == 1084938783)
    {
        C = Value_neighbor;
        //Serial.print(" Value function neighbor from DA1F = ");
        //Serial.println(C);
        goto out_Value;
    }
if(DL_sensor == 1085895347)
    {
        D = Value_neighbor;
        //Serial.print(" Value function neighbor from 72B3 = ");
        //Serial.println(D);
    }

```

```

        goto out_Value;
    }
    if(DL_sensor == 1085895338)
    {
        E = Value_neighbor;
        //Serial.print(" Value function neighbor from 72AA = ");
        //Serial.println(E);
        goto out_Value;
    }
    if(DL_sensor == 1084938838)
    {
        F = Value_neighbor;
        //Serial.print(" Value function neighbor from DA56 = ");
        //Serial.println(F);
        goto out_Value;
    }
    else
    {
        G = Value_neighbor;
        //Serial.print(" Value function neighbor from 7281 = ");
        //Serial.println(G);
        goto out_Value;
    }
    out_Value:
    //Serial.println(A);Serial.println(B);Serial.println(C);
    Serial.println(D);Serial.println(E);Serial.println(F);Serial.println(G);
    Serial.print(" Value function neighbor = ");
    Serial.println(Value_neighbor);
    full_value = 1;

} else if (xbee.getResponse().getApiId() == MODEM_STATUS_RESPONSE) {
xbee.getResponse().getModemStatusResponse(msr);
// the local XBee sends this response on certain events, like association/dissociation

if (msr.getStatus() == ASSOCIATED)
{
    // yay this is great. flash led
    flashLed(statusLed, 10, 10);
} else if (msr.getStatus() == DISASSOCIATED) {
    // this is awful.. flash led to show our discontent
    flashLed(errorLed, 10, 10);
} else {

        // another status
        flashLed(statusLed, 5, 10);
    }
} else {
    // not something we were expecting
    flashLed(errorLed, 1, 25);
}
} else if (xbee.getResponse().isError())

```



```

        {           //nss.print("Error reading packet. Error code: ");
                   //nss.println(xbee.getResponse().getErrorCode());
        }

        full_value = 0;
    }
}

```

Scalarized Q multi objective reinforcement learning algorithm:

```

#include <XBee.h>
#include <EEPROM.h>
#include <SPI.h>
#include <SD.h>
// create the XBee object
XBee xbee = XBee();
uint8_t payload[9];

// SH + SL Address of receiving XBee
XBeeAddress64 addr64 = XBeeAddress64(0x0, 0xFFFF);
ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
ZBTxStatusResponse txStatus = ZBTxStatusResponse();

//For receive command packet
XBeeResponse response = XBeeResponse();
// create reusable response objects for responses we expect to handle
ZBRxResponse rx = ZBRxResponse();
ModemStatusResponse msr = ModemStatusResponse();

File myFile;
const int chipSelect = 4;
int LUX_Voltage = 0;
int statusLed = 13;
int errorLed = 13;
int dataLed = 13;
int RELAY = 8;
int addr = 0;
int value = 0;
int Status = 0
int State = 0;
long Time_step = 0;
int LDR_itself = 0;
int Reward1 = 0;
int Reward2 = 0;
double Sum_reward1 = 0;
double Sum_reward2 = 0;
double Ave_reward1 = 0;
double Ave_reward2 = 0;
int Value_func1 = 0;

```

```

int Value_func2 = 0;
int Neigh_fac = 0;
const float Alpha = 0.1;
const float Gamma = 0.2;
int full_value = 0;
int Value_neighbor1 = 0;
int Value_neighbor2 = 0;
int Value_neighbor_all1 = 0;
int Value_neighbor_all2 = 0;
int A_1=0,B_1=0,C_1=0,D_1=0,E_1=0,F_1=0,G_1=0;
int A_2=0,B_2=0,C_2=0,D_2=0,E_2=0,F_2=0,G_2=0;
int W1=0.5;
int W2=0.5;

//Creat Q table
int Q1[4][2] = {26,45,28,46,182,197,167,200};
int Q2[4][2] = {0,-16,0,-16,-151,-168,-151,-168};
int SQ[4][2] = {161,175,352,364,188,193,242,320};
/*int Q1[4][2] = {0,0,0,0,0,0,0,0};
int Q2[4][2] = {0,0,0,0,0,0,0,0};
int SQ[4][2] = {0,0,0,0,0,0,0,0};*/

void flashLed(int pin, int times, int wait)
{
    for (int i = 0; i < times; i++)
    {
        digitalWrite(pin, HIGH);
        delay(wait);
        digitalWrite(pin, LOW);
        if (i + 1 < times)
        {
            delay(wait);
        }
    }
}

void setup()

{
    pinMode(statusLed, OUTPUT);
    pinMode(errorLed, OUTPUT);
    pinMode(dataLed, OUTPUT);
    pinMode(RELAY, OUTPUT);

    Serial.begin(9600);
    xbee.setSerial(Serial);

```

```

randomSeed(analogRead(0));
Serial.print("Initializing SD card...");
pinMode(4, OUTPUT);
if (!SD.begin(chipSelect))
    {
        Serial.println("initialization failed!");
        return;
    }
Serial.println("initialization done.");
flashLed(statusLed, 3,25);
}

void loop()
{
    Serial.println("");
    Serial.print("Time_step = ");
    Serial.print(Time_step);

    if(Time_step < 100)
        {
            Serial.println("Choose action by random(e-greedy)");
            Status = random(2);
            if (Status == 0)
                {
                    Serial.println("Turned off the lamp");
                    digitalWrite(RELAY, LOW);
                }
            else
                {
                    Serial.println("Turned on the lamp");
                    digitalWrite(RELAY, HIGH);
                }
        }

    if (Time_step >= 100)
        {
            Serial.println("Choose action by greedy");
            if(SQ[State][0]>SQ[State][1])
                {
                    Status = 0;
                    Serial.println("Turned off the lamp");
                    digitalWrite(RELAY, LOW);
                }
            else
                {
                    Status = 1;
                    Serial.println("Turned on the lamp");
                    digitalWrite(RELAY, HIGH);
                }
        }

    Serial.print("Status = ");

    Serial.println(Status);
    delay(5000);
}

```

```

LUX_Voltage = analogRead(0);
payload[0] = LUX_Voltage >> 8 & 0xff;
payload[1] = LUX_Voltage & 0xff;

if(LUX_Voltage >= 0 && LUX_Voltage <= 32){State = 0; goto out_quanti;}
if(LUX_Voltage > 32 && LUX_Voltage <= 55){State = 1; goto out_quanti;}
if(LUX_Voltage > 55 && LUX_Voltage <= 194){State = 2; goto out_quanti;}
else{State = 3; goto out_quanti;}
out_quanti:

//Calculate reward
if(Status == 0){LDR_itself = 0;}
else{LDR_itself = 168;}
Reward1 = LUX_Voltage;
Reward2 = -LDR_itself;
Sum_reward1 = Sum_reward1 + Reward1;
Sum_reward2 = Sum_reward2 + Reward2;
Ave_reward1 = Sum_reward1/Time_step;
Ave_reward2 = Sum_reward2/Time_step;

myFile = SD.open("datalog.txt",FILE_WRITE);
if (myFile)
    {
        myFile.print(Time_step); // สั่งให้เขียนข้อมูล
        myFile.print("\t");
        myFile.print(Status);
        myFile.print("\t");
        myFile.print(LUX_Voltage);
        myFile.print("\t");
        myFile.print(State);
        myFile.print("\t");
        myFile.print(Ave_reward1);
        myFile.print("\t");
        myFile.println(Ave_reward2);
        myFile.close(); // ปิดไฟล์
        //Serial.println("done.");
    } else {
        // ถ้าเปิดไฟล์ไม่สำเร็จ ให้แสดง error
        Serial.println("error opening datalog.txt");
    }

//Find value function from Neighbor
if(Q1[State][0]>Q1[State][1]){Value_func1 = Q1[State][0];}
else{Value_func1 = Q1[State][1];}
if(Q2[State][0]>Q2[State][1]){Value_func2 = Q2[State][0];}

else{Value_func2 = Q2[State][1];}

```

```

payload[5] = Value_func1 >> 8 & 0xff;
payload[6] = Value_func1 & 0xff;
payload[7] = Value_func2 >> 8 & 0xff;
payload[8] = Value_func2 & 0xff;

//Calculate neighbor of factor
Neigh_fac = 1/4;
Value_neighbor_all1 = Neigh_fac*(A_1+B_1+C_1+D_1);
Value_neighbor_all2 = Neigh_fac*(A_2+B_2+C_2+D_2);

//Calculate Q value
Q1[State][Status] = ((1-
Alpha)*Q1[State][Status])+((Alpha)*(Reward1+(Gamma*Value_neighbor_all1)));
Q2[State][Status] = ((1-
Alpha)*Q2[State][Status])+((Alpha)*(Reward2+(Gamma*Value_neighbor_all2)));

SQ[State][Status] = (W1*Q1[State][Status])+(W2*Q2[State][Status]);

if(digitalRead(RELAY)==0){payload[2] = 144 & 0xff;}
if(digitalRead(RELAY)==1){payload[2] = 145 & 0xff;}

Serial.print(" analogRead A0 = ");
Serial.println(value);
Serial.print(" State = ");
Serial.println(State);
Serial.print(" Reward = ");
Serial.print(Reward1);
Serial.print("\t");
Serial.println(Reward2);
Serial.print(" Value function = ");
Serial.println(Value_func1);
Serial.print(" Q value = ");
Serial.print(Q1[State][Status]);
Serial.print("\t");
Serial.println(Q2[State][Status]);

Time_step = Time_step + 1;
payload[3] = Time_step >> 8 & 0xff;
payload[4] = Time_step & 0xff;

for(int i=0;i<=9;i++)
{
    xbee.send(zbTx);
    // flash TX indicator
    flashLed(statusLed, 1, 100);
    // after sending a tx request, we expect a status response

```

```

// wait up to half second for the status response
if (xbee.readPacket(500))
    {
        // got a response!
        // should be a znet tx status
        if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE)
            {
                xbee.getResponse().getZBTxStatusResponse(txStatus);
                // get the delivery status, the fifth byte
                if (txStatus.getDeliveryStatus() == SUCCESS)
                    {
                        // success. time to celebrate
                        flashLed(statusLed, 5, 50);
                    }
                else {
                    // the remote XBee did not receive our packet. is it powered on?
                    flashLed(errorLed, 3, 500);
                }
            }
        }
    }
    } else if (xbee.getResponse().isError()) {
        //nss.print("Error reading packet. Error code: ");
        //nss.println(xbee.getResponse().getErrorCode());
    }
    } else {
// local XBee did not provide a timely TX Status Response -- should not happen
flashLed(errorLed, 2, 50);
    }
}

delay(1000);
//Recive value function from neighbor
xbee.readPacket();
if (xbee.getResponse().isAvailable())
    {
        // got something
        if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
            {
                // got a zb rx packet
                Serial.println(" ");
                Serial.println("got a zb rx packet");
                // now fill our zb rx class
                xbee.getResponse().getZBRxResponse(rx);
                if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED)
                    {
                        // the sender got an ACK
                        flashLed(statusLed, 10, 10);
                    }
                else {
                    // we got it (obviously) but sender didn't get an ACK
                    flashLed(errorLed, 2, 20);
                }
            }
    }

XBeeAddress64 senderLongAddress = rx.getRemoteAddress64();
uint32_t DH_sensor = senderLongAddress.getMsb();
Serial.print(DH_sensor,HEX);
Serial.print(" ");
uint32_t DL_sensor = senderLongAddress.getLsb();
Serial.println(DL_sensor);

```

```

// READ LUX Voltage
uint8_t V_High1 = rx.getData(5);
uint8_t V_Low1 = rx.getData(6);
uint8_t V_High2 = rx.getData(7);
uint8_t V_Low2 = rx.getData(8);

Value_neighbor1 = ((V_Low1 << 0) & 0xFF) + ((V_High1 << 8) & 0xFF00);
Value_neighbor2 = ((V_Low2 << 0) & 0xFF) + ((V_High2 << 8) & 0xFF00);

if(DL_sensor == 1086054566)
    {
        A_1 = Value_neighbor1; A_2 = Value_neighbor2;
        //Serial.print(" Value function neighbor from EOA6 = ");
        //Serial.println(A);
        goto out_Value;
    }
if(DL_sensor == 1085895355)
    {
        B_1 = Value_neighbor1; B_2 = Value_neighbor2;
        //Serial.print(" Value function neighbor from 72BB = ");
        //Serial.println(B);
        goto out_Value;}
if(DL_sensor == 1084938783)
    {
        C_1 = Value_neighbor1; C_2 = Value_neighbor2;
        //Serial.print(" Value function neighbor from DA1F = ");
        //Serial.println(C);
        goto out_Value;
    }
if(DL_sensor == 1085895347)
    {
        D_1 = Value_neighbor1; D_2 = Value_neighbor2;
        //Serial.print(" Value function neighbor from 72B3 = ");
        //Serial.println(D);
        goto out_Value;
    }
if(DL_sensor == 1085895338)
    {
        E_1 = Value_neighbor1; E_2 = Value_neighbor2;
        //Serial.print(" Value function neighbor from 72AA = ");
        //Serial.println(E);
        goto out_Value;
    }
if(DL_sensor == 1084938838)
    {
        F_1 = Value_neighbor1; F_2 = Value_neighbor2;
        //Serial.print(" Value function neighbor from DA56 = ");
        //Serial.println(F);
        goto out_Value;
    }
else
    {
        G_1 = Value_neighbor1; G_2 = Value_neighbor2;
        //Serial.print(" Value function neighbor from 7281 = ");

```

```

//Serial.println(G);

    goto out_Value;}
    out_Value:
    full_value = 1;
}
else if (xbee.getResponse().getApiId() == MODEM_STATUS_RESPONSE) {
    xbee.getResponse().getModemStatusResponse(msr);
    // the local XBee sends this response on certain events,

if (msr.getStatus() == ASSOCIATED)
    {
        // yay this is great. flash led
        flashLed(statusLed, 10, 10);
    } else if (msr.getStatus() == DISASSOCIATED) {
        // this is awful.. flash led to show our discontent
        flashLed(errorLed, 10, 10);
    } else {
        // another status
        flashLed(statusLed, 5, 10);
    }
} else {
    // not something we were expecting
    flashLed(errorLed, 1, 25);
}
} else if (xbee.getResponse().isError())
    {
        //nss.print("Error reading packet. Error code: ");
        //nss.println(xbee.getResponse().getErrorCode());
    }

    full_value = 0;
}
}
}

```




APPENDIX B

**FINAL POLICY POSSIBLE IN AUTOMATIC LIGHTING
CONTROL**

มหาวิทยาลัยเทคโนโลยีสุรนารี

In appendix B we explain the automatic lighting control experimental position setting. The automatic lighting control system consists of sensor nodes, each of which are equipped with a wireless communication module with XBee Series 2, a microcontroller part with Arduino Uno R3 and additional external memory unit for recording measurements for control purposes, a light dependent resistor (LDR) to measure the light intensity at every light bulb in the system. Each sensor node has the ability to measure the intensity of light within their own area, exchange information between the neighboring node sensors and collect the data inside the memory as shown in figure B.1.

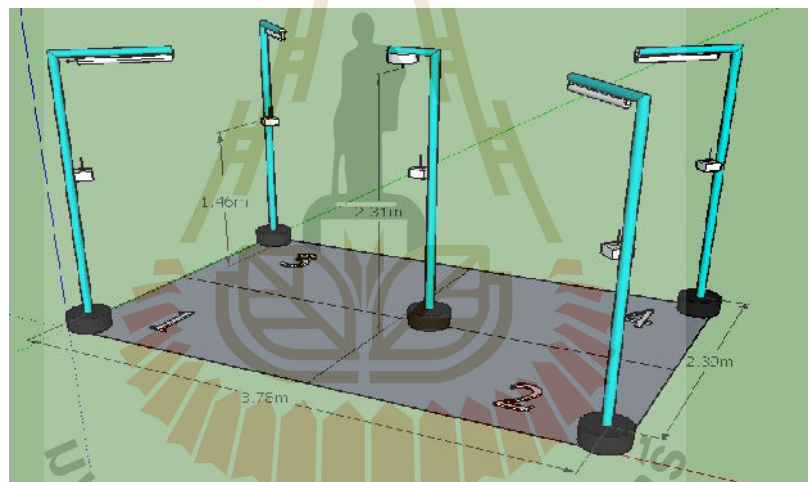
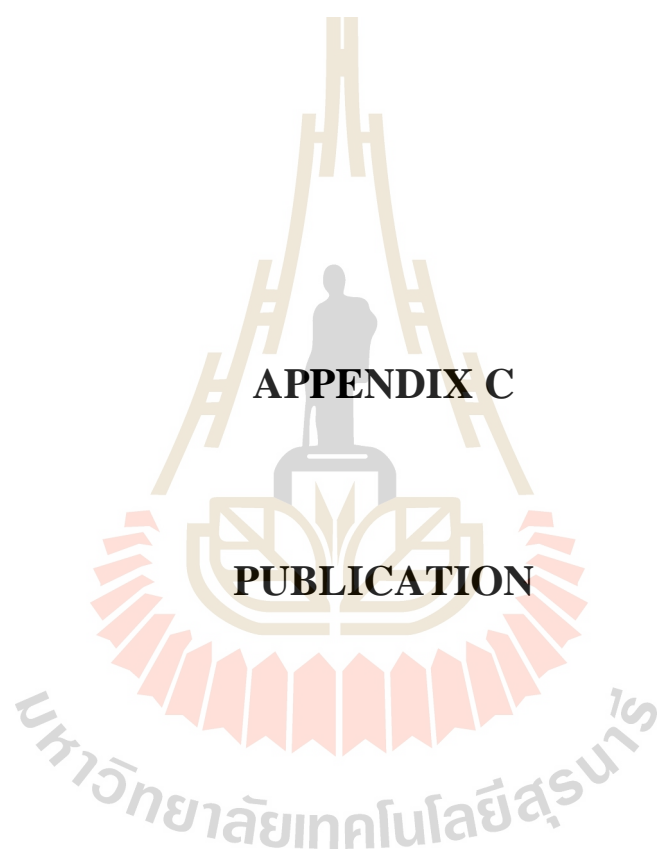


Figure B.1 The location of the five node sensors.

Figure B.1 represent the distance and height of each sensor node. Since there are experimental room has 5 sensor nodes, each sensor node has 2 actions. Thus, there are 32 possible policies. The optimal policy that is appropriate for this room is the 16th policy, where the 4 sensors in the corner of the room are turned on and sensor node in the middle of the room is turned off as show in table B.1

Table B.1 Final policy possible of five sensor node.

Policy	Sensor nodes				
	E0A6 [0]	72BB [1]	DA1F [2]	72B3 [3]	72AA [4]
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	1	1
5	0	0	1	0	0
6	0	0	1	0	1
7	0	0	1	1	0
8	0	0	1	1	1
9	0	1	0	0	0
10	0	1	0	0	1
11	0	1	0	1	0
12	0	1	0	1	1
13	0	1	1	0	0
14	0	1	1	0	1
15	0	1	1	1	0
16	0	1	1	1	1
17	1	0	0	0	0
18	1	0	0	0	1
19	1	0	0	1	0
20	1	0	0	1	1
21	1	0	1	0	0
22	1	0	1	0	1
23	1	0	1	1	0
24	1	0	1	1	1
25	1	1	0	0	0
26	1	1	0	0	1
27	1	1	0	1	0
28	1	1	0	1	1
29	1	1	1	0	0
30	1	1	1	0	1
31	1	1	1	1	0
32	1	1	1	1	1



List of Publications

Phuphanin, A., Usaha, W. (2016). A Multi-Agent Scheme for Energy-Efficient Coverage Control in Wireless Sensor Networks, **Proceedings of ICITS International Conference on Information Technology and Science. Tokyo, Japan**



A Multi-Agent Scheme for Energy-Efficient Coverage Control in Wireless Sensor Networks

Akkachai Phuphanin and Wipawee Usaha *

School of Telecommunication Engineering, Suranaree University of Technology,
 111 University Avenue, Muang District, Nakhon Ratchasima 30000 Thailand

Abstract. The underlying aim of this paper is to maximize the energy efficiency in the coverage control scheme in a wireless sensor network (WSN) by selecting the minimal number of working nodes while still maintaining network coverage area. The proposed algorithm is based on a self-adaptive multi-agent system (MAS) coverage control scheme whereby sensor nodes learn to adjust their own coverage to achieve the network-wide coverage. This paper proposes a variation of an existing MAS scheme called the distributed value function (DVF) which differs from the original scheme in the use of cost function which is a function of redundant coverage area. Performance evaluation were compared with a guaranteed complete coverage method, i.e., the optimal geographical density control (OGDC) scheme, and a partial area coverage scheme, i.e., the Probing Environment and Adaptive Sleeping (PEAS) scheme. Results show that modified DVF can achieve a nearly full coverage with only 13-64% of active sensor nodes whereas the OGDC and PEAS required 14-68% and 16-76% of active sensor nodes, respectively for high to low node densities. Results suggests that the MAS coverage control scheme can achieve efficient coverage control, is self-adaptive and therefore suitable for coverage control applications in WSNs such as lighting control in smart offices.

Keywords: coverage control, multi-agent systems, wireless sensor networks, reinforcement learning

1. Introduction

Wireless sensor networks (WSNs) are a collection of numerous cheap sensory devices installed within a particular environment to gather the physical parameters of interest. Measurements of these sensor devices are then acquired and relayed through the network to be processed or collected at the base station. Such data acquisition gives the ability to continuously monitor the particular surroundings of interest and respond quickly to any changes that may incur. WSNs have emerged in biomedical, military, agricultural monitoring and control applications [1], [2], [3]. In smart homes or buildings, lighting control have been a particular application which coverage control is needed to reduce energy consumption while maintaining a required level of light intensity.

Coverage control problems have been a significant issues arising in wireless sensor networks with the aim to extend the longevity of network lifetime and efficient energy consumption in the network due to the limited on-board battery power of a sensor node [4]. Refs. [5] and [6] proposed a distributed wireless sensor network with adjustable sensing radii enabling a flexible and efficient coverage. In [7], the authors proposed a Probing Environment and Adaptive Sleeping (PEAS) scheme which is a coverage maintenance scheme that increases the network lifetime by maintaining a necessary number of working nodes and shutting down the rest as reserve. By querying neighboring nodes, a particular working node can determine the status of neighboring working and sleeping nodes prior to deciding on its own status. In [8], the optimal geographical density control (OGDC) scheme was proposed as a guaranteed full coverage control scheme based on grid redundancy check and sequential node activation. The grid redundancy requires that each

* Corresponding author. Tel.: + (66) 4422 4392 ; fax: +(66) 4422 4603.
 E-mail address: wusaha@ieee.org.

sensor node maintain a list of the grid points it covers. The sequential node activation requires that each active node sends out activation messages to neighboring nodes to reset their timers.

The aforementioned coverage control methods are non-learning schemes which require reconfiguration should the environment change. Due to node deployment in potentially wide areas, direct access to reconfigure the nodes may not be feasible. On the other hand, multi-agent system (MAS) technologies have shown to be promising due to their flexibility and self-adaptability which caters autonomous self-awareness at sensor nodes [11]. In a multi-agent system (MAS), nodes act as agents which have the ability to learn and adjust their coverage in a distributed manner thereby enabling a light weight self-adaptive coverage control. The nodes in a MAS decide their actions in a cooperative manner to achieve a mutual goal of maximizing the network coverage by using the minimal amount of energy. To do so, a cost function based on a function of redundant coverage areas of a sensor node is introduced.

The contribution of this paper is thus twofold: 1) a modified multi-agent coverage control scheme based on a redundancy coverage area cost function; 2) comparison of the scheme with non-learning coverage control schemes, i.e., OGDC and PEAS. The objective is to maximize the coverage control efficiency by maximizing the obtained coverage control per unit of energy consumed. Our results suggests the suitability of applying MAS in coverage control in WSNs.

2. Multi-Agent Coverage Control

2.1. Distributed Value Function Scheme

A multi-agent coverage control scheme called the Distributed Value Function (DVF) has been proposed to co-ordinately and cooperatively improve the coverage control performance in wireless sensor networks [9]. In this method, each node communicates and exchanges information about its value function. A value function is a function that quantifies how well the agent (sensor node) performs at a given state $s \in S$ where S is a discrete set of all possible states of the sensor network. Let $a \in A$ be the action selected by an agent, where A is the discrete set of all possible actions available at each state. The decision rule of an agent, so called policy π , is defined as a rule which the agent selects an action as a function of its state. In other words, it is the mapping from a state $s \in S$ and action $a \in A$ to the probability of selecting action a at state s . The

value function of state s under a given policy π is formally defined by $V^\pi(s) = E^\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}$,

where r_{t+1} is the reward of taking a particular action in a given state s at time t , γ is the discount factor and $E^\pi \{ \cdot \}$ is the expectation operator. Similarly, we define the action value function of taking action a at a given state under policy π by

$$Q^\pi(s, a) = E^\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\}.$$

The objective is to find a policy π^* such that $\pi^* = \underset{\pi}{\operatorname{argmax}} Q^\pi(s, a)$. To achieve this objective, each agent i (node) in the DVF algorithm performs an update of its own action value function. The update rule at time step t for agent i is given by [9]:

$$Q_{i,t+1}^i(s_t^i, a_t^i) = (1 - \alpha) Q_{i,t}^i(s_t^i, a_t^i) + \alpha (r_{i,t+1}^i(s_t^i) + \gamma \sum_{j \in \operatorname{Neigh}(i)} f^j(j) V_j^i(s_{t+1}^i)) \quad (1)$$

$$V_{i,t+1}^i(s_t^i) = \max_{a \in A} Q_{i,t+1}^i(s_t^i, a) \quad (2)$$

where α is the learning rate, $f^j(j)$ are factors that weigh the value functions of the neighbors of agent i such that

$$f^j(j) = \begin{cases} \frac{1}{|\operatorname{Neigh}(i)|} & , \text{if } \operatorname{Neigh}(i) \neq \emptyset \\ 1 & , \text{otherwise} \end{cases} \quad (3)$$

where $j \in \operatorname{Neigh}(i)$ is the set of neighbors of node i [9].

2.2. Modified DVF Framework for Coverage Control

Consider a wireless sensor network comprising multiple light sensor nodes. For a particular sensor node i , the local state and actions taken are defined as follows.

Local agent state: Each sensor node i can sense the level of coverage in its area. Its local state s^i is state of each agent based on its mode and coverage area.

Local agent actions: Each sensor node i has the ability to take one of the following actions in any state it lands in. The action space A^i is the set of all possible actions for each state $A^i = \{Action_0, Action_1\}$ where *Action_0* (*Action_1*) refers to sensor node i turning off (on).

Each action decided by sensor node i results in a reward, denoted as $r^i(s_t^i)$ which is a function of sensor node i 's state s^i at time t defined by

$$r^i(s_t^i) = G^i(s_t^i) - C^i \quad (4)$$

where $G^i(s_t^i)$ is a function of the number of cells within the coverage area of sensor node i such that

$$G^i(s_t^i) = Area_coverage(a^i) \times GAIN_CELL_BRIGHT, \quad (5)$$

and C^i is the area overlapped as a result from the action taken by sensor node i at time t .

3. Performance Evaluation

To compare the coverage control performance of a WSN, we considered a gridded area of 1000 x 1000 sq.m. containing a number of sensor nodes ranging from 100,200,300,400,500 sensor nodes placed randomly in the area.

The objective is for the sensor node to learn to cooperate with one another in order to completely coverage area in an energy-efficient way, i.e. minimize the number of sensor node turned on. The coverage area of node (agent) i was given within a transmission range of 100m. The initial energy of each sensor node was 10 Joule. Comparison was based on the number of working nodes, coverage percentage and coverage lifetime. The DVF was compared with OGDC, PEAS8 and PEAS9, where the latter two are PEAS with probing ranges of 80 and 90m, respectively.

According to equations (1) and (4), the value of the learning rate $\alpha = 0.4$, the discount factor $\gamma = 0.7$ and the $GAIN_CELL_BRIGHT = 0.5$. The values of the learning rate and discount factor were obtained from experimenting a range of values and selecting the parameters which received the best performance in terms of average accumulated reward in (4). The simulation results were averaged over 10 runs to achieve the desired accuracy.

Fig 1 depicts the number of working nodes against the number of deployed nodes in each algorithm. Note that OGDC, PEAS8 and PEAS9 consistently use a gradually increasing number of working nodes which is higher than DVF. The reason is due to DVF determines the working nodes which achieves the best coverage while saving energy consumption.

Fig 2. shows the percentage of coverage achieved by all algorithms against the number of deployed nodes. Note that above 200 deployed nodes, all algorithms can achieve full coverage with DVF attaining 99.2% coverage at 200 nodes and 99.8% at 500 deployed nodes. The reason is because DVF must conservatively select working nodes so as to reduce the amount of energy consumption. Even so, the modified DVF can achieve a nearly full coverage with only 13-64% of active sensor nodes whereas the OGDC and PEAS required 14-68% and 16-76% of active sensor nodes, respectively for high to low node densities.

Fig. 3. illustrates the efficiency in terms of coverage area per working node for each algorithm. The rationale is from the fact that coverage area attained is a trade-off with the energy dissipated by the working node. Results show that DVF achieved the highest coverage per working node, followed by OGDC and PEAS.

Fig. 4,5,6 depict the area coverage lifetime for each algorithm for various numbers of deployed nodes. The area coverage lifetime is defined by the duration from the start of network operation until the coverage

requirement is no longer satisfied. It is evident that the coverage lifetime for DVF is prolonged the most in terms of number of time steps due to the least number of working nodes. PEAS8 attained the least coverage lifetime. This is because PEAS requires acknowledgement messages in addition to the higher number of working nodes than OGDC and DVF.

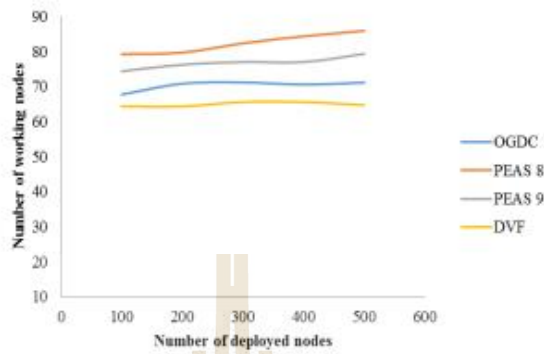


Fig. 1: Average number of working nodes against the number of deployed nodes.

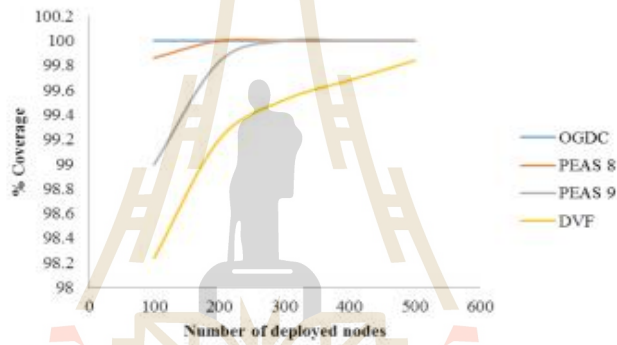


Fig. 2: Percentage of coverage against the number of deployed nodes.

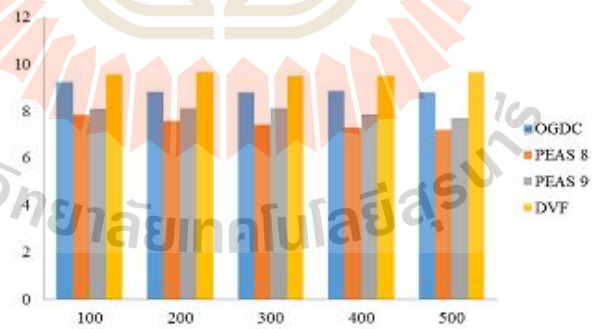


Fig. 3: Average coverage area per working node against number of deployed nodes.

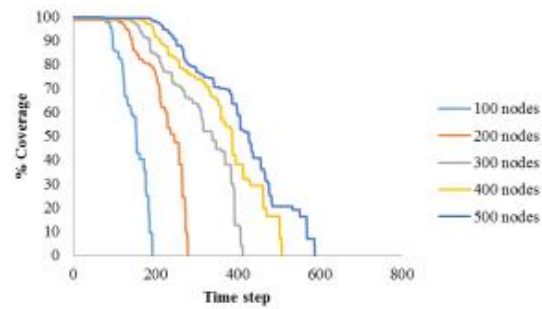


Fig. 4: Coverage lifetime for DVF.

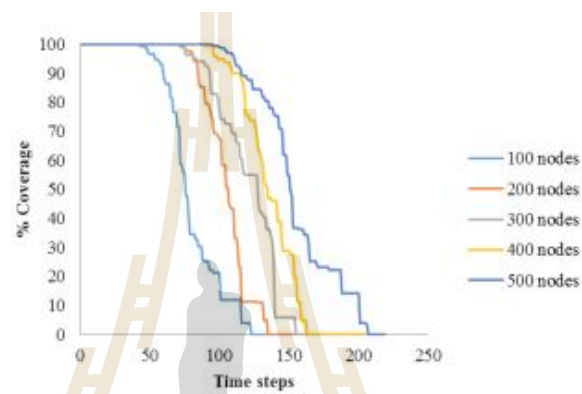


Fig. 5: Coverage lifetime for OGDC.

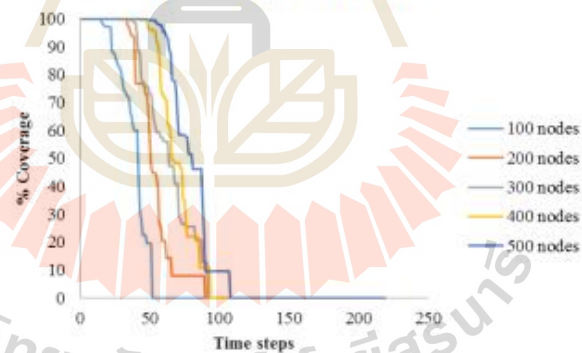


Fig. 6: Coverage lifetime for PEASS.

4. Conclusions

In this paper, we have proposed a cost function-modified distributed value function (DVF) scheme which is a multi-agent scheme aimed at energy-efficient coverage control in wireless sensor networks. Results were compared with two non-learning coverage control schemes i.e., PEAS which is a partial coverage control scheme and OGDC which is a guaranteed coverage control scheme. Results showed

that the proposed modified DVF attained the least working nodes of all while still achieving nearly complete coverage. Therefore, DVF outperformed PEAS and OGDC in terms of area coverage energy efficiency and area coverage lifetime. Results suggest the suitability of employing MAS for coverage control problems in WSNs. In the future, we plan to apply the DVF framework to lighting control applications in smart offices.

5. Acknowledgements

This research is supported by the National Research Council of Thailand.

6. References

- [1] A. Alaiad and L. Zhou, "Patients' Behavioral Intentions toward Using WSN based Smart Home Healthcare Systems: An Empirical Investigation," Proceedings of the 2015 Hawaii International Conference on System Sciences (HICSS), p.824-833, January, 2015.
- [2] Md.A.Hussain, P. Khan, and K.K. Sup, "WSN Research Activities for Military Application", Proceedings of 2009 International Conference on Advanced Communication Technology, p.271-274, February, 2009.
- [3] Santoshkumar, and Udaykumar R.Y, "Development of WSN System for Precision Agriculture", Innovations in Information, Embedded and Communication Systems (ICHIECS), 2015 International Conference on, p.1-5, March 2015.
- [4] B. Wang, "Coverage Problems in Sensor Networks: A Survey," ACM Computing Surveys, 43(4), Article 32, 53 pages, October, 2011.
- [5] C. T. Vu, Z. Cai, and Y. Li, "Distributed Energy-Efficient Algorithms to Maximize Network lifetime for Coverage Problem in Adjustable Sensing Radii Wireless Sensor Networks", submitted to Discrete Mathematics, Algorithms and Applications (DMAA), 1(3):299-317, 2009.
- [6] C. T. Vu, G. Chen, Y. Zhao, and Y. Li, "A Universal Framework for α -Coverage Problem in Wireless Sensor Network", Proceedings of the 2009 IEEE International Conference on Network Protocols (ICNP).
- [7] F. Ye, H. Zhang, S. Lu, L. Zhang and J. Hou, "A Randomized Energy-Conservation Protocol for Resilient Sensor Networks," Wireless Networks, 12(5), p.637-652, 2006.
- [8] H. Zhang, J. C. Hou, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks", Ad Hoc & Sensor Wireless Networks, 1(1), pp.89-124, 2005.
- [9] C.K. Tham, and J.C. Renaud, "Multi-Agent Systems in Sensor Networks: A Distributed Reinforcement Learning Approach", Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, December, 2005.
- [10] A. Phuphanin and W. Usaha, "Secure Coverage Control in Wireless Sensor Networks with Malicious Nodes using Multi-Agents" Proceedings of the 2011 IFIP International Conference on Embedded and Ubiquitous Computing (EUC), October, 2011.
- [11] M Vinyals, JA Rodriguez-Aguilar, J. Cerquides, "A Survey on Sensor Networks from a Multiagent Perspective," The Computer Journal, 54(3), p.455-470, 2011.

Authors' background

Your Name	Title*	Research Field	Personal website
Akkachai Phuphanin	Phd candidate	Coverage control, smart lighting, multiagents	
Wipawee Usaha	assistant professor	Wireless sensor networks applications, resource allocation, anomaly detection, signal extraction	

BIOGRAPHY

Mr. Akkachai Phuphanin was born on October 29, 1986 in Kalasin province, Thailand. He finished high school education from Kalasinpittayasan School, Kalasin province. He received her Bachelor's Degree in Engineering (Telecommunication) from Suranaree University of Technology in 2009. For her post-graduate, he continued to study with a Master's degree in the Telecommunication Engineering Program, Institute of Engineering, Suranaree University of Technology. During Master's degree education, she was a visiting at Centre for Dynamic Intelligent Communication (CIDCOM), Department of Electrical and Electronic Engineering, University of Strathclyde, Scotland in topic of wireless sensor networks. Then he is currently pursuing him Ph.D program in Telecommunication Engineering, School of Telecommunication Engineering, Suranaree University of Technology. He current research interests concern the design and simulation of network area coverage control in wireless sensor networks.