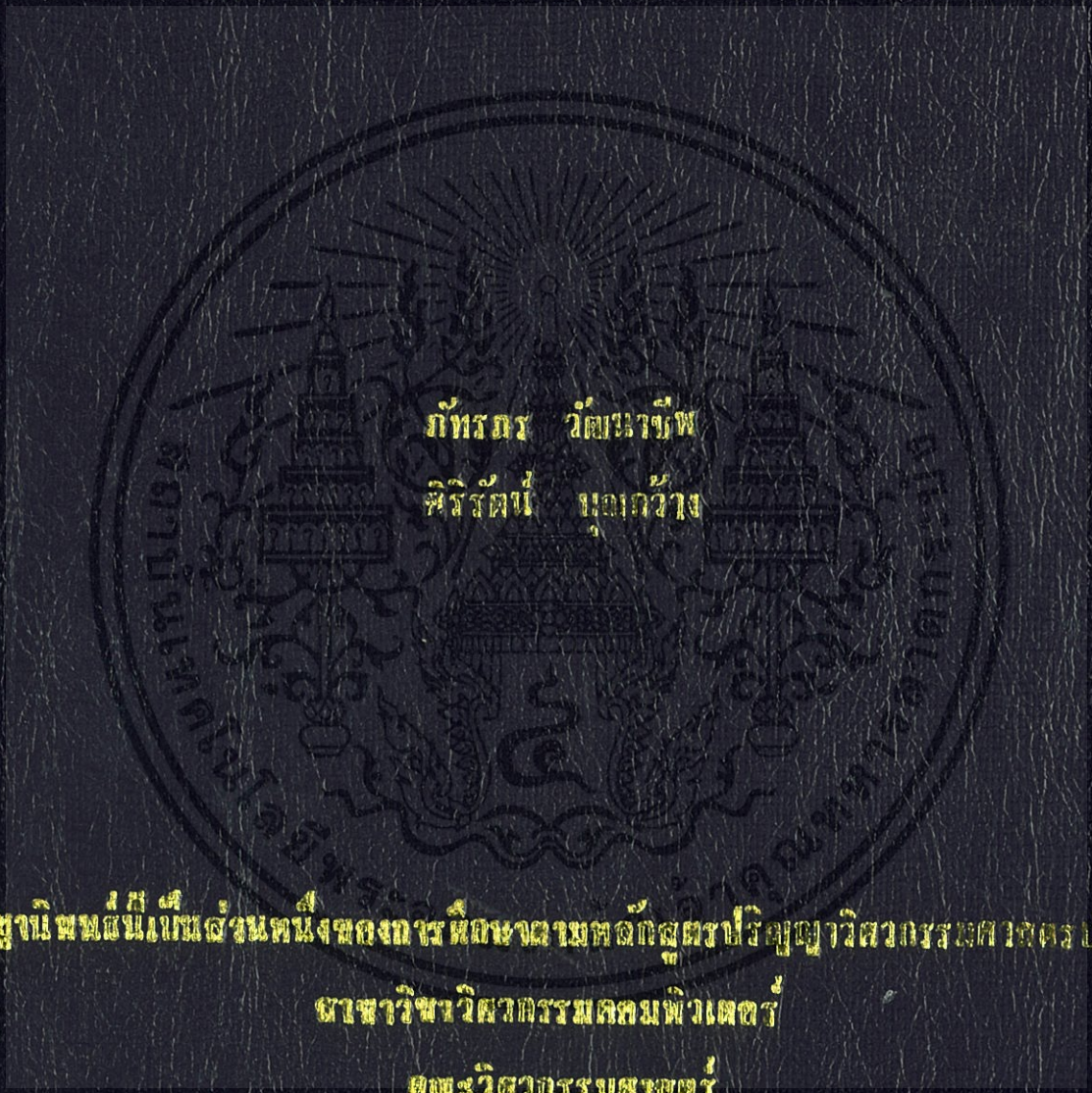
การเรียนรู้ของเครื่อง
MACHINE LEARNING

ภัทรภร วัฒนาชีพ
ศิริรัตน์ บุญกว้าง



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

การเรียนรู้ของเครื่อง

MACHINE LEARNING

ภัทรภร วัฒนาชีพ
ศิริรัตน์ บุญกว้าง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

ปริญญานิพนธ์ปีการศึกษา 2555

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การเรียนรู้ของเครื่อง

MACHINE LEARNING

ผู้จัดทำ

1. นางสาวภัทรภร	วัฒนาชีพ	รหัสนักศึกษา	52010894
2. นางสาวศิริรัตน์	บุญกว้าง	รหัสนักศึกษา	52011194

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ กฤตวัน ศิริบูรณ์)

การเรียนรู้ของเครื่อง

นางสาว ภัทรภร	วัฒนาชีพ	52010894
นางสาว ศิริรัตน์	บุญกว้าง	52011194
รศ. กฤตวัน	ศิริบูรณ์	อาจารย์ที่ปรึกษา

ปีการศึกษา 2555

บทคัดย่อ

การเรียนรู้ของเครื่องในรูปแบบของ การเรียนรู้แบบเสริมกาลัง (Reinforcement Learning) เป็นการเรียนรู้และตัดสินใจการกระทำจากประสบการณ์ที่ผ่านมา

ในปริญญานิพนธ์ฉบับนี้ผู้จัดทำได้นำมาทดลองกับเกมประเภทกระดาน ซึ่งเป็นเกมค้นหาเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดสุดท้าย โดยในการทดลองจะสร้างตัวเดินที่สามารถเรียนรู้จดจำ วิเคราะห์และค้นหาเส้นทาง จนกระทั่งค้นพบเส้นทางที่สั้นที่สุดได้ หากเราสามารถทำให้อัลกอริทึมที่สร้างขึ้นค้นหาเส้นทางที่สั้นที่สุดได้ จะเป็นเรื่องที่น่าสนใจและเป็นประโยชน์ต่อผู้ที่สนใจเป็นอย่างมาก อัลกอลิทึมที่ผู้จัดทำได้ศึกษาและทดลองใช้นั้น ได้แก่ อัลกอริทึมมัลติคาโล อัลกิริทึมทีดีคิวเลิร์นนิ่ง และอัลกอริทึมวัตกินส์คิว

MACHINE LEARNING

Ms. Pattaraporn	Wattanacheep	52010894
Ms. Sirirath	Boonkwang	52011194
Assoc. Prof. Kritawan	Siriboon	
Academic Year 2012		

ABSTRACT

Reinforcement learning is a machine learning technique which learns from past experiences and uses past experiences to act in the present. The reinforcement learning is applied to board types of games, which finds out the shortest distance from start to finish. We will design the agent that can learn, remember, and find out the shorter distance. If we can use the algorithm to get the shortest distance, it will be very useful to people who are interested. The Algorithm that we used are the Monte Carlo Method, Qlearning Method and Watkins's Q() Method.

กิตติกรรมประกาศ

ขอขอบคุณ รศ.กฤตวัน ศิริบูรณ์ ที่ให้คำปรึกษาและคำแนะนำในการพัฒนาโครงงานนี้ และช่วยให้ความรู้แก่ผู้จัดทำโครงงาน

ขอขอบคุณ รศ.ดร.บุญธีร์ เครือตราชู ที่ให้ความรู้และคำแนะนำในการพัฒนาโครงงานนี้ และช่วยให้คำชี้แนะแก่ผู้จัดทำโครงงาน

นางสาว ภัทรภร วัฒนาชีพ

นางสาว ศิริรัตน์ บุญกว้าง

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญรูป VII

บทที่ 1 บทนำ 1

1.1 ความสำคัญและที่มาของโครงงาน 1

1.2 วัตถุประสงค์ของโครงงาน 2

1.3 ขอบเขตของโครงงาน 2

1.4 วิธีการดำเนินการ 2

1.5 ประโยชน์ที่คาดว่าจะได้รับ 2

1.6 ส่วนประกอบของปริญญานิพนธ์ 3

บทที่ 2 ทฤษฎีบทที่เกี่ยวข้อง 4

2.1 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) 4

2.2 องค์ประกอบของการเรียนรู้แบบเสริมกำลัง 5

2.2.1 สถานะ (State) 5

2.2.2 ค่าตอบแทน (Reward) 5

2.2.3 ค่าผลกำไร (Return) 5

2.2.4 การกระทำ (Action) 5

2.2.5 ความน่าจะเป็นในการกระทำ (Policy) 5

2.2.6 ค่าของสถานะ (State Value, $V\pi$) 6

2.2.7 ค่าของการกระทำ (Action Value, $Q\pi$) 6

2.3 ปัญหาของการเรียนรู้แบบเสริมกำลัง (Reinforcement Leaning Problem) 6

2.4 กระบวนการตัดสินใจแบบมาร์คอฟ (Markov Property) 8

2.5 ฟังก์ชันมูลค่า 9

สารบัญ (ต่อ)

หน้า

2.6 การประเมินค่าตอบแทน 10
2.7 Monte Carlo Method 11
2.7.1 Monte Carlo Estimation of Action 12
2.8 Temporal Difference Learning(TD) 13
2.8.1 Q() 14

บทที่ 3 การออกแบบ 15
3.1 การออกแบบแบบจำลอง 15
3.1.1 ลักษณะแบบจำลอง 15
3.1.2 รูปแบบกระดาน 15
3.1.3 วิธีการใช้แบบจำลอง 15
3.2 การออกแบบหน้าต่างโปรแกรม 16
3.2.1 ส่วนกระดานเกม 16
3.2.2 ส่วนแสดงจำนวนรอบที่ผู้เรียนกำลังเรียน 17
3.2.3 ส่วนของช่องที่เราต้องกรอก 17
3.2.4 ส่วนของปุ่มกด 17
3.3 การหาเส้นทางและการตัดสินใจ 18
3.3.1 การเลือกเส้นทางโดยใช้ ε-greedy 18
3.3.2 การตัดสินใจในการเลือกเส้นทาง 19
3.4 การออกแบบอัลกอริทึม 19
3.4.1 อัลกอริทึม Monte Carlo 19
3.4.2 อัลกอริทึม Qlearning 20
3.4.3 อัลกอริทึม Watkins's Q() 21

บทที่ 4 การทดลองและผลการทดลอง 23
4.1 การทดลอง 23
4.1.1 การทดลองรูปแบบกระดานแบบที่ 1 23

สารบัญ (ต่อ)

หน้า

4.1.2 การทดลองรูปแบบกระดานแบบที่ 2 29
4.2 การเปรียบเทียบผลการทดลอง 35
4.2.1 เปรียบเทียบระหว่างกำหนดค่า Q เริ่มต้น เป็น 0 และ 30 โดยวิธี Monte Carlo 35
4.2.2 เปรียบเทียบกำหนดค่า e-greedy ต่างกัน โดยวิธี Qlearning 37
4.2.3 เปรียบเทียบการเรียนรู้ทั้ง 3 วิธี 39

บทที่ 5 บทสรุป 43
5.1 บทสรุป 43
5.2 ปัญหาอุปสรรคและแนวทางการแก้ไข 43
5.3 แนวทางในการพัฒนาต่อ 44

บรรณานุกรม 45

สารบัญรูป

รูปที่ **หน้า**

2.1 การโต้ตอบระหว่างผู้เรียนและสภาพแวดล้อมในการเรียนรู้แบบเสริมกำลัง 6

2.2 แบ็คอัพค่าผลตอบแทนสำหรับ (a) $V\pi$ และ (b) $Q\pi$ 10

2.3 An E-soft on-policy Monte Carlo control algorithm 13

3.1 หน้าต่างโปรแกรม 16

3.2 กระดานเกม 16

3.3 ส่วนแสดงจำนวนรอบที่ผู้เรียนกำลังเรียน 17

3.4 ส่วนแสดงช่องให้กรอกข้อมูลต่างๆ 17

3.5 ส่วนแสดงปุ่มกด 18

3.6 โครงสร้างต้นไม้ในการเลือกเส้นทางของผู้เรียนรู้ 18

3.7 แผนภาพการทำงานของโปรแกรม โดยใช้ Monte Carlo 20

3.8 แผนภาพการทำงานของโปรแกรม โดยใช้ Qlearning 21

3.9 แผนภาพการทำงานของโปรแกรม โดยใช้ Watkins's Q() 22

4.1 รูปแสดงรูปแบบกระดานแบบที่ 1 23

4.2 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 100 รอบ 24

4.3 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 1000 รอบ 24

4.4 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 10000 รอบ 25

4.5 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 100 รอบ 26

4.6 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 1000 รอบ 26

4.7 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 10000 รอบ 27

4.8 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 100 รอบ 28

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

4.9 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 1000 รอบ 28
4.10 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 10000 รอบ 29
4.11 รูปแสดงรูปแบบกระดานแบบที่ 2 29
4.12 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 100 รอบ 30
4.13 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 1000 รอบ 30
4.14 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 10000 รอบ 31
4.15 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 100 รอบ 32
4.16 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 1000 รอบ 32
4.17 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 10000 รอบ 33
4.18 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 100 รอบ 34
4.19 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 1000 รอบ 34
4.20 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 10000 รอบ 35
4.21 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 100 รอบ 36
4.22 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 1,000 รอบ 37

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

4.23 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 1,000 รอบ 38
4.24 รูปแสดงรูปแบบกระดานแบบที่ 3 39
4.25 กราฟแสดงการเรียนรู้ ใช้จำนวนรอทั้งหมด 100 รอบ 40
4.26 กราฟแสดงการเรียนรู้ ใช้จำนวนรอทั้งหมด 1,000 รอบ 41

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

เนื่องจากในปัจจุบันเทคโนโลยีที่เกี่ยวกับคอมพิวเตอร์ได้มีการพัฒนาขึ้นอย่างมาก ทำให้เทคโนโลยีคอมพิวเตอร์เข้ามาเป็นมีส่วนอยู่ในเครื่องมือ เครื่องจักรต่าง ๆ มากมาย ทั้งนี้ไม่ว่าจะเป็นด้านอุตสาหกรรม ด้านการค้าการตลาด รวมไปถึงการดำเนินชีวิตของมนุษย์โดยทั่วไป ได้มีการนำเอาเครื่องมือ เครื่องจักรเหล่านั้น มาใช้กันอย่างแพร่หลาย เพื่อช่วยอำนวยความสะดวกในด้านต่างๆ โดยได้มีการนำมาใช้ทำงานหลายอย่างแทนมนุษย์ ซึ่งเมื่อเปรียบเทียบกับมนุษย์แล้ว พบว่าเครื่องมือเครื่องจักร สามารถทำงานได้เร็วและดีกว่าการใช้มนุษย์มาก หากแต่เครื่องมือ เครื่องจักรไม่สามารถเข้ามาทำงานแทนมนุษย์ได้ทั้งหมด ซึ่งมนุษย์ยังคงต้องเป็นผู้ควบคุม ตัดสินใจ และแก้ปัญหา จึงทำให้เกิดแนวคิดที่จะทำให้เครื่องมือ เครื่องจักร สามารถตัดสินใจ เรียนรู้การแก้ปัญหาแทนมนุษย์ได้ ซึ่งเป็นที่มาของการเกิดสาขาวิชาที่เรียกว่า “ปัญญาประดิษฐ์”

ปัญญาประดิษฐ์ (A I : Artificial Intelligence) หมายถึง การทำให้คอมพิวเตอร์สามารถคิดหาเหตุผลได้ เรียนรู้ได้ทำงานได้เหมือนสมองมนุษย์ ปัญญาประดิษฐ์เป็นสาขาหนึ่งของคอมพิวเตอร์ที่เกี่ยวข้องกับการนำคอมพิวเตอร์ ทำให้คอมพิวเตอร์สามารถ ทำงาน ประมวลผลในลักษณะของการคิดหาเหตุผล การตัดสินใจใน การแก้ปัญหาได้เหมือนมนุษย์ การที่จะให้คอมพิวเตอร์ทำงานเหล่านี้ได้จะต้องพัฒนาคอมพิวเตอร์ให้มีความเร็วในการประมวลผลสูง และสามารถทำการประมวลผลได้อย่างมีประสิทธิภาพ ปัญญาประดิษฐ์นั้นจึงจะต้องมีความรู้อยู่ด้วย เมื่อนำความรู้มาประมวลผลด้วยระบบคอมพิวเตอร์ และระบบนี้ก็สามารถแก้ปัญหาได้เช่นเดียวกับมนุษย์ แต่การที่จะทำให้คอมพิวเตอร์มีความรู้ความสามรถในการประมวลผลความรู้ไม่ใช่เรื่องง่าย เพราะการแก้ปัญหาของมนุษย์ก็เป็นกระบวนการที่ซับซ้อน

เนื่องจากปัจจุบันโปรแกรมประยุกต์ทางด้านปัญญาประดิษฐ์กลายเป็นสิ่งที่นิยมใช้กันอย่างกว้างขวางผู้จัดทำจึงเกิดความสนใจที่จะศึกษาและพัฒนาปัญญาประดิษฐ์ขึ้น ซึ่งปัญญาประดิษฐ์ที่ผู้จัดทำเลือก เป็นการเรียนรู้ของเครื่องในรูปแบบของ การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) ซึ่งเป็นการเรียนรู้และตัดสินใจการกระทำจากประสบการณ์ที่ผ่านมา โดยผู้จัดทำได้สร้างแบบจำลองที่ใช้มีลักษณะเป็นแบบกระดาน ซึ่งเป็นเกมค้นหาเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดสุดท้าย โดยในการทดลองจะสร้างตัวเดินที่สามารถเรียนรู้จดจำและค้นหาเส้นทาง จนกระทั่งค้นพบ

เส้นทางที่สั้นที่สุดได้ ซึ่งภาษาที่ใช้ในการพัฒนาตัวเกมนั้น ผู้จัดทำเลือกใช้ภาษา C++ ในการเขียน เนื่องจากเป็นภาษาระดับสูง ประมวลผลเร็ว และง่ายต่อการจัดการกราฟิกต่างๆ

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาทฤษฎีการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)
2. เพื่อนำความรู้การเขียนโปรแกรมด้วยภาษา C++ มาพัฒนาแบบจำลอง
3. เพื่อนำทฤษฎีการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) มาประยุกต์ใช้ในแบบจำลองที่พัฒนาขึ้น

1.3 ขอบเขตของโครงงาน

1. ศึกษาทฤษฎีการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)
2. ศึกษาการเขียนโปรแกรมด้วยภาษา C++ โดยใช้ไลบรารี MFC
3. พัฒนาแบบจำลองเป็นเกมประเภทกระดาน เป็นเกมค้นหาเส้นทางที่สั้นที่สุด มีลักษณะกระดานขนาด 10x10 ช่อง

1.4 วิธีการดำเนินการ

1. ศึกษาทฤษฎีการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)
2. ศึกษาการเขียนโปรแกรมด้วยภาษา C++ และไลบรารี MFC
3. ออกแบบแบบจำลอง
4. นำกระบวนการเรียนรู้มาใช้ในแบบจำลอง
5. ทดลองและเก็บข้อมูล
6. นำข้อมูลสรุปผลและแนวทางการพัฒนาต่อ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้เรื่องทฤษฎีการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)
2. ได้แบบจำลองที่ถูกพัฒนาขึ้นมาด้วยภาษา C++
3. ได้แบบจำลองที่นำทฤษฎีการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) มาประยุกต์ใช้

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกัน คือ

บทที่ 1 บทนำ กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงานวิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)

บทที่ 3 การออกแบบและพัฒนา กล่าวถึงการออกแบบแบบจำลองและอัลกอริทึมที่ใช้ในการทดลองการเรียนรู้ของผู้เรียน

บทที่ 4 การทดลอง ผลการทดลอง และการเปรียบเทียบผล

บทที่ 5 บทสรุป กล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน ข้อจำกัด รวมถึงปัญหาอุปสรรคต่างๆ ของโครงงาน และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 2
ทฤษฎีบทที่เกี่ยวข้อง

2.1 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)

การเรียนรู้แบบเสริมกำลัง คือ การเรียนรู้ว่าจะต้องทำอะไร ให้เหมาะสมกับสถานการณ์ โดยผลของการกระทำจะให้ค่าตอบแทน (Reward) ในรูปแบบตัวเลข ผู้เรียนจะไม่ถูกบอกว่าจะต้องกระทำอะไรถึงจะได้ค่าตอบแทนที่มากที่สุด แต่ผู้เรียนต้องค้นพบการกระทำที่จะได้รับค่าตอบแทนค่าตอบแทนที่มากที่สุดจากการกระทำของผู้เรียนเอง การกระทำอาจไม่ได้ส่งผลต่อค่าตอบแทนในทันที แต่การค้นหา การทดลอง ข้อผิดพลาดและค่าตอบแทนล่าช้า ก็เป็นสิ่งสำคัญที่ทำให้การเรียนรู้แบบเสริมกำลังมีคุณสมบัติที่แตกต่างกันด้วย

การเรียนรู้แบบเสริมกำลัง ไม่ได้ถูกกำหนดการพัฒนาให้พัฒนาจากอัลกอริทึมการเรียนรู้ แต่ถูกพัฒนาจากปัญหาที่ได้เรียนว่า อัลกอริทึมใดที่เหมาะสมที่สุดในการแก้ปัญหา ที่เราตัดสินใจนำมาเป็นอัลกอริทึมการเรียนรู้แบบเสริมกำลัง ส่วนใหญ่รูปแบบของปัญหาการเรียนรู้แบบเสริมกำลังในแง่ของการควบคุมที่เหมาะสมเป็นกระบวนการตัดสินใจในรูปแบบของมาร์คอฟ (Markov Decision Process : MDP) ซึ่งแนวคิดพื้นฐานนั้นมีส่วนประกอบคือ ผู้เรียน (Agent), ชุดการกระทำ (Action) และ สภาพแวดล้อม (Environment) ซึ่งในการเรียนรู้แบบเสริมกำลังนั้น กระบวนการส่วนใหญ่จะเกี่ยวข้องกับความสัมพันธ์ระหว่าง ผู้เรียน และสภาพแวดล้อมเพื่อให้บรรลุเป้าหมาย ซึ่งผู้เรียนต้องสามารถรู้ขอบเขตและการกระทำที่เป็นไปได้ทั้งหมดที่จะส่งผลต่อสภาพแวดล้อมปัจจุบัน(State) หลังจากนั้น การกระทำที่ถูกเลือก จะมีผลทำให้ สภาพแวดล้อม เปลี่ยนแปลงไปและผู้เรียนก็จะได้รับค่าตอบแทน (Reward) ซึ่งขึ้นอยู่ว่าการกระทำดังกล่าวมีผลให้สภาพแวดล้อมเปลี่ยนแปลงไปในทางใด (ดีขึ้นหรือแย่ลง ถ้าดีขึ้นค่าตอบแทนอาจจะมาก แต่ถ้าแย่ลงค่าตอบแทนอาจจะน้อย) จากค่าตอบแทนที่ได้รับ ผู้เรียนจะพยายามค้นหา นโยบาย (Policy) ในการเลือก การกระทำ ในสภาพแวดล้อมใดๆ เพื่อให้ค่าตอบแทน ในระยะยาวนั้นมีค่ามากที่สุด

การเรียนรู้แบบเสริมกำลังนี้แตกต่างจากการเรียนรู้แบบมีผู้สอน (Supervised learning) ตรงที่ตัวผู้เรียนเองจะไม่ได้เรียนรู้จากชุดของข้อมูลตัวอย่าง (training set) แต่จะทำการโต้ตอบกับสภาพแวดล้อมที่ผู้เรียนกำลังทำงานอยู่โดยตรง ดังนั้นข้อมูลเดียวที่ผู้เรียนสามารถใช้ในการเรียนรู้ได้ก็คือค่าตอบแทนที่ได้รับเมื่อมีการเลือกการกระทำใดการกระทำหนึ่ง ซึ่งเราสามารถมองการเรียนรู้แบบนี้ได้เป็น การเรียนรู้แบบลองผิดลองถูก (trial-and-error) โดยส่วนใหญ่แล้วจะมีการนำการเรียนรู้แบบนี้ไปใช้ในงานที่เกี่ยวข้องกับการควบคุมหรือเกม เช่น การควบคุมหุ่นยนต์ หมากรุก เป็นต้น

2.2 องค์ประกอบของการเรียนรู้แบบเสริมกำลัง

2.2.1 สถานะ (State)

สถานะของปัญญาประดิษฐ์มีความจำเป็นมากในการตัดสินใจทำสิ่งต่างๆ โดยสถานะของปัญญาประดิษฐ์นั้นจะถูกตัดสินโดยสภาพแวดล้อมของปัญญาประดิษฐ์นั้นๆ ในสถานะจะประกอบด้วยค่าตอบแทน ค่าผลกำไร การกระทำต่างๆ และโอกาสในการทำการกระทำต่างๆ ซึ่งค่าเหล่านี้จะเป็นตัวตัดสินใจกำหนดการกระทำของปัญญาประดิษฐ์ ซึ่งจะกล่าวในหัวข้อถัดไป

2.2.2 ค่าตอบแทน (Reward)

ค่าตอบแทนจะได้มาจากการกระทำต่างๆ ในสถานะนั้นๆ เมื่อปัญญาประดิษฐ์ได้ทำการกระทำอย่างใดอย่างหนึ่งก็จะได้รับค่าตอบแทนมา และเปลี่ยนสถานะไปยังสถานะถัดไป

2.2.3 ค่าผลกำไร (Return)

ปัญญาประดิษฐ์จะทราบได้อย่างไรว่า ควรจะทำการกระทำใดจึงจะเกิดผลตอบแทนสูงสุด ซึ่งผลตอบแทนที่ต้องคำนึงนั้น ไม่ใช่แค่เพียงค่าตอบแทน (Reward) เพียงอย่างเดียว แต่ต้องคำนึงถึงค่าผลกำไร (Return) ด้วย

2.2.4 การกระทำ (Action)

ในหนึ่งสถานะสามารถของปัญญาประดิษฐ์มีได้หลายการกระทำ ซึ่งเมื่อทำการกระทำแล้วสิ่งที่ปัญญาประดิษฐ์จะได้รับก็คือค่าตอบแทนของการกระทำนั้น และไปสู่สถานะถัดไป โดยในการตัดสินใจทำการกระทำนั้นจะขึ้นอยู่กับความน่าจะเป็นของการกระทำนั้นๆ

2.2.5 ความน่าจะเป็นในการกระทำ (Policy)

ความน่าจะเป็นของการกระทำในสถานะหนึ่ง ความเป็นไปได้ในการกระทำ จะถูกเก็บไว้ในรูปแบบของเซตคู่อันดับ

$$\pi_t(s, a) \qquad (2.1)$$

ซึ่งจะบ่งบอกว่า การกระทำใดมีความน่าจะเป็นที่จะทำเท่าใด โดยอาจกำหนดเป็นค่าคงที่หรือตัวแปรก็ได้

2.2.6 ค่าของสถานะ (State Value, V^{π})

ในสถานะหนึ่ง จะมีค่าที่บ่งชี้โอกาสได้รับผลกำไรเก็บอยู่ ซึ่งหากค่านี้มีค่ามากนั่นหมายความว่าในสถานะนั้นๆ ปัญญาประดิษฐ์จะสามารถแสวงหาผลกำไรได้มากจากการกระทำต่างๆ ในสถานะนั้นๆ

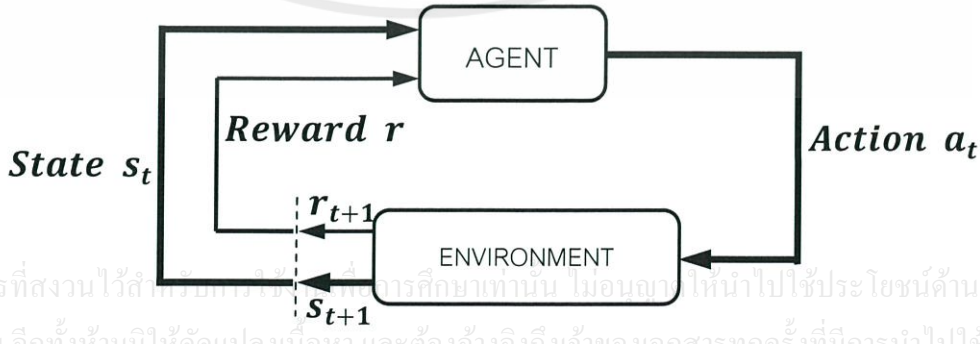
2.2.7 ค่าของการกระทำ (Action Value, Q^{π})

ค่าของการกระทำจะจะเป็นค่าที่บ่งบอกว่า ในการกระทำนั้นๆ จะมีโอกาสแสวงหาผลกำไรได้มากเพียงใด ซึ่งในสถานะหนึ่งสถานะ จะมีค่าของการกระทำหลายค่า เท่ากับจำนวนการกระทำที่สามารถทำได้ ซึ่งค่าของสถานะ (V^{π}) และค่าของการกระทำ (Q^{π}) นั้นรวมกันเรียกว่า ค่าผลกำไรที่คาดหวัง (Expect Return)

2.3 ปัญหาของการเรียนรู้แบบเสริมกำลัง (Reinforcement Leaning Problem)

โจทย์ปัญหาเกี่ยวกับการเรียนรู้แบบเสริมกำลังเปรียบเสมือนกับโครงสร้างพื้นฐานของปัญหาเกี่ยวกับการเรียนรู้ผ่านการโต้ตอบเพื่อให้บรรลุเป้าหมายที่ต้องการระบบที่ทำการตัดสินใจและเรียนรู้เรียกว่า ผู้เรียน (agent) ซึ่งเป็นระบบที่โต้ตอบกับสภาพแวดล้อม (environment) หรือระบบที่อยู่ภายนอกผู้เรียนการโต้ตอบระหว่างกันของผู้เรียนกับสภาพแวดล้อมจะเป็นไปอย่างต่อเนื่องจากการเลือกกระทำของผู้เรียนและการตอบสนองต่อการกระทำเหล่านั้นของสภาพแวดล้อมด้วยการให้ข้อมูลเกี่ยวกับสถานการณ์ปัจจุบัน

นอกจากนี้สภาพแวดล้อมยังเป็นสิ่งที่กำหนดรางวัลที่ผู้เรียนจะได้รับซึ่งเป็นผลของการตัดสินใจของผู้เรียนตลอดอายุการเรียนรู้ของรางวัลที่มีค่าสูงที่สุดคือสิ่งที่ผู้เรียนพยายามเรียนรู้และปรับตัวเพื่อให้ได้มารายละเอียดทั้งของสภาพแวดล้อมเป็นสิ่งที่กำหนดทาสก์ (Task) ซึ่งเป็นหนึ่งในองค์ประกอบของโจทย์ปัญหาเกี่ยวกับการเรียนรู้แบบเสริมกำลัง



รูปที่ 2.1 การโต้ตอบระหว่างผู้เรียนและสภาพแวดล้อมในการเรียนรู้แบบเสริมกำลัง

รูป 2.1 แสดงแผนภาพการโต้ตอบระหว่างผู้เรียนกับสภาพแวดล้อมตามลำดับขั้นเวลา t = 0, 1, 2,... ในแต่ละขั้นเวลา (time step) ผู้เรียนจะได้รับสภาพแวดล้อมปัจจุบันมาในรูปของ state, $s_t \in S$, โดยที่ S คือเซตของสถานะที่เป็นไปได้ทั้งหมด จากนั้นผู้เรียนรู้ต้องทำการเลือกการกระทำ $a_t \in A(s_t)$ หมายถึงเซตของการกระทำที่มีอยู่ของสถานะ s_t และจะได้รับผลตอบแทนเป็นเลขจำนวนหนึ่งกลับมา $r_t \in R$ และพาตัวเองไปยังสถานะใหม่ $(s_t + 1)$ ต่อไป การตัดสินใจเลือกการกระทำของผู้เรียนในแต่ละลำดับขั้นเวลาจะเป็นไปตามฟังก์ชัน เรียกว่า โพลิซี (policy) ซึ่งแทนด้วยสัญลักษณ์ π_t โดย $\pi_t(s, a)$ หมายถึงความน่าจะเป็นที่ $a_t = a$ เมื่อ $s_t = s$ วิธีการต่างๆ ในการเรียนรู้แบบเสริมกำลังจะกำหนดการปรับเปลี่ยนของผู้เรียนผ่านประสบการณ์ของผู้เรียนเอง เป้าหมายของผู้เรียนคือการให้ได้มาซึ่งค่าคาดหมายของผลตอบแทน (expected return) ที่สูงที่สุด ในที่นี้ผลตอบแทน R_1 หมายถึงฟังก์ชันของลำดับของรางวัลที่ได้รับหลังจากลำดับขั้นเวลา t ซึ่งสามารถเขียนได้ในรูปแบบของเรขาคณิต $r_t, r_{t+1}, r_{t+2}, \ldots$ ฟังก์ชัน R_1 สามารถเขียนให้อยู่ในรูปทั่วไป ดังนี้

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{k+t+1} \qquad (2.2)$$

เมื่อ γ เป็นค่าอัตราการลดค่า (discount rate) และT เป็นค่าของลำดับขั้นเวลาขั้นสุดท้าย ในกรณีที่ทาสก์ของผู้เรียนสามารถแบ่งเป็นเอพิโสด (episode) ได้ γ จะมีค่าเป็น 1 และ Tจะต้องมีค่าจำกัดซึ่งไม่เท่ากับ ∞ ในกรณีที่ทาสก์ของผู้เรียนเป็นแบบต่อเนื่อง γ จะมีค่าอยู่ในช่วง (0,1) และ Tจะมีค่าเข้าใกล้ ∞ เนื่องจากเป็นทาสก์ที่ไม่มีสถานะจบและไม่มีการเริ่มเอพิโสดใหม่

ตัวอย่างที่เป็นแบบเอพิโสด เช่น การแข่งขันเทนนิส อาจจะให้แต้มครั้งหนึ่งของการเล่นเป็นหนึ่งเอพิโสด ภายในเอพิโสดอาจจะมีสถานะว่าตอนนี้อยู่ที่จุดไหนของคอร์ด และการกระทำก็คือการการตีลูกไปลงจุดไหน เมื่อมีผลแพ้ชนะในแต้มนั้นก็จบเอพิโสดแล้วก็นำค่ารีเทิร์น (return) นี้อัพเดตแต่ละสถานะแล้วก็เริ่มเอพิโสดใหม่

ตัวอย่างที่เป็นแบบต่อเนื่อง เช่น หุ่นยนต์ที่คอยประคองไม้ที่ตั้งอยู่บนต้นไม้ไม่ให้ล้ม สถานะก็คือไม้เอียงไปทางไหนมากเท่าไหร่ ส่วนการกระทำก็คือต้องเลื่อนตัวหุ่นยนต์ไปทางไหน ระยะทางเท่าไหร่จึงจะทำให้ไม้ที่ตั้งอยู่ไม่ล้ม กรณีแบบนี้จะไม่มีวันจบ ไม่มีเอพิโสด ดังนั้นค่าที่จะต้องปรับไปเรื่อยๆ โดยค่า γ จะลดลงไปตามระยะทาง ซึ่งก็คือส่วนที่กำลังทำอยู่ จะได้รับค่ารีเทิร์นไปมากกว่าส่วนที่อยู่หลังๆ ที่ผ่านมานานแล้ว พอไกลมากๆ ก็จะได้รับค่ารีเทิร์นที่เข้าใกล้ศูนย์ หรือเรียกได้ว่าไม่ได้รับค่ารีเทิร์นเลย

2.4 กระบวนการตัดสินใจแบบมาร์คอฟ (Markov Property)

ในการศึกษาการเรียนรู้แบบเสริมกำลัง (Reinforcement learning) ผู้เรียนจะทำการตัดสินใจเลือกการกระทำ โดยขึ้นอยู่กับสภาพปัจจุบันของสภาพแวดล้อมปัจจุบัน ที่เรียกว่าสภาพแวดล้อมปัจจุบัน (environment) และคุณสมบัติทุกอย่างของสภาพแวดล้อมปัจจุบันที่ผู้เรียนรู้ให้ความสนใจ จะถูกเรียกอย่างทางการว่า Markov Property

สภาพแวดล้อมปัจจุบัน โดยปกติแล้วจะหมายถึงเหตุการณ์หรือสิ่งที่เซน (Sense) ของผู้เรียนรู้ รับรู้ได้ในปัจจุบัน แต่บางครั้งมันก็สามารถถูกประกอบไปด้วยข้อมูลในอดีตด้วยก็ได้ เช่น การที่เราได้ยินคำตอบว่า “ใช่” ตอบกลับมา การที่เราจะสามารถแยกแยะได้ว่ามันอยู่ในสภาพปัจจุบันไหนนั้นขึ้นอยู่กับข้อมูลในอดีต ซึ่งก็คือคำถามที่เราถามไปก่อนหน้าด้วย

ในทางตรงกันข้าม สภาพแวดล้อมปัจจุบัน ไม่ควรจะมีข้อมูลทุกๆอย่างเสมอไป แม้สิ่งนั้นจะมีประโยชน์และช่วยให้ผู้เรียนเลือกการกระทำนั้น ได้อย่างเหมาะสมขึ้นก็ตาม เช่น ถ้าผู้เรียนกำลังเล่นไพ่อยู่ เราต้องไม่คาดหวังว่า ผู้เรียนจะรู้ไพ่ใบต่อไปที่อยู่บนโต๊ะ เนื่องจากข้อมูลนั้นถือว่าเป็นข้อมูลที่ถูกซ่อนไว้ของสภาพแวดล้อมปัจจุบัน

สภาพแวดล้อมปัจจุบันที่มีการรวมเอาข้อมูลสารสนเทศ (Information) ในอดีตทั้งหมด และมาสรุปเป็นข้อมูลสารสนเทศของสภาพแวดล้อมปัจจุบันได้นั้น เราจะเรียกสภาพแวดล้อมปัจจุบันนั้นว่า Markov หรือเรียกว่า Markov Property เช่น การเล่นหมากรุก ตำแหน่งปัจจุบันของหมากทุกตัวบนกระดานก็ถือว่าเป็นข้อมูลที่เพียงพอแล้ว ถือว่าเป็น Markov state เพราะถือว่าแม้มันไม่ได้มีข้อมูลของประวัติเก่าอยู่เลย แต่มีเพียงข้อมูลที่มีอยู่ปัจจุบัน ก็เพียงพอให้ผู้เรียนเลือก และตัดสินใจได้แล้ว

เราสามารถแทน Markov Property ของปัญหาของการเรียนรู้แบบเสริมกำลังสมการทางคณิตศาสตร์อย่างง่ายได้ โดยสมมติให้มีสถานะและค่าของผลตอบแทนที่จำกัดได้ ซึ่งกรณีปกติเราจะได้สมการความน่าจะเป็นที่ต้องอ้างถึงเวลาในอดีตด้วย ดังนี้

$$P_r\{s_{t+1} = s', r_{t+1} = r \,|s_t, a_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0|,\} \qquad (2.3)$$

แต่ถ้าสถานะมี Markov Propertyจะทำให้สิ่งแวดล้อมปัจจุบันที่ตอบสนองที่เวลา $t+1$ จะขึ้นกับเพียงสถานะและการกระทำของเวลา t เท่านั้น จึงทำให้เราสามารถแทนสมการข้างต้นได้ด้วยสมการดังนี้

$$P_r\{s_{t+1} = s', r_{t+1} = r|s_t, a_t|,\} \qquad (2.4)$$

เราสามารถสรุปได้ว่า ถ้าสภาพแวดล้อมปัจจุบันมี Markov Property จะทำให้เราสามารถอาศัยเพียง one-step dynamics ($r|s_t, a_t|$) (next state) และผลตอบแทนที่คาดหวังต่อไป (next expected reward) เพื่อหาการกระทำที่เหมาะสมได้แล้ว สรุปนั่นคือ Markov State จะช่วยในการเลือกการกระทำที่เป็นไปได้ และเหมาะสมที่สุดของแต่ละสถานะ ได้โดยอาศัยการสรุปประวัติเหตุการณ์โดยรวม มารวมไว้เป็นข้อมูลของสถานะปัจจุบัน ซึ่งถือว่าเป็น policy ที่ดีและเหมาะสมที่สุด แม้บางครั้งสถานะจะไม่เป็น Markov Property แต่หากเป็นไปได้ เราก็ควรที่จะพยายามแปลงสถานะนั้นให้อยู่ในรูป Markov State โดยเราอาจจะทำการสรุปเฉพาะข้อมูลที่สำคัญๆ เท่าที่หาได้มาเป็นข้อมูลสารสนเทศของสถานะปัจจุบันก็พอ เพราะหากอยู่ในรูปของ Markov จะทำให้ประสิทธิภาพของระบบดีขึ้นด้วย

Markov Propertyถือว่ามีส่วนสำคัญมากต่อปัญหาของการเรียนรู้แบบเสริมกำลัง เนื่องจากการตัดสินใจและการเลือกทำการกระทำ จะอาศัยเพียงข้อมูลจากสถานะปัจจุบันเท่านั้น ทำให้มันสามารถใช้เป็นพื้นฐานในการนำไปประยุกต์ใช้กับปัญหาต่างๆ ทั้ง Markov และ non Markov ทีมีความซับซ้อนได้เป็นอย่างดี

2.5 ฟังก์ชันมูลค่า

ฟังก์ชันมูลค่า คือฟังก์ชันที่ใช้ในอัลกอริทึมเกี่ยวกับการเรียนรู้แบบเสริมกำลังในการประเมินมูลค่าหรือประโยชน์ของการอยู่ในสถานะที่กำหนด หรือประโยชน์ของการเลือกการกระทำใดๆ ในสถานะที่กำหนด ซึ่งเป็นฟังก์ชันของสถานะ หรือฟังก์ชันของคู่ของสถานะและการกระทำซึ่งมูลค่าในที่นี้กำหนดโดยค่าคาดหมายของผลตอบแทนและแปรเปลี่ยนไปตามโพลิซีที่ใช้มูลค่าของสถานะ $s \in S$ ภายใต้โพลิซี π ซึ่งแทนด้วยสัญลักษณ์ $V_{\pi(s)}$ หมายถึงค่าคาดหมายของผลตอบแทน ซึ่งนับเริ่มต้นจากสถานะ s และดำเนินไปตามโพลิซี π ซึ่งเป็นตัวกำหนดความน่าจะเป็น $\pi_{(s,a)}$ ในการเลือกการกระทำ $a \in A(s)$ เมื่ออยู่ในสถานะ s การคำนวณค่า $V_{\pi(s)}$ สามารถเขียนในรูปของสมการสำหรับ MDP ได้ดังนี้

$$V^{\pi}(s) = E_{\pi(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s)} \qquad (2.5)$$

เมื่อ E_{π} หมายถึง ค่าคาดหมายของผลตอบแทนเมื่อกำหนดให้ผู้เรียนดำเนินตามโพลิซี π และ t หมายถึงลำดับขั้นเวลาใดๆในกรณีที่ทาสก์ (task) มีสถานะสิ้นสุดมูลค่าของสถานะสิ้นสุดจะมีค่าเป็นศูนย์เสมอ ฟังก์ชัน V_{π} นี้เรียกว่าฟังก์ชันมูลค่าของสถานะตามโพลิซี π (state-value function for policy) ในทำนองเดียวกัน มูลค่าของการเลือกกระทำ a ในสถานะ s ภายใต้โพลิซี π การ

คำนวณค่า $Q_{\pi(s,a)}$ หมายถึงค่าคาดหมายของผลตอบแทน ซึ่งนับเริ่มต้นจากสถานะ s ผ่านการตัดสินใจเลือกการกระทำของ a และดำเนินไปตามโพลิซี π การคำนวณค่า $Q_{\pi(s,a)}$ สามารถเขียนในรูปของสมการสำหรับ MDP ได้ดังนี้

$$Q^{\pi}(s,a) = E_{\pi}(R_t|s_t = s, a_t = a) = E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a) \qquad (2.6)$$

ฟังก์ชัน Q_{π} นี้เรียกว่าฟังก์ชันมูลค่าของการกระทำของโพลิซี π (Action-value function for policy π)

รูปที่ 2.2 แบ็คอัพค่าผลตอบแทนสำหรับ (a) V_{π} และ (b) Q_{π}

รูป 2.2 แบ็คอัพค่าผลตอบแทนสำหรับ (a) V_{π} และ (b) Q_{π} โหนดบนสุดคือรูท (root) ด้านล่างคือความเป็นไปได้ของการกระทำและสถานะทั้งหมดที่ปรากฏ จะเห็นว่าค่าของ V_{π} จะเป็นตัวบอกว่าสถานะนั้นดีหรือเปล่า ส่วนค่า Q_{π} จะเป็นตัวบอกว่าที่สถานะนี้ทำการกระทำนี้ดีหรือเปล่า

2.6 การประเมินค่าตอบแทน

คุณสมบัติสำคัญที่สุดของการเรียนรู้แบบเสริมกำลังแตกต่างจากการเรียนรู้ประเภทอื่นๆ คือ ใช้ข้อมูลที่ได้การฝึกอบรมมาประเมินการกระทำมากกว่าการสั่งให้กระทำในสิ่งที่ถูก นี่คือสิ่งที่สร้างความจำเป็นในการสำรวจสำหรับการทดลองและค้นหาข้อผิดพลาดเพื่อการกระทำที่ดีโดยทั่วไปวิธีการตามแนวทางการเรียนรู้แบบเสริมกำลังจะมุ่งเน้นไปที่การแก้ปัญหาสองเรื่องที่มีความสัมพันธ์กันคือ ปัญหาการทำนายค่า (prediction problem) ซึ่งเป็นการเรียนรู้ฟังก์ชันมูลค่าเมื่อดำเนินไปตามโพลิซีใดๆและปัญหาการควบคุม (control problem) ซึ่งเป็นการปรับเปลี่ยนหรือค้นหาโพลิซีเพื่อให้ได้มาซึ่งโพลิซีที่เหมาะสมที่สุด สิ่งที่สำคัญสำหรับการได้มาซึ่งได้ผลตอบแทนที่สูงที่สุดในการเรียนรู้แบบ

เสริมกำลังคือการปรับสมดุลระหว่างการสำรวจและการใช้ประโยชน์จากความรู้เดิม ซึ่งเป็นผลจากการตัดสินใจเลือกการกระทำของผู้เรียน วิธีการเลือกการกระทำแบบ เอพซิลอน-กรีดี (ε-greedy) เป็นวิธีหนึ่งที่เป็นที่นิยมที่สุดในวิธการตัดสินใจเลือกการกระทำในการเรียนรู้แบบเสริมกำลัง ซึ่งใช้การเลือกการกระทำที่มีมูลค่าที่สูงที่สุดเป็นส่วนใหญ่ เพื่อใช้ประโยชน์จากความรู้ที่มีอยู่เดิม แต่ในบางครั้งจะใช้การสุ่มเลือกการกระทำเพื่อสำรวจการกระทำอื่นๆไปด้วย โดยกำหนดให้ความน่าจะเป็นในการสุ่มเลือกการกระทำที่มีมูลค่าสูงที่สุดมีค่าเป็น $1-\varepsilon+\varepsilon/|A(s)|$ และความน่าจะเป็นในการสุ่มเลือกการกระทำอื่นๆมีค่าเป็น $\varepsilon/|A(s)|$ เมื่อ A หมายถึงจำนวนการกระทำที่เป็นไปได้ และ ε เป็นอัตราการสำรวจ ซึ่งมักกำหนดด้วยค่าน้อยๆ แต่ไม่เท่ากับศูนย์

$$\pi(s,a) = \begin{cases} 1-\epsilon+\frac{\epsilon}{A(s)} & if\ a = a^* \\ \frac{\epsilon}{A(s)} & if\ a \neq a^* \end{cases} \qquad (2.7)$$

โดยที่ a^* คือ การกระทำที่ให้ค่าตอบแทนสูงที่สุด

2.7 Monte Carlo Method

เป็น Method ที่ใช้ในการประมาณหาค่า value function (V^{π} หรือ Q^{π}) และหาค่า optimal policies แต่ method นี้ ไม่ได้คาดหวังว่าจะต้องรู้ทุกสถานะ(state)ที่เป็นไปได้ หรือสิ่งแวดล้อมทั้งหมดของระบบ Monte Carlo Method อาศัยเพียงประสบการณ์ หรือสถานะบางสถานะ บางการกระทำ และบางค่าตอบแทนที่ได้รับมาจากการกระทำเหล่านั้นเท่านั้น ซึ่งเป็นประสบการณ์ที่ได้จากการกระทำนั้นทันทีแบบออนไลน์ (on-line) เมื่อประสบการณ์ที่ได้รับจากการจำลองสถานการณ์หรือจำลองสิ่งแวดล้อมแล้วเรียนรู้

การเรียนรู้จากประสบการณ์ คือ การเรียนรู้จากการกระทำโดยตรงกับสิ่งแวดล้อม ผู้เรียนไม่ต้องมีความรู้เบื้องต้นเกี่ยวกับการเปลี่ยนแปลงไปของสภาพแวดล้อม แต่ก็สามารถหา optimal policy ได้เช่นเดียวกับการเรียนรู้ (จากสภาพแวดล้อมปัจจุบันจำลองก่อน) จากประสบการณ์จำลอง ถึงแม้ว่ายังต้องการต้นแบบ(model) ต้นแบบก็เพียงแค่แสดงการเปลี่ยนสถานะบางสถานะ ไม่ได้แสดงการเปลี่ยนไปยังสถานะต่างๆทั้งหมดทุกสถานะที่เป็นไปได้ อย่างเช่นใน Dynamic Programming Method

Monte Carlo Method เป็น method ที่แก้ปัญหาทางการเรียนรู้แบบเสริมกำลังโดยการหาค่าเฉลี่ยของ return เป็นการรับประกันว่าจะได้รับค่า return กลับมา Monte Carlo มักจะกำหนดสำหรับ episodic task ซึ่งถือว่าการเรียนรู้แต่ละครั้งคือแต่ละ experience สำหรับหนึ่งเอพิโสดและ

ทุกๆ เอพิโสดจะต้องมีสถานะสุดท้าย(terminal state) ไม่ว่าในตอนนั้นจะเลือกการกระทำใดบ้างก็ตาม ค่าโดยปริมาณและ policies จะเปลี่ยนแปลงไปตาม ตอนค่า Value จะเปลี่ยนแปลงหลังจบแต่ละตอนเท่านั้น ไม่ได้เปลี่ยนทุกๆครั้งหลังกระทำหรือได้รับค่าตอบแทนเช่นmethodอื่นๆของการเรียนรู้แบบเสริมกำลัง

คำว่า Monte Carlo มักหมายถึง method สำหรับหาค่าเฉลี่ยของค่าที่ได้รับจากการสุ่ม ในที่นี้ Monte Carlo Method เป็น method ที่หาค่าเฉลี่ยของค่า return ที่ได้มาแบบสุ่ม จากการกระทำใดๆในตอนนั้นๆ ถึงแม้ว่า Monte Carlo และ DP. Method จะแตกต่างกัน แต่ทั้งสอง method ก็มีจุดมุ่งหมายเดียวคือหาค่า value function (V^{π} หรือ Q^{π}) และหา optimal policy สำหรับเลือกกระทำให้ได้การกระทำที่ก้อให้เกิดค่า return กลับมามากที่สุด

2.7.1 Monte Carlo Estimation of Action

ถ้าไม่มีแบบการคำนวณคิดการกระทำจะดีกว่าคิดจากสถานะ เพราะการมีต้นแบบคำนวณจากสถานะอย่างเดียวก็สามารถใช้หา policy ที่เหมาะสมได้โดยการมองไปข้างหน้า 1 ขั้น ว่าทำการกระทำใดแล้วให้ค่าตอบแทนและ Value ของสถานะถัดไปกลับมาสูงสุด แต่การไม่มีต้นแบบการคำนวณค่าของสถานะอย่างเดียว ไม่เพียงพอต่อการที่จะหาค่า Value สำหรับแต่ละการกระทำ เพื่อใช้ค่า valueที่ได้ในการหา policy ที่เหมาะสมที่สุด หรือเพื่อใช้ค่า Value ที่ได้เป็นแนวทางในการเลือกการกระทำที่ดีที่สุด ดังนั้นหนึ่งในเป้าหมายสำคัญของ Monte Carlo คือ ประมาณค่า Q^{π}

ปัญหาของการคำนวณค่า policy สำหรับคำนวณค่าการกระทำ Value คือ $Q^{\pi}(s,a)$ เป็นค่าตอบแทนที่คาดหวังที่ได้จากการเริ่มต้น สถานะการกระทำ a แล้วหลังจากนั้นก็ดำเนิน policy เช่นเดียวกับ Monte Carlo Method สำหรับประมาณค่า state value ที่แบ่งได้ 2 method

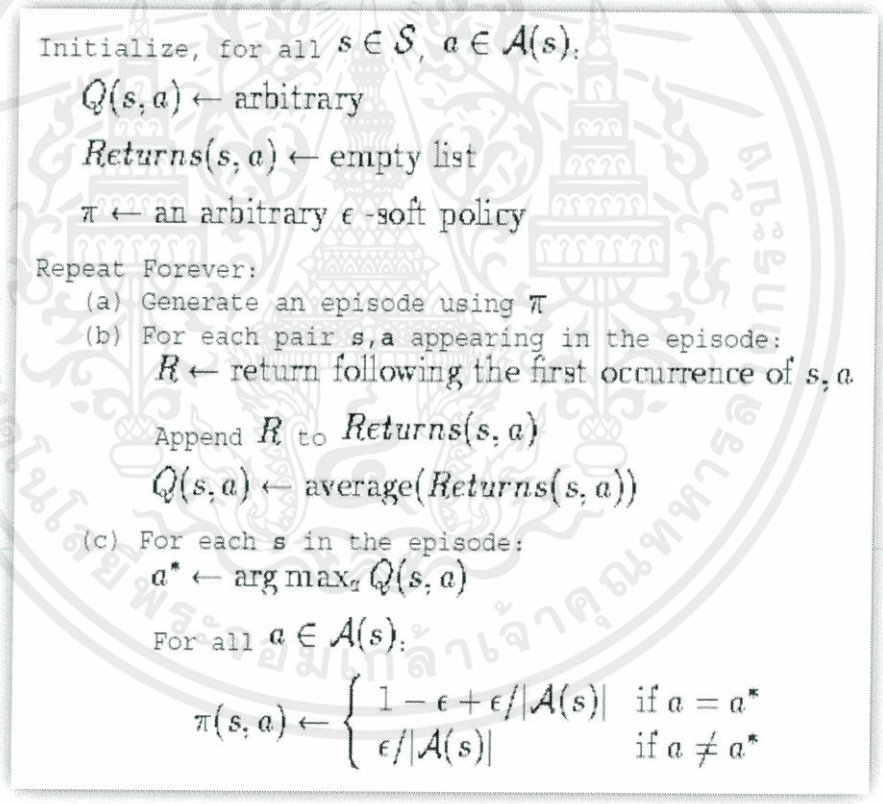
1. Every-visit MC method : ประมาณค่า value ของ state-action pair เฉลี่ยค่า Return ที่ได้ทุกๆ return ที่ได้หลังจากการ visit state นั้นแล้วทำการกระทำนั้น
2. First-visit MC method : เฉลี่ยค่า Return ที่ได้จากการ first-visit state นั้นแล้วคอยการกระทำนั้นในแต่ละสถานะ

ทั้งสอง Method ลู่เข้าสู่ค่าที่คาดหวัง ถ้าจำนวนในการพบแต่ละ state-action pair เข้าใกล้อนันต์

ปัญหาที่สำคัญอย่างหนึ่งสำหรับการคำนวณค่าโดยใช้ $Q^{\pi}(s,a)$ คือถ้า π เป็น greedy policy การเลือกการกระทำตาม policy π จะทำให้แต่ละ state เลือกทำการกระทำใดๆ อยู่เพียงการกระทำนั้นการกระทำเดียวตลอดทุกๆตอนทำการกระทำอื่นๆในสถานะที่ไม่ถูกเลือกและไม่มีค่าเฉลี่ย Return มาหาค่า value ของ state-action pair นั้นทำให้จุดประสงค์ของการหา

ค่าประมาณของทุกๆการกระทำ policy evaluation สำหรับค่าการกระทำเพื่อที่จะแก้ปัญหาที่บางสถานะการกระทำที่ไม่ถูกพบ เราต้องหาเส้นทางใหม่บ้างโดยทำได้ 2 วิธี

1. Exploring start : ในการเริ่มต้นแต่ละตอนให้ทุกคู่ของการกระทำ มีความน่าจะเป็น (probability) ที่จะถูกเลือกเป็นคู่ของการกระทำ เป็นการรับประกันว่าทุกๆคู่ของการกระทำจะถูกพบและเริ่มแต่นี้บางครั้งก็ใช้งานได้ดี แต่ไม่สามารถใช้ได้กับทุกๆกรณี โดยเฉพาะการเรียนรู้โดยตรงจากการโต้ตอบกับสิ่งแวดล้อม ในกรณีเช่นนี้การกำหนดเงื่อนไขตั้งต้นไม่ค่อยได้ผล
2. เป็นวิธีที่ใช้ได้โดยทั่วไป นั่นคือ เปลี่ยน policy ไปเรื่อยๆ (stochastic policy) และความน่าจะเป็นในการเลือกแต่ละ state-action pair ไม่เท่ากับศูนย์



Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s,a) \leftarrow$ arbitrary

$Returns(s,a) \leftarrow$ empty list

$\pi \leftarrow$ an arbitrary ϵ-soft policy

Repeat Forever:

(a) Generate an episode using π

(b) For each pair s,a appearing in the episode:

$R \leftarrow$ return following the first occurrence of s,a

Append R to $Returns(s,a)$

$Q(s,a) \leftarrow \text{average}(Returns(s,a))$

(c) For each s in the episode:

$a^* \leftarrow \arg\max_a Q(s,a)$

For all $a \in \mathcal{A}(s)$:

$$\pi(s,a) \leftarrow \begin{cases} 1-\epsilon+\epsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \epsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

รูปที่ 2.3 An E-soft on-policy Monte Carlo control algorithm

2.8 Temporal Difference Learning(TD)

เป็นวิธีการที่ประสมประสานระหว่างมอนติคาโลกับไดนามิกโปรแกรมมิ่ง TD เป็นวิธีการเรียนรู้ที่ไม่ขึ้นกับรูปลักษณะของสิ่งแวดล้อม แต่จะสนใจในเรื่องของการปรับปรุงประสบการณ์

ตลอดเวลาโดยไม่ต้องรอการบูทสแตรป (bootstrap) สิ่งที่สนใจในเนื้อหาส่วนของ TD คือ policy หรือการ Prediction Problem เพื่อใช้ในการควบคุมปัญหาจาก state ของสิ่งแวดล้อมที่ไม่คงที่ ทุกครั้งที่ผู้เรียนพบกับ state นั้น ก็จะมีการปรับปรุงข้อมูลทันที

2.8.1 Q()

เป็นกระบวนการที่ควบคุมปัญหาบน TD โดยการคำนวณค่า Policy (Q) โดยคำนวณเป็นคู่ตามลำดับของ state ปัจจุบันกับ state ถัดไป ค่า

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \quad (2.8)$$

แอลฟา คือค่าคงที่ของมัลติคาโร เป็นตัวบอกขนาดพารามิเตอร์ที่จะบูทสแตรป ค่าแกรมมา ที่เหมาะสม ควรมีค่าระหว่างศูนย์ถึงหนึ่ง

บทที่ 3
การออกแบบ

3.1 การออกแบบแบบจำลอง

3.1.1 ลักษณะแบบจำลอง

รูปแบบของแบบจำลองเป็นลักษณะเกมประเภทกระดาน ที่ค้นหาเส้นทางที่สั้นที่สุด โดยตัวเดินจะต้องเริ่มเดินจากจุดเริ่มต้น เพื่อไปยังจุดสุดท้าย โดยจะต้องใช้เส้นทางที่มีระยะทางสั้นที่สุด และตัวเดินไม่สามารถเดินผ่านกำแพงได้

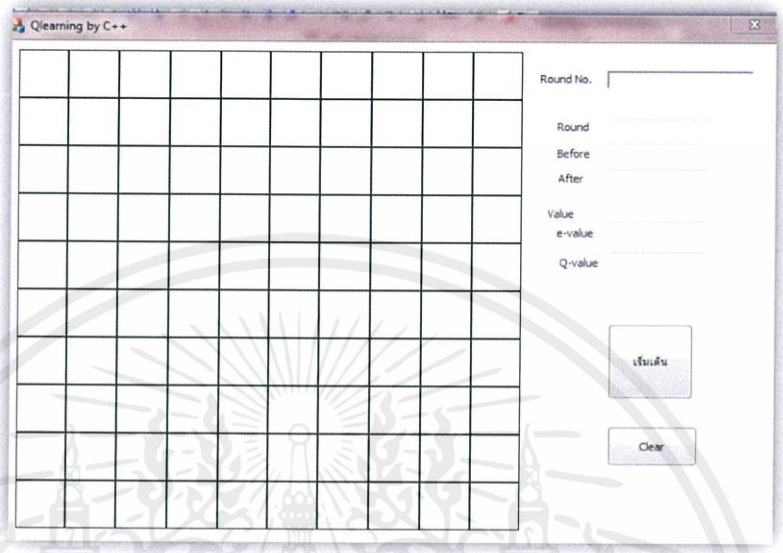
3.1.2 รูปแบบกระดาน

กระดานมีขนาด 10x10 ช่อง โดยจุดเริ่มต้น และจุดสุดท้าย จะเป็นช่องสีเขียว และช่องสีแดงตามลำดับ ส่วนกำแพงจะเป็นช่องสีดำ ซึ่งเป็นช่องที่ไม่สามารถเดินผ่านได้ และช่องสีขาวคือช่องที่ตัวเดินสามารถเดินผ่านได้

3.1.3 วิธีการใช้แบบจำลอง

1. เลือกกำหนดจุดเริ่มต้น จุดสุดท้ายและกำแพง ซึ่งในส่วนนี้สามารถกำหนดเอง
2. กรอกจำนวนรอบที่เราต้องการให้ผู้เรียนเรียนรู้ที่ช่อง “Round”
3. กรอกจำนวนรอบที่เราต้องการเห็นผู้เรียนเดินในช่วงแรกที่ช่อง “Before”
4. กรอกจำนวนรอบที่เราต้องการเห็นผู้เรียนเดินในช่วงท้ายที่ช่อง “After”
5. กรอกค่า e-value
6. กรอกค่า Q-value
7. จากนั้นกดปุ่ม “เริ่มเดิน”

3.2 การออกแบบหน้าต่างโปรแกรม

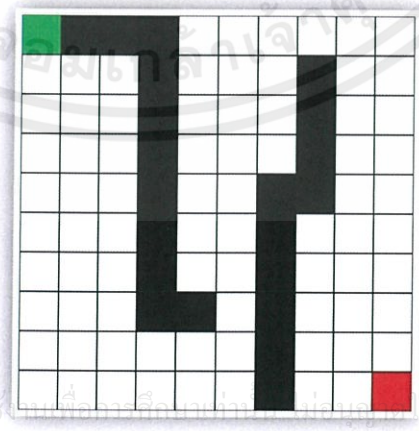


รูปที่ 3.1 หน้าต่างโปรแกรม

ในการออกแบบหน้าต่างโปรแกรม ก็จะประกอบด้วยกระดานเกม และปุ่มกดต่างๆ ดังนี้

3.2.1 ส่วนกระดานเกม

ส่วนกระดานเกม ส่วนนี้ก็จะประกอบไปด้วย จุดเริ่มต้น (สีเขียว), จุดสุดท้าย (สีแดง), กำแพง (สีดำ) และส่วนของช่องที่ด้วยเดินสามารถเดินได้ คือ ช่องสีขาวที่เหลือ ดังที่แสดงในรูปที่ 3.2



รูปที่ 3.2 กระดานเกม

3.2.2 ส่วนแสดงจำนวนรอบที่ผู้เรียนกำลังเรียน

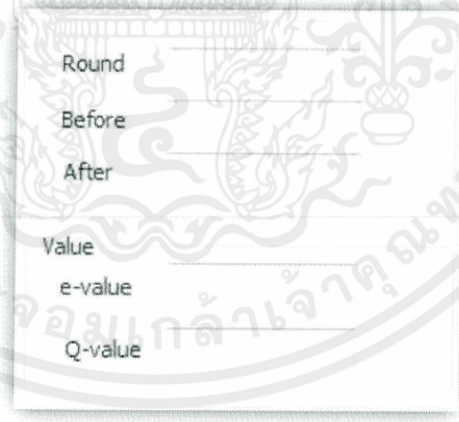
เป็นส่วนแสดงจำนวนรอบ ว่าขณะนี้ผู้เรียนเรียนเป็นรอบที่เท่าไร

รูปที่ 3.3 ส่วนแสดงจำนวนรอบที่ผู้เรียนกำลังเรียน

3.2.3 ส่วนของช่องที่เราต้องกรอก

ส่วนต่างๆ ดังที่แสดงในรูปที่ 3.4 ประกอบด้วย

1. “Round” เป็นช่องกรอกจำนวนรอบที่เราต้องการให้ผู้เรียนเรียนรู้
2. “Before” เป็นช่องกรอกจำนวนรอบที่เราต้องการเห็นผู้เรียนเดินในช่วงแรก
3. “After” เป็นช่องกรอกจำนวนรอบที่เราต้องการเห็นผู้เรียนเดินในช่วงท้าย
4. “e-value” เป็นช่องกรอกค่า e-value คือค่า ε-greedy
5. “Q-value” เป็นช่องกรอกค่า Q-value คือค่า Q เริ่มต้น



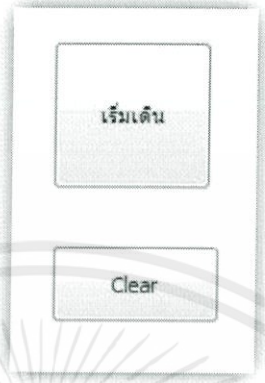
รูปที่ 3.4 ส่วนแสดงช่องให้กรอกข้อมูลต่างๆ

3.2.4 ส่วนของปุ่มกด

ส่วนต่างๆ ประกอบด้วย

1. ปุ่ม “เริ่มเดิน” กดปุ่มนี้เพื่อเริ่มเล่นเกม โดยการให้ปัญญาประดิษฐ์เป็นผู้เล่น และทำการเรียนรู้หาเส้นทางที่สั้นที่สุดเอง

2. ปุ่ม “Clear” กดปุ่มนี้เพื่อลบจุดเริ่มต้น จุดสุดท้าย และกำแพงในตารางทั้งหมด

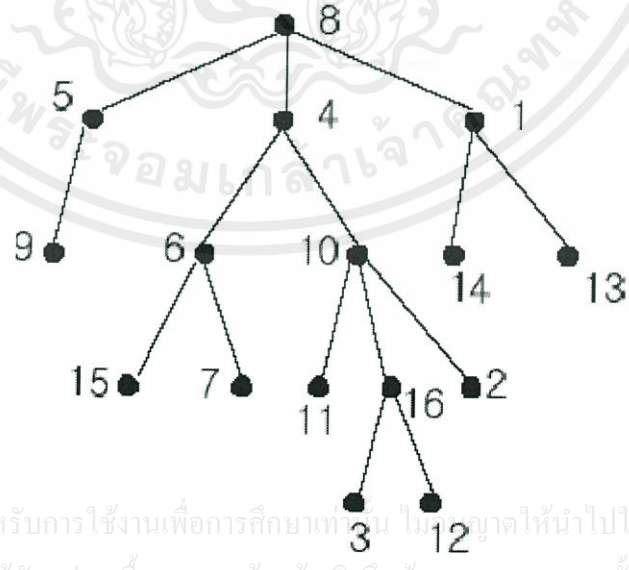


รูปที่ 3.5 ส่วนแสดงปุ่มกด

3.3 การหาเส้นทางและการตัดสินใจ

3.3.1 การเลือกเส้นทางโดยใช้ ε-greedy

ในการทดลองเป็นการหาเส้นทางจากโมเดลประสบการณ์ที่ผู้เรียนรู้ได้กระทำมาโดยผู้เรียนรู้จะเลือกกระทำกับโหนดที่มีค่าสูงที่สุดโดยสร้างทางเลือกที่เหมาะสมที่สุดในแต่ละขั้น เพื่อหาคำตอบที่เหมาะสมที่สุดในแต่ละสถานการณ์



รูปที่ 3.6 โครงสร้างต้นไม้ในการเลือกเส้นทางของผู้เรียนรู้

จากรูป เมื่อผู้เรียนรู้อยู่ตรงจุด State ที่มีค่า Q-value เท่ากับ 10 ก็จะมีการสำรวจเส้นทาง พร้อมทั้งหาค่า Q ของลูกสามโหนด ซึ่งจะเห็นได้ว่า โหนดที่มีค่า Q-value 16 มีค่ามากที่สุดก็จะเลือกไปตามเส้นทางนั้น

3.3.2 การตัดสินใจในการเลือกเส้นทาง

เนื่องจากโหนดบางโหนดเป็นโหนดที่ขาดประสบการณ์ หรือผู้เรียนต้องการสำรวจ เพราะฉะนั้นจึงต้องมี Policy เป็นตัวควบคุมพฤติกรรมของผู้เรียน โดยกำหนดให้เป็น 90 และ 10 หากผู้เรียนสุ่มได้น้อยกว่า 90 ผู้เรียนจะเลือกเส้นทางที่มีค่า Q มากที่สุด แต่หากไม่ใช่ ผู้เรียนจะนำทุกเส้นทางมาสุ่มและเลือกเดินเส้นทางนั้น

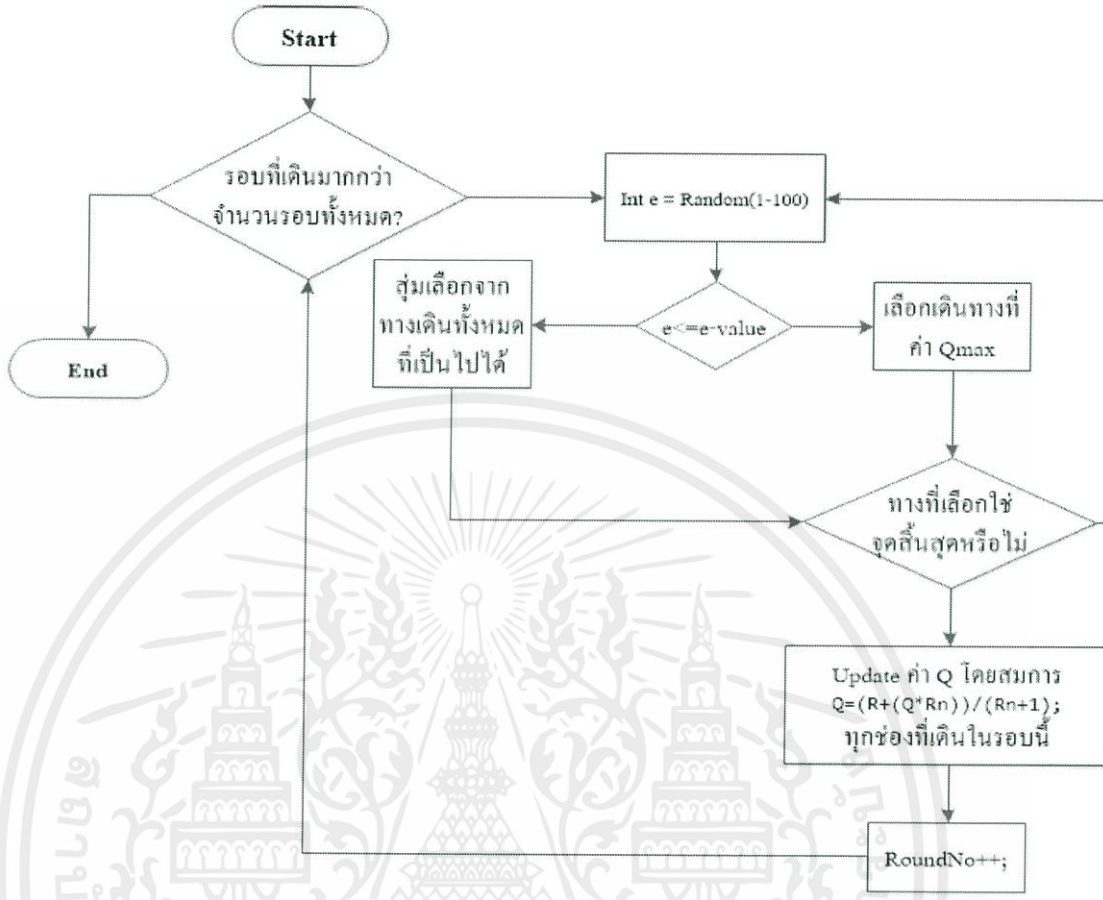
3.4 การออกแบบอัลกอริทึม

ในส่วนของการออกแบบอัลกอริทึม ผู้จัดทำได้เลือกใช้อัลกอริทึมในการทดลองทั้งหมด 3 วิธี ได้แก่ Monte Carlo, Qlearning และ Watkins's Q() โดยรายละเอียดการออกแบบเป็นดังนี้

3.4.1 อัลกอริทึม Monte Carlo

อัลกอริทึม Monte Carlo เป็นการเรียนรู้โดยใช้ชุดข้อมูลจากประสบการณ์ที่ผ่านมาใช้วิเคราะห์ และตัดสินใจ การอัพเดตค่าต่างๆ จะเกิดขึ้นเมื่อผู้เรียนเดินจากจุดเริ่มต้นจนกระทั่งถึงจุดสุดท้าย นั่นคือการอัพเดตค่าจะเกิดขึ้นเมื่อมีเดินจนสิ้นสุดเส้นทาง ลำดับของการกำหนดค่าเริ่มต้นและการอัพเดตค่า เป็นดังนี้

1. กำหนดค่าเริ่มต้นให้ค่า Q สำหรับทุกค่า โดยในเราสามารถกรอกค่าเริ่มต้นนี้ในหน้าโปรแกรมที่ช่อง Q-value และสร้างอาร์เรย์เก็บค่า Return โดยเริ่มต้นให้เป็นศูนย์
2. เริ่มเดินจากจุดเริ่มต้น แล้วทำการเลือกเส้นทางโดยการสุ่มค่า 1-100 หากสุ่มได้ 1-90 ตัวเดินจะเลือกเดินไปในช่องที่มีค่า Q สูงที่สุด หากสุ่มได้ 91-100 ตัวเดินจะเลือกสุ่มทางเดินจากเส้นทางทั้งหมดที่สามารถเดินไปได้
3. เลือกเดินไปช่องที่เลือกได้ จากนั้นทำการเลือกเส้นทางโดยการสุ่มค่า 1-100 หากสุ่มได้ 1-90 ตัวเดินจะเลือกเดินไปในช่องที่มีค่า Q สูงที่สุด หากสุ่มได้ 91-100 ตัวเดินจะเลือกสุ่มทางเดินจากเส้นทางทั้งหมดที่สามารถเดินไปได้ ทำข้อนี้วนซ้ำจนกว่าจะเจอว่าช่องที่เลือกนั้นเป็นจุดสุดท้าย
4. จากนั้นทำการอัพเดตค่า Q ตลอดเส้นทางที่เดินผ่านมา โดยค่า Q เท่ากับค่าเฉลี่ยของค่ารีเทิร์นทุกรอบที่ผ่าน ณ ช่องนั้นในทิศทางนั้น



รูปที่ 3.7 แผนภาพการทำงานของโปรแกรม โดยใช้ Monte Carlo

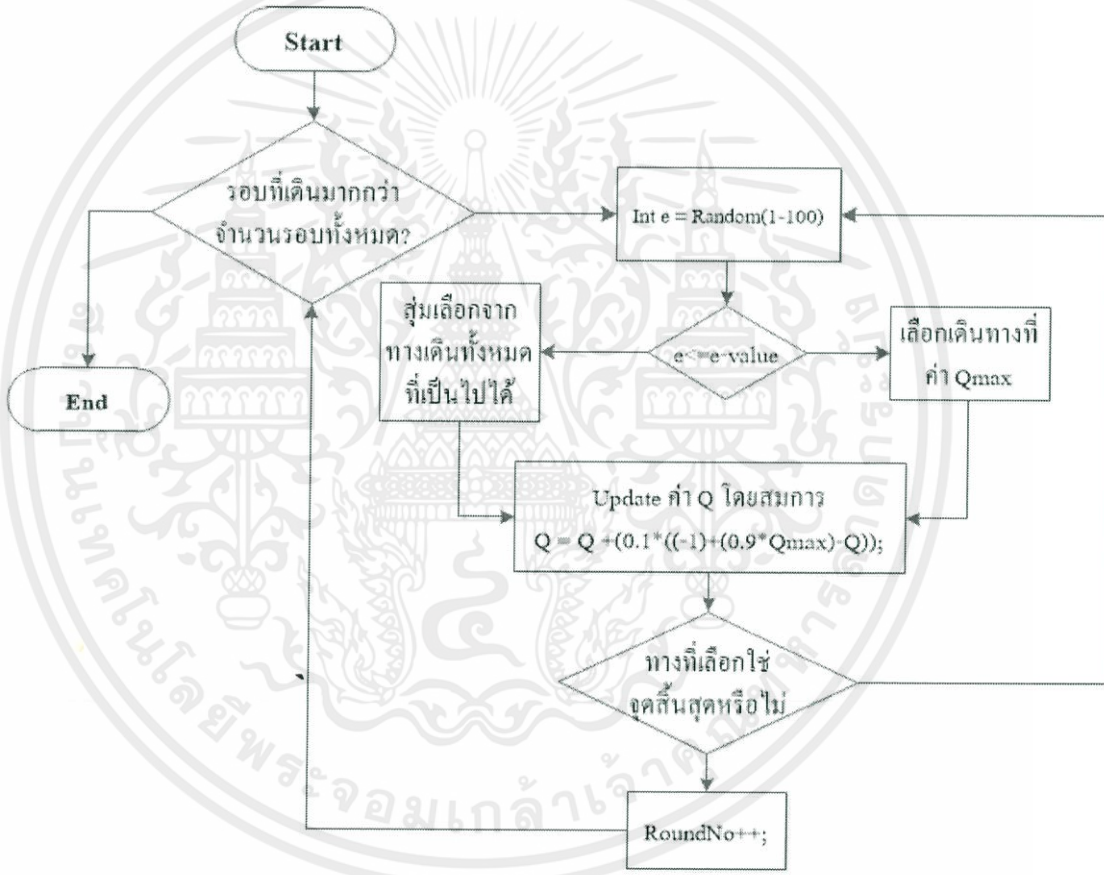
3.4.2 อัลกอริทึม Qlearning

อัลกอริทึม Qlearning เป็นวิธีการเรียนรู้ที่ผสมผสานกันระหว่างแนวคิดของ Monte Carlo กับไดนามิกโปรแกรมมิ่ง มีลักษณะวิธีการคล้ายกับ Monte Carlo คือการเรียนรู้โดยตรงจากประสบการณ์ช่วงแรกๆ โดยไม่ขึ้นกับรูปแบบการเปลี่ยนแปลงของสภาพแวดล้อม และคล้ายกับไดนามิกโปรแกรมมิ่ง คือวิธีการนี้จะปรับปรุงค่าประสบการณ์อยู่ตลอดเวลา โดยไม่จำเป็นต้องรอให้จบเอพิโสด โดยจะมีการอัพเดตค่าทุกครั้งที่มีการเดิน หรือเคลื่อนที่ไปช่องใหม่ ลำดับของการกำหนดค่าเริ่มต้นและการอัพเดตค่า เป็นดังนี้

1. กำหนดค่าเริ่มต้นให้ค่า Q สำหรับทุกค่า โดยในเราสามารถกรอกค่าเริ่มต้นนี้ในหน้าโปรแกรมที่ช่อง Q-value
2. เริ่มเดินจากจุดเริ่มต้น แล้วทำการเลือกเส้นทางโดยการสุ่มค่า 1-100 หากสุ่มได้ 1-90 ตัวเดินจะเลือกเดินไปในช่องที่มีค่า Q สูงที่สุด หากสุ่มได้ 91-100 ตัวเดินจะเลือกสุ่ม

ทางเดินจากเส้นทางทั้งหมดที่สามารถเดินไปได้ เมื่อเลือกทางเดินแล้วทำการอัพเดตค่าตามสมการในแผนภาพที่แสดงในรูปที่ 3.8

3. เลือกเดินไปช่องที่เลือกได้ จากนั้นทำการเลือกเส้นทางโดยการสุ่มค่า 1-100 หากสุ่มได้ 1-90 ตัวเดินจะเลือกเดินไปในช่องที่มีค่า Q สูงที่สุด หากสุ่มได้ 91-100 ตัวเดินจะเลือกสุ่มทางเดินจากเส้นทางทั้งหมดที่สามารถเดินไปได้ เมื่อเลือกทางเดินแล้วทำการอัพเดตค่าตามสมการในแผนภาพที่แสดงในรูปที่ 3.8 ทำข้อนี้วนซ้ำจนกว่าจะเจอว่าช่องที่เลือกนั้นเป็นจุดสุดท้าย

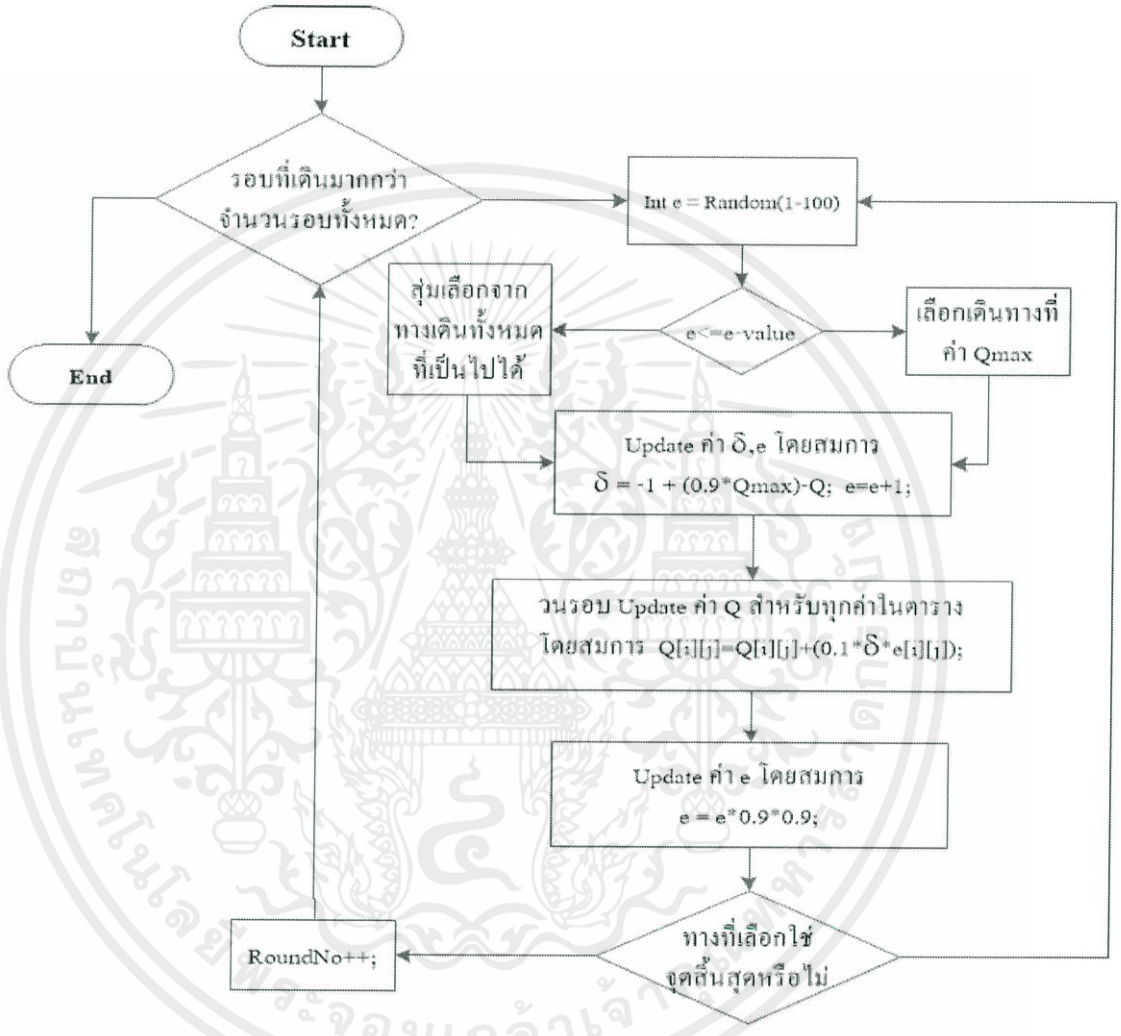


รูปที่ 3.8 แผนภาพการทำงานของโปรแกรม โดยใช้ Qlearning

3.4.3 อัลกอริทึม Watkins's Q()

อัลกอริทึม Watkins's Q() เป็นการผสมผสานกันระหว่าง 2 วิธีการ คือ Eligibility Traces กับ Qlearning รายละเอียดการเลือกเดินเริ่มต้นทำการเลือกเส้นทางโดยการสุ่มค่า 1-100 หากสุ่มได้ 1-90 ตัวเดินจะเลือกเดินไปในช่องที่มีค่า Q สูงที่สุด หากสุ่มได้ 91-100 ตัวเดินจะเลือกสุ่มทางเดิน

จากเส้นทางทั้งหมดที่สามารถเดินไปได้ เมื่อเลือกทางเดินแล้วทำการอัพเดตค่าตามสมการในแผนภาพที่แสดงในรูปที่ 3.9 ในวิธีนี้จะมีการอัพเดตการต่างๆ หลายค่า และในการเดินแต่ละครั้งจะทำการอัพเดตค่า Q ในช่องอื่นๆ ในตารางด้วย



รูปที่ 3.9 แผนภาพการทำงานของโปรแกรม โดยใช้ Watkins's Q()

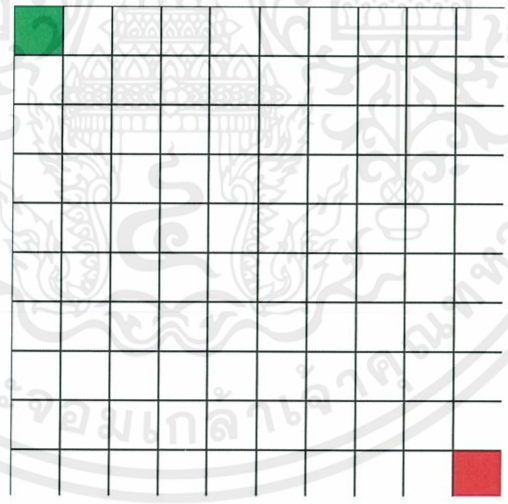
บทที่ 4
การทดลองและผลการทดลอง

4.1 การทดลอง

ผู้จัดทำได้ทำการทดลองโดยใช้รูปแบบกระดานในการทดลองทั้งหมด 2 รูปแบบ และมีการกำหนดค่าจำนวนรอบทั้งหมด เป็น 100, 1000 และ 10000 รอบ โดยใช้อัลกอริทึมต่างๆ ผลที่ได้จะแสดงในรูปแบบกราฟความสัมพันธ์ระหว่างจำนวนรอบที่เดินและจำนวนก้าวที่เดินแต่ละรอบ ซึ่งแกนนอนคือจำนวนรอบที่เดิน และแกนตั้งคือจำนวนก้าวที่เดินแต่ละรอบ

4.1.1 การทดลองรูปแบบกระดานแบบที่ 1

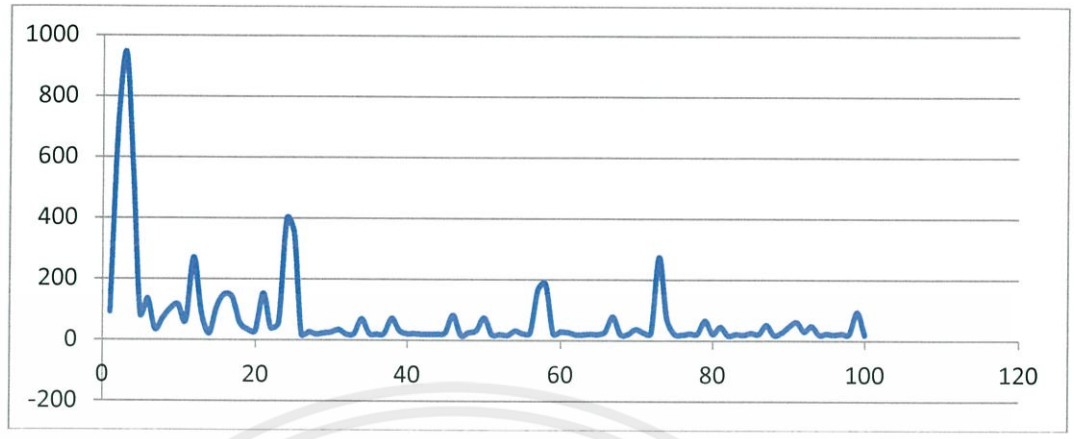
รูปแบบกระดานแบบที่ 1 ผู้จัดทำได้เลือกรูปแบบกระดานที่ไม่มีกำแพงหรือที่กั้นระหว่างจุดเริ่มต้นและจุดสุดท้าย ดังรูปที่ 4.1



รูปที่ 4.1 รูปแสดงรูปแบบกระดานแบบที่ 1

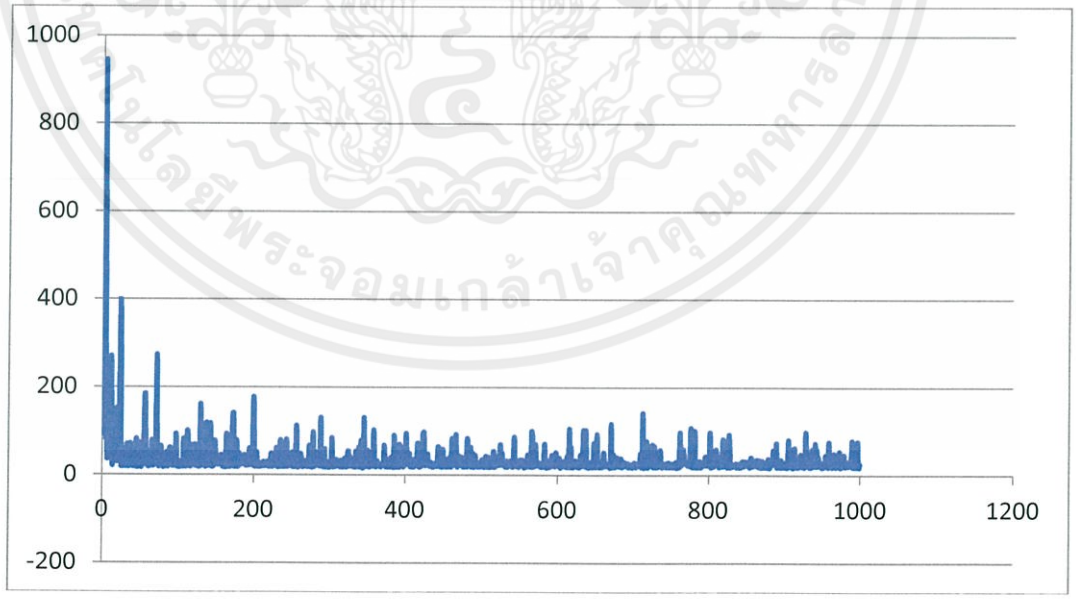
4.1.1.1 การทดลองโดยใช้ Monte Carlo

ในการทดลองกำหนดจำนวนรอบเดินทั้งหมดเป็น 100 รอบ 1000 รอบ และ 10000 รอบ ส่วนค่า e-greedy และค่า Q เริ่มต้น กำหนดเป็น 90 และ 0 ตามลำดับ



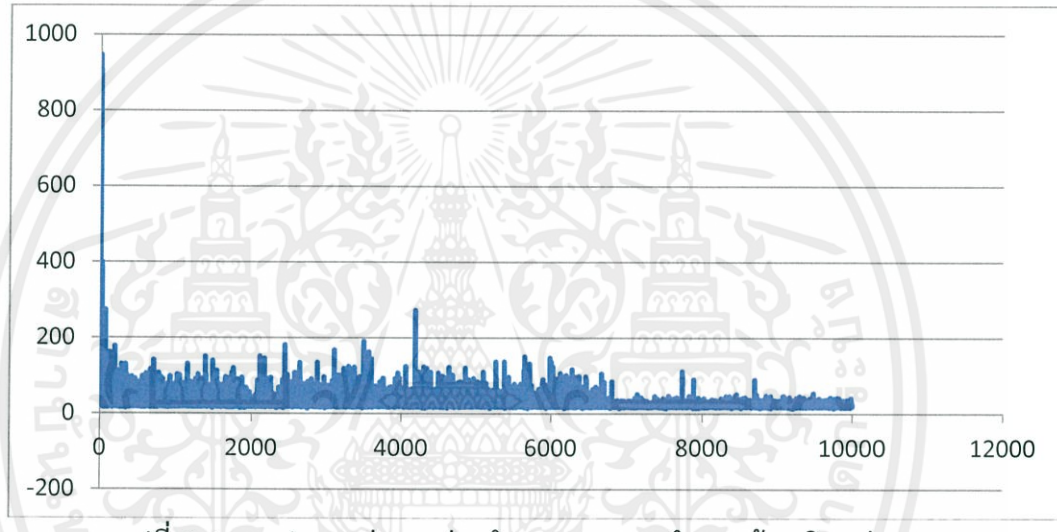
รูปที่ 4.2 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 100 รอบ

จากรูปที่ 4.2 พบว่าการเดินในช่วงรอบที่ 1-25 จำนวนก้าวที่เดินในแต่ละรอบ จำนวนก้าวที่มากที่สุดมีค่าสูงที่สุดถึง 945 ก้าว และหลายๆ รอบในช่วงนี้ก็ยังคงอยู่ในจำนวนหลักร้อยเป็นส่วนใหญ่ ส่วนช่วงหลัง คือรอบที่ 26-100 พบว่าค่าของจำนวนก้าวจะอยู่ที่ไม่เกินร้อยเป็นส่วนใหญ่ จะมีบ้างที่อยู่ในช่วงหลักร้อย ซึ่งช่วงหลังนี้จำนวนก้าวที่น้อยที่สุดมีค่าเท่ากับ 17 ก้าว ทั้งนี้จากรูประยะทางที่สั้นที่สุดคือ 10 ก้าว ดังนั้นวิธีนี้แสดงให้เห็นว่าผู้เรียนมีการเรียนรู้ แต่ยังไม่สามารถค้นหาทางที่สั้นที่สุดได้



รูปที่ 4.3 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 1000 รอบ

จากรูปที่ 4.3 พบว่าการเดินในช่วงแรก จำนวนก้าวที่เดินในแต่ละรอบ จำนวนก้าวที่มากที่สุดมีค่าสูงที่สุดเกือบถึง 1,000 ก้าว และหลายๆ รอบในช่วงนี้ก็ยังคงอยู่ในจำนวนหลักร้อยเป็นส่วนใหญ่ ส่วนช่วงหลัง พบว่าค่าของจำนวนก้าวจะอยู่ที่ไม่เกินร้อยเป็นส่วนใหญ่ จะมีบางรอบที่อยู่ในช่วงหลักร้อย ซึ่งช่วงหลังนี้จำนวนก้าวที่น้อยที่สุดมีค่าเท่ากับ 17 ก้าว แต่การจำนวนก้าวที่เดินในรอบต่างๆ ทั้งที่เรียนไปเกือบ 1,000 รอบแล้ว บางรอบยังมีจำนวนก้าวอยู่ในหลักร้อยหรือมากกว่า 200 ซึ่งแสดงให้เห็นว่าผู้เรียนได้เรียนรู้ให้เดินระยะทางสั้นลง แต่ผู้เรียนไม่สามารถที่จะเดินให้ได้ระยะทางสั้นๆ ในทุกรอบ

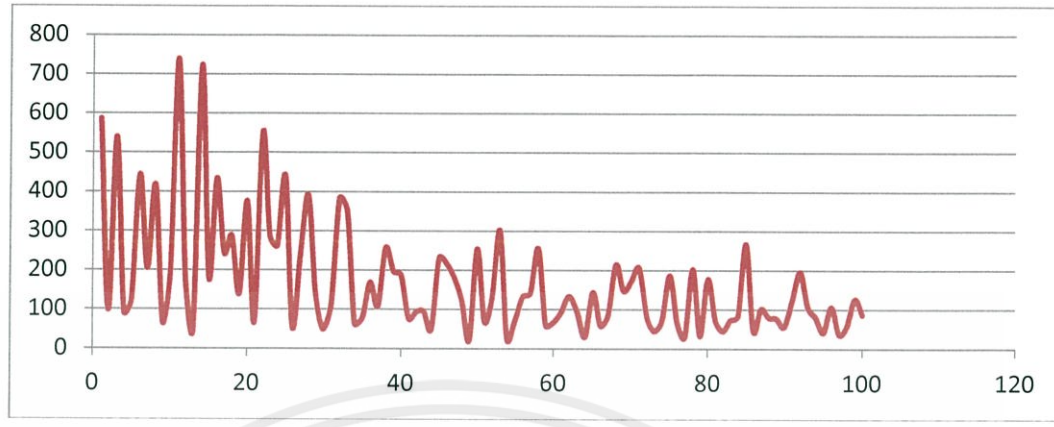


รูปที่ 4.4 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 10000 รอบ

จากรูปที่ 4.4 พบว่าการเดินในช่วงแรก และช่วงรอบที่เริ่มหลักพัน พบว่ามีลักษณะเหมือนกับรูปที่ 4.3 หากแต่ช่วงหลังรอบที่ 7,000 ผู้เรียนเริ่มที่จะใช้จำนวนก้าวในแต่ละรอบต่ำกว่า 100 ซึ่งช่วงหลังนี้จำนวนก้าวที่น้อยที่สุดมีค่าเท่ากับ 17 ก้าว วิธีนี้ยังไม่สามารถหาเส้นทางที่สั้นที่สุดได้

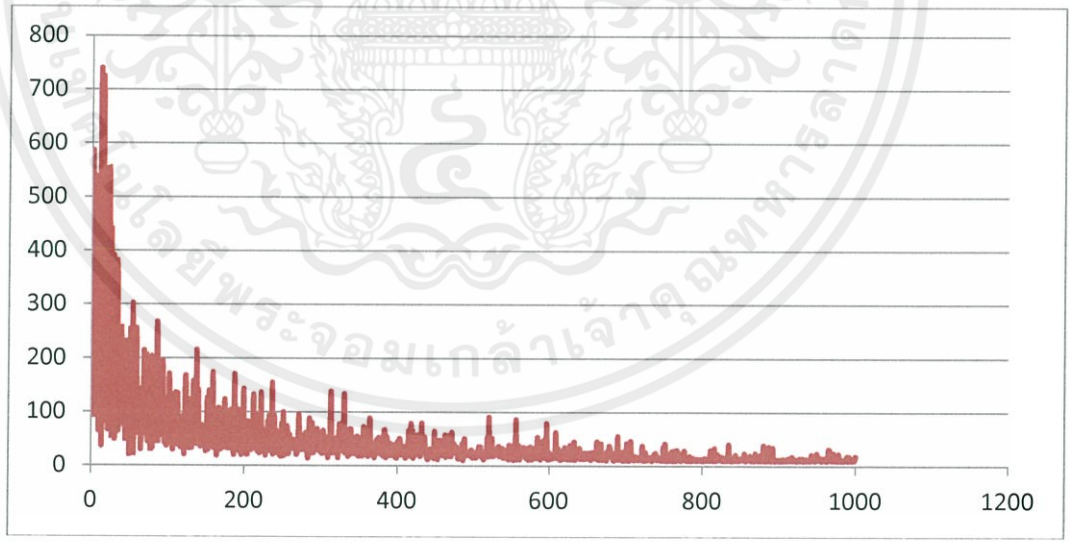
4.1.1.2 การทดลองโดยใช้ Qlearning

ในการทดลองกำหนดจำนวนรอบเดินทั้งหมดเป็น 100 รอบ 1,000 รอบ และ 10,000 รอบ ส่วนค่า e-greedy และค่า Q เริ่มต้น กำหนดเป็น 90 และ 0 ตามลำดับ



รูปที่ 4.5 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 100 รอบ

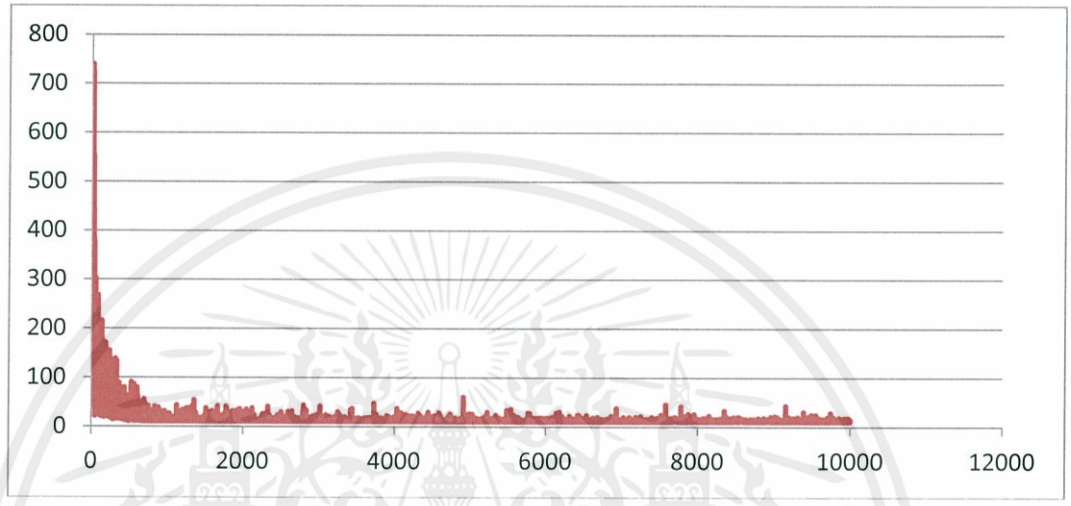
จากรูปที่ 4.5 พบว่าการเดินในช่วงแรกจำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 300 ก้าว ส่วนช่วงหลังจากเกือบรอบที่ 40 จะมีค่าต่ำกว่า 300 แต่จำนวนก้าวที่เดินจนกระทั่งเกือบรอบที่ 100 ก็ยังคงใช้จำนวนก้าวที่มากกว่า 100 ก้าวอยู่ ถึงแม้ว่าค่าต่ำสุดที่ผู้เรียนใช้จะอยู่ที่ 24 ก้าว แต่มีเพียงไม่กี่รอบเท่านั้น หากดูโดยรวมพบว่าวิธีนี้เรียนรู้ช้าอยู่ และยังไม่สามารถค้นหาทางที่สั้นที่สุดได้เลย



รูปที่ 4.6 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 1000 รอบ

จากรูปที่ 4.6 พบว่าการเดินในช่วงแรกรอบที่ 0-200 จำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 100 ก้าว ส่วนช่วงหลังจากแนวโน้มของกราฟเริ่มลดลง จนช่วงเกือบถึงรอบที่ 1,000 ผู้เรียนจะใช้

จำนวนก้าวเดินทั้งหมดอยู่ระหว่าง 10-20 ก้าว และพบว่าบ่อยครั้งที่ผู้เรียนเดินด้วยจำนวนก้าวทั้งมด คือ 10 ก้าว ซึ่งนั้นคือระยะทางที่สั้นที่สุด จากแนวโน้มของกราฟเห็นได้ว่าผู้เรียนเริ่มมีพัฒนาการดีขึ้นเรื่อยๆ

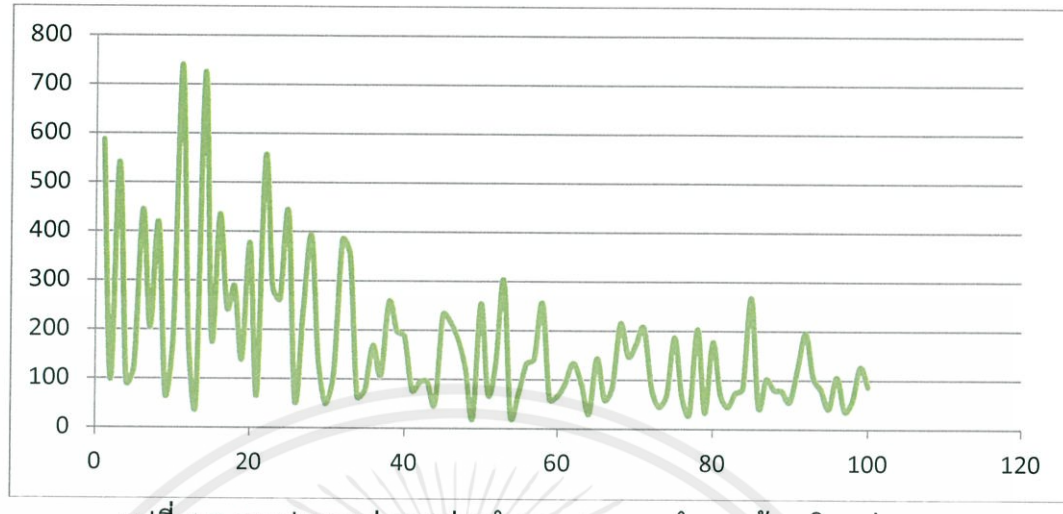


รูปที่ 4.7 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 10000 รอบ

จากรูปที่ 4.7 พบว่าการเดินในช่วงแรกรอบที่ 0-1,000 จำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 100 ก้าว ส่วนช่วงหลังจากแนวโน้มของกราฟเริ่มลดลง จนช่วงเกือบถึงรอบที่ 2,000-10,000 แนวโน้มของกราฟจะลดลงเรื่อยๆ จำนวนก้าวที่เดินส่วนใหญ่จะไม่เกิน 20 ก้าว และช่วงหลังผู้เรียนจะใช้จำนวนก้าวเดินอยู่ที่ 10-12 ก้าว

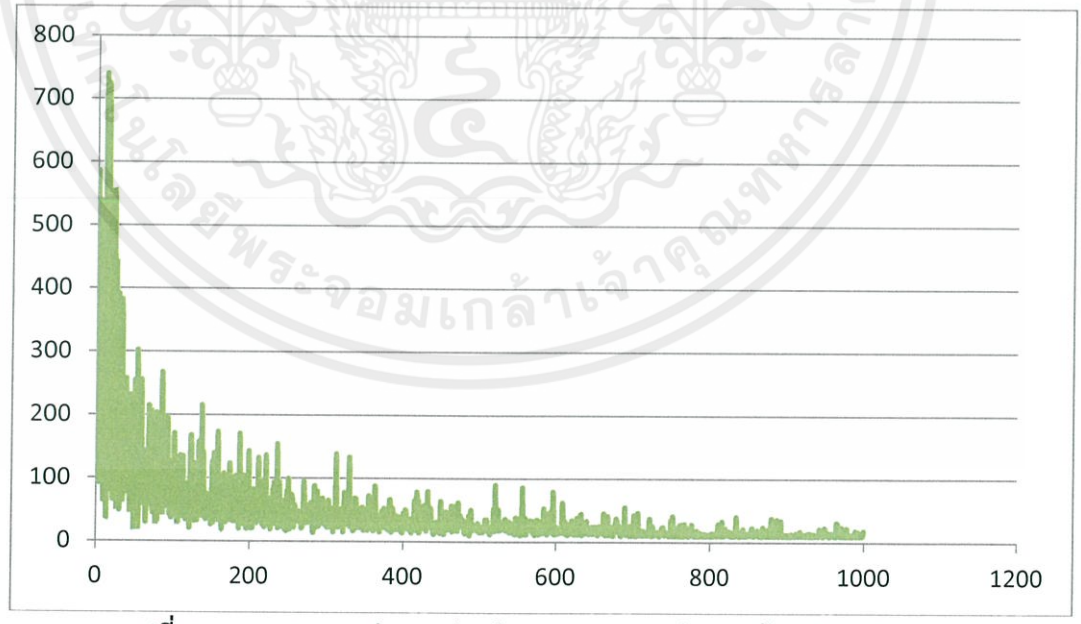
4.1.1.2 การทดลองโดยใช้ Watkins's Q()

ในการทดลองกำหนดจำนวนรอบเดินทั้งหมดเป็น 100 รอบ 1000 รอบ และ 10000 รอบ ส่วนค่า e-greedy และค่า Q เริ่มต้น กำหนดเป็น 90 และ 0 ตามลำดับ



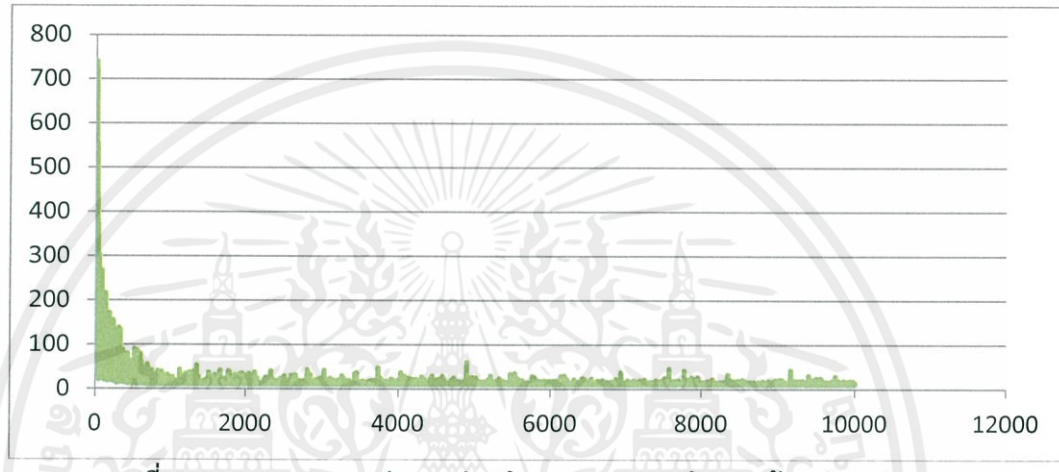
รูปที่ 4.8 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ
โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 100 รอบ

จากรูปที่ 4.8 พบว่าการเดินในช่วงแรกจำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 300 ก้าว ส่วนช่วงหลังจากเกือบรอบที่ 40 จะมีค่าต่ำกว่า 300 แต่จำนวนก้าวที่เดินจนกระทั่งเกือบรอบที่ 100 ก็ยังคงใช้จำนวนก้าวที่มากกว่า 100 ก้าวอยู่ ถึงแม้ว่าค่าต่ำสุดที่ผู้เรียนใช้จะอยู่ที่ 33 ก้าว แต่มีเพียงไม่กี่รอบเท่านั้น หากดูโดยรวมพบว่าวิธีนี้เรียนรู้ช้าอยู่ และยังไม่สามารถค้นหาทางที่สั้นที่สุดได้เลย



รูปที่ 4.9 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ
โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 1000 รอบ

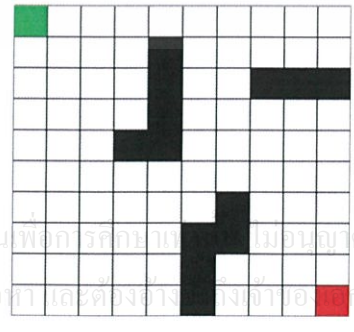
จากรูปที่ 4.9 พบว่าการเดินในช่วงแรกรอบที่ 0-200 จำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 100 ก้าว ส่วนช่วงหลังจากแนวโน้มของกราฟเริ่มลดลง จนช่วงเกือบถึงรอบที่ 1,000 ผู้เรียนจะใช้จำนวนก้าวเดินทั้งหมดอยู่ระหว่าง 10-20 ก้าว และพบว่าบ่อยครั้งที่ผู้เรียนเดินด้วยจำนวนก้าวทั้งมดคือ 10 ก้าว ซึ่งนั้นคือระยะทางที่สั้นที่สุด จากแนวโน้มของกราฟเห็นได้ว่าผู้เรียนเริ่มมีพัฒนาการดีขึ้นเรื่อยๆ



รูปที่ 4.10 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 10000 รอบ

จากรูปที่ 4.10 พบว่าการเดินในช่วงแรกรอบที่ 0-1,000 จำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 100 ก้าว ส่วนช่วงหลังจากแนวโน้มของกราฟเริ่มลดลง จนช่วงเกือบถึงรอบที่ 2,000-10,000 แนวโน้มของกราฟจะลดลงเรื่อยๆ จำนวนก้าวที่เดินส่วนใหญ่จะไม่เกิน 20 ก้าว และช่วงหลังผู้เรียนจะใช้จำนวนก้าวเดินอยู่ที่ 10-12 ก้าว

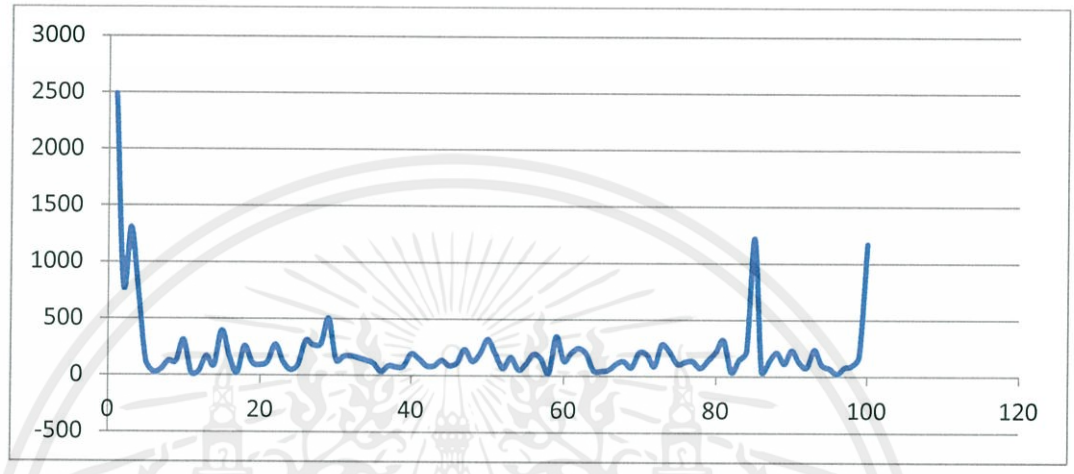
4.1.2 การทดลองรูปแบบกระดานแบบที่ 2



รูปที่ 4.11 รูปแสดงรูปแบบกระดานแบบที่ 2

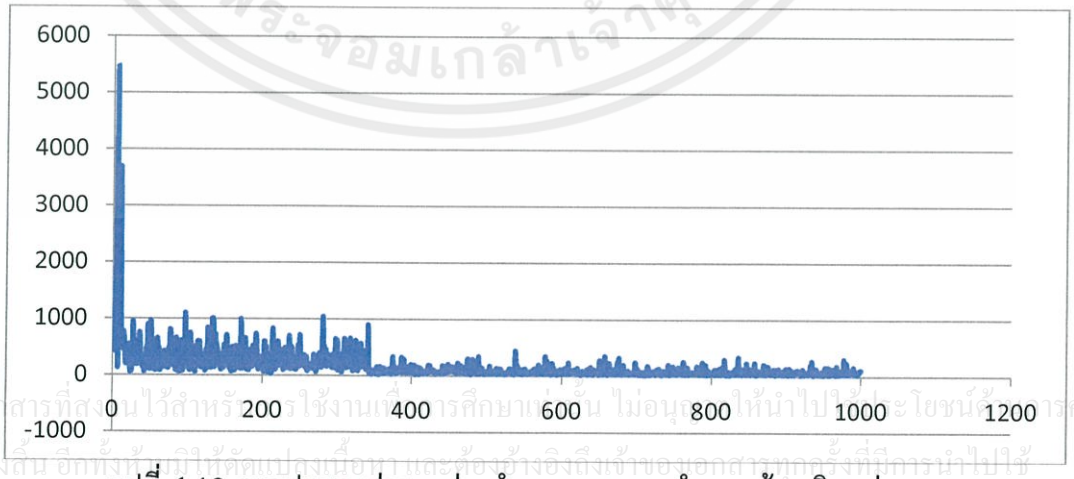
4.1.2.1 การทดลองโดยใช้ Monte Carlo

ในการทดลองกำหนดจำนวนรอบเดินทั้งหมดเป็น 100 รอบ 1000 รอบ และ 10000 รอบ ส่วนค่า e-greedy และค่า Q เริ่มต้น กำหนดเป็น 90 และ 0 ตามลำดับ



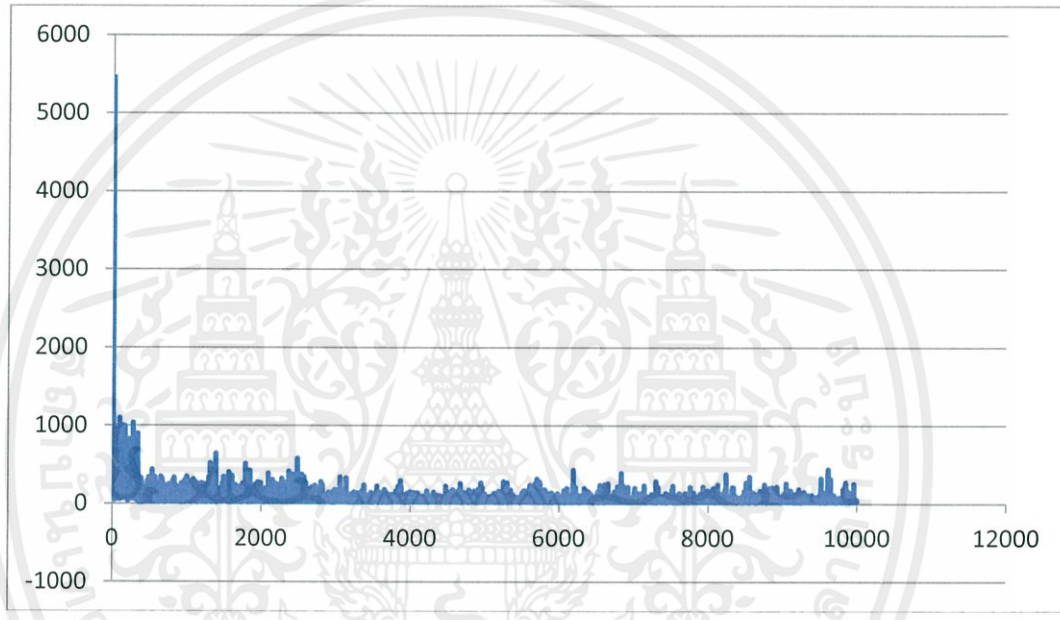
รูปที่ 4.12 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 100 รอบ

จากรูปที่ 4.12 พบว่าการเดินในรอบแรกจำนวนรอบมากที่สุดมีค่าสูงที่สุดถึง 945 ก้าว และหลายๆ รอบในช่วงนี้ก็ยังคงอยู่ในจำนวนหลักร้อยเป็นส่วนใหญ่ ส่วนช่วงหลังพบว่ายังมีหลายรอบที่ยังเดินเกิน 1,000 ก้าว ในการเดินทั้งหมดจำนวนก้าวที่น้อยที่สุดมีค่าเท่ากับ 23 ก้าว ทั้งนี้จากรูประยะทางที่สั้นที่สุดคือ 13 ก้าว ดังนั้นวิธีนี้แสดงให้เห็นว่าผู้เรียนมีการเรียนรู้ แต่การเดินส่วนใหญ่ยังให้จำนวนก้าวมากอยู่ การเรียนร้อยรอบในรูปแบบกระดานนี้ยังไม่ดีเมื่อเทียบกับกระดานที่หนึ่ง



รูปที่ 4.13 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 1000 รอบ

จากรูปที่ 4.13 พบว่าการเดินในช่วงแรก จำนวนก้าวที่เดินในแต่ละรอบ จำนวนก้าวที่มากที่สุดมีค่าสูงที่สุดเกือบถึง 6,000 ก้าว และหลายๆ รอบในช่วงนี้ก็ยังคงอยู่ในจำนวนหลักร้อยเกือบถึงพันเป็นส่วนใหญ่ ส่วนช่วงหลัง พบว่าค่าของจำนวนก้าวจะอยู่ที่ไม่เกินร้อยเป็นส่วนใหญ่ จะมีบางรอบที่อยู่ในช่วงหลักร้อย ซึ่งช่วงหลังนี้จำนวนก้าวที่น้อยที่สุดมีค่าเท่ากับ 25 ก้าว แต่การจำนวนก้าวที่เดินในรอบต่างๆ ทั้งที่เรียนไปเกือบ 1,000 รอบแล้ว บางรอบยังมีจำนวนก้าวอยู่ในหลักร้อย ซึ่งแสดงให้เห็นว่าผู้เรียนได้เรียนรู้ให้เดินระยะทางสั้นลง แต่ผู้เรียนไม่สามารถที่จะเดินให้ได้ระยะทางสั้นได้

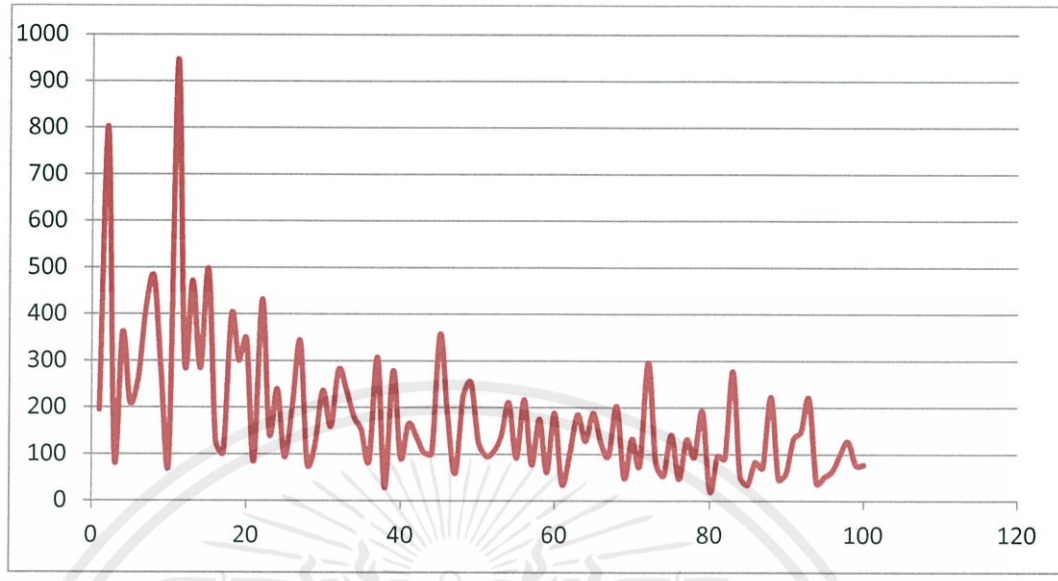


รูปที่ 4.14 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 10000 รอบ

จากรูปที่ 4.14 พบว่าการเดินในช่วงแรก และช่วงรอบที่เริ่มหลักพัน หากแต่ช่วงรอบหลังผู้เรียนเริ่มที่จะใช้จำนวนก้าวลดลง ซึ่งช่วงหลังนี้จำนวนก้าวที่น้อยที่สุดมีค่าเท่ากับ 25 ก้าว วิธีนี้ยังไม่สามารถหาเส้นทางที่สั้นที่สุด ในรูปแบบกระดานนี้ได้

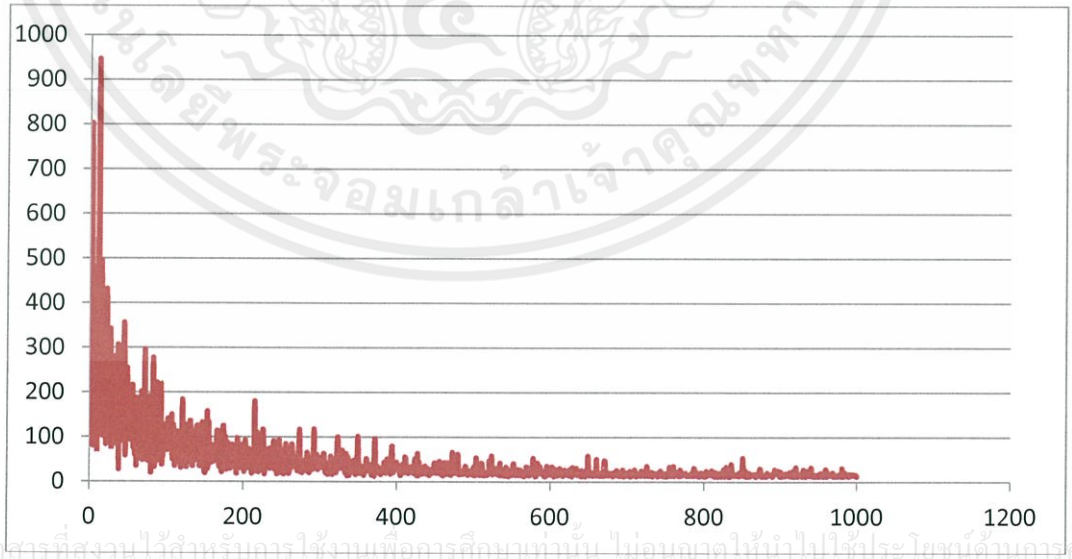
4.1.1.2 การทดลองโดยใช้ Qlearning

ในการทดลองกำหนดจำนวนรอบเดินทั้งหมดเป็น 100 รอบ 1,000 รอบ และ 10,000 รอบ ส่วนค่า e-greedy และค่า Q เริ่มต้น กำหนดเป็น 90 และ 0 ตามลำดับ



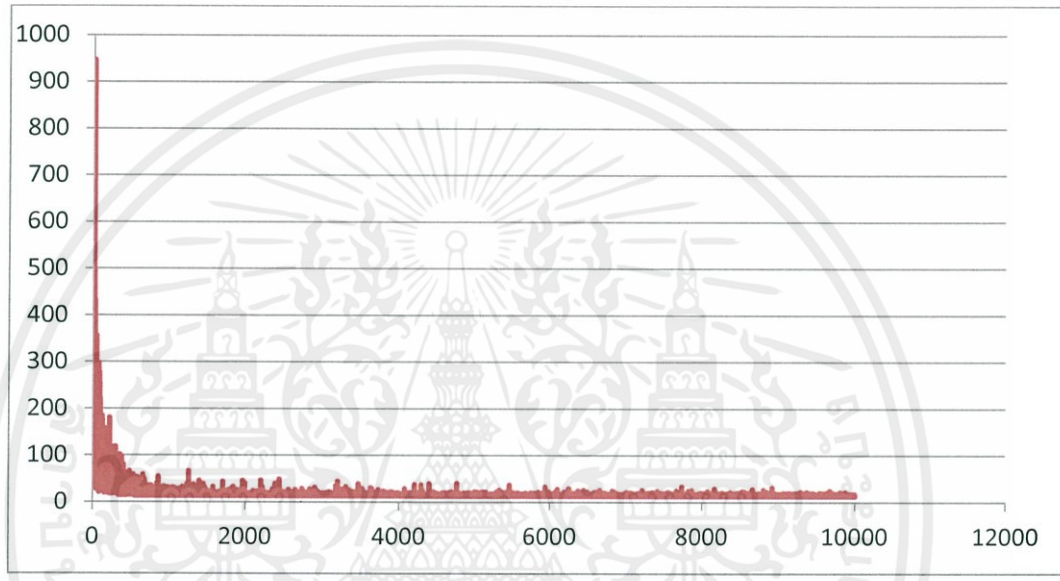
รูปที่ 4.15 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 100 รอบ

จากรูปที่ 4.15 พบว่าการเดินในช่วงแรกจำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 400 ก้าว ส่วนช่วงหลังจากเกือบรอบที่ 30 จะมีค่าลดลง แต่จำนวนก้าวที่เดินจนกระทั่งเกือบรอบที่ 100 ก็ยังคงใช้จำนวนก้าวที่มากกว่า 100 ก้าวอยู่ ถึงแม้ว่าค่าต่ำสุดที่ผู้เรียนใช้จะอยู่ที่ 28 ก้าว แต่มีเพียงไม่กี่รอบเท่านั้น หากดูโดยรวมพบว่าวิธีนี้เรียนรู้ช้าอยู่ และยังไม่สามารถค้นหาทางที่สั้นที่สุดได้เลย



รูปที่ 4.16 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 1000 รอบ

จากรูปที่ 4.16 พบว่าการเดินในช่วงแรกรอบที่ 0-400 จำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 100 ก้าว ส่วนช่วงหลังจากแนวโน้มของกราฟเริ่มลดลง จนช่วงเกือบถึงรอบที่ 1,000 ผู้เรียนจะใช้จำนวนก้าวเดินทั้งหมดอยู่ระหว่าง 13-30 ก้าว และพบว่าบ่อยครั้งที่ผู้เรียนเดินด้วยจำนวนก้าวทั้งมดคือ 13 ก้าว ซึ่งนั้นคือใกล้ระยะทางที่สั้นที่สุด จากแนวโน้มของกราฟเห็นได้ว่าผู้เรียนเริ่มมีพัฒนาการดีขึ้นเรื่อยๆ

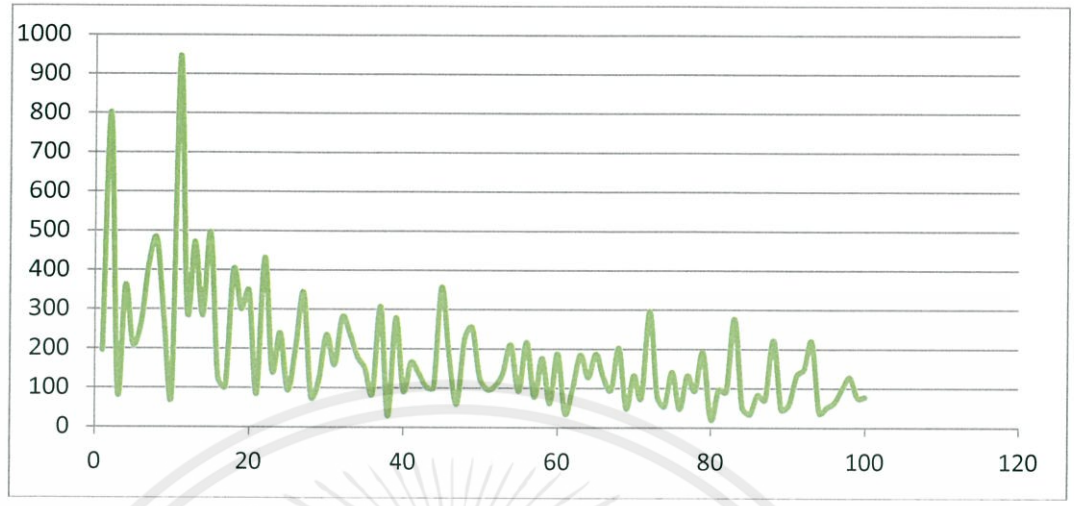


รูปที่ 4.17 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 10000 รอบ

จากรูปที่ 4.17 พบว่าการเดินในช่วงแรกรอบที่ 0-1,000 จำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 100 ก้าว ส่วนช่วงหลังจากแนวโน้มของกราฟเริ่มลดลง จนช่วงเกือบถึงรอบที่ 2,000-10,000 แนวโน้มของกราฟจะลดลงเรื่อยๆ จำนวนก้าวที่เดินส่วนใหญ่จะไม่เกิน 20 ก้าว และช่วงหลังผู้เรียนจะใช้จำนวนก้าวเดินอยู่ที่ 13-17 ก้าว

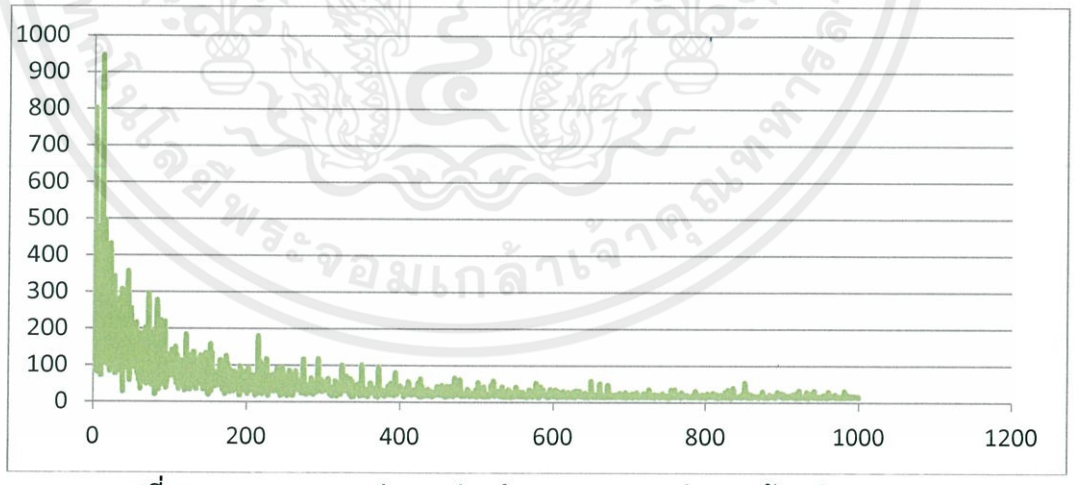
4.1.1.2 การทดลองโดยใช้ Watkins's Q()

ในการทดลองกำหนดจำนวนรอบเดินทั้งหมดเป็น 100 รอบ 1000 รอบ และ 10000 รอบ ส่วนค่า e-greedy และค่า Q เริ่มต้น กำหนดเป็น 90 และ 0 ตามลำดับ



รูปที่ 4.18 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 100 รอบ

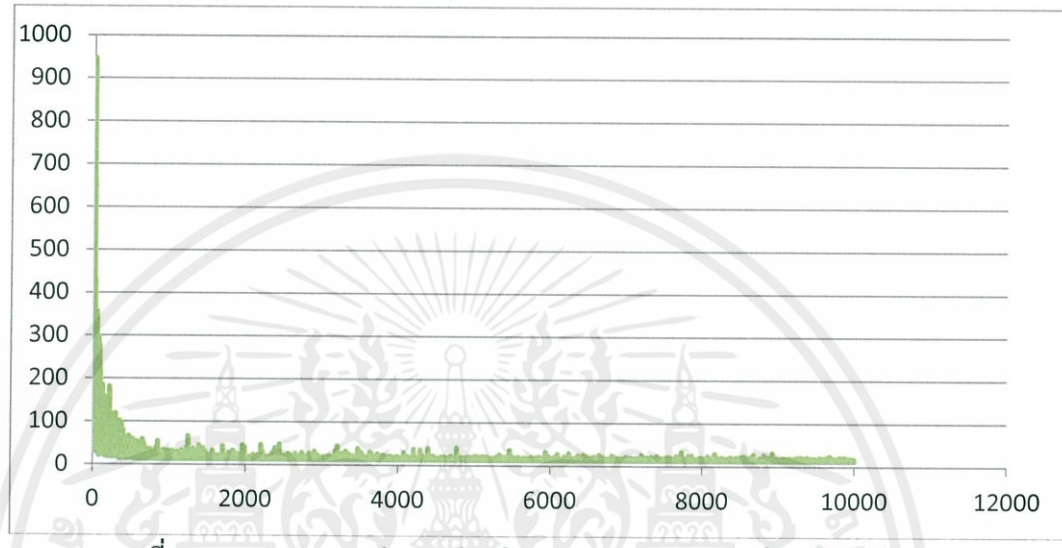
จากรูปที่ 4.18 พบว่าการเดินในช่วงแรกจำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 300 ก้าว ส่วนช่วงหลังจากเกือบรอบที่ 40 จะมีค่าต่ำกว่า 300 แต่จำนวนก้าวที่เดินจนกระทั่งเกือบรอบที่ 100 ก็ยังคงใช้จำนวนก้าวที่มากกว่า 200 ก้าวอยู่ ถึงแม้ว่าค่าต่ำสุดที่ผู้เรียนใช้จะอยู่ที่ 21 ก้าว แต่มีเพียงไม่กี่รอบเท่านั้น หากดูโดยรวมพบว่าวิธีนี้เรียนรู้ช้าอยู่ และยังไม่สามารถค้นหาทางที่สั้นที่สุดได้เลย



รูปที่ 4.19 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 1000 รอบ

จากรูปที่ 4.19 พบว่าการเดินในช่วงแรกรอบที่ 0-300 จำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 100 ก้าว ส่วนช่วงหลังจากแนวโน้มของกราฟเริ่มลดลง จนช่วงเกือบถึงรอบที่ 1,000 ผู้เรียน

จะใช้จำนวนก้าวเดินทั้งหมดอยู่ระหว่าง 13-32 ก้าว และพบว่าบ่อยครั้งที่ผู้เรียนเดินด้วยจำนวนก้าวทั้งมดคือ 13-14 ก้าว ซึ่งนั้นคือระยะทางที่สั้นที่สุด จากแนวโน้มของกราฟเห็นได้ว่าผู้เรียนเริ่มมีพัฒนาการดีขึ้นเรื่อยๆ



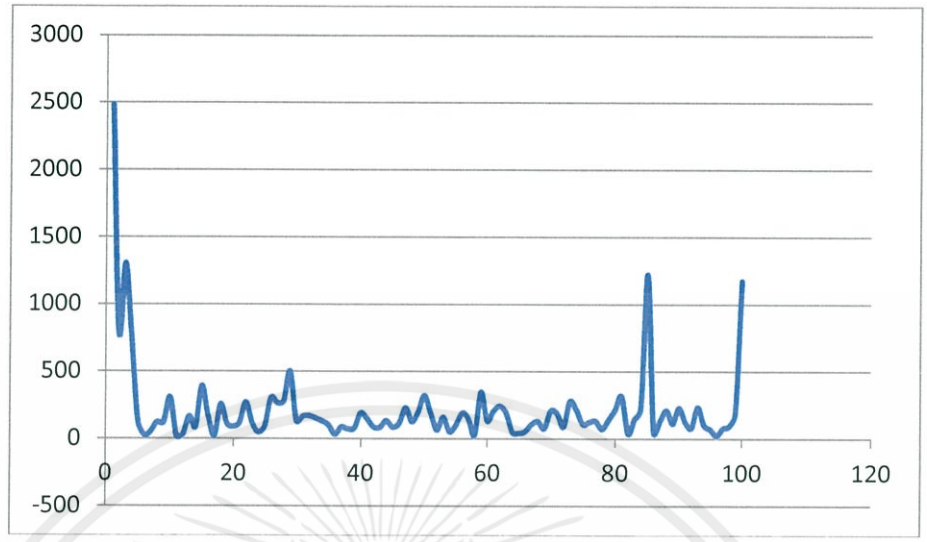
รูปที่ 4.20 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Watkins's Q() และใช้จำนวนรอบทั้งหมด 10000 รอบ

จากรูปที่ 4.20 พบว่าการเดินในช่วงแรกรอบที่ 0-1,000 จำนวนก้าวเดินส่วนมากจะมีค่ามากกว่า 100 ก้าว ส่วนช่วงหลังจากแนวโน้มของกราฟเริ่มลดลง จนช่วงเกือบถึงรอบที่ 8,000-10,000 แนวโน้มของกราฟจะลดลงเรื่อยๆ จำนวนก้าวที่เดินส่วนใหญ่จะไม่เกิน 22 ก้าว และช่วงหลังผู้เรียนจะใช้จำนวนก้าวเดินอยู่ที่ 13-15 ก้าว เห็นได้ว่าช่วงหลังผู้เรียนได้เกิดการเรียนรู้และจดจำเส้นทางที่สั้นที่สุดได้

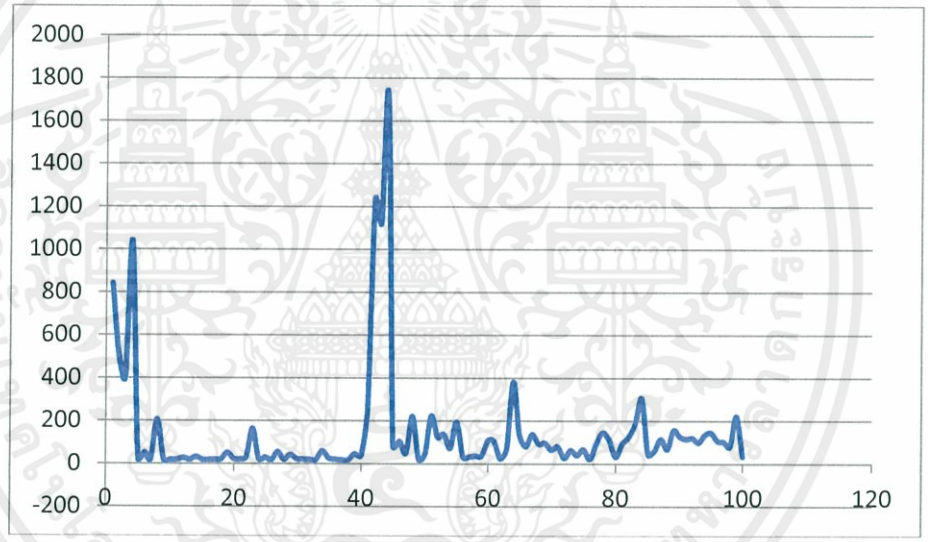
4.2 การเปรียบเทียบผลการทดลอง

4.2.1 เปรียบเทียบระหว่างกำหนดค่า Q เริ่มต้น เป็น 0 และ 30 โดยวิธี Monte Carlo

การทดลองเราได้จะเปรียบเทียบว่าการกำหนดค่า Q เริ่มต้น มีผลต่อการเรียนรู้โดยใช้วิธี Monte Carlo หรือไม่ โดยกำหนดค่า e-greedy เท่ากับ 90:10 ในทดลองในรูปแบบกระดานแบบที่ 2 ใช้จำนวนรอบเดินทั้งหมด 100 รอบ และ 1,000 รอบ



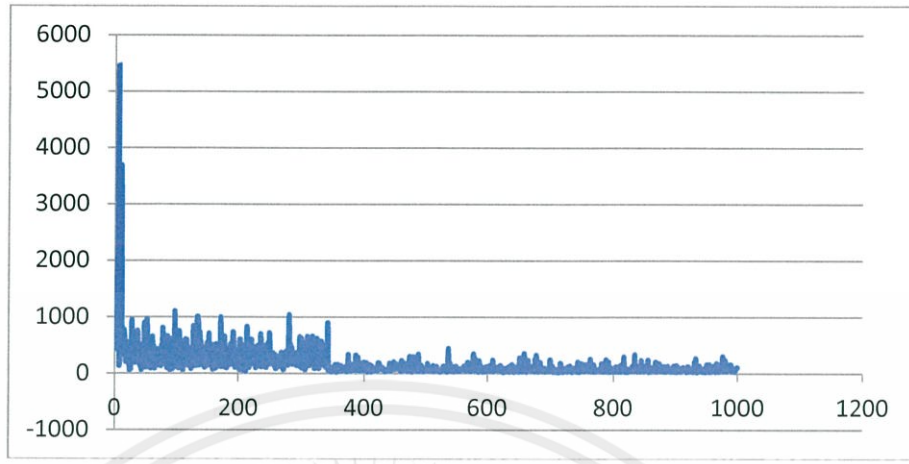
(ก)



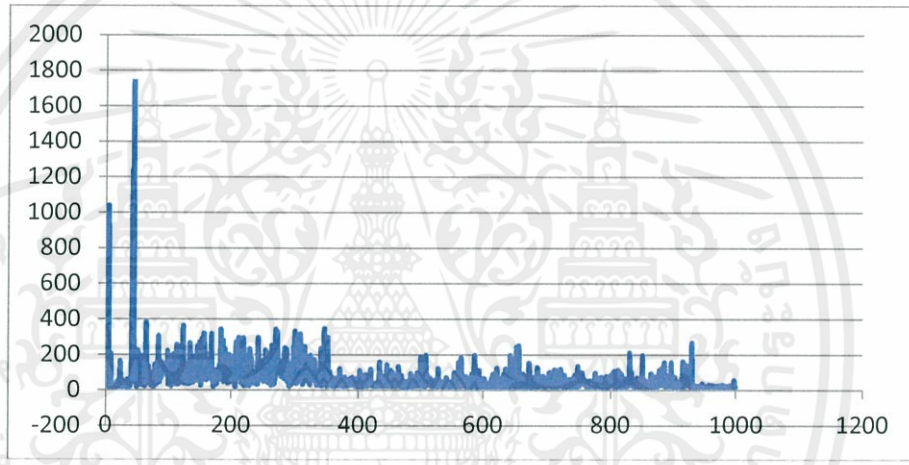
(ข)

รูปที่ 4.21 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 100 รอบ
(ก) กำหนดค่า Q เริ่มต้น คือ 0
(ข) กำหนดค่า Q เริ่มต้น คือ 30

จากรูปที่ 4.21 เปรียบเทียบระหว่าง (ก) กับ (ข) พบว่าช่วงแรก (ก) ใช้จำนวนก้าวเดินในบางรอบเกิน 70,000 ก้าว ในขณะที่ (ข) โดยภาพรวมมีค่าอยู่ไม่เกิน 2,000 ก้าว หากแต่แนวโน้มช่วงหลังจะไม่ต่างกันมาก ซึ่งพบว่าจำนวนก้าวเดินในช่วงหลังส่วนใหญ่มีค่า ประมาณ 100-300 ก้าว



(ก)



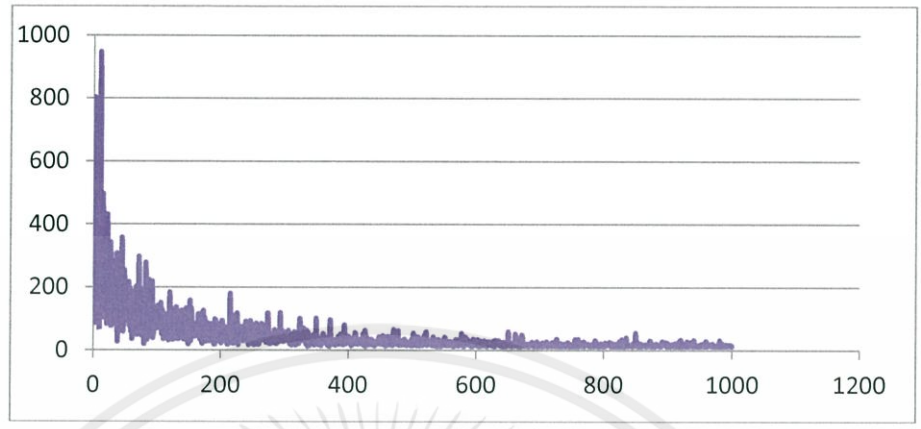
(ข)

รูปที่ 4.22 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Monte Carlo และใช้จำนวนรอบทั้งหมด 1,000 รอบ (ก) กำหนดค่า Q เริ่มต้น คือ 0 (ข) กำหนดค่า Q เริ่มต้น คือ 30

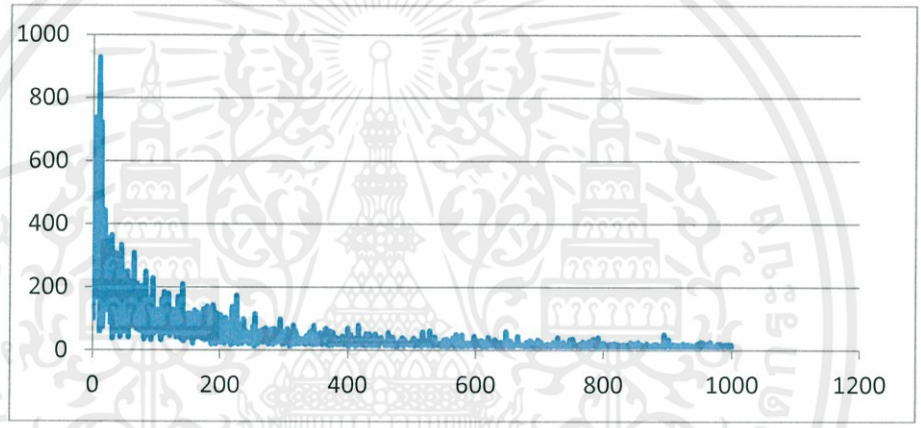
จากรูปที่ 4.22 เมื่อเทียบกันในช่วงรอบที่ 0-400 พบว่า (ก) ใช้จำนวนก้าวส่วนใหญ่เกือบถึง 1,000 ก้าว ส่วน (ข) จำนวนก้าวส่วนใหญ่จะอยู่ในช่วงไม่เกิน 400 ก้าว และเมื่อพิจารณาดูช่วงหลัง ซึ่งพบว่า (ก) จะใช้จำนวนก้าวเดินในช่วง 50-100 ก้าวเป็นส่วนใหญ่ ส่วน (ข) จะใช้จำนวนก้าวเดินในช่วง 15-30 ทำให้เห็นได้ชัดว่าการกำหนดค่า Q เริ่มต้นที่ต่างกันมีผลต่อการเรียนรู้อย่างมาก

4.2.2 เปรียบเทียบกำหนดค่า e-greedy ต่างกัน โดยวิธี Qlearning

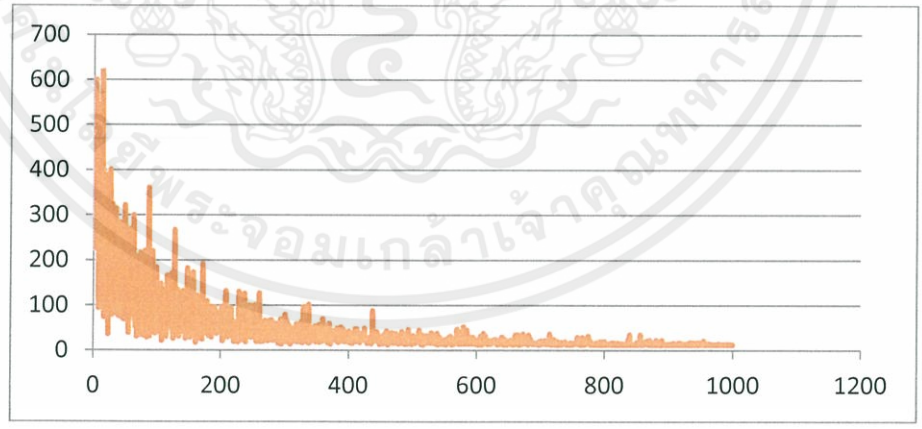
การทดลองจะแบ่งเป็น 3 กราฟ โดยใช้จำนวนรอบทั้งหมดคือ 1,000 รอบ มีตัวแปรที่ต่างกัน คือค่า e-greedy เท่ากับ 90:10 , 95:5 และ 99:1 และทดลองให้กระดานรูปแบบที่ 2 (ดังรูปที่ 4.11)



(ก)



(ข)



(ค)

รูปที่ 4.23 กราฟแสดงค่าระหว่างจำนวนรอบและจำนวนก้าวเดินแต่ละรอบ โดยวิธี Qlearning และใช้จำนวนรอบทั้งหมด 1,000 รอบ

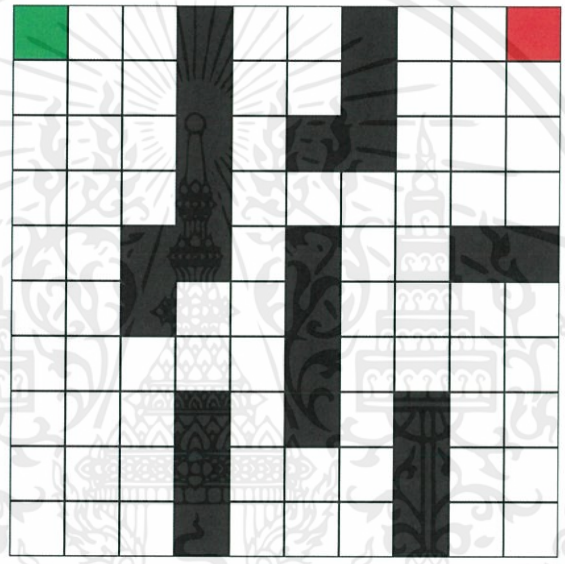
(ก) ค่า e-greedy เท่ากับ 90 : 10

(ข) ค่า e-greedy เท่ากับ 95 : 5

(ค) ค่า e-greedy เท่ากับ 99 : 1

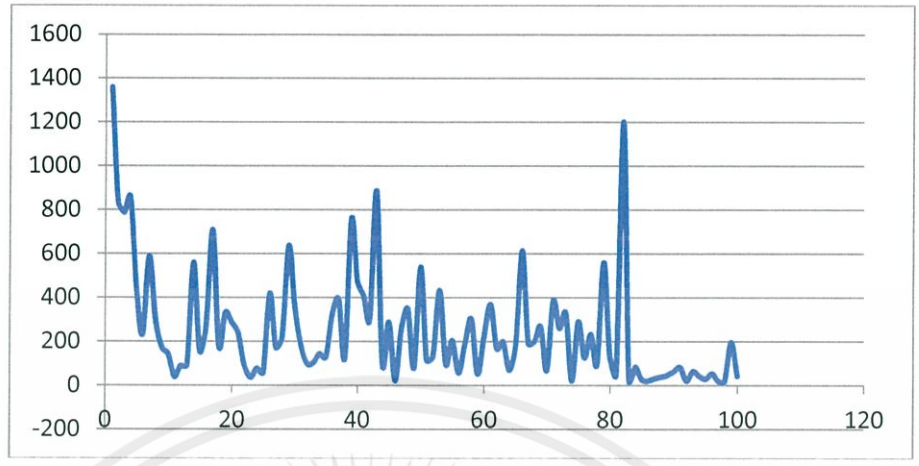
จากรูปที่ 4.23 พบว่ากราฟมีแนวโน้มลดลงในลักษณะเดียวกัน หากแต่จะต่างกันในช่วงหลัง ๆ โดยที่ในกราฟ (ค) กำหนดค่า e-greedy 99:1 ค่าช่วงหลังจะคงที่โดยเดินจำนวน 13 ก้าว บ่อยที่สุด ซึ่งมีเพียงไม่กี่รอบเท่านั้นที่จะเดิน 14-15 ก้าว รองลงมาจะเป็น 95:5 และ 90:10 ทั้งนี้ก็เนื่องจากเมื่อผู้เรียนรู้จักเส้นทางที่สั้นที่สุดแล้ว ในการเลือกเส้นทางเดินแต่ละครั้ง หากค่า e-greedy สูง ทำให้ทางที่ผู้เรียนเดินมักจะเลือกไปในทางที่ดีที่สุดถึง 99%

4.2.3 เปรียบเทียบการเรียนรู้ทั้ง 3 วิธี

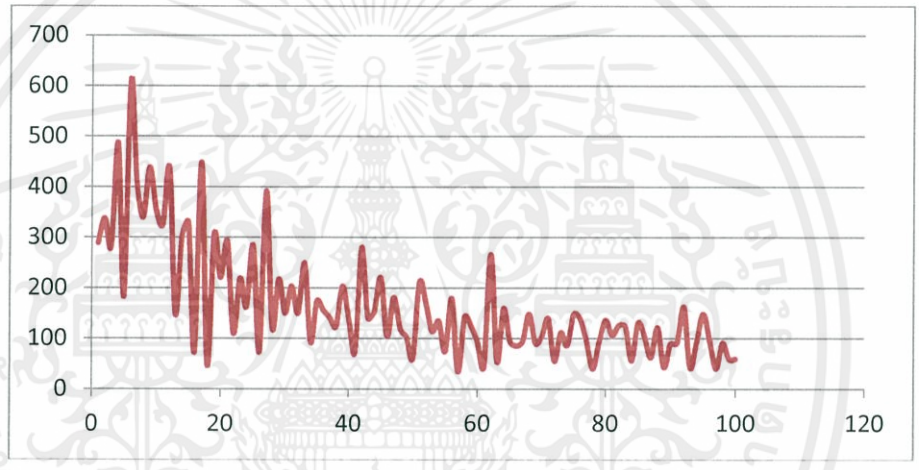


รูปที่ 4.24 รูปแสดงรูปแบบกระดานแบบที่ 3

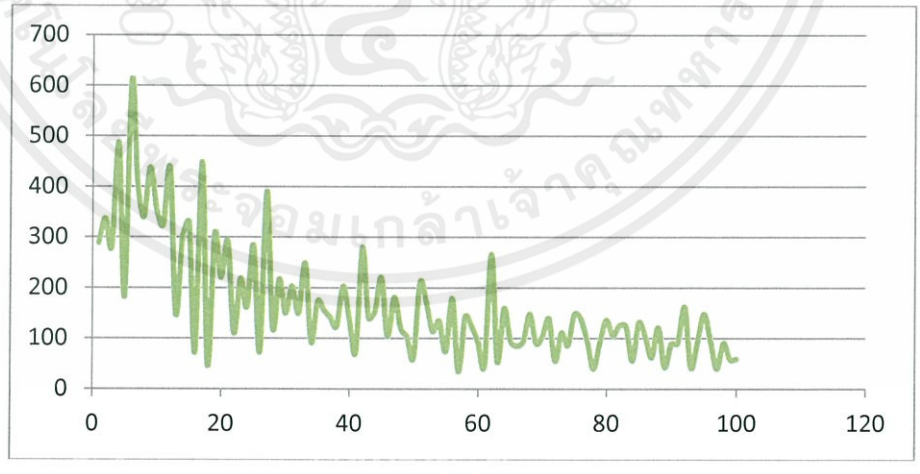
ในการทดลองเปรียบเทียบการเรียนรู้ทั้ง 3 วิธี ผู้จัดทำเลือกทดลองโดยใช้รูปแบบกระดานที่ 3 ซึ่งเป็นรูปแบบที่มีกำแพงซับซ้อนมากกว่าในรูปแบบที่ 1กำหนดจำนวนรอบทั้งหมดคือ 100 และ 1,000 รอบ และให้ค่า e-greedy เท่ากับ 90:10 ส่วนค่า Q เริ่มต้นให้เป็น 0



(ก)



(ข)



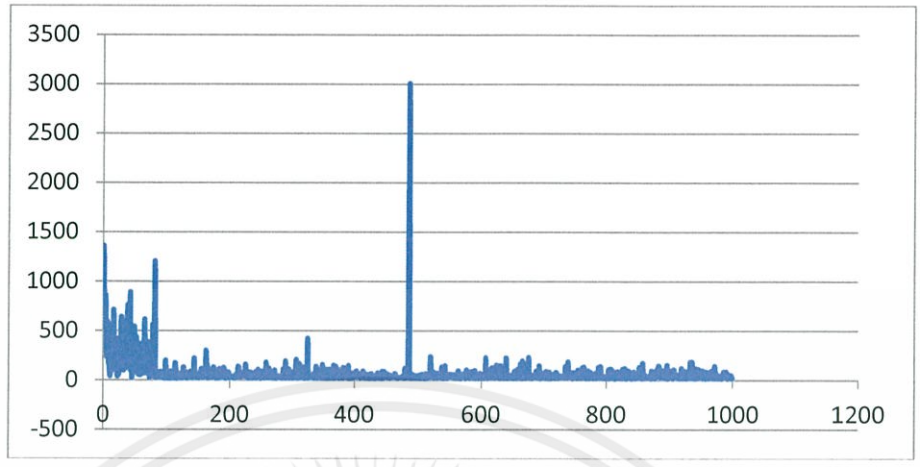
(ค)

รูปที่ 4.25 กราฟแสดงการเรียนรู้ ใช้จำนวนรอบทั้งหมด 100 รอบ

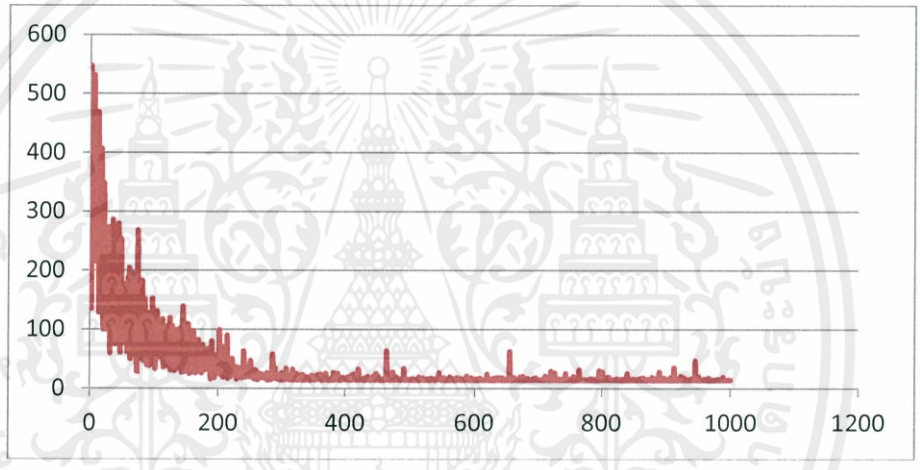
(ก) โดยวิธี Monte Carlo

(ข) โดยวิธี Qlearning

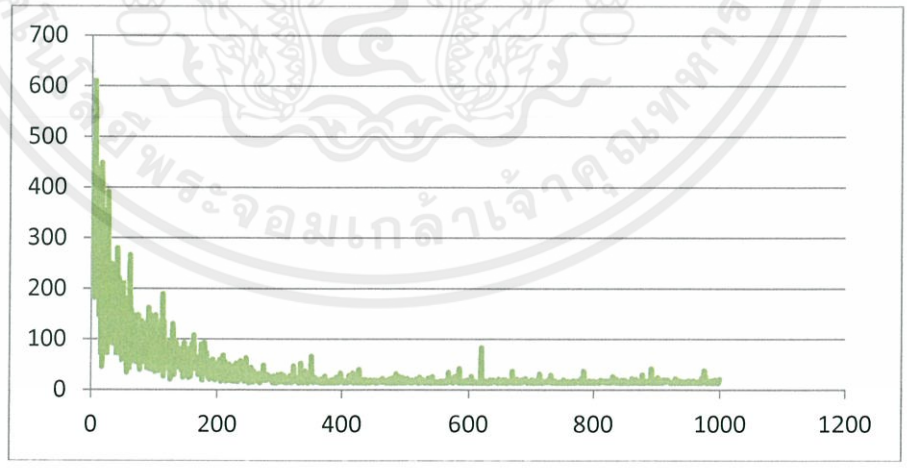
(ค) โดยวิธี Watkins's Q()



(ก)



(ข)



(ค)

รูปที่ 4.26 กราฟแสดงการเรียนรู้ ใช้จำนวนรอบทั้งหมด 1,000 รอบ

(ก) โดยวิธี Monte Carlo

(ข) โดยวิธี Qlearning

(ค) โดยวิธี Watkins's Q()

จากรูปที่ 4.25 (ก) จำนวนก้าวที่ใช้ที่มากที่สุด มีค่าเข้าใกล้ 1400 และลักษณะกราฟมีแนวโน้มไม่คงที่ บางช่วงลดลง แต่ช่วงหลังจากรอบที่ 80 นั้นยังมีค่าของจำนวนก้าวเดินสูงถึง 1200 ก้าว และลดลงต่ำ ซึ่งรอบหลังๆ จำนวนก้าวที่เดินอยู่ระหว่าง 20-100 ส่วน (ข) และ (ค) มีลักษณะกราฟที่คล้ายกัน โดยมีแนวโน้มของกราฟลดลงเรื่อยๆ ช่วงหลังจำนวนก้าวที่เดินอยู่ระหว่าง 40-150 ก้าว การเปรียบเทียบรูปนี้วิธี Monte Carlo สามารถเดินได้เส้นทางที่สั้นกว่าวิธีอื่นในช่วงหลัง

จากรูปที่ 4.26 (ก) มีแนวโน้มของกราฟลดลงแบบไม่ต่อเนื่อง เนื่องจากประมาณรอบที่ 500 วิธีนี้ใช้จำนวนก้าวทั้งหมดถึง 3,000 ก้าว และช่วงหลัง 18-110 วิธีนี้สามารถเดินเส้นทางสั้นที่สุดคือ 18 ก้าว แต่มีเพียงไม่กี่รอบเท่านั้น (ข) และ (ค) มีแนวโน้มของกราฟลดลงสม่ำเสมอ ซึ่งแสดงให้เห็นว่ายิ่งเรียนมาก ความสามารถในการเดินให้ได้ระยะทางสั้นๆ จะมากขึ้น หากดูค่าจำนวนก้าวต่ำสุดที่สองวิธีนี้ใช้ จะมีค่าต่ำสุดเท่ากันคือ 14 ก้าว แต่เมื่อพิจารณาดูในช่วงหลัง (ข) จะมีการเดินในเส้นทางระยะ 14 ก้าว มีความถี่สูงกว่า (ค) ทำให้เห็นว่าการเรียนแบบ Qlearning มีผลออกมาดีกว่าวิธีอื่น

บทที่ 5
บทสรุป

5.1 บทสรุป

โครงงานนี้ได้ทำการศึกษาหลักการของการเรียนรู้แบบเสริมกำลัง โดยได้พัฒนาเกมด้วยภาษา C++ และไลบรารี MFC โดยใช้ Visual Studio 2010 ในการพัฒนาเกม ทำให้ได้เกมกระดานซึ่งเป็นเกมหาเส้นทางการเดินจากจุดเริ่มต้น ถึงจุดสุดท้ายที่สั้นที่สุด จากนั้นทำการออกแบบให้สามารถประยุกต์ใช้ปัญญาประดิษฐ์ โดยการใช้หลักการการเรียนรู้แบบเสริมกำลัง เพื่อให้ได้อัลกอริทึมที่ทำได้สามารถเรียนรู้ จดจำ และค้นหาเส้นทางที่สั้นที่สุดเองได้

จากการทดลองพบว่าการเรียนรู้ที่ใช้ทั้งสามวิธีมีความสามารถพัฒนาการเรียนรู้ และหาเส้นทางที่ระยะทางสั้นลงได้ นอกจากนั้นการกำหนดค่า Q เริ่มต้น และค่า e-greedy ที่ต่างกัน ทำให้ส่งผลต่อการเรียนรู้ด้วย จากการทดลองกำหนดค่า Q เริ่มต้น เป็น 0 และ และ 30 ในวิธี Monte Carlo พบว่าพัฒนาการเรียนรู้ของการทดลองที่กำหนดค่า Q เริ่มต้น เป็น 30 สามารถเรียนรู้ได้ดีมากกว่าถึงสองเท่า และจากการทดลองกำหนดค่า e-greedy ต่างกันในวิธี Qlearning กับ Watkins's Q() พบว่าการทดลองใช้ค่า 99:1 ส่งผลให้การเรียนรู้ระยะยาวมีผลการเดินที่คงที่ และเดินไปในเส้นทางที่สั้นที่สุด ส่วนการเปรียบเทียบผลการเรียนรู้ทั้งสามวิธี ทำให้พบว่า Qlearning เหมาะสมการการเรียนรู้ในแบบจำลองนี้มากที่สุด รองลงมาคือ Watkins's Q() และ Monte Carlo ตามลำดับ

5.2 ปัญหาอุปสรรคและแนวทางการแก้ไข

1. การพัฒนาเกมโดยใช้ไลบรารี MFC เป็นความรู้ใหม่สำหรับผู้จัดทำ ทำให้ยังไม่เข้าใจการทำงานของฟังก์ชันต่างๆ มากนัก ซึ่งผู้จัดทำได้ศึกษาเพิ่มเติมจนสามารถพัฒนาแบบจำลองได้สำเร็จ
2. การออกแบบการเรียนรู้แบบเสริมกำลัง มีการนำหลายอัลกอริทึมมาใช้ในแบบจำลอง ทำให้มีการใช้ตัวแปร อาร์เรย์ต่างกันในการเก็บค่า ผู้จัดทำได้แก้ไขโดยออกแบบให้ตัวแปรที่มีลักษณะการเก็บค่าคล้ายกัน กำหนดให้ใช้ชื่อตัวแปร และอาร์เรย์เดียวกันสำหรับในแต่ละวิธี

5.3 แนวทางในการพัฒนาต่อ

1. สามารถนำการเรียนรู้แบบเสริมกำลังในอัลกอริทึมต่างๆมาลองประยุกต์ใช้
2. สามารถขยายขนาดตาราง และเพิ่มความซับซ้อนของรูปแบบกระดาน
3. ในแบบจำลองที่ใช้ควร สามารถเลือกรูปแบบอัลกอริทึมได้หลากหลาย
4. ออกแบบการทดลองให้หลากหลายขึ้น

บรรณานุกรม

[1] กรวิชญ์ พัววรานุเคราะห์ และคณะ. **"โปรแกรมซื้อขายค่าเงินอัตโนมัติ"** ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2553.

[2] จีระพงษ์ ศรีไชย, ชัยวัฒน์ ตั้งตรงสุนทร. **"การเรียนรู้ของเครื่อง2"** ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2554.

[3] ยุทธนา ลีลาศวัฒนกุล. **คู่มือการเขียนโปรแกรมวินโดว์ด้วย Visual C++ 2010** กรุงเทพมหานคร : ห้างหุ้นส่วนจำกัดไทยเจริญการพิมพ์. 2555.

[4] ศุภณัฐ กิตติวรการชัยม, แสงระวี พินิจมนตรี. **"การเรียนรู้ของเครื่อง2"** ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2554.

[5] Richard S., Sutton. **"Reinforcement Learning."** [Online]. Available : http://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html. 2012.