การเรียนรู้ของเครื่อง

Machine Learning

พิเชษฐ์ พูลสวัสดิ์

กวิศ อิ่มปศิรส

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

การเรียนรู้ของเครื่อง

Machine Learning

พิเชษฐ์ พูลสวัสดิ์

ภวิศ ลิมป์ศิระ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การเรียนรู้ของเครื่อง

Reinforcement Learning

ผู้จัดทำ

1. นายพิเชษฐ์ พูลสวัสดิ์ รหัสนักศึกษา 53011127

2. นายภวิศ ลิมป์ศิระ รหัสนักศึกษา 53011199

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์กฤตวัน ศิริบูรณ์)

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร.บุญธีร์ เครือตาชู)

การเรียนรู้ของเครื่อง

นายพิเชษฐ์	พูลสวัสดิ์	53011127
นายภวิศ	ลิมป์ศิระ	53011199
รศ.กฤตวัน	ศิริบูรณ์	อาจารย์ที่ปรึกษา
รศ. ดร.บุญธีร์	เครือตราชู	อาจารย์ที่ปรึกษา

ปีการศึกษา 2556

บทคัดย่อ

ในปัจจุบันการนำคอมพิวเตอร์มาประยุกต์ใช้ในการเรียนรู้และตัดสินใจแก้ปัญหาต่างๆ แทนมนุษย์นั้นได้รับความนิยมมากขึ้น การเรียนรู้ของเครื่องนั้น เป็นวิธีการต่างๆ ที่ทำให้คอมพิวเตอร์สามารถเรียนรู้และแก้ไขปัญหาเองได้ ซึ่งการเรียนรู้แบบเสริมกำลังนั้นเป็นอีกแนวคิดหนึ่งที่ถูกใช้อย่างแพร่หลาย สำหรับโครงงานนี้จะเป็นการนำการเรียนรู้เสริมกำลัง ซึ่งเป็นรูปแบบการเรียนรู้ที่ใช้ประสบการณ์ในอดีตมาแก้ไขและปรับใช้ในการเลือกการกระทำต่อไปในอนาคต มาแก้ไขปัญหาเกมเขาวงกต ซึ่งในโครงงานนั้นจะใช้การเรียนรู้เสริมกำลังเป็นหลัก โดยใช้สภาพแวดล้อม ณ เวลาปัจจุบันเป็นสถานะปัจจุบัน และใช้สถานะปัจจุบันในการเลือกการกระทำเท่านั้น โดยที่ไม่ขึ้นกับสถานะ ณ เวลาอื่นๆ และผลลัพธ์ของการกระทำหนึ่งๆนั้นจะคงเดิม โดยโครงงานนี้จะทำให้คอมพิวเตอร์นั้นสามารถเรียนรู้ปัญหาและวิธีแก้ไข ตลอดจนสามารถแก้ไขปัญหาเกมเขาวงกตได้อย่างชาญฉลาด

MACHINE LEARING

Mr.Pichet	Poonsawat	53011127
Mr.Pavis	Limsira	53011199
Assoc.Prof.Kritawan	Siriboon	Advisor
Assoc.Prof.Dr.Boontee	Kruatrachue	Advisor

Academic year 2013

ABSTACT

Nowadays, the application of computer to learn and to solve problems instead of human is more popular. Machine learning is used in various ways for learning and solving their own problems. The reinforcement learning is one of concepts that has been wildly used. This project used reinforcement learning, the machine learning method that learn from the past experience to adjust choice of action in the future, to solve a maze game. By use surrounding environment at that time to be a current state. And use that current state to determine action, not any others state. However, the set of action must remain the same. This project will teach computer to learn problem and solution of maze game until it can solve the game intelligently.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สามารถสำเร็จได้ด้วยดี จากความอนุเคราะห์ของคณะอาจารย์และครอบครัวของผู้จัดทำ ผู้จัดทำขอขอบคุณ รศ.กฤตวัน ศิริบูรณ์ และรศ.ดร.บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษาโครงการ ที่ให้คำแนะนำในการทำโครงงาน เสนอแนวคิดที่สามารถนำมาปรับปรุงแก้ไขในโครงการ ให้ตัวอย่างที่สำคัญ รวมทั้งสอนทฤษฎีที่เกี่ยวข้อง ขอขอบคุณสาขาวิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เพื่อให้โครงการเป็นไปได้อย่างสะดวก

คณะผู้จัดทำจึงขอขอบคุณมา ณ ที่นี้

นายพิเชษฐ์ พูลสวัสดิ์

นายภวิศ ลิมป์ศิระ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

หน้า

สารบัญตาราง

หน้า

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

การเรียนรู้ของเครื่อง (Machine Learning) คือการที่เครื่องคอมพิวเตอร์นั้น สามารถเรียนรู้ปัญหาคลอดจนสามารถแก้ไข้ปัญหาดังกล่าวได้อย่างชาญฉลาด โดยที่การเรียนรู้ของเครื่องนั้นสามารถทำได้หลากหลายวิธี ซึ่งวิธีที่แพร่หลายที่สุดคือการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) โดยผลักดันให้ผู้เรียนรู้มีความสามารถที่จะเรียนรู้จากประสบการณ์ในอดีต และนำมาปรับปรุงแก้ไขเพื่อใช้ในการแก้ไขปัญหาดังกล่าวต่อไป

การเรียนรู้ของเครื่อง ถูกนำไปประยุกต์ใช้อย่างแพร่หลาย ไม่ว่าจะเป็นเกม หุ่นยนต์ ตลอดจนธุรกิจต่างๆ ดังนั้น การศึกษารูปแบบการเรียนรู้แบบเสริมกำลัง จึงเป็นประโยชน์และเหมาะแก่การศึกษาให้เข้าใจ เพื่อให้สามารถนำไปประยุกต์ใช้ในชีวิตจริงได้ และเพื่อเรียนรู้วิธีที่จะทำให้เครื่องสามารถแก้ไขปัญหาได้ดีและมีประสิทธิภาพต่อไป

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อให้เข้าใจหลักแนวคิดของวิธีการแก้ไขปัญหาต่างๆ ในการเรียนรู้แบบเสริมกำลัง
2. เพื่อให้เข้าใจการทำงานของระบบที่มีการเรียนรู้
3. เพื่อศึกษาวิธีการแก้ปัญหาของเกมเขาวงกตด้วยการเรียนรู้แบบเสริมกำลัง
4. เพื่อให้สามารถนำการเรียนรู้ของเครื่องไปใช้ในการแก้ไขปัญหาต่างๆ ในชีวิตจริง

1.3 ขอบเขตของโครงงาน

1. คอมพิวเตอร์สามารถเรียนรู้การทำงานของระบบเองได้
2. คอมพิวเตอร์สามารถหาวิธีการที่ดีเพียงพอ ในการทำงานของระบบได้เอง
3. คอมพิวเตอร์มีหน่วยความจำและหน่วยประมวลผลที่มีประสิทธิภาพสูง

1.4 วิธีการดำเนินการ

1. ศึกษาทฤษฎีการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)
2. ศึกษากระบวนการตัดสินใจแบบมาร์คอฟ (Markov Decision Process)
3. ศึกษาทฤษฎีการเรียนรู้เสริมกำลังแบบ Q-Learning
4. ทดลองการใช้ Q-Learning ในการแก้ปัญหาเขาวงกต
5. ทดลองสร้างเกมเขาวงกตจากที่ทำการศึกษาและทดลองที่ผ่านมา

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. เข้าใจหลักแนวคิดของการเรียนรู้แบบเสริมกำลัง
2. สามารถนำการเรียนรู้แบบเสริมกำลังไปประยุกต์ใช้เป็นการเรียนรู้ในหลายๆ เรื่องได้
3. สามารถวิเคราะห์ได้ว่าแต่ละปัญหาควรใช้วิธีการใดในการแก้ไขปัญหาได้อย่างเหมาะสม
4. สามารถใช้หลักการของการเรียนรู้แบบเสริมกำลังไปใช้ในการแก้ไขปัญหาต่างๆในชีวิตจริงได้

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาโดยทั่วไปออกเป็น 5 บทด้วยกัน

บทที่ 1 บทนำ กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงาน ทฤษฎี หลักการ ความรู้ที่เกี่ยวข้องกับโครงงาน

บทที่ 3 การออกแบบและพัฒนา กล่าวถึงรายละเอียดการออกแบบและการพัฒนาโครงงานนี้ บรรยายส่วนของการทำงานของระบบ

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึงการเตรียมการทดลอง ทั้งการจัดเตรียมส่วนฮาร์ดแวร์ ส่วนซอฟต์แวร์ สภาวะแวดล้อมในการทำการทดลอง ข้อมูลทดลองหรือทดสอบ การทำงานหรือการจำลองการทำงานของระบบ ผลการทดลอง ค่าสมรรถนะของระบบ การวัดประสิทธิภาพของระบบ และการวิเคราะห์ผลการทดลองหรือผลการทำงาน

บทที่ 5 บทสรุป กล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน ข้อจำกัด รวมถึงปัญหาอุปสรรคต่างๆ ของโครงงาน และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

ภาคผนวก

บทที่ 2

ทฤษฎีที่เกี่ยวของ

2.1 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)

การเรียนรู้แบบเสริมกำลัง คือ การเรียนรู้ว่าจะต้องทาอะไร ให้เหมาะสมกับสถานการณ์ โดยผลของการกระทำจะให้ค่าตอบแทน (Reward) ในรูปแบบตัวเลข ผู้เรียนจะไม่ถูกบอกว่าจะต้องกระทำอะไรถึงจะได้ค่าตอบแทนที่มากที่สุด แต่ผู้เรียนต้องค้นพบการกระทำที่จะได้รับค่าตอบแทนค่าตอบแทนที่มากที่สุดจากการกระทำของผู้เรียนเอง การกระทำอาจไม่ได้ส่งผลต่อค่าตอบแทนในทันที แต่การค้นหา การทดลอง ข้อผิดพลาดและค่าตอบแทนล่าช้า ก็เป็นสิ่งสำคัญที่ทาให้การเรียนรู้แบบเสริมกำลังมีคุณสมบัติที่แตกต่างกันด้วย

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

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

2.2 องค์ประกอบของการเรียนรู้แบบเสริมกำลัง

2.2.1 สถานะ (State)

สถานะของปัญญาประดิษฐ์มีความจาเป็นมากในการตัดสินใจทาสิ่งต่างๆ โดยสถานะของปัญญาประดิษฐ์นั้นจะถูกตัดสินโดยสภาพแวดล้อมของปัญญาประดิษฐ์นั้นๆ ในสถานะจะประกอบด้วยค่าตอบแทน ค่าผลกาไร การกระทำต่างๆ และโอกาสในการทาการกระทำต่างๆ ซึ่งค่าเหล่านี้จะเป็นตัวตัดสินใจกำหนดการกระทำของปัญญาประดิษฐ์ ซึ่งจะกล่าวในหัวข้อถัดไป

2.2.2 ค่าตอบแทน (Reward)

ค่าตอบแทนจะได้มาจากการกระทำต่างๆ ในสถานะนั้นๆ เมื่อปัญญาประดิษฐ์ได้ทาการกระทำอย่างใดอย่างหนึ่งก็จะได้รับค่าตอบแทนมา และเปลี่ยนสถานะไปยังสถานะถัดไป

2.2.3 ค่าผลกาไร (Return)

ปัญญาประดิษฐ์จะทราบได้อย่างไรว่า ควรจะทาการกระทำใดจึงจะเกิดผลตอบแทนสูงสุด ซึ่งผลตอบแทนที่ต้องคำนึงนั้น ไม่ใช่แค่เพียงค่าตอบแทน (Reward) เพียงอย่างเดียว แต่ต้องคำนึงถึงค่าผลกาไร (Return) ด้วย

2.2.4 การกระทำ (Action)

ในหนึ่งสถานะสามารถของปัญญาประดิษฐ์มีได้หลายการกระทำ ซึ่งเมื่อทาการกระทำแล้วสิ่งที่ปัญญาประดิษฐ์จะได้รับก็คือค่าตอบแทนของการกระทำนั้น และไปสู่สถานะถัดไป โดยในการตัดสินใจทาการกระทำนั้นจะขึ้นอยู่กับความน่าจะเป็นของการกระทำนั้นๆ

2.2.5 ความน่าจะเป็นในการกระทำ (Policy)

ความน่าจะเป็นของการกระทำในสถานะหนึ่ง ความเป็นไปได้ในการกระทำ จะถูกเก็บไว้ในรูปแบบของเซตคู่อันดับ $\pi_t(s, a)$ ซึ่งจะบ่งบอกว่า การกระทำใดมีความน่าจะเป็นที่จะทาเท่าใด โดยอาจกำหนดเป็นค่าคงที่หรือตัวแปรก็ได้

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

ในสถานะหนึ่ง จะมีค่าที่บ่งชี้โอกาสได้รับผลกาไรเก็บอยู่ ซึ่งหากค่านี้มีค่ามากนั่นหมายความว่าในสถานะนั้นๆ ปัญญาประดิษฐ์จะสามารถแสวงหาผลกาไรได้มากจากการกระทำต่างๆ ในสถานะนั้นๆ

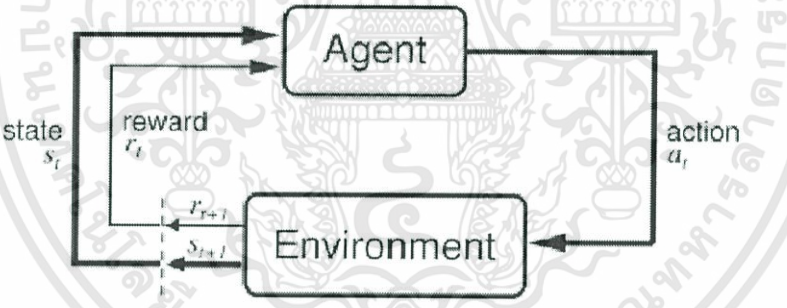
2.2.7 ค่าของการกระทำ (Action Value, Q^{π})

ค่าของการกระทำจะจะเป็นค่าที่บ่งบอกว่า ในการกระทำนั้นๆ จะมีโอกาสแสวงหาผลกาไรได้มากเพียงใด ซึ่งในสถานะหนึ่งสถานะ จะมีค่าของการกระทำหลายค่า เท่ากับจานวนการกระทำที่สามารถทาได้ ซึ่งค่าของสถานะ ( ) และค่าของการกระทำ ( ) นั้นรวมกันเรียกว่า ค่าผลกาไรที่คาดหวัง (Expect Return)

2.3 ปัญหาของการเรียนรู้แบบเสริมกำลัง (Reinforcement Leaning Problem)

โจทย์ปัญหาเกี่ยวกับการเรียนรู้แบบเสริมกำลังเปรียบเสมือนกับโครงสร้างพื้นฐานของปัญหาเกี่ยวกับการเรียนรู้ผ่านการโต้ตอบเพื่อให้บรรลุเป้าหมายที่ต้องการระบบที่ทาการตัดสินใจและเรียนรู้เรียกว่า ผู้เรียน (agent) ซึ่งเป็นระบบที่โต้ตอบกับสภาพแวดล้อม (environment) หรือระบบที่อยู่ภายนอกผู้เรียนการโต้ตอบระหว่างกันของผู้เรียนกับสภาพแวดล้อมจะเป็นไปอย่างต่อเนื่องจากการเลือกกระทำของผู้เรียนและการตอบสนองต่อการกระทำเหล่านั้นของสภาพแวดล้อมด้วยการให้ข้อมูลเกี่ยวกับสถานการณ์ปัจจุบัน

นอกจากนี้สภาพแวดล้อมยังเป็นสิ่งที่กำหนดรางวัลที่ผู้เรียนจะได้รับซึ่งเป็นผลของการตัดสินใจของผู้เรียนตลอดอายุการเรียนรู้ของรางวัลที่มีค่าสูงที่สุดคือสิ่งที่ผู้เรียนพยายามเรียนรู้และปรับตัวเพื่อให้ได้มารายละเอียดทั้งของสภาพแวดล้อมเป็นสิ่งที่กำหนดทาสก์ (Task) ซึ่งเป็นหนึ่งในองค์ประกอบของโจทย์ปัญหาเกี่ยวกับการเรียนรู้แบบเสริมกำลัง



รูป 2.1 การโต้ตอบระหว่างผู้เรียนและสภาพแวดล้อมในการเรียนรู้แบบเสริมกำลัง

รูป 2.1 แสดงแผนภาพการโต้ตอบระหว่างผู้เรียนกับสภาพแวดล้อมตามลาดับขั้นเวลา t = 0, 1, 2,... ในแต่ละขั้นเวลา (time step) ผู้เรียนจะได้รับสภาพแวดล้อมปัจจุบันมาในรูปของ state, S_t , $\in$ S โดยที่ S คือเซตของสถานะที่เป็นไปได้ทั้งหมด จากนั้นผู้เรียนรู้ต้องทาการเลือกการกระทำ $a_t \in a(s_t)$ หมายถึงเซตของการกระทำที่มีอยู่ของสถานะ S_t และจะได้รับผลตอบแทนเป็นเลขจานวนหนึ่งกลับมา $\mathbf{r}_t \in$ R และพาตัวเองไปยังสถานะใหม่ (S_{t+1}) ต่อไป การตัดสินใจเลือกการกระทำของผู้เรียนในแต่ละลาดับขั้นเวลาจะเป็นไปตามฟังก์ชัน เรียกว่า โพลิซี (policy) ซึ่งแทนด้วยสัญลักษณ์ π_t โดย $\pi_t(s, a)$ หมายถึงความน่าจะเป็นที่ $a_t = a$ เมื่อ $s_t = s$ วิธีการต่างๆ ในการเรียนรู้แบบเสริมกำลังจะกำหนดการปรับเปลี่ยนของผู้เรียนผ่าน

ประสบการณ์ของผู้เรียนเอง เป้าหมายของผู้เรียนคือการให้ได้มาซึ่งค่าคาดหมายของผลตอบแทน (expected return) ที่สูงที่สุด ในที่นี้ผลตอบแทน R_1 หมายถึงฟังก์ชันของลาดับของรางวัลที่ได้รับหลังจากลาดับขั้นเวลา t ซึ่งสามารถเขียนได้ในรูปแบบของเรขาคณิต ฟังก์ชัน สามารถเขียนให้อยู่ในรูปทั่วไป ดังนี้

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

เมื่อ γ เป็นค่าอัตราการลดค่า (Discount rate) และ T เป็นค่าของลำดับชั้นเวลาขั้นสุดท้ายในกรณีที่ทาสก์ของผู้เรียนสามารถแบ่งเป็นเอพิโสด (Episode) ได้ γ จะมีค่าเป็น 1 และ T จะต้องมีค่าจำกัดซึ่งไม่เท่ากับ ∞ ในกรณีที่ทาสก์ของผู้เรียนเป็นแบบต่อเนื่อง γ จะมีค่าอยู่ในช่วง (0,1) และ T จะมีค่าเข้าใกล้ ∞ เนื่องจากเป็นทาสก์ที่ไม่มีสถานะจบและไม่มีการเริ่มต้นเอพิโสดใหม่

ตัวอย่างที่เป็นแบบเอพิโสด เช่น การแข่งขันเทนนิส อาจจะให้แต้มครั้งหนึ่งของการเล่นเป็นหนึ่งเอพิโสด ภายในเอพิโสดอาจจะมีสถานะว่าตอนนี้อยู่ที่จุดไหนของคอร์ด และการกระทำก็คือการการตีลูกไปลงจุดไหน เมื่อมีผลแพ้ชนะในแต้มนั้นก็จบเอพิโสดแล้วก็นาค่ารีเทิร์น (return) นี้อัพเดตแต่ละสถานะแล้วก็เริ่มเอพิโสดใหม่

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

2.4 คุณสมบัติมาร์คอฟ (The Markov Property)

ผู้เรียนจะทาการตัดสินใจบนพื้นฐานของสัญญาณจากสิ่งแวดล้อม โดยการกำหนดคุณสมบัติของสิ่งแวดล้อมและสัญญาณสถานะ (State signal) ซึ่งมีการให้ความสนใจโดยเฉพาะเรียกว่า คุณสมบัติมาร์คอฟ ซึ่งจะให้ความสำคัญเฉพาะข้อมูลที่เป็นประโยชน์ต่อผู้เรียน กล่าวคือสิ่งที่พิจารณาเป็นสำคัญคือ การกระทำใดที่จะทาเมื่อไหร่ก็ตามที่สัญญาณสถานะสามารถใช้ได้ สัญญาณสถานะประกอบด้วยสิ่งที่เซ็นเซอร์ของผู้เรียนรับรู้ได้ในปัจจุบัน แต่บางครั้งยังรวมไปถึงข้อมูลของอดีตอีกด้วย ตัวอย่างเช่น การที่คนเราได้ยินคาตอบว่า

"ใช่" ตอบกลับมา การที่เราจะสามารถแยกแยะได้ว่า มันอยู่ในสถานะไหนนั้น ขึ้นอยู่กับข้อมูลในอดีตซึ่งก็คือคำถามที่ได้ถามไปก่อนหน้านั้น

สัญญาณสถานะที่มีการสรุปข้อมูลในอดีตที่มีความเกี่ยวข้องเก็บไว้ จะเรียกสัญญาณสถานะนั้นว่าเป็นมาร์คอพหรือเรียกว่ามีคุณสมบัติแบบมาร์คอพ โดยสามารถแทนคุณสมบัติแบบมาร์คอพได้ด้วยสมการทางคณิตศาสตร์อย่างง่าย ซึ่งสมมุติให้มีสถานะและค่าของรางวัลที่จำกัดได้ ซึ่งในกรณีปกติเราจะได้สมการความน่าจะเป็นที่ต้องอ้างถึงเวลาดังนี้

$$\Pr = \{S_{t+1} = s', r_{t+1} = r \,|s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_0, s_0, a_0\}$$

ถ้าสถานะมีคุณสมบัติแบบมาร์คอพ จะทำให้สิ่งแวดล้อมที่ตอบสนองที่เวลา t+1 จะขึ้นอยู่กับ State และ Action ของเวลา t เท่านั้น จึงทำให้เราสามารถแทนสมการข้างต้นได้ด้วยสมการ

$$\Pr = \{s_{t+1} = s', r_{t+1} = r \,|s_t, a_t$$

2.5 กระบวนการตัดสินใจมาร์คอพ (Markov Decision Process)

กระบวนการตัดสินใจมาร์คอพเป็นกระบวนการที่ช่วยเสริมให้การเรียนรู้แบบเสริมกำลังและกำหนดการพลวัต สามารถเลือกเส้นทางที่เหมาะสมและดีขึ้น โดยเป็นการสร้างกรอบการตัดสินใจด้วยการสร้างแบบจำลองที่ผลของสถานการณ์หนึ่งเป็นแบบสุ่มและอีกส่วนหนึ่งอยู่ภายใต้การควบคุมของตัวตัดสินใจและกำหนดรายละเอียดทั้งหมดเกี่ยวกับสภาพแวดล้อมในการเรียนรู้แบบเสริมกำลัง และมีส่วนสำคัญในการทำความเข้าใจวิธีแก้ปัญหาที่เกี่ยวกับการเรียนรู้แบบเสริมกำลัง เนื่องจากสามารถอธิบายการเปลี่ยนแปลงต่าง ๆ ที่เกิดขึ้นในการเรียนรู้แบบเสริมกำลังได้เป็นอย่างดี

กระบวนการตัดสินใจแบบมาร์คอพ (Markov Decision Process : MDP) ซึ่งประกอบด้วย เซตของสถานะที่เป็นไปได้ เซตของการกระทำทั้งหมดที่เป็นไปได้ ความน่าจะเป็นของการเปลี่ยนสถานะ (Transition Probabilities) และค่าคาดหมายของรางวัลที่จะได้รับโดยทันที (Expected Immediate Reward) ความน่าจะเป็นของการเปลี่ยนสถานะจากสถานะไปสถานะ เมื่อเลือกการกระทำ สามารถเขียนในรูปทั่วไปได้ดังนี้

$$P^a_{ss'} = Pr\{S_{t+1} = s' \,|s_t = s, a_t = a\}$$

และค่าคาดหมายของรางวัลที่จะได้รับโดยทันทีหลังจากเปลี่ยนสถานะจากสถานะไปสถานะ จากการเลือกการกระทำ สามารถเขียนในรูปทั่วไปได้ดังนี้

$$R_{ss'}^{a} = E\{r_{t+1}|s_t = s, a_t = a, s_{t+1} = s\}$$

2.6 Q-learning

เป็นวิธีการสำหรับใช้ในการทำให้เครื่องคอมพิวเตอร์สามารถเรียนรู้การแก้ปัญหา ซึ่ง Q-learning เป็นหนึ่งในเทคนิคของ Reinforcement Learning โดยใช้หลักการในการลองผิดลองถูกเพื่อที่จะหาการกระทำที่ทำให้ได้รับสิ่งตอบแทนที่พึงพอใจซึ่งเป็นผลจากการกระทำนั้นๆ โดย Q-learning ถือได้ว่าเป็นวิธีการที่นำมาหาการกระทำที่ให้ผลลัพธ์ได้ดีที่สุด ณ สถานะหนึ่งๆในปัญหาภายใต้ข้อจำกัดของ Markov decision process ซึ่งจะใช้หลักการของการตัดสินใจเลือกการกระทำจากฟังก์ชันคำนวณค่าของการกระทำโดยฟังก์ชันนี้จะทำการค้นหาว่าการกระทำใดมีค่าสูงที่สุดจากการกระทำที่เป็นไปได้ทั้งหมด ณ สถานะนั้นๆโดยค่าของการกระทำดังกล่าวจะถูกเรียกว่าค่าคิว จากนั้นก็จะเลือกการกระทำดังกล่าวมาใช้ ณ สถานะนั้นๆ เนื่องจากค่าคิวในตอนแรกนั้นยังไม่สามารถบ่งบอกได้ว่าการกระทำใดที่ควรจะถูกเลือก เราต้องทำให้ฟังก์ชันดังกล่าวเรียนรู้ว่าควรจะให้เปลี่ยนแปลงค่าคิวของการกระทำต่างๆอย่างไร โดยเรียนรู้จากการลองผิดลองถูกไปก่อนเพื่อที่จะสังเกตว่าผลลัพธ์จากการกระทำนั้นๆส่งผลอะไรย้อนกลับมา ถ้าการกระทำนั้นส่งผลในทางบวกของปัญหานั้นๆก็จะได้รับสิ่งตอบแทนที่ดี ในทางกลับกันถ้าส่งผลไปในทางลบก็จะได้รับสิ่งตอบแทนที่ไม่ดีย้อนกลับ จากนั้นฟังก์ชันก็จะทำการปรับค่าคิวของการกระทำที่เป็นต้นเหตุของผลลัพธ์ให้สอดคล้องกับสิ่งตอบแทนเพื่อให้รอบต่อมาที่กลับมา ณ สถานะนี้อีกครั้งก็สามารถที่จะเรียนรู้ว่าควรจะเลือกการกระทำใดจากค่าคิวที่มีการปรับค่ามาจากสิ่งที่เคยทำผ่านมา ทำให้เมื่อมีการลองผิดลองถูกไปนานๆก็ยิ่งสามารถที่จะหาการกระทำที่ดีที่สุด ณ สถานะนั้นๆของปัญหา

จะเห็นได้ว่าข้อดีของ Q-learning นั้นสามารถที่จะเลือกการกระทำที่ดีที่สุด ณ สถานะต่างๆได้จากค่าคิวของแต่ละการกระทำที่เป็นไปได้ทั้งหมดเพียงอย่างเดียว ไม่จำเป็นต้องอาศัยปัจจัยอื่นๆเข้ามาคำนึงทำให้การคำนวณเป็นไปได้อย่างรวดเร็ว ซึ่งวิธีการนี้ได้รับรองมาถึงปัจจุบันว่าเป็นวิธีการที่สามารถค้นหาวิธีการที่ดีที่สุดสำหรับปัญหาภายใต้ข้อจำกัด Markov decision process

2.6.1 Algorithm

เนื่องจากปัญหาที่อยู่ภายใต้ข้อจำกัด Markov decision process นั้นประกอบด้วย ตัวกระทำหรือเอเจนต์ สถานะ S และ เซทของการกระทำที่เป็นไปได้ของแต่ละสถานะ A โดยแต่ละการการะทำ a เป็นสมาชิกภายใต้เซท A โดยเอเจนต์สามารถเคลื่อนย้ายจากสถานะหนึ่งไปอีกสถานะหนึ่งได้โดยผ่านการกระทำที่เลือก โดยแต่ละครั้งที่เปลี่ยนสถานะจะมีการให้ค่ารางวัลเป็นผลตอบแทนจากการกระทำที่ส่งผลให้มีการ

เปลี่ยนสถานะโดยค่ารางวัลดังกล่าวจะคำนึงจากสถานะต่อไปที่เกิดจากการกระทำดังกล่าว โดยเป้าหมายของเอเจนต์คือการเลือกเส้นทางหรือลำดับการกระทำที่ส่งผลให้ได้ค่ารางวัลรวมสูงสุด ซึ่งการที่จะสามารถหาลำดับการกระทำดังกล่าวได้นั้นจำเป็นต้องมีการเรียนรู้จนถึงขั้นที่สามารถหาได้ว่าการกระทำใดที่ดีที่สุดสำหรับแต่ละสถานะในปัญหานั้นๆ โดยฟังก์ชันที่นำมาคำนวณค่าคิวของการกระทำของแต่ละสถานะนั้นเป็นไปตามสมการดังนี้

$$Q : S \times A \longrightarrow R$$

โดยค่าคิวก่อนมีการเรียนรู้จะขึ้นอยู่กับค่าเริ่มต้นที่ตั้งค่าโดยผู้ออกแบบ จากนั้นเมื่อเอเจนต์มีการเลือกการกระทำและดูผลตอบแทนที่ได้รับกลับมาหรือค่ารางวัลที่เกิดจากการเปลี่ยนไปอีกสถานะหนึ่ง โดยสถานะต่อไปจึงขึ้นอยู่กับสถานะปัจจุบันกับการกระทำที่เลือกใช้เพื่อเปลี่ยนไปอีกสถานะอื่น ซึ่งหัวใจหลักของหลักการของ Q-learning คือการวนซ้ำปรับค่าให้กับค่าคิวของแต่ละการกระทำของแต่ละสถานะทุกครั้งที่มีการเลือกการกระทำ โดยคำนึงจากปัจจัยสองอย่างหลักๆคือ ค่าคิวเดิมของการกระทำ และ ผลลัพธ์ใหม่ที่ได้จากการกระทำนั้นๆ ซึ่งเป็นไปตามสมการดังนี้

$$Q_{t+1}(s_t, a_t) = \underbrace{(1 - \alpha_t(s_t, a_t))}_{\text{inverse learning rate}} \times \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \overbrace{\left[\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}\right]}^{\text{learned value}}$$

โดย R_{t+1} คือค่ารางวัลที่ได้ตอบกลับจากการกระทำ a_t ณ สถานะ s_t โดย $\alpha(s,a)$ ซึ่งมีค่าในช่วง $0 < \alpha \leq 1$ เป็นค่าอัตราการเรียนรู้ ซึ่งจะเป็นตัวที่เข้ามาคำนวณว่าผลจากการกระทำใหม่นั้นจะส่งผลต่อการเปลี่ยนแปลงข้อมูลเดิมที่สะสมมาตั้งแต่เริ่มต้นมากน้อยเพียงใด โดยยิ่งค่าน้อยจะส่งผลน้อย ซึ่งถ้าเป็น 0 ก็จะทำให้เอเจนต์ไม่มีการเรียนรู้อะไรเพิ่มเลยก็จะเชื่อแค่สิ่งที่เคยเจอมาเท่านั้น ในทางกลับกันถ้ายิ่งค่ามาก ก็จะทำให้เอเจนต์ให้ความสำคัญกับผลตอบแทนใหม่ที่ได้รับมามากกว่าค่าเดิมที่มีอยู่ ซึ่งถ้ามีค่าเท่ากับ 1 ก็จะทำให้เอเจนต์เชื่อลบความเชื่อเก่าๆที่เคยเรียนมาทั้งหมด แล้วเชื่อความรู้ใหม่ที่เกิดจากการกระทำครั้งล่าสุดเท่านั้น ซึ่งในทางปฏิบัตินิยมตั้งค่า α เป็นค่าคงที่ไม่เปลี่ยนแปลงไปตามเวลา โดยค่าจะขึ้นอยู่กับปัญหาที่ต้องแก้เป็นหลัก และอีกค่าหนึ่งที่เป็นปัจจัยสำคัญคือ γ ซึ่งเรียกว่าค่าปัจจัยการลดซึ่งมีค่าอยู่ในช่วง $0 \leq \gamma \leq 1$ โดยตัวแปรนี้เป็นส่วนมาเข้ามาคำนึงถึงความสำคัญระหว่างผลตอบแทนในภายหน้าคือมองไป ณ สถานะต่อไปว่าจะได้รับผลตอบแทนอะไรกลับมาซึ่งคำนวณจากค่าคิวสูงสุดของการกระทำในสถานะต่อไป โดยยิ่งค่า γ ยิ่งน้อยหรือเท่ากับ 0 ก็จะทำให้เอเจนต์มีวิสัยทัศน์แคบคือไม่มองการณ์ไกลมองแค่ปัจจัย ในที่นี้คือคำนึงแค่ค่ารางวัลที่ได้รับมาเท่านั้นไม่มองว่า สถานะที่ไปต่อนั้นมันดีหรือไม่ ในทางกลับกันถ้าค่า γ ยิ่งสูงก็จะทำให้เอเจนต์ให้ความสำคัญกับสิ่งที่

อาจจะเกิดขึ้นในภายหน้าหรือมองการณ์ไกล โดยจะให้ความสำคัญกับผลตอบแทนในระยะยาวมากกว่าระยะสั้น

ในทางปฏิบัตินั้นแต่ละฉากของการแก้ปัญหาจะจบเมื่อเอเจนต์ถึงสถานะที่เป็นเป้าหมายจากนั้นก็ทำการเริ่มใหม่โดยยังคงเก็บค่าคิวของแต่ละการกระทำไว้ แต่อย่างไรก็ตามเอเจนต์สามารถแก้ปัญหาในรูปแบบที่ไม่สิ้นสุดได้เช่นเดียวกันโดยมีข้อจำกัดว่าค่า γ ต้องมีค่าต่ำกว่า 1 เนื่องจากค่าคิวของการกระทำจะลู่เข้าจนมีค่านิ่งถึงแม้ว่าเป็นการแก้ปัญหาที่ไม่มีวันสิ้นสุดก็ตาม

2.7 Epsilon Greedy Policy

เป็นวิธีการที่จะนำมาใช้กับการแก้ปัญหาที่ยังไม่รู้ว่าคำตอบที่ดีที่สุดนั้นคืออะไร จึงต้องมีการค้นหาโดยการสุ่มโดยคาดหวังว่าค่าที่สุ่มมานั้นอาจเจออะไรที่ดีกว่าเดิมก็เป็นไปได้นั่นเอง โดยหลักการของวิธีการนี้ที่นำมาประยุกต์ใช้กับการเรียนรู้ของเครื่องคือการเลือกการกระทำที่เป็นไปได้ ณ สถานะหนึ่งๆโดยการสุ่ม โดยใช้หลักการของความน่าจะเป็นเข้ามาใช้ ซึ่งเรียกค่าความน่าจะเป็นในการใช้หลักการของ Epsilon Greedy ว่าความน่าจะเป็นของเอปไซลอน(ε) ซึ่งความน่าจะเป็นที่เหลือคือ 1- ε จะเป็นส่วนที่เลือกการกระทำที่มีค่าดีที่สุด ณ ขณะนั้นๆ จะเห็นได้ว่าหลักการนี้เข้ามาช่วยให้สามารถเกิดความสมดุลระหว่างการค้นหาคำตอบใหม่ๆ และการเข้าหาเป้าหมายให้ไวที่สุด

2.8 กำหนดการพลวัต (Dynamic Programming)

กำหนดการพลวัตมีขึ้นมาเพื่อใช้ในการคำนวณนโยบายดีเยี่ยม โดยกำหนดแบบจำลอง สิ่งแวดล้อมที่สมบูรณ์ให้เป็นการตัดสินใจมาร์คอฟ ให้สามารถแก้ปัญหาได้อย่างมีประสิทธิภาพสูงสุด โดยใช้ฟังก์ชั่นมูลค่าในการค้นหานโยบายที่ดี ซึ่งจะสามารถได้รับนโยบายดีเยี่ยมเมื่อพบฟังก์ชั่นมูลค่าดี เยี่ยม โดยจะเป็นการหาค่าต่างๆ ของสถานะ โดยค่าที่สนใจนั่นคือค่า π, $Q\pi$ และ $V\pi$

$$V\pi\ s = maxa\ Rt \mid Rt = Return$$

2.8.1 การหาค่าความน่าจะเป็นการทำการกระทำที่ดีที่สุด

ความน่าจะเป็นในการทำการกระทำจะดีที่สุดก็ต่อเมื่อ มีการกระทำที่ทำให้ได้ค่าของการ กระทำในสถานะนั้นๆ สูงที่สุดนั่นคือ

$$\pi\ s = argmaxa\ Q(s,a)$$

2.9 Heuristic

เป็นวิธีการหรือเทคนิคที่นำมาใช้ในแก้ไขปัญหาซึ่งได้มาของหนทางหรือคำตอบที่ไม่รองรับว่าเป็นคำตอบที่ดีที่สุดเนื่องจากวิธีการดังกล่าวนี้จะใช้หลักการบนเทคนิคแบบ Experience-based ซึ่งเป็นเทคนิคที่จะอิงจากประสบการณ์ที่เคยเรียนรู้มาแล้วนั้นมาประยุกต์ใช้แก้ไขปัญหาใหม่ที่คล้ายคลึงกัน สาเหตุที่เทคนิคแบบฮิวริสติกถูกนำมาใช้นั้นเนื่องจากในการแก้ไขปัญหาที่มีโดเมนของคำตอบที่เป็นไปได้ทั้งหมดขนาดใหญ่แล้วนั้นการที่จะใช้วิธีการแก้ปัญหาเพื่อที่จะได้คำตอบที่ดีที่สุดก็มีเพียงวิธีการเดียวคือ Brute force โดยเทคนิคนี้จะนำมาซึ่งการค้นหาคำตอบแบบ Exhaustive Search ซึ่งเป็นวิธีการค้นหาคำตอบที่จะเข้าไปดูคำตอบที่เป็นไปได้ทั้งหมดของปัญหานั้นๆ แล้วเลือกคำตอบที่ดีที่สุดออกมาได้อย่างแน่นอน แต่เมื่อนำมาใช้กับปัญหาขนาดใหญ่นั้นในทางปฏิบัติแทบจะเป็นไปไม่ได้เลยที่จะนำวิธีการนี้มาใช้เนื่องจากตัวแปรที่เราไม่สามารถเข้าไปแก้ไขได้และมีผลมากกระทบที่สุดคือ เวลา ทำให้ต้องคิดค้นวิธีการอื่นขึ้นมาแทนซึ่งหนึ่งในนั้นคือฮิวริสติก โดยวิธีการนี้จะเข้ามาลดตัวแปรในเชิงเวลาลงแต่ต้องแลกกับคำตอบที่ไม่สามารถการันตีได้ว่าดีที่สุดแต่ยอมรับได้ โดยในปัจจุบันมีหลากหลายวิธีการที่นำมารองรับเทคนิดของฮิวริสติก ตัวอย่างเช่น Rule of Thumb, Educated Guess หรือไม่ว่าจะเป็นการใช้จิตใต้สำนึกก็ถือว่าเป็นหลักการภายใต้เทคนิคฮิวริสติก และวิธีการที่ถือได้ว่าเป็นพื้นฐานที่สุดของฮิวริสติกก็คือการลองผิดลองถูกกับปัญหาดังกล่าวเพื่อเรียนรู้ว่าอะไรคือสิ่งที่ควรทำหรือไม่ควรทำจนในที่สุดเมื่อเราเรียบรู้ไปเรื่อยๆแล้วนั้นก็จะสามารถรู้ได้ว่าเมื่อเจอเหตุการณ์แบบไหนควรจะทำอะไรและสามารถนำไปใช้กับปัญหาอื่นๆที่คล้ายกันต่อไป

ในเชิงจิตวิทยาแล้วนั้นได้อธิบายเกี่ยวกับฮิวริสติกว่าเป็นหลักการที่นำมาอธิบายการตัดสินใจภายใต้กระบวนการคิดของมนุษย์ซึ่งใช้ในการแก้ปัญหาที่ซับซ้อนรวมถึงข้อมูลที่มีให้นั้นไม่ครบถ้วน โดยวฮิวริสติกนั้นถูกนำไปใช้ทดลองกับมนุษย์ภายใต้สถานการณ์ต่างๆแล้วค้นพบว่าสามารถใช้ได้ดี แต่สำหรับปัญหาที่ต้องการความแม่นยำสูงนั้นถือว่ายังไม่สามารถนำมาใช้ได้ ซึ่งถ้านำหลักการนี้นั้นมาใช้กับการแก้ปัญหาของเครื่องคอมพิวเตอร์ร่วมกับวิธีที่ทำให้คอมพิวเตอร์เรียนรู้ด้วยตัวเองแล้วนั้นก็สามารถสร้างกฎเกณฑ์ต่างๆได้เองภายใต้หลักการของฮิวริสติกแล้วนำไปใช้กับปัญหาอื่นๆที่คล้ายคลึงกันเพื่อให้สามารถแก้ปัญหาได้รวดเร็วยิ่งขึ้น

บทที่ 3

การออกแบบและพัฒนา

3.1 กฎของตัวเกม

ในการออกแบบตัวเกม ได้มีการกำหนดให้เป็นเกมแก้ปัญหาเขาวงกต โดยจะมีองค์ประกอบหลักคือ ตัวผู้เล่น ตัวผู้ร้าย ทางออก และกำแพง โดยมีข้อกำหนดและกติกาดังนี้

1. ผู้เล่นจะสามารถเดินได้ครั้งละ 1 ช่อง ตาละ 1 ครั้ง โดยสามารถเดินได้เพียง 4 ทิศทาง
2. ผู้ร้ายจะสามารถเดินได้ครั้งละ 1 ช่อง ตาละ 2 ครั้ง โดยสามารถเดินได้เพียง 4 ทิศทาง ในแต่ละครั้ง
3. ผู้ร้ายจะเดินหาผู้เล่นโดยคำนวณจากระยะห่างในแนวแกน X ก่อน แล้วจึงคำนวณในแนวแกน Y หากไม่สามารถเดินได้ (ติดกำแพง) จะไม่เดิน
4. เกมจะจบเมื่อผู้เล่นเดินถึงทางออก หรือ ผู้ร้ายเดินถึงผู้เล่น

3.2 การออกแบบหน้าต่างเกม

การแสดงผลของโปรแกรมจะแบ่งออกเป็นสองส่วนหลักๆ คือ ส่วนที่เป็น Console ซึ่งใช้ในการแสดงผลเพื่อ Debug และส่วนที่เป็นหน้าจอแสดงผลแบบกราฟิกซึ่งจะมีขนาดแสดงผล 800 x 600 Pixels โดยส่วนหน้าจอแสดงผลแบบกราฟิกจะแบ่งได้อีก 4 ส่วน คือ

3.2.1 หน้าจอหลัก (Main Menu)

หน้าจอหลักทำหน้าที่เลือกการทำงานต่างๆ ของโปรแกรมจึงไม่มีรายละเอียดมากมีเพียง อักษรแสดงตัวเลือกการทำงานต่างๆของโปรแกรม

3.2.2 หน้าจอตอนเล่น (Playing Interface)

การแสดงผลขณะผู้ใช้เล่นเกมได้ทำการออกแบบโดยมีพื้นที่ขนาดใหญ่ด้านซ้ายเป็นกระดานที่ใช้แสดงแผนที่ที่กำลังเล่น ตำแหน่งของตัวผู้เล่น ของมิโนทอร์ และทางออก โดยผู้เล่นจะสามารถเลือกแผนที่ที่จะเล่น

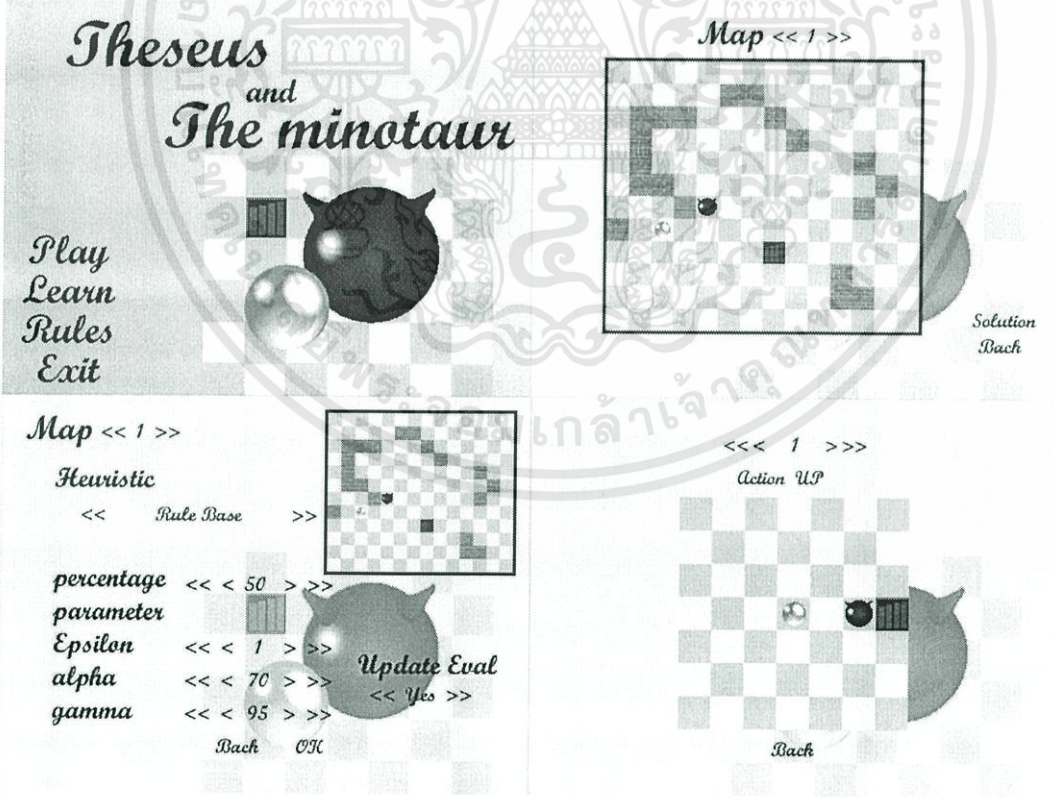
ได้จากตัวเลือกแผนที่ด้านบนของกระดาน และมีตัวเลือกเพื่อช่วยและตัวเลือกเพื่อกลับสู่หน้าจอหลักทางด้านขวา

3.2.3 หน้าจอปรับค่าเพื่อการเรียนรู้ของ AI (AI Configuration Menu)

ในส่วนของการเรียนรู้ของ AI จะต้องปรับค่าต่างๆ เพื่อปรับเปลี่ยนพฤติกรรมการเรียนรู้ของเครื่อง รวมทั้งเลือกแผนที่ที่เครื่องจะเรียนรู้ จึงออกแบบให้มีแผนที่แสดงเล็กๆทางด้านบนขวา และค่าต่างๆที่สามารถปรับได้อยู่ทางด้านซ้ายและด้านล่างของแผนที่ เมื่อเลือกเสร็จเครื่องจะทำการเรียนรู้และแสดงผลทั้งส่วนของ Console ในระหว่างการเรียนรู้และแสดงผลเส้นทางที่เรียนรู้มาในส่วนของการแสดงผลแบบกราฟิก

3.2.4 หน้าจอแสดงกฎต่างที่เรียนรู้มาของ AI (AI's Learned Rules)

ในส่วนของการแสดงกฎต่างๆของ AI จะแสดงกระดานขนาด 7 x 7 เพื่อแสดงรูปแบบการเรียงตัวของตัวละครหลักทั้ง 3 และกำแพง และทางเลือกที่ AI จะเลือกตามรูปแบบนั้นๆ

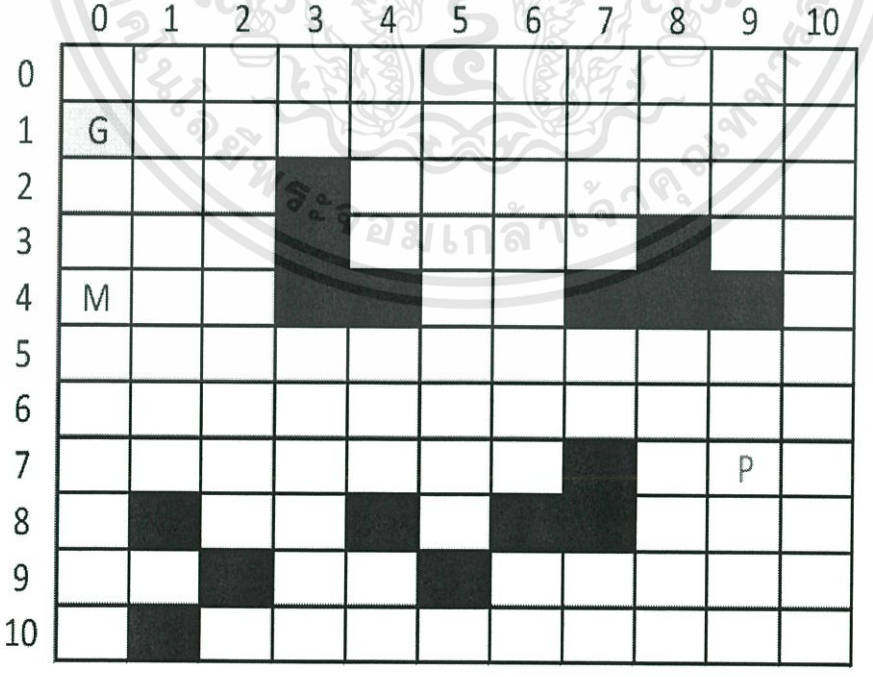


รูป 3.1 รูปแสดงหน้าจอแสดงผลต่างๆของเกม

3.4 การออกแบบส่วนของการเรียนรู้แบบเสริมกำลัง

การออกแบบระบบจะเน้นในเรื่องของการพัฒนาด้านซอฟต์แวร์เป็นหลักในเรื่องของการนำเรื่องการเรียนรู้ของเครื่องแบบเสริมกำลัง (Reinforcement Learning) โดยนำมาแก้ปัญหาในเรื่องของเขาวงกตแต่จะมีตัวเสริมเข้ามาคือตัวร้ายที่คอยเดินเข้ามาฆ่าตัวหลักหรือเอเจนต์ เนื่องจากการเรียนรู้แบบเสริมกำลังนั้นเน้นไปที่ให้เอเจนต์ทำการลองผิดลองถูกแล้วรอดูผลลัพธ์จากการกระทำนั้นๆว่าจะได้รับผลตอบแทนอะไรกับมา จากนั้นเมื่อทำการลองไปเรื่อยๆก็จะสามารถหาการกระทำที่ดีที่สุดออกมาได้นั่นเอง

ในส่วนของการออกแบบจะใช้วิธีการของ Q-learning ซึ่งเป็นหนึ่งในหลักการของการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) โดยวิธีการดังกล่าวจะใช้หลักการของการให้ผลตอบแทนหรือรางวัลกับเอเจนต์เมื่อสามารถทำในสิ่งที่สามารถเข้าถึงเป้าหมายที่กำหนดไว้ได้ ในทางกลับกันเมื่อทำในสิ่งที่ไม่ถูกต้องก็จะให้ผลตอบแทนเช่นกันแต่จะได้ค่าที่บ่งบอกว่าไม่ควรเลือกการกระทำดังกล่าวเสมือนเป็นการลงโทษ ทำให้เมื่อเอเจนต์ลองผิดลองถูกไปเรื่อยๆก็จะสามารถเลือกการกระทำที่ดีที่สุดที่สามารถเข้าถึงเป้าหมายได้ในที่สุดโดยการออกแบบจะอ้างอิงจากหลักการของ Markov property นั่นคือปัญหาที่นำมาเป็นตัวอย่างในการเรียนรู้นั้นต้องอยู่ภายใต้เงื่อนไขคือสถานะ S ต้องมีขอบเขตที่จำกัด และเนื่องจากปัญหาที่อยู่ภายใต้หลักการของ Markov property นั้นคือสามารถเลือกการกระทำได้โดยดูจากสถานะปัจจุบันเท่านั้นก็เพียงพอที่จะสามารถไป ณ สถานะต่อไปได้โดยไม่ต้องย้อนกลับไปดูสิ่งที่เคยทำผ่านมา โดยปัญหาที่เรานำมาแก้นั้นเป็นรูปแบบของเกมซึ่งเป็นรูปแบบของการเขาวงกตที่มีตัวร้ายตามล่า เป้าหมายหลักที่ต้องทำคือออกจากเขาวงกตโดยไม่ถูกตัวร้ายฆ่าให้ได้ โดยภาพรวมของเกมเป็นดังรูป



รูป 3.2 รูปแสดงตัวอย่างรูปแบบแผนที่

โดยจากรูปจะประกอบไปด้วย G คือทางออกจากเขาวงกต P คือเอเจนต์ และ M คือตัวร้ายที่ตามฆ่าเอเจนต์ ซึ่งจากรูปแบบของเกมดังรูปทำให้เราได้ออกแบบสถานะ S ซึ่งประกอบไปด้วย

- Px คือค่าพิกัดในแนวแกน x ของเอเจนต์
- Py คือค่าพิกัดในแนวแกน y ของเอนเจนต์
- Mx คือค่าพิกัดในแนวแกน x ของตัวร้าย
- My คือค่าพิกัดในแนวแกน y ของตัวร้าย
- Gx คือค่าพิกัดในแนวแกน x ของทางออกเขาวงกต
- Gy คือค่าพิกัดในแนวแกน y ของทางออกของเขาวงกต
- setAction คือเซทของการกระทำที่เป็นไปได้ในสถานะนั้นๆ
- Qvalue คือค่าคิวของแต่ละการกระทำใน setAction

จะได้ว่า State ที่เป็นไปได้ทั้งหมดคือขนาด $(XY)^2$ ซึ่งตามรูปคือ 121^2 สถานะ โดยหลักการที่ใช้คือ Q-learning ซึ่งค่าตอบแทนที่ได้รับจะมีอยู่สองค่าหลักๆคือ R_{max} และ R_{min} โดยค่า R_{max} จะได้รับก็ต่อเมื่อสามารถไปถึงสถานะที่ Px เท่ากับ Gx และ Py เท่ากับ Gy หรือถึงทางออกเข้าวงกตนั่นเอง ในทางตรงกันข้าม R_{min} จะให้ก็ต่อเมื่อเอเจนต์ถูกตัวร้ายฆ่า ซึ่งก็คือสถานะที่ Px เท่ากับ Mx และ Py เท่ากับ My ส่วนสถานะที่เหลือค่าตอบแทนจะเท่ากับศูนย์และเราได้มีการออกแบบให้การกระทำเป็นรูปแบบเดียวกับเกมคือเซทของการกระทำที่เป็นไปได้จะเหมือนกันทุกๆสถานะซึ่งมีทั้งหมด 5 การการะทำประกอบด้วย เดินบน เดินล่าง เดินซ้าย เดินขวา และ หยุดรอ

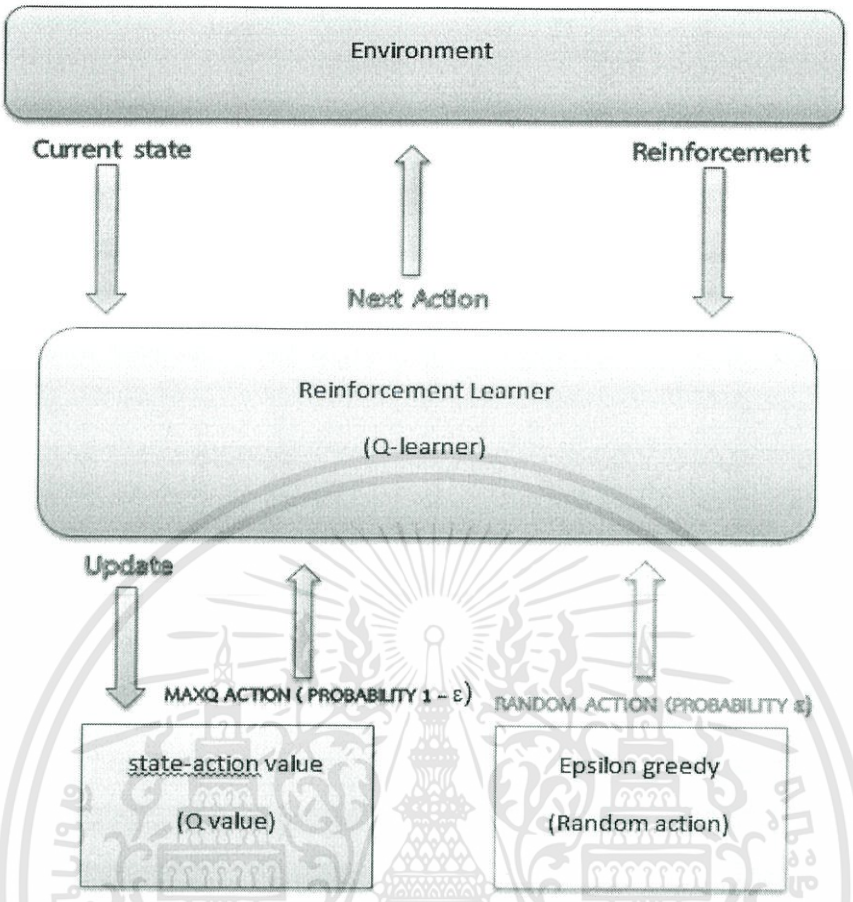
จากสูตรการหาค่าคิวซึ่งมีสมการดังนี้

$$Q_{t+1}(s_t, a_t) = \underbrace{(1-\alpha_t(s_t, a_t))}_{\text{inverse learning rate}} \times \underbrace{Q_t(s_t, a_t)}_{\text{old value}} - \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \overbrace{\left[\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}\right]}^{\text{learned value}}$$

จะได้ว่าค่าคิวใหม่สามารถหาจากสองส่วนหลักๆคือ คำนวณจากค่าคิวเดิมก็คือ $Q_t(s_t,a_t)$ และประสบการณ์ใหม่ที่เกิดจากการกระทำ a_t ก็คือส่วนของ R_{t+1} และ $\max Q_t(s_t+1,a)$ โดยค่าเราออกแบบให้ค่าคิวมองว่าค่าที่เคยเจอมาก่อนมีความสำคัญมากกว่าจึงให้ค่าอัตราการเรียนรู้ α มีค่าอยู่ที่ 0.1 และในส่วนของค่าของการเรียนรู้ซึ่งคือ ค่าผลตอบแทนบวกด้วยค่าคิวสูงสุดของสถานะต่อไปแต่จะมีตัวแปร γ เข้ามาเพื่อให้เกิดสมดุลระหว่างผลตอบแทนกับการมองการณ์ไกลไป ณ สถานะต่อไปที่เป็นผลจากการกระทำ a_t ว่ามีเป็นสถานะที่ควรที่จะไปหรือไม่ซึ่งดูได้จากค่าคิวสูงสุดเพียงอย่างเดียว เนื่องจากปัญหาดังกล่าวอาจมีการเรียนรู้แบบไม่

สิ้นสุดจึงออกแบบให้ค่า γ มีค่าน้อยกว่า 1 ซึ่งในการทดลองใช้เพียง 0.8 เนื่องจากสถานะส่วนใหญ่ไม่มีการได้รับค่าตอบแทนหรือมีค่าเป็นศูนย์ โดยสถานะที่มีการได้รับค่าตอบแทนมีเพียงสถานะที่ไปถึงทางออก และสถานะที่เดินไปสถานะที่ถูกฆ่า แต่เราออกแบบให้สถานะที่เป็นจุดสิ้นสุดทั้งสองแบบทั้งแบบสถานะที่ถึงจุดหมายและสถานะที่ถูกฆ่าไม่จำเป็นต้องมีการกระทำใดๆต่อจึงให้ค่าคิวของทุกๆการกระทำที่เป็นไปได้เป็นศูนย์ทั้งหมด จะได้ว่าสถานะก่อนที่จะถึงสถานะสิ้นสุดจะได้รับค่าการเรียนรู้เฉพาะในส่วนของผลตอบแทน (R_{t+1}) ส่วนค่าคิวสูงสุดของสถานะต่อไป ($maxQ_t(s_{t+1},a)$) จะเป็นศูนย์เนื่องจากเป็นสถานะสิ้นสุดซึ่งทั้งสองแบบจะได้รับค่าตอบแทนเป็นค่าสูงสุดและต่ำสุดตามลำดับ ส่วนสถานะที่เหลือจะได้รับค่าผลตอบแทนเป็นศูนย์ทั้งหมด แต่จะมองไปที่สถานะต่อไปว่าค่าคิวสูงสุดเป็นเท่าใดทำให้เอเจนต์สามารถที่จะเรียนรู้ลำดับการกระทำที่สามารถพาไปถึงเป้าหมายหรือทางออกได้ และรู้ว่าการกระทำใดทำให้พาไปถึงสถานะที่ถูกฆ่า ผลคือเอเจนต์สามารถเรียนรู้ลำดับการกระทำที่ดีที่สุด ณ ขณะนั้นๆได้จากค่าคิวของการกระทำของสถานะปัจจุบันนั่นเอง และเนื่องจากเราออกแบบให้สถานะที่เหลือคำนวณค่าคิวใหม่จากค่าคิวสูงสุดของสถานะต่อไปรวมกับทั้งมีค่าปัจจัยการลด(discount factor γ) ทำให้เอเจนต์สามารถรู้ว่าการกระทำใดสามารถไปถึงจุดมุ่งหมายได้เร็วที่สุด เนื่องจากยิ่งลำดับการกระทำที่พาไปถึงจุดมุ่งหมายมีความยาวมากก็ยิ่งทำให้ค่า γ มีผลมากขึ้นตามลำดับจากจุดมุ่งหมายออกมา ผลคือค่าคิวของการกระทำของสถานะที่อยู่ไกลจากจุดมุ่งหมายจะยิ่งมีค่าลดน้อยลงเรื่อยๆ แต่ถ้าอีกการกระทำมีลำดับที่สั้นกว่าแต่อยู่ในสถานะเดียวกันก็จะมีค่าคิวที่สูงกว่าเนื่องจากค่า γ ส่งผลจำนวนครั้งน้อยกว่า ดังนั้นเอเจนต์สามารถเลือกการกระทำที่เป็นคำตอบที่ดีที่สุดได้จากค่าคิวสูงของของการกระทำที่เป็นได้นั่นเอง

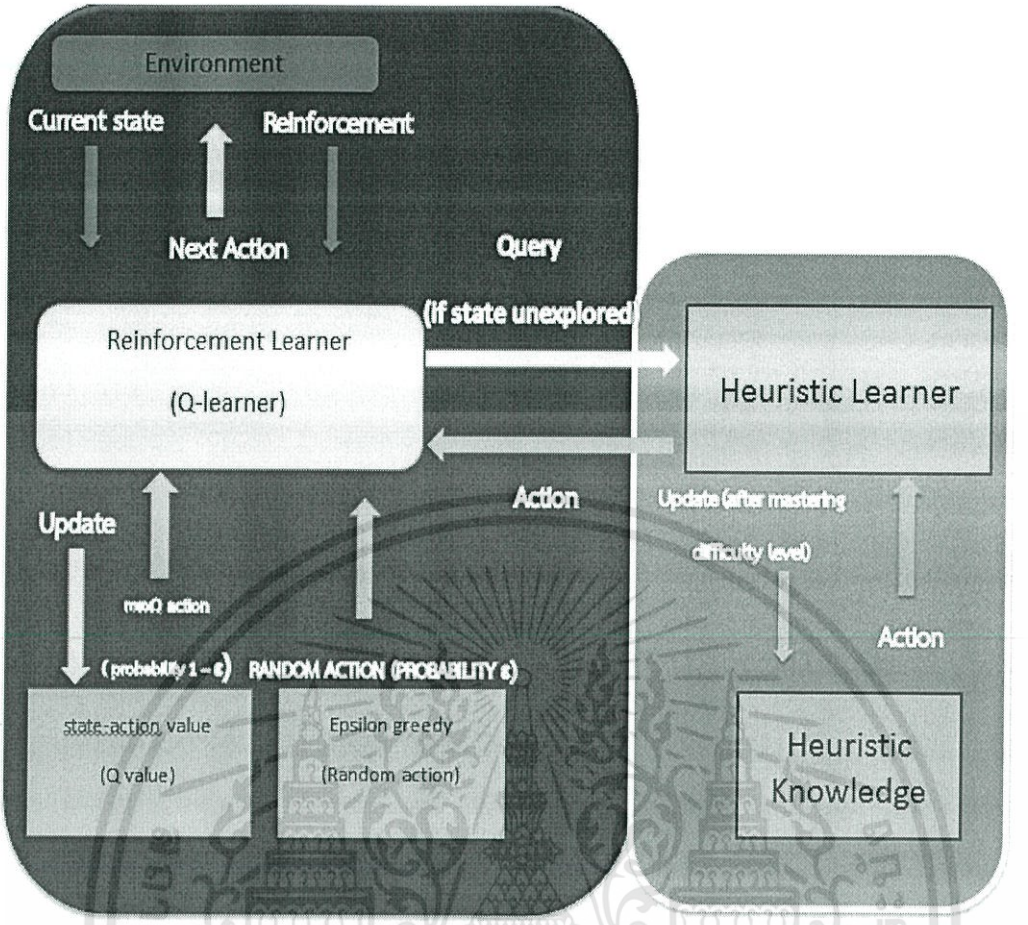
เนื่องจากเกมนี้มีการเล่นเป็นรอบๆ โดยจบเมื่อถูกฆ่าหรือถึงทางออก ดังนั้นเราจึงออกแบบให้เอเจนต์เรียนรู้ไปเรื่อยๆ ซึ่งจำนวนรอบเราไม่ได้จำกัดไว้โดยให้คาดหวังว่ายิ่งเอเจนต์เรียนรู้นานเท่าไหร่ยิ่งได้ผลลัพธ์ที่ดีกว่า เราจึงให้เอเจนต์แก้ปัญหาเดิมวนซ้ำไปเรื่อยๆ แต่เนื่องจากหลักการของ Q-learning ธรรมดานั้นไม่สามารถหาคำตอบที่ดีที่สุดของปัญหาได้เนื่องจากจะเชื่อค่าคิวสูงสุดเสมอทำให้การกระทำที่เหลือที่ยังไม่ได้ทดลองทำไม่สามารถรู้ว่าอาจจะมีลำดับการกระทำที่ดีกว่าก็เป็นได้ จากสาเหตุดังกล่าวเราจึงใช้หลักการของ Epsilon Greedy เข้ามาผสมกับ Q-learning เนื่องจากการเรียนรู้จะมีประสิทธิภาพต้องมีการค้นหาสิ่งที่ดีกว่าเดิม แต่ในที่นี้จะใช้วิธีการสุ่มการกระทำที่เป็นไปได้ ซึ่งการสุ่มดังกล่าวจะมีค่าความน่าจะเป็นเรียกว่าค่าความน่าจะเป็นของเอปไซลอน ε จะได้ว่าความน่าจะเป็นที่เหลือคือ 1- ε เป็นโอกาสที่จะเชื่อค่าคิวสูงสุด ในที่นี้เราออกแบบให้ค่า ε เท่ากับ 0.001 หรือ 0.1 เปอร์เซ็นต์ ส่วน 99.9 เปอร์เซ็นต์จะเป็นการเลือกการกระทำที่มีค่าคิวสูงสุด ณ ปัจจุบัน จากหลักการนี้จะเห็นได้ว่าถ้าให้เอเจนต์เรียนไปนานๆ จะทำให้สามารถค้นหาสิ่งทางที่ดีกว่าไปเรื่อยๆด้วยความน่าจะเป็นของ ε ซึ่งเป็นไปตามโมเดลการทำงานดังนี้



รูป 3.3 รูปแสดงการทำงานของโปรแกรม

3.5 การออกแบบส่วนเสริมของการเรียนรู้เสริมกำลัง

ในส่วนที่ได้ออกแบบเพิ่มเข้ามาเพื่อช่วยให้การเรียนรู้รวดมีประสิทธิภาพสูงขึ้นก็คือส่วนของ Heuristic โดยส่วนนี้จะดูแลในส่วนของความรู้เกิดจากการเรียนรู้จากปัญหาที่เคยเจอมาแล้วในอดีตปรับเปลี่ยนให้อยู่ในรูปแบบที่สามารถนำไปประยุกต์ใช้กับปัญหาใหม่ๆที่คล้ายคลึงกันเพื่อให้การเรียนรู้ในสิ่งที่เคยผ่านมาแล้วรวดเร็วขึ้น พร้อมทั้งสามารถค้นหาคำตอบได้เร็วกว่าเดิมที่ต้องเริ่มลองผิดลองถูกใหม่ทั้งหมด



รูป 3.4 รูปแสดงการทำงานของโปรแกรม

ในส่วนของตัวช่วยตัดสินใจ (Heuristic) จะประกอบไปด้วยส่วนสองหลักๆคือ ส่วนการเรียนรู้ของตัวช่วยตัดสินใจ (Heuristic Learner) และฐานความรู้ช่วยตัดสินใจ (Heuristic Knowledge) โดยส่วนของผู้เรียนรู้เพื่อช่วยตัดสินใจจะทำหน้าที่ในการติดต่อระหว่างผู้เรียนแบบเสริมกำลังและฐานความรู้ช่วยตัดสินใจ เนื่องจากรูปแบบในการแสดงข้อมูลแตกต่างกันจึงจำเป็นต้องมีตัวกลางในการแปลงข้อมูลให้สอดคล้องกัน ส่วนฐานความรู้ช่วยตัดสินใจจะเป็นส่วนที่เก็บข้อมูลของความรู้ที่เคยเรียนรู้มาจากปัญหาที่เคยแก้แล้วในอดีต ซึ่งก่อนที่จะนำความรู้มาอัพเดทนั้นจำเป็นต้องให้ตัวเอเจนต์หรือผู้เรียนทำการเรียนรู้ปัญหานั้นๆจนเชี่ยวชาญระดับหนึ่งจึงนำความรู้มาอัพเดทกับฐานข้อมูลในฐานความรู้ช่วยตัดสินใจเพื่อให้ได้ข้อมูลที่มีประสิทธิภาพมากที่สุด เนื่องจากการเรียนรู้ของเอเจนต์เป็นแบบลองผิดลองถูก (Reinforcement Learning) เพื่อค้นหาลำดับของการกระทำที่ดีที่สุด แต่เนื่องจากเราไม่ได้ให้เอเจนต์ลองทุกวิธีที่เป็นไปได้ของลำดับการกระทำเนื่องจากโดเมนของคำตอบที่เป็นไปได้นั้นใหญ่เกินไปอาจต้องใช้เวลานานเกินไปในทางปฏิบัติ เราจึงให้เอเจนต์เรียนรู้ด้วยตัวเองโดยลองผิดลองถูกจนเจอเส้นทางที่ดีขึ้นเรื่อยๆแต่ไม่การันตีว่าจะดีที่สุด ขึ้นกับว่าเวลาที่ให้เอเจนต์เรียนนั้นยาวนานแค่นั้นรวมถึงค่า e-greedy ที่เป็นเปอร์เซ็นต์ที่เอเจนต์จะลองการกระทำที่ไม่เคยทำมาก่อนแทนที่จะเลือกการกระทำที่ดีที่สุดในตอนนั้นเพื่อเป็นการค้นหาลำดับการกระทำที่ดีกว่า ดังนั้นยิ่งเอเจนต์มีการ

เรียนรู้นานรวมทั้งค่า e-greedy ที่เหมาะสมก็จะทำให้เอเจนต์มีโอกาสที่จะค้นหาวิธีการที่จะแก้ปัญหาได้ดียิ่งขึ้น ซึ่งนี่คือสิ่งที่เอามาวัดว่าเอเจนต์เชี่ยวชาญในการแก้ปัญหา ณ สถานการณ์นั้นๆแล้วหรือยัง ก่อนที่จะนำความรู้นั้นมาอัพเดทลงฐานข้อมูลเพื่อนำไปประยุกต์ใช้กับปัญหาที่คล้ายคลึงกันต่อไป

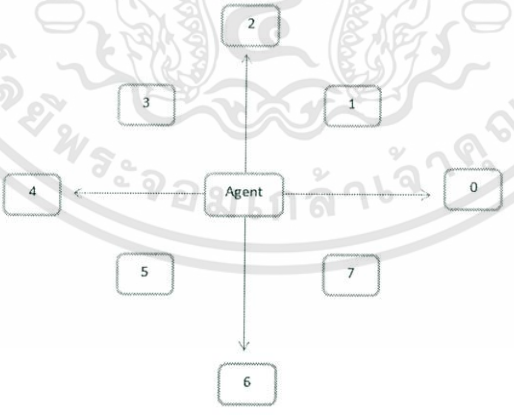
3.5.1 การออกแบบรูปแบบการเก็บข้อมูลของฐานความรู้ช่วยตัดสินใจ

ในส่วนของฐานความรู้ช่วยตัดสินใจเราได้ออกแบบรูปแบบของข้อมูลนำเสนอของข้อมูลเพื่อช่วยตัดสินใจ(กฎ)(Heuristic Knowledge Representation) ออกเป็น 7 ส่วนหลักๆดังนี้

1. ทิศทางของทางออก (Goal Direction)

   เป็นค่าที่จะแสดงถึงว่าทิศทางของทางออกของเขาวงกตของปัญหาที่ให้เอเจนต์เรียนรู้นั้นอยู่ไปทางทิศใดของเอเจนต์ ซึ่งจะเก็บในรูปแบบของ Integer โดยแต่ละทิศจะแทนด้วยค่าดังนี้

   - ทางออก อยู่ไปทิศทางทำมุม 0 องศา : Value = 0
   - ทางออก อยู่ในทิศทางทำมุม > 0 และ < 90 องศา : Value = 1
   - ทางออก อยู่ในทิศทางทำมุม 90 องศา : Value = 2
   - ทางออก อยู่ในทิศทางทำมุม > 90 และ < 180 องศา : Value = 3
   - ทางออก อยู่ในทิศทางทำมุม 180 องศา : Value = 4
   - ทางออก อยู่ในทิศทางทำมุม > 180 และ < 270 องศา : Value = 5
   - ทางออก อยู่ในทิศทางทำมุม 270 องศา : Value = 6
   - ทางออก อยู่ในทิศทางทำมุม > 270 และ < 360 องศา : Value = 7



รูป 3.5 รูปแสดงตัวเลขแทนทิศที่ใช้

2. ทิศทางของมิโนทอร์ (Minotaur Direction)

   เป็นค่าที่แสดงถึงทิศทางของ Minotaur จากตัวเอเจนต์ ซึ่งเก็บเป็นค่าเช่นเดียวกับ Goal Direction

3. ตัวละครที่ใกล้ที่สุด (Nearest Distant)

เป็นค่าที่จะเป็นว่า Minotaur หรือ Goal อยู่ห่างจากเอเจนต์ใกล้กว่ากันหรือเท่ากันโดยไม่คำนึงถึงทิศทางโดยค่าจะเก็บในรูปแบบของ Integer ดังนี้

- Goal's Distant < Minotaur's Distant : Value = 0
- Goal's Distant = Minotaur's Distant : Value = 1
- Goal's Distant > Minotaur's Distant : Value = 2

4. กำแพงทิศเหนือ (Top Brick)
   เป็นค่าที่แสดงถึงช่องบนของเอเจนต์ ณ จุดๆนั้นว่ามีสิ่งกีดขวางหรือไม่ โดยค่าจะเก็บในรูปแบบของ Integer ซึ่งถ้ามีสิ่งกีดขวางจะมีค่าเป็น 1 แต่ถ้าไม่มีจะมีค่าเป็น 0
5. กำแพงทิศใต้ (Bottom Brick)
   เป็นค่าที่แสดงถึงช่องล่างของเอเจนต์ ณ จุดๆนั้นว่ามีสิ่งกีดขวางหรือไม่ ซึ่งค่าจะเก็บในรูปแบบเดียวกับ Top Brick
6. กำแพงทิศตะวันตก (Left Brick)
   เป็นค่าที่แสดงถึงช่องซ้ายของเอเจนต์ ณ จุดๆนั้นว่ามีสิ่งกีดขวางหรือไม่ ซึ่งค่าจะเก็บในรูปแบบเดียวกับ Top Brick
7. กำแพงทิศตะวันออก (Right Brick)
   เป็นค่าที่แสดงถึงช่องขวาของเอเจนต์ ณ จุดๆนั้นว่ามีสิ่งกีดขวางหรือไม่ ซึ่งค่าจะเก็บในรูปแบบเดียวกับ Top Brick

3.5.2 การออกแบบฐานข้อมูลความรู้ช่วยตัดสินใจ

ในส่วน Heuristic Knowledge เราได้ออกแบบให้เก็บข้อมูลแบบของฐานข้อมูลเชิงสัมพันธ์ (Relational Database) โดยเราได้แบ่งฐานข้อมูลออกเป็นสองตาราง คือ เงื่อนไขแบบกฎ (Rule Base) และเงื่อนไขกฎความถี่ (Max Frequency Base) โดยเงื่อนไขแบบกฎนั้นจะเก็บเฉพาะการกระทำที่ทุกๆปัญหาที่เคยเรียนรู้ผ่านมาในอดีตสอดคล้องกันเมื่ออยู่ในรูปแบบของกฎเดียวกันเปรียบเสมือนกฎตายตัวว่าถ้าเจอสถานการณ์แบบนี้จะต้องทำแบบนี้ ส่วนของเงื่อนไขแบบกฎความถี่จะเก็บทุกๆการกระทำไม่ว่าจะสอดคล้องกันหรือไม่แต่จะเพิ่มในส่วนของค่าความถี่ในการกระทำเข้ามาด้วยเพื่อบ่งบอกว่าการกระใด ณ รูปแบบของกฎต่างๆ มีค่าความถี่ในการกระทำสูงสุดหรือหมายความจำนวนปัญหาที่เคยเรียนผ่านมานั้นเลือกใช้การกระทำดังกล่าวมากสุดเพื่อบ่งบอกว่าการกระทำดังกล่าวน่าเชื่อถือมากที่สุดแต่ โดยรูปแบบของตารางเป็นดังนี้

ตาราง 3.1 ตารางแสดงเค้าร่างฐานข้อมูลเก็บเงื่อนไขแบบกฎ

Goal Direction	Minotaur Direction	Nearest Distant	Top Brick	Down Brick	Left Brick	Right Brick	Action

จะเห็นได้ว่าเราได้นำค่าของ Heuristic Knowledge Representation มาเก็บในส่วนของ 7 Columns แรก และที่เหลือคือ Action ซึ่งจะแสดงถึงการกระทำที่ใช้

ตาราง 3.2 ตารางแสดงเค้าร่างฐานข้อมูลเก็บเงื่อนไขแบบกฎความถี่

Goal rection	Minotaur Direction	Nearest Distant	Top Brick	Down Brick	Left Brick	Right Brick	Action	Frequency

จะเห็นได้ว่าการเก็บข้อมูลจะคล้ายกับฐานข้อมูลเงื่อนไขแบบกฎแต่จะต่างตรงที่มีการเก็บความถี่ของการกระทำนั้นๆเพิ่มขึ้นมา

3.5.3 การออกแบบส่วนการเรียนรู้เพื่อช่วยตีดสินใจ (Heuristic Learning)

เป็นตัวกลางที่แปลงข้อมูลเพื่อช่วยตัดสินใจเชิงนำเสนอในฐานความรู้ให้อยู่ในรูปแบบของข้อมูลที่ ส่วนการเรียนรู้แบบเสริมกำลังสามารถเข้าใจได้ เนื่องจากรูปแบบของข้อมูลที่ส่วนการเรียนรู้แบบเสริมกำลังใช้นั้นจะอยู่ในรูปแบบของ ค่าสถานะ (State) ดังนั้นในกรณีที่ส่วนเสริมกำลังต้องการดึงค่าการกระทำจากฐานความรู้เพื่อช่วยตัดสินใจไปใช้ในยามที่ไปเยือนสถานะที่ไม่เคยไปมาก่อนก็จำเป็นต้องแปลงรูปแบบของข้อมูลจากสถานะเป็นข้อมูลเพื่อช่วยตัดสินใจแบบนำเสนอผ่านตัวส่วนการเรียนรู้เพื่อช่วยตัดสินใจจากนั้นตัวส่วนการเรียนรู้เพื่อช่วยตัดสินใจก็จะทำการดูว่าในฐานความรู้เพื่อช่วยตัดสินใจนั้นมีค่าที่ตรงกับ ส่วนการเรียนรู้แบบเสริมกำลังต้องการหรือไม่ ถ้ามีก็จะส่งค่าการกระทำที่สอดคล้องกับไปให้

อีกหน้าที่หนึ่งที่ส่วนการเรียนรู้เพื่อการตัดสินใจทำคือเมื่อเอเจนต์สามารถเรียนรู้ปัญหาจนชำนาญพร้อมที่จะอัพเดทไปที่ฐานความรู้เพื่อช่วยตัดสินใจแล้วส่วนการเรียนรู้แบบเสริมกำลังจะส่งค่าทุกๆสถานะที่เอเจนต์เคยเรียนรู้ไปให้กับส่วนการเรียนรู้เพื่อช่วยตัดสินใจเพื่อทำการคำนวณและสร้างกฎต่างๆขึ้นจากสถานะทั้งหมดจากนั้นก็ทำการเพิ่มกฎดังกล่าวเข้าไปในฐานความรู้เพื่อช่วยในการตัดสินใจ

3.6 การแก้ไขปัญหาผู้เรียนเดินไปกลับ

ปัญหาที่พบนั้นคือเอเจนต์จะเดินวนลูปเนื่องจากเดินไป ณ สถานะที่ไม่เคยไปมาก่อนแล้วไม่มีการเรียนรู้ใดๆเกิดขึ้นเนื่องจากไม่มีการตายจากมิโนทอร์, เดินถึงทางออก หรือจะเป็นการเดินชนกำแพง ทำให้เอเจนต์ไม่รู้เลยว่าตอนนี้ควรจะทำอย่างไร ได้แต่ต้องสุ่มการกระทำไปเรื่อยๆจนกว่าจะได้รับผลตอบแทนกลับมาเพื่ออัพเดทค่าคิวของการกระทำ ทำให้เอเจนต์สุ่มการกระทำจนวนลูปไปโดยไม่รู้ตัวจนเสียเวลาในการเรียนรู้ จากปัญหาดังกล่าวเราได้ออกแบบวิธีการแก้ปัญหาดังกล่าวมาโดยเราจะทำการนับจำนวนครั้งที่ไปเยือนสถานะต่างๆไว้ด้วย ทำให้เอเจนต์สามารถรู้ได้ว่าเคยไป ณ สถานะ นี้บ่อยแค่ไหน ด้วยสาเหตุดังกล่าวเราจึงออกแบบให้เอเจนต์เดินไป ณ สถานะที่มีค่าความถี่ในการไปเยือนต่ำสุดเพื่อให้เอเจนต์ไม่เดินไป ณ สถานะเดิมๆจนวนลูปในกรณีที่ต้องสุ่มการกระทำ และเมื่อเอเจนต์สามารถเปลี่ยนค่ารอบการเรียนเนื่องจากสามารถเดินถึงทางออกก็จะทำการเคลียร์ค่าความถี่ในการไปเยือนของแต่ละสถานะใหม่ทุกครั้ง

3.7 รหัสเทียม (Pseudo Code)

```
/*
Heuristic_Percentage = Percentage that you want to rely on heuristic knowledge database.
TypeOfHeuristic = Heuristic's type -> Rule base or Max frequency base.
e-greedy_Percentage = e-greedy value * 100.
*/


LevelOfProblem = First level of problem;
StartState = new state(CurrentProblem);

while(true)
{

    bool EpochEnd = false;
    CurrentState = StartState;
    /* Repeat learning until agent can master in the current problem before change to higher level.
    */
    for( int loop =  ; loop < iteration_of_epoch ; loop++)
    {
        do{
            int randPercentage = random();

            /*  In case of agent has not learned anything at this state or
                never visited this state before, so we put him to query action
                that most appropriate from heuristic knowledge database.
            */
            if(CurrentState->GetMaxQValue() <=  )
            {

                if(randPercentage %  >= (  - Heuristic_Percentage) )
                    Action = Heuristic_Learner->getAction(CurrentState,TypeOfHeuristic);
                else
                    Action = randomAction();

                NextState = CurrentState->TakeAction(Action);

            }
            /* In case of agent's Q value of action in this state has updated
                due to has learned something before at this state, so we put him
                to make a action base on e-greedy algorithm.
            */
            else
            {
                if(randPercentage % 100 >= e-greedy_Percentage)
                    Action = CurrentState->getActionMaxQvalue();
                else
                    Action = randomAction();

            }
```

```
        /* Update Q value base on NextState that is the result due to action that CurrentState has taken. */

          /*  To prevent agent to do the action that waste a time, we need to check weather agent doesn't still in the same state after
              taken a action if CurrentState equal to NextState,it mean as if agent hasn't done anything -> it waste
              a time so we give the negative reward to agent.
          */
        if(CurrentState != NextState)
        {
            if(NextState->Agent_pX == NextState->Minotaur_pX && NextState->Agent_pY == NextState->Minotaur_pY)
            {
                /*In case of NextAction is the state that Minotaur can reach agent's position, we put agent to
                  update his action's Q value base on Q-learning equation with reward's value equal to minimum reward
                    and NextState's maximum Qvalue equal to zero, so learned value equal to minimum reward.
                */
                CurrentState->Qvalue[Action] = ((1-alpha) * CurrentState->aQvalue[Action]) + ( alpha * minReward );
                EpochEnd = true;
            }
            else if(NextState->Agent_pX == NextState->Goal_pX && NextState->Agent_pY == NextState->Goal_pY)
            {

                /*In case of NextAction is the state that agent can reach goal position, we put agent to
                  update his action's Q value base on Q-learning equation with reward's value equal to maximum reward
                    and NextState's maximum Qvalue equal to zero, so learned value equal to maximum reward
                */
                CurrentState->Qvalue[Action] = ((1-alpha) * CurrentState->Qvalue[Action]) + ( alpha * maxReward );
                EpochEnd = true;
            }
            else
            {
                /*In case of NextAction is the state that agent neither reach goal position nor minotaur can reach agent position,
                  we put agent to update his action's Q value base on Q-learning equation with learned value equal to
                  alpha * ( gamma * NextState's maximum Q value) due to reward at state like this equal to zero .
                */
                CurrentState->Qvalue[Action] = ((1-alpha) * CurrentState->Qvalue[Action]) + (alpha * ( gamma * NextState->getMaxQvalue()));


                /* Change to next state and repeat until reach the end of epoch.
                */
                CurrentState = NextState;
            }

        }

        /* In case of agent still in the same state after has taken the action, We give the negative reward (wasteReward) to agent.
        */
        else
        {
            CurrentState->Qvalue[Action] = ((1-alpha) * CurrentState->Qvalue[Action]) + (alpha * wasteReward);
        }


    }while(!EpochEnd);

    /* Start at begin state again in next epoch.
    */
    CurrentState = StartState;
}

/*Update Heuristic Knowledge database.
*/
UpdateHeuristicKnowledge();

/*Change Level of problem to higher level.
*/
if(LevelOfProblem < HighestLevelOfProblem)
{
    ChangelevelOfProblem();
    ClearState();
    StartState = new state(CurrentProblem);
}
else
    break;
}
```

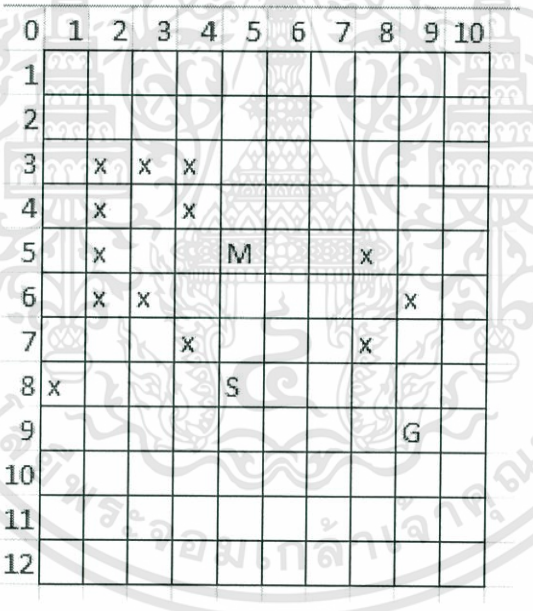
บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดลอง Epsilon Policy

ผู้จัดทำได้ทำการทดลองโดยใช้รูปแบบเขาวงกตในการทดลองทั้งหมด 2 รูปแบบ และมีการกำหนดค่าจำนวนรอบทั้งหมด เป็น 2000 รอบ โดยมีการใช้ Epsilon Policy กับไม่ใช้ ผลที่ได้จะแสดงในรูปแบบกราฟแสดงความสัมพันธ์ระหว่างจำนวนรอบกับจำนวนก้าวในแต่ละรอบ กับจำนวนรอบที่ตายจนถึงทางออกในแต่ละครั้ง

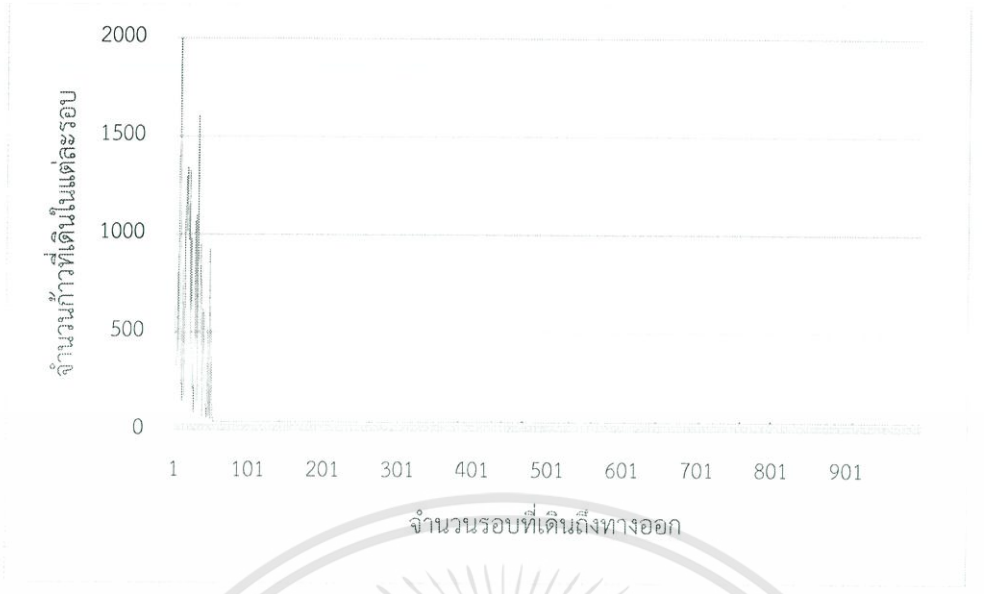
4.1.1 การทดลองรูปแบบเขาวงกตแบบ ที่ 1



รูปที่ 4.1 รูปแสดงรูปแบบของเขาวงกตแบบที่ 1

4.1.1.1 การทดลองโดยใช้ Epsilon Policy

ในการทดลองกำหนดรอบที่เดินทั้งหมด 2000 รอบ โดยมีค่า Epsilon greedy = 0.1 และค่า Q เริ่มต้นเป็น 0



รูปที่ 4.2 กราฟแสดงค่าระหว่างจำนวนรอบที่ถึงทางออกและจำนวนก้าวเดินแต่ละรอบ

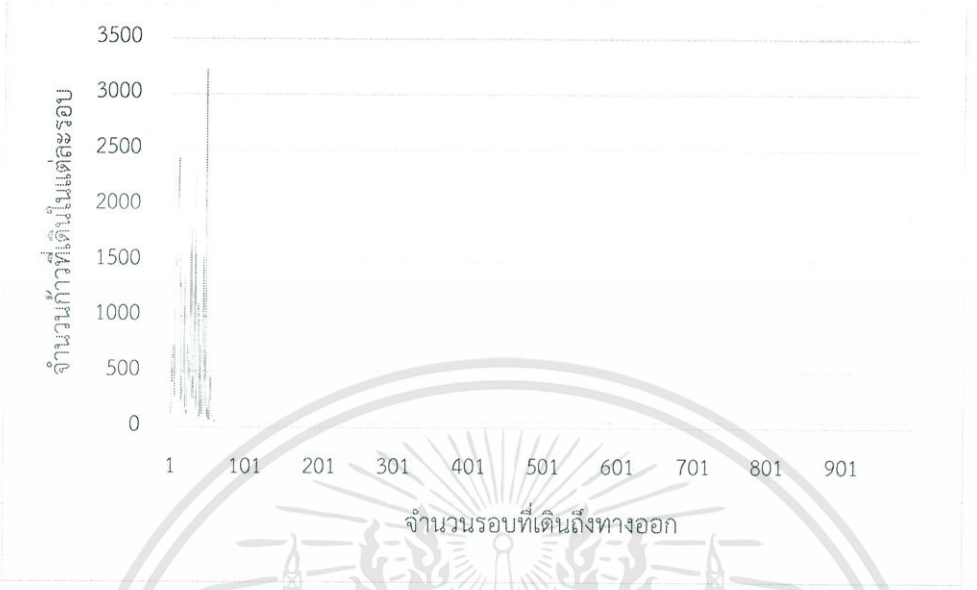
จากรูป 4.2 พบว่าสามารถเดินจนถึงทางออกได้ 1484 รอบ โดยใช้จำนวนก้าวมากที่สุดถึง 2329 ก้าว และน้อยที่สุด 45 ก้าว

รูปที่ 4.3 กราฟแสดงค่าระหว่างจำนวนรอบที่ตายและจำนวนก้าว

จากรูป 4.3 พบว่าการเดินจนกระทั่งตายนั้นในช่วงแรกจะสามารถเดินได้น้อยมาก และจะมากขึ้นสูงสุดที่ช่วงกลางแต่จะลดลงตอนท้ายเนื่องจากการที่หาเส้นทางอื่นที่เป็นไปได้เพิ่มเติม โดยสามารถเดินได้มากที่ 3107 ก้าว และน้อยที่สุดเพียง 1 ก้าว โดยมีค่าเฉลี่ยที่ 280.603 ก้าวก่อนจะตาย

4.1.1.2 การทดลองโดยไม่ใช้ Epsilon Policy

ในการทดลองกำหนดรอบที่เดินทั้งหมด 2000 รอบ โดยมีค่า Epsilon greedy = 0 และค่า Q เริ่มต้นเป็น 0



รูปที่ 4.4 กราฟแสดงค่าระหว่างจำนวนรอบที่ถึงทางออกและจำนวนก้าวเดินแต่ละรอบ

จากรูป 4.4 พบว่าสามารถเดินจนถึงทางออกได้ 1590 รอบ โดยใช้จำนวนก้าวมากที่สุดถึง 3236 ก้าว และน้อยที่สุด 50 ก้าว ซึ่ง 50 ก้าวนั้นไม่ใช่ทางที่สั้นที่สุด แต่เนื่องจากไม่มีการตั้งค่า Epsilon Greedy จึงทำให้เมื่อสามารถถึงทางออกได้ครั้งแรกแล้ว ผู้เรียนจะเดินบนเส้นทางเดิมตลอดทำให้ไม่สามารถหาทางที่สั้นที่สุดได้

รูปที่ 4.5 กราฟแสดงค่าระหว่างจำนวนรอบที่ตายและจำนวนก้าว

จากรูป 4.5 พบว่าการเดินจนกระทั่งตายนั้นในช่วงแรกจะสามารถเดินได้น้อยมาก และจะมากขึ้นสูงสุดที่เรื่อยๆ แต่เมื่อพบทางที่สามารถเดินถึงทางออกได้แล้ว ก็จะไม่ตายอีก โดยสามารถเดินได้มากที่ 3409 ก้าว และน้อยที่สุดเพียง 1 ก้าว โดยมีค่าเฉลี่ยที่ 310.214 ก้าวก่อนจะตาย

4.1.2 การทดลองรูปแบบเขาวงกตแบบ ที่ 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0										x	x	x	x	x	x	x	x	x	x	x	x
1				x						x										x	
2			x		x		G			x		x	x				x	x			
3										x			x				x				
4										x		x	x				x			x	
5				x																x	
6			x									x						x		x	
7				x					x			x						x	x	x	
8										x	x	x								x	
9												M			x			S			

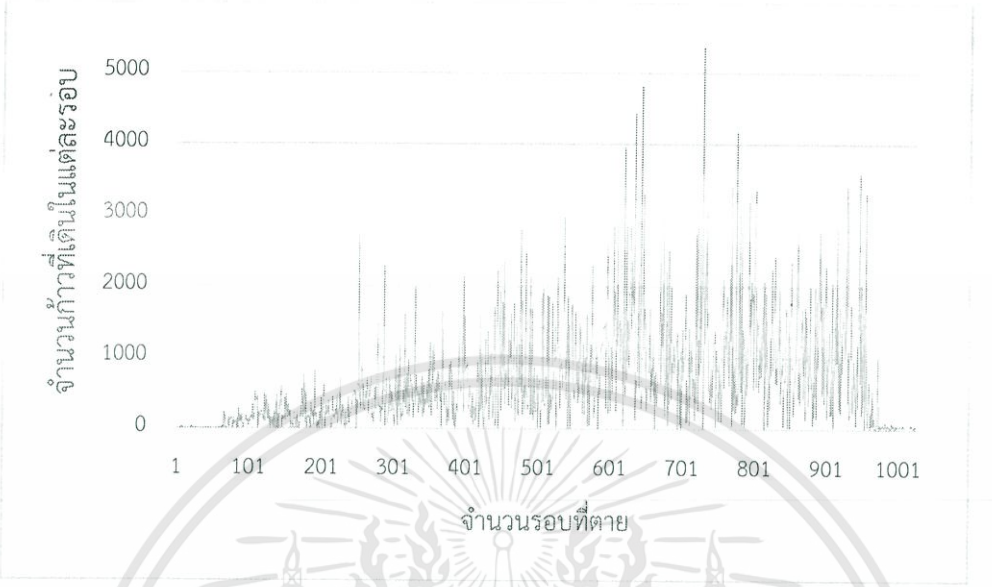
รูปที่ 4.6 รูปแสดงรูปแบบของเขาวงกตแบบที่ 2

4.1.2.1 การทดลองโดยใช้ Epsilon Policy

ในการทดลองกำหนดรอบที่เดินทั้งหมด 2000 รอบ โดยมีค่า Epsilon greedy = 0.1 และค่า Q เริ่มต้นเป็น 0

รูปที่ 4.7 กราฟแสดงค่าระหว่างจำนวนรอบที่ถึงทางออกและจำนวนก้าวเดินแต่ละรอบ

จากรูป 4.7 พบว่าสามารถเดินจนถึงทางออกได้ 977 รอบ โดยใช้จำนวนก้าวมากที่สุดถึง 3523 ก้าว และน้อยที่สุด 82 ก้าว

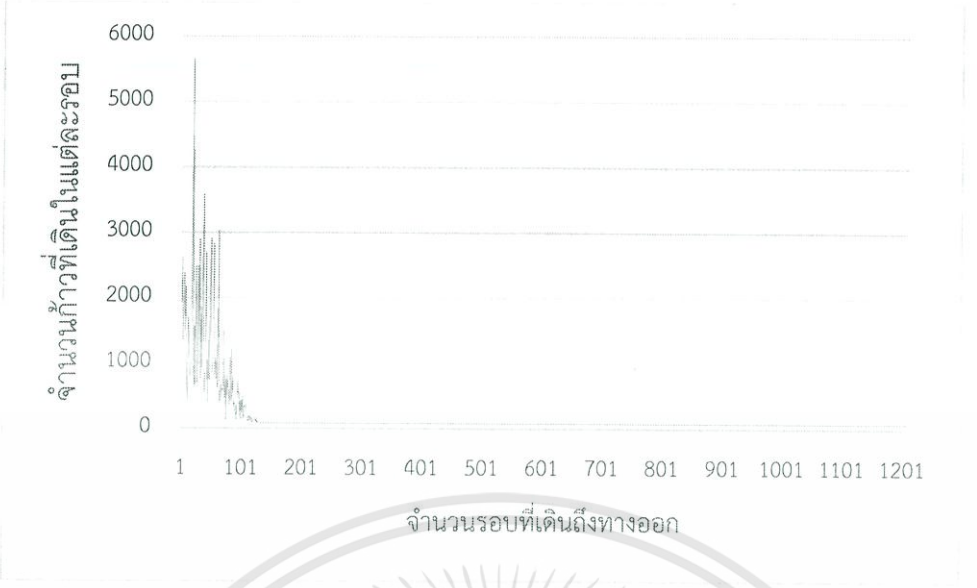


รูปที่ 4.8 กราฟแสดงค่าระหว่างจำนวนรอบที่ตายและจำนวนก้าว

จากรูป 4.8 พบว่าการเดินจนกระทั่งตายนั้นในช่วงแรกจะสามารถเดินได้น้อยมาก และจะมากขึ้นสูงสุดที่ช่วงกลางแต่จะลดลงตอนท้ายเนื่องจากการที่หาเส้นทางอื่นที่เป็นไปได้เพิ่มเติม โดยสามารถเดินได้มากที่ 5307 ก้าว และน้อยที่สุดเพียง 3 ก้าว โดยมีค่าเฉลี่ยที่ 683.266 ก้าวก่อนจะตาย

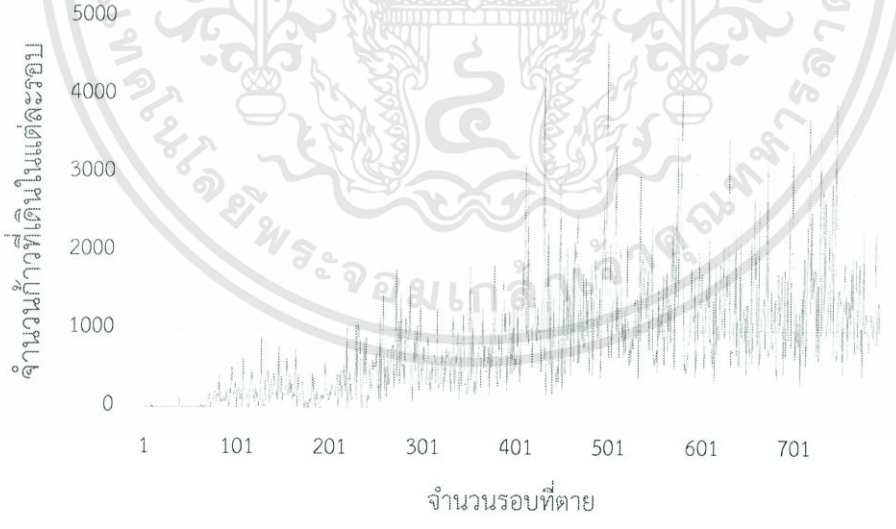
4.1.2.2 การทดลองโดยไม่ใช้ Epsilon Policy

ในการทดลองกำหนดรอบที่เดินทั้งหมด 2000 รอบ โดยมีค่า Epsilon greedy = 0 และค่า Q เริ่มต้นเป็น 0



รูปที่ 4.9 กราฟแสดงค่าระหว่างจำนวนรอบที่ถึงทางออกและจำนวนก้าวเดินแต่ละรอบ

จากรูป 4.9 พบว่าสามารถเดินจนถึงทางออกได้ 1590 รอบ โดยใช้จำนวนก้าวมากที่สุดถึง 3236 ก้าว และน้อยที่สุด 50 ก้าว ซึ่ง 50 ก้าวนั้นไม่ใช่ทางที่สั้นที่สุด แต่เนื่องจากไม่มีการตั้งค่า Epsilon Greedy จึงทำให้เมื่อสามารถถึงทางออกได้ครั้งแรกแล้ว ผู้เรียนจะเดินบนเส้นทางเดิมตลอดทำให้ไม่สามารถหาทางที่สั้นที่สุดได้

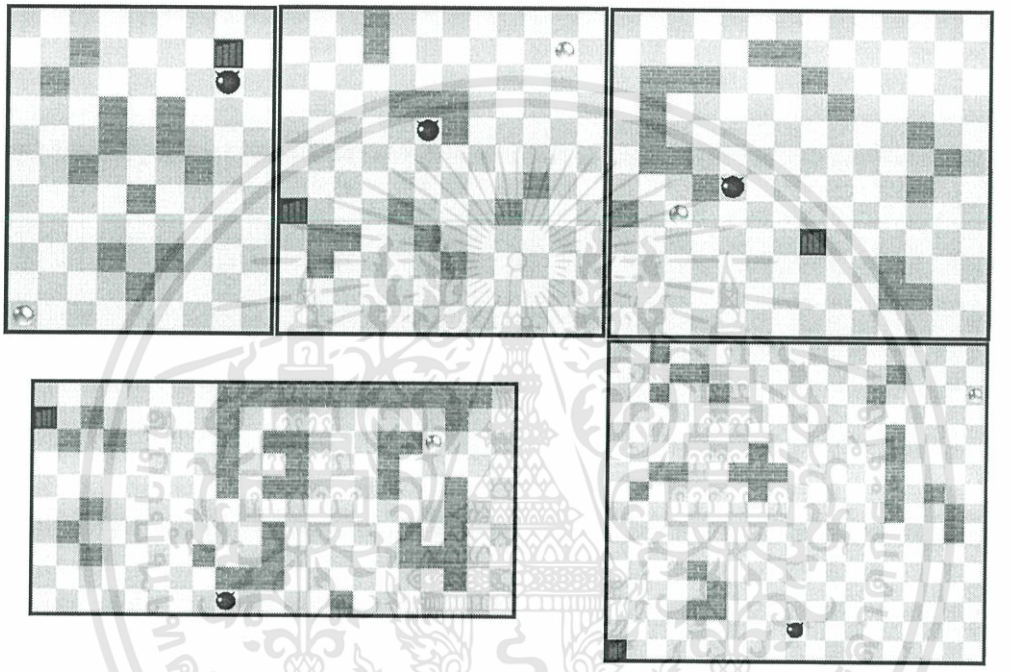


รูปที่ 4.10 กราฟแสดงค่าระหว่างจำนวนรอบที่ตายและจำนวนก้าว

จากรูป 4.10 พบว่าการเดินจนกระทั่งตายนั้นในช่วงแรกจะสามารถเดินได้น้อยมาก และจะมากขึ้นสูงสุดที่เรื่อยๆ แต่เมื่อพบทางที่สามารถเดินถึงทางออกได้แล้ว ก็จะไม่ตายอีก โดยสามารถเดินได้มากที่ 4714 ก้าว และน้อยที่สุดเพียง 3 ก้าว โดยมีค่าเฉลี่ยที่ 833.476 ก้าวก่อนจะตาย

4.2 การทดลอง Heuristic Algorithm

ผู้จัดทำได้ทำการทดลองโดยใช้รูปแบบเขาวงกตในการทดลองทั้งหมด 5 รูปแบบ โดยในการทดลองนี้จะใช้รูปแบบของ Heuristic Algorithm 3 รูปแบบคือ Rule Base, Max Sequence Base และไม่ใช้ Heuristic Algorithm ซึ่งจะทำการทดลอง 10 ครั้งเพื่อหาค่าเฉลี่ยของจำนวนรอบ(epoch)ที่ใช้ในการเดินถึงทางออกครั้งแรกของแต่ละอัลกอริทึมในแต่ละแผนที่



รูปที่ 4.11 รูปแสดงรูปแบบของเขาวงกต (เรียงลำดับรูปแบบ 1 2 3 จากซ้ายบน และ 4 5 จากซ้ายล่าง)

การทดลองเริ่มโดยการที่ให้ AI เรียนรู้แผนที่รูปแบบที่ 1 เพื่อสร้างกฎทั้งสองแบบขึ้นมาก่อนโดยมีการให้ผู้เรียนเรียนแผนที่รูปแบบที่ 1 ซ้ำทั้งหมด 5 ครั้งเพื่อลดความผิดพลาดของกฎที่สร้างขึ้นจากนั้นนำความรู้ที่ได้ไปทดลองใช้กับแผนที่รูปแบบที่ 2 และ 3 โดยการเรียนรู้จากแผนที่รูปแบบที่ 1 นั้นผู้เรียนได้สร้างเงื่อนไขสำหรับ Rule Base ขึ้นมา 26 ข้อ และ Max Sequence Base มา 67 ข้อ และได้ผลดังตารางที่ 4.1

ตารางที่ 4.1 ตารางแสดงผลการทดลองใช้ Heuristic Algorithm แผนที่รูปแบบที่ 1

แผนที่รูปแบบที่	ไม่ใช้กฎ	Rule Base	Max Sequence Base
2	280.913 รอบ	166.464 รอบ	245.897 รอบ
3	464.269 รอบ	399.556 รอบ	430.574 รอบ

จากนั้นให้ผู้เรียนได้เรียนรู้เพื่อสร้างเงื่อนไขจากแผนที่รูปแบบที่ 2 เพิ่มเติม โดยได้เงื่อนไขสำหรับ Rule Base และ Max Sequence Base มา 37 และ 72 ข้อตามลำดับโดยได้ผลการทดลองหลังจากที่ได้เพิ่มเงื่อนไขเพิ่มเติมแล้วได้ดังตารางที่ 4.2

ตารางที่ 4.2 ตารางแสดงผลการทดลองใช้ Heuristic Algorithm แผนที่รูปแบบที่ 2

แผนที่รูปแบบที่	ไม่ใช้กฎ	Rule Base	Max Sequence Base
3	458.904 รอบ	333.269 รอบ	369.694 รอบ
4	516.174 รอบ	260.303 รอบ	317.901 รอบ

จากนั้นให้ผู้เรียนได้เรียนรู้เพื่อสร้างเงื่อนไขจากแผนที่รูปแบบที่ 3 เพิ่มเติม โดยได้เงื่อนไขสำหรับ Rule Base และ Max Sequence Base มา 19 และ 90 ข้อตามลำดับโดยได้ผลการทดลองหลังจากที่ได้เพิ่มเงื่อนไขเพิ่มเติมแล้วได้ดังตารางที่ 4.3

ตารางที่ 4.3 ตารางแสดงผลการทดลองใช้ Heuristic Algorithm แผนที่รูปแบบ 3

แผนที่รูปแบบที่	ไม่ใช้กฎ	Rule Base	Max Sequence Base
4	508.87 รอบ	255.886 รอบ	261.481 รอบ
5	2440.77 รอบ	2109.32 รอบ	1824.99 รอบ

จากนั้นให้ผู้เรียนได้เรียนรู้เพื่อสร้างเงื่อนไขจากแผนที่รูปแบบที่ 4 เพิ่มเติม โดยได้เงื่อนไขสำหรับ Rule Base และ Max Sequence Base มา 32 และ 90 ข้อตามลำดับโดยได้ผลการทดลองหลังจากที่ได้เพิ่มเงื่อนไขเพิ่มเติมแล้วได้ดังตารางที่ 4.3

ตารางที่ 4.4 ตารางแสดงผลการทดลองใช้ Heuristic Algorithm แผนที่รูปแบบที่ 4

แผนที่รูปแบบที่	ไม่ใช้กฎ	Rule Base	Max Sequence Base
5	2473.61 รอบ	1435.64 รอบ	1580.26 รอบ

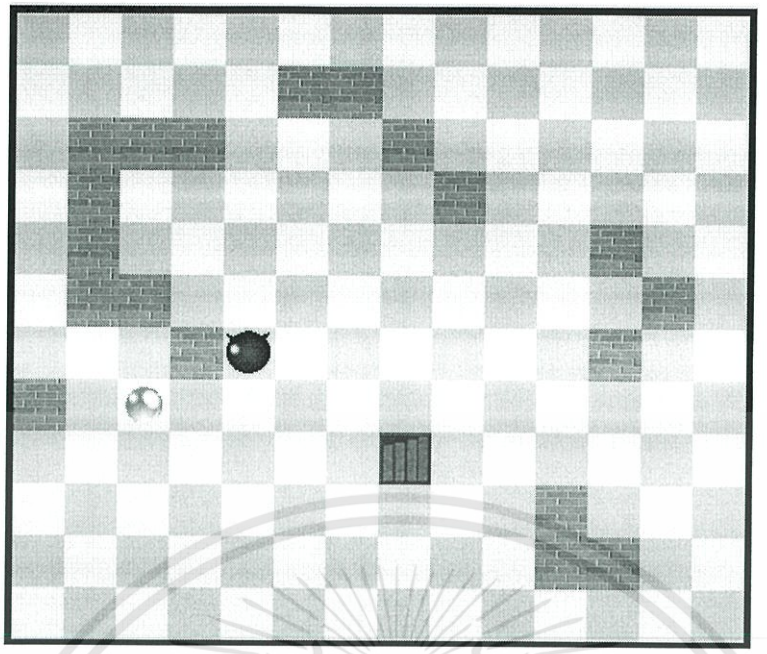
จากการทดลองจะเห็นได้ว่าการใช่ Heuristic Algorithm นั้นสามารถลดเวลาที่ใช้ในการเรียนรู้แผนที่ในรูปแบบต่างๆได้จริง โดยประสิทธิภาพที่ได้นั้นขึ้นอยู่กับความรู้ที่ได้เรียนมาก่อนหน้า ซึ่งจากการทดลองได้แสดงให้เห็นว่าเงื่อนไขแบบกฎนั้นสามารถลดเวลาได้มากที่สุด แต่ในการทดลองกับแผนที่รูปแบบที่ 5 นั้นกลับลดเวลาลงได้น้อยกว่าแบบกฎความถี่เนื่องจากยังมีกฎที่สามารถใช้ได้กับรูปแบบปัจจุบันน้อยแต่หลังจากที่ได้เรียนรู้เพื่อเพิ่มกฎจากแผนที่รูปแบบที่ 4 เข้าไปเพิ่มแล้ว การแก้ปัญหาแผนที่ในรูปแบบที่ 5 ก็กลับมาเป็นว่าแบบกฎสามารถลดเวลาได้มากกว่าโดยสามารถลดลงได้อย่างมาก

ตารางที่ 4.5 ตารางแสดงจำนวนกฎที่สร้างได้หลังจากที่เรียนแผนที่ในรูปแบบต่างๆ

รูปแบบแผนที่ที่ทำการเรียน	Rule Base		Max Sequence Base	
	เพิ่ม	รวม	เพิ่ม	รวม
1	26	26	67	67
2	37	63	72	139
3	19	82	90	229
4	32	114	90	319

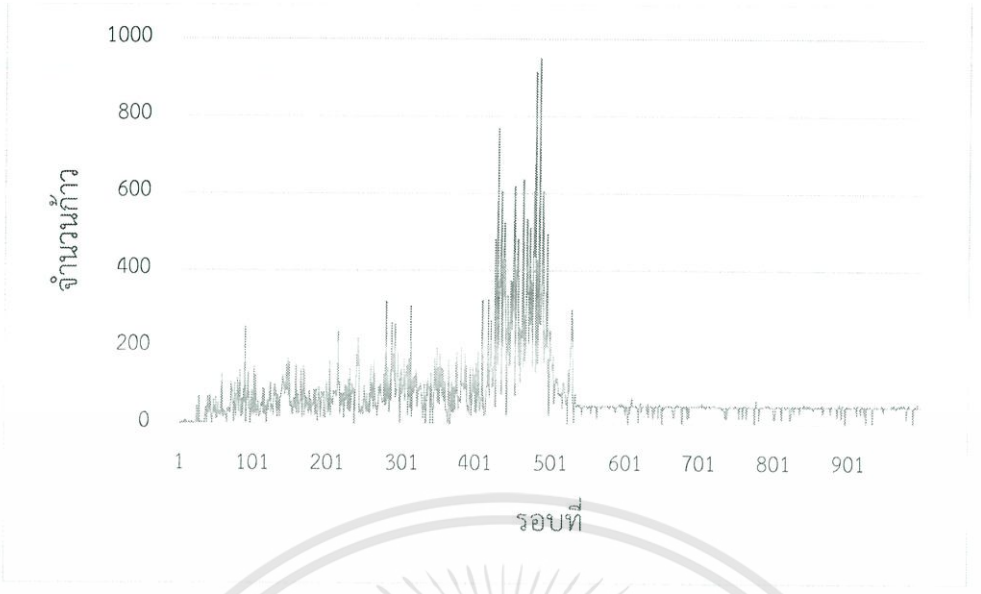
4.3 การทดลองทดสอบการแก้ปัญหา Loop

เนื่องจากอัลกอริทึมที่ใช้ในช่วงแรกของโครงงานมีข้อบกพร่องในเรื่องของปัญหาบางครั้งผู้เรียนเลือกที่จะเดินไปเดินกลับ ผู้จัดทำจึงได้วิเคราะห์และเปลี่ยนอัลกอริทึมใหม่อีกครั้ง โดยการทดลองนี้จัดทำเพื่อแสดงให้เห็นว่าปัญหาการติดลูปนั้นสามารถแก้ไขได้โดยการเปลี่ยนอัลกอริทึมเป็นการนับครั้งที่ก้าวผ่านในแต่ละ State โดยในการทดลองนี่มีการใช้แผนที่ 2 รูปแบบโดยแบบแรกจะใช้อัลกอริทึมเดิม และแบบที่สองจะใช้อัลกอริทึมที่ปรับปรุงแล้ว



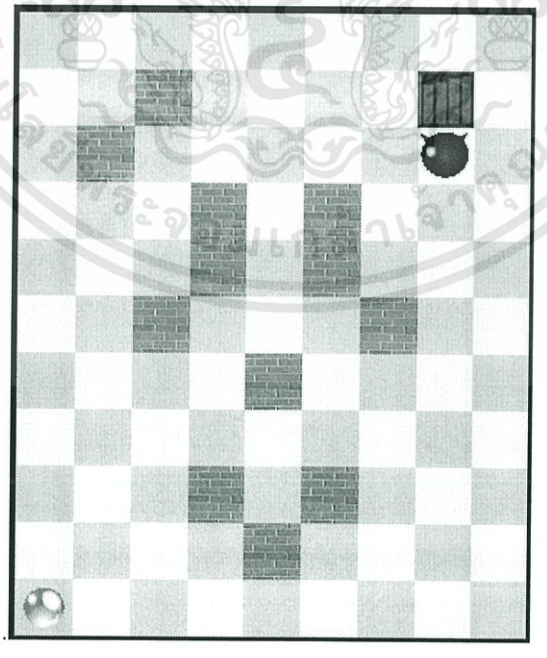
รูปที่ 4.12 รูปแผนที่รูปแบบที่ 1

รูปที่ 4.13 กราฟแสดงจำนวนก้าวที่ใช้ในแต่ละรอบ (แผนที่รูปแบบที่ 1 อัลกอริทึมเดิม)

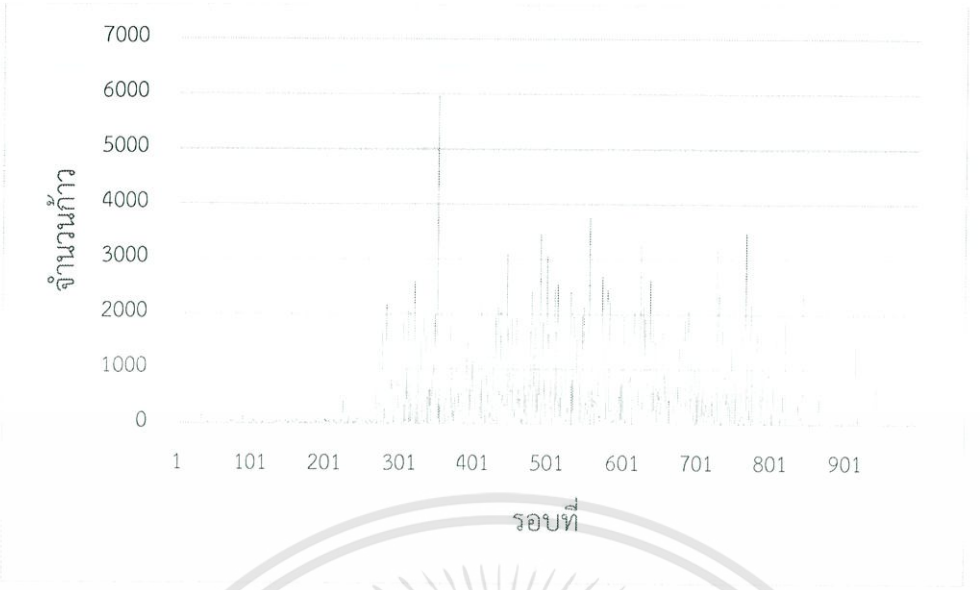


รูปที่ 4.14 กราฟแสดงจำนวนก้าวที่ใช้ในแต่ละรอบ (แผนที่รูปแบบที่ 1 อัลกอริทึมที่ปรับปรุงแล้ว)

จากการทดลองกับแผนที่รูปแบบที่ 1 นั้น การใช้อัลกอริทึมแบบเดิมนั้นมีจำนวนก้าวต่อรอบสูงสุดอยู่ที่ 2689 ก้าว ในขณะที่จำนวนก้าวต่อรอบสูงสุดอยู่ที่ 952 ก้าว การใช้อัลกอริทึมแบบปรับปรุงแล้วยังทำให้การเรียนรู้เร็วขึ้นอีกด้วยโดยจำนวนรอบที่ใช้เมื่อเจอคำตอบแรกของอัลกอริทึมแบบเดิมและแบบปรับปรุงอยู่ที่ 670 และ 534 รอบตามลำดับ



รูปที่ 4.15 รูปแผนที่รูปแบบที่ 2



รูปที่ 4.16 กราฟแสดงจำนวนก้าวที่ใช้ในแต่ละรอบ (แผนที่รูปแบบที่ 2 อัลกอริทึมเดิม)

รูปที่ 4.17 กราฟแสดงจำนวนก้าวที่ใช้ในแต่ละรอบ (แผนที่รูปแบบที่ 2 อัลกอริทึมปรับปรุงแล้ว)

จากการทดลองกับแผนที่รูปแบบที่ 2 นั้น การใช้อัลกอริทึมแบบเดิมนั้นมีจำนวนก้าวต่อรอบสูงสุดอยู่ที่ 5979 ก้าว ในขณะที่จำนวนก้าวต่อรอบสูงสุดอยู่ที่ 4308 ก้าว การใช้อัลกอริทึมแบบปรับปรุงแล้วยังทำให้การเรียนรู้เร็วขึ้นอีกด้วยโดยจำนวนรอบที่ใช้เมื่อเจอคำตอบแรกของอัลกอริทึมแบบเดิมและแบบปรับปรุงอยู่ที่ 804 และ 431 รอบตามลำดับ

บทที่ 5

บทสรุป

5.1 บทสรุป

การเรียนรู้เสริมกำลังเป็นหลักการทางการเรียนรู้ของเครื่องซึ่งเลียนแบบพฤติกรรมการเรียนรู้ของสิ่งมีชีวิต ซึ่งการเรียนรู้เสริมกำลังนั้นสามารถนำมาประยุกต์ใช้ในชีวิตประจำวันได้มากมาย ไม่ใช่แค่นำมาประยุกต์กันการแก้ปัญหาเขาวงกตเท่านั้น แต่ยังสามารถนำไปประยุกต์ใช้ก็ตามแค่ต้องสามารถที่จะกำหนดกฎเกณฑ์และผลลัพธ์ที่ต้องการและให้เวลาเครื่องได้ทำการเรียนรู้ลองผิดลองถูกเพื่อให้เครื่องได้เรียนรู้รวมถึงสร้างสถานการณ์ใหม่ๆ จนได้การกระทำที่ดีที่สุดเพื่อใช้ในการแก้ปัญหา โดยโครงงานนี้ได้นำการเรียนรู้เสริมกำลังนี้มาใช้ในการแก้ไขปัญหาเขาวงกตโดยได้สร้างส่วนเสริมของการเรียนรู้โดยให้ผู้เรียนสร้างกฎขึ้นใช้เพิ่มเพื่อให้สามารถใช้ความรู้ที่เรียนมาใช้กับปัญหาอื่นที่ใกล้เคียงกันโดยผลรับจะทำให้การเรียนรู้เร็วขึ้น

จากการทดลองพบว่ารูปแบบการเรียนรู้ที่ใช้มีความสามารถพัฒนาการเรียนรู้ และหาเส้นทางที่ใช้ระยะทางที่สั้นที่สุดได้ แต่สามารถลดเวลาที่ใช้ในการเรียนรู้ได้โดยใช้กฎที่สร้างขึ้นจากการเรียนรู้ที่ผ่านมาเป็นความรู้มาเป็นตัวเลือกในสถานะที่ไม่เคยเจอทำให้สามารถลดเวลาในการเรียนรู้เพื่อหาทางแก้ไขปัญหาแรกได้ แต่ในกรณีที่ต้องการทางแก้ปัญหาที่ดีที่สุดนั้นยังต้องใช้เวลาอยู่แต่ก็ยังใช้เวลาน้อยกว่าแบบไม่ใช้กฎดังกล่าว

5.2 ปัญหาและอุปสรรค

1. ทฤษฎีที่ทำการศึกษานั้นไม่มีกฎเกณฑ์ตายตัวว่าจะต้องทำการออกแบบหรือเขียนโปรแกรมอย่างไร เพราะมีแค่รูปแบบแนวคิด และแนวทางในการแก้ไขปัญหาเท่านั้น ทำให้ต้องศึกษาแนวคิดให้เข้าใจและสมาชิกในกลุ่มช่วยกันเรียนรู้และเรียบเรียงวิธีการเพื่อใช้ในการออกแบบและเขียนโปรแกรมตามความรู้ที่ได้ศึกษามา

2. การจัดเก็บค่าต่างๆ ที่ได้จากการเรียนรู้ และการนำค่านั้นโหลดมาใช้จำเป็นต้องมีรูปแบบวิธีการจัดเก็บที่ดีซึ่งใช้เวลาในการจัดเก็บและโหลดกลับมาใช้ ซึ่งการจัดเก็บในรูปไฟล์ใช้เวลาและไม่สามารถเข้าถึงกฎที่ต้องการได้ทันที จึงได้เปลี่ยนการจัดเก็บมาอยู่ในรูปของฐานข้อมูลซึ่งสามารถลดเวลาลงได้

3. การออกแบบและสร้างส่วนติดต่อกับผู้ใช้แบบกราฟิก ได้ออกแบบและสร้างขึ้นมาที่หลังทำให้การรวมส่วนติดต่อผู้ใช้แบบกราฟิกกับส่วนคำนวณทำได้ยาก จึงได้มีการออกแบบให้ทั้งสองส่วนทำงานร่วมกันผ่านเทรดแทนการควบรวมโปรแกรม

5.3 ข้อเสนอแนะ

1. ออกแบบรูปแบบแผนที่เพิ่มเติม

2. ปรับปรุงส่วนการควบคุมด้วยแป้นพิมพ์ให้สามารถใช้ได้ทั้งโปรแกรม

3. ปรับปรุงแก้ไขอัลกอริทึมในการสร้างเงื่อนไขฮิวริสติกเพื่อให้สามารถส่งต่อความรู้เพื่อใช้งานในปัญหาอื่นๆได้ดีขึ้น

บรรณานุกรม

[1] ภัทรภร วัฒนาชีพ, ศิริรัตน์ บุญกว้าง. "**การเรียนรู้ของเครื่อง**" ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2555

[2] จุติ นรพัลลภ, เฉลิม แซ่ลิ้ม. "**การเรียนรู้แบบเสริมกำลังในเกมหมากฮอส**" ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2555

[3] Ethem Alpaydin. Introduction to Machine Learning. Cambridge, MA : MIT Press, 2004. [2] Hojjat Adeli and Shin-Lin Hung. Machine Learning. New York : John Wiley & Sons, Inc, 1995. [4] ณัฐพงษ์ วารีประเสริฐ และ ณรงค์ ล่ำดี. ปัญญาประดิษฐ์. กรุงเทพมหานคร : บริษัท เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด. พ.ศ. 2552.

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