การเพิ่มประสิทธิภาพของการเรียนรู้แบบเสริมกำลัง โดยใช้การเรียนรู้จากทางเลือกของฮิวริสติก

IMPROVING EFFICIENCY OF THE REINFORCEMENT LEARNING ALGORITHM BY LEARNING CHOICE GENERATED FROM HEURISTIC GUIDES

ธีรธร ชูวงศ์
TEERATORN CHOOWONG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2555
KMITL 2012-EN-M-070-149

IMPROVING EFFICIENCY OF THE REINFORCEMENT LEARNING ALGORITHM BY LEARNING CHOICE GENERATED FROM HEURISTIC GUIDES

TEERATORN CHOOWONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTER ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2012

KMITL 2012-EN-M-070-149

หัวข้อวิทยานิพนธ์	การเพิ่มประสิทธิภาพของการเรียนรู้แบบเสริมกำลังโดยใช้การเรียนรู้จากทางเลือกของฮิวริสติก
นักศึกษา	นายธีรธร ชูวงศ์
รหัสนักศึกษา	52611004
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2555
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.บุญธีร์ เครือตราชู

บทคัดย่อ

ลักษณะของการสุ่มเลือกการกระทำด้วยวิธีกรีดี ε (ε-greedy) เป็นเทคนิคที่ใช้ความน่าจะเป็นในการค้นหาแบบสุ่ม(random search) ถ้าหากนำฟังก์ชันฮิวริสติกมาประยุกต์เพื่อให้คำแนะนำ อาจจะเพิ่มประสิทธิภาพของเอเจนต์ให้สามารถค้นหาคำตอบที่เหมาะสม หรือค้นพบได้เร็วขึ้นทั้งนี้ขึ้นอยู่กับฟังก์ชันฮิวริสติกที่ใช้ ถ้าหากคำตอบที่เหมาะสมไม่อยู่ในคำแนะนำของฮิวริสติกหรือฮิวริสติกแนะนำทางเลือกที่ไม่เหมาะสม อาจทำให้เอเจนต์ติดอยู่บนปริภูมิคำตอบแบบโลคอล (Local optima) แต่ในทางกลับกันถ้าคำตอบที่เหมาะสมอยู่ในส่วนหนึ่งที่ฟังก์ชันฮิวริสติกแนะนำก็จะช่วยลดทางเลือกในการค้นหาจุดเหมาะสมได้ ปัญหาคือเราไม่ทราบว่าฮิวริสติกที่ใช้ สามารถแนะนำคำตอบที่เหมาะสมได้หรือไม่ (มีคำตอบที่เหมาะสมอยู่ในเซตของคำแนะนำหรือไม่) หรืออีกนัยหนึ่งคือ ควรที่จะเชื่อหรือไม่เชื่อฮิวริสติก หากไม่เชื่อคำแนะนำเหล่านั้น ก็จะต้องค้นหาด้วยกลไกตามปกติโดยการค้นหาจากปริภูมิคำตอบที่เป็นไปได้ทั้งหมด ซึ่งต้องอาศัยเวลาค่อนข้างมาก ดังนั้นในวิทยานิพนธ์เล่มนี้ จึงนำเสนอแนวคิดของการประยุกต์ใช้คำแนะนำของฮิวริสติกร่วมกับน้ำหนักของการเรียนรู้จริง ด้วยเทคนิคของเซตทางเลือกที่แนะนำ เอเจนต์สามารถเรียนรู้คำแนะนำเหล่านั้นจากการลองเลือกตาม หากพบว่า คำแนะนำให้ผลการเรียนรู้ที่ไม่ดี เอเจนต์สามารถรู้จำได้ด้วยกลไกการปรับปรุงความรู้ตามปกติ และหลีกเลี่ยงเส้นทางดังกล่าวในครั้งถัดไป และในลักษณะของการแก้ปัญหาที่มีเส้นทางการค้นหาคำตอบค่อนข้างลึก เช่น การแก้ปัญหาการเลือกโปรโตไทป์ จะพบว่าการใช้คำแนะนำจากฮิวริสติกจะสามารถช่วยให้เอเจนต์สามารถค้นหาคำตอบที่เหมาะสมได้พบ อีกทั้งการรู้จำตัวเลือกที่ไม่ดีก็จะช่วยให้เอเจนต์หลีกเลี่ยงเส้นทางของคำแนะนำที่เป็นคำตอบแบบโลคอลได้ ในขณะที่กระบวนการเรียนรู้แบบเสริมกำลังแบบปกติไม่สามารถค้นหาได้พบ หรือใช้จำนวนขั้นในการเรียนรู้ที่ค่อนข้างสูงกว่า

Thesis Title	IMPROVING EFFICIENCY OF THE REINFORCEMENT LEARNING ALGORITHM BY LEARNING CHOICE GENERATED FROM HEURISTIC GUIDES
Student	Mr. Teeratorn Choowong
Student ID.	52611004
Degree	Master of Engineering
Program	Computer Engineering
Year	2012
Thesis Advisor	Assoc.Prof. Dr. Boontee Kruatrachue

ABSTRACT

In a problem with large search space (NP-complete problem), It is hard for normal e-greedy used in RL to find an optimal or even near optimal answer. In order to improve the searching time, an heuristic can be applied to limit the search space only to the choice suggested by heuristics. The problem is the quality of the heuristic. If the optimal point or near optimal point is not included in the heuristic choice, RL that believe can't find the points. Then, should RL believe or not to believe the heuristic? In this thesis, the heuristic is used in suggestion choice favored by the heuristic to limit the search space. At the same time we also applied normal Q-learning and ε-greedy of RL to correct the improper choice suggested by the heuristic. Since the bad suggested choice will be revealed by the reward of RL during normal Q-learning. So, the use of heuristic to normal Q-learning with ε-greedy can help improve both the quality of the solution generated from RL and also the time taken to obtain the solution. In order to investigate the effectiveness of the techniques with the depth of solution path, the prototype selection problem is used as a test problem. The "improper" heuristic is also used to the problem, the proposed algorithm also find the near optimal point nearer than suggestion by the heuristic.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จลุลวงไปด้วยดี เป็นเพราะความกรุณาจากท่านอาจารย์ รศ.ดร.บุญธีร์ เครือตราชู ที่ได้กรุณาให้เกียรติเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ท่านทั้งให้คำปรึกษา, สร้างเสริมพื้นฐาน, แนะนำแนวคิดที่เป็นประโยชน์ ตลอดจนตรวจทานและแก้ไขข้อบกพร่องต่างๆจนทำให้วิทยานิพนธ์เล่มนี้ประสบความสำเร็จโดยสมบูรณ์ ข้าพเจ้าต้องขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบพระคุณ กรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้กรุณาให้คำแนะนำในเรื่องต่าง ๆ ที่เสริมสร้างให้วิทยานิพนธ์เล่มนี้มีความสมบูรณ์ของเนื้อหางานวิจัยในแง่มุมต่างๆ ครบถ้วนยิ่งขึ้น

ขอขอบคุณ ปิยวรรณ ศรีสมานมิตร สำหรับกำลังใจและทุกๆคำปรึกษา ตลอดจนให้การสนับสนุน ผลักดันในทุก ๆ เรื่อง เพื่อให้ข้าพเจ้าสามารถสร้างสรรค์วิทยานิพนธ์เล่มนี้สำเร็จลุล่วงได้ด้วยดี

ขอกราบขอบพระคุณ บิดามารดาและครอบครัวของข้าพเจ้า ที่ช่วยเสริมสร้างลักษณะนิสัยของผู้ใฝ่รู้และรักการศึกษา ส่งผลให้ข้าพเจ้าสามารถมาถึง ณ จุดนี้ได้

ขอขอบคุณสมาชิกห้อง 804 ทั้งในอดีตและปัจจุบันทุกท่าน สำหรับมิตรภาพที่ดี แนวคิดและคำแนะนำต่างๆ ทั้งในด้านการศึกษาและการดำเนินชีวิต ขอขอบคุณพี่รัตติกร, พี่นพพล, พี่ศรชัย สำหรับทุกๆ ความช่วยเหลือ ขอขอบคุณ อรรถนิติ, ชานนท์, จตุรนต์ และ น้องเฉียบวุฒิ ที่ได้สร้างความสนุกสนาน มอบรอยยิ้ม และเสียงหัวเราะ รวมไปถึงการแลกเปลี่ยนความรู้ในหลายๆครั้งและหลายๆโอกาส

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์และบัณฑิตศึกษาคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้เปิดโอกาสให้ข้าพเจ้าได้เข้ารับการศึกษา ณ สถาบันแห่งนี้ ตลอดจนให้การสนับสนุนและได้มอบทุนสนับสนุนการทำวิทยานิพนธ์เพื่อให้วิทยานิพนธ์เล่มนี้สำเร็จลุล่วงไปได้ด้วยดี

คุณประโยชน์อันใดที่พึงเกิดจากวิทยานิพนธ์เล่มนี้ต่อท่านผู้มีความสนใจ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่านที่กล่าวมาแล้วข้างต้น และหากวิทยานิพนธ์เล่มนี้ เกิดข้อผิดพลาดประการใด ข้าพเจ้าจักขอน้อมรับไว้แต่เพียงผู้เดียว

ธีรธร ชูวงศ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)[1] หรืออัลกอริธึม RL ถือเป็นกระบวนการเรียนรู้ของเครื่องคำนวณที่ใช้แนวคิดของกระบวนการตัดสินใจมาร์คอฟ (Markov decision process)[2] หรือ MDP ที่กำหนดให้ผู้เรียนรู้หรือเอเจนต์ (Agent) เรียนรู้ที่จะแก้ปัญหาด้วยการลองผิดลองถูกบนสถานะ (State) ของสภาพแวดล้อม (Environment) เพื่อให้ได้รางวัลรวมสูงสุดที่คาดว่าจะได้รับซึ่งมีชื่อเรียกว่า ฟังก์ชันมูลค่า (Value Function) ในการเรียนรู้ระยะยาว[2] และคำตอบของปัญหาจะได้จากเส้นทางการค้นหาที่เอเจนต์ได้ผ่านไปในหนึ่งรอบการแก้ปัญหา

แนวทางในการเลือกการกระทำเพื่อแก้ปัญหาของเอเจนต์หรือเรียกอีกอย่างหนึ่งว่าโพลิซี (Policy) ซึ่งนิยมใช้โพลิซีแบบกรีดี ε (ε-greedy) และเนื่องจาก ε เป็นอัตราร้อยละ จะส่งผลโดยตรงต่อความลึกของเส้นทางการค้นหาคำตอบ หากว่าค่า ε มีค่าน้อยเกินไป ส่งผลให้เอเจนต์เลือกการกระทำด้วยการกรีดีเป็นส่วนมาก ถ้าหาคำตอบที่ดีที่สุดไม่พบแต่แรก เอเจนต์มีโอกาสน้อยมากที่จะพบคำตอบที่ดีที่สุดในภายหลังได้ และถ้าค่า ε มีค่ามากเกินไป เอเจนต์จะเรียนรู้ด้วยการสำรวจเป็นหลัก ทำให้มีโอกาสน้อยมากที่เอเจนต์จะสามารถกลับมายังคำตอบที่อยู่ในเส้นทางการค้นหาเดิมในปัจจุบันได้

ดังนั้นการใช้โพลิซีแบบ ε-greedy เพียงอย่างเดียวจึงไม่เหมาะสมกับการเรียนรู้ในทุกๆ ปัญหา และเพื่อเป็นการปรับปรุงโพลิซีการเรียนรู้ จึงได้มีการประยุกต์ใช้การแนะนำจากฟังก์ชันฮิวริสติกเพื่อช่วยให้เอเจนต์ นำมาประกอบการตัดสินใจเลือกการกระทำ เพื่อเป็นการควบคุมเส้นทางการค้นหาบนปริภูมิคำตอบให้มีทิศทางที่ถูกต้องเหมาะสม มีผลทำให้ความเร็วในการเรียนรู้ลู่เข้าสู่คำตอบที่ดีเร็วขึ้น

วิทยานิพนธ์เล่มนี้ จึงเสนอแนวคิดในการปรับปรุงความเร็วของกระบวนการเรียนรู้แบบเสริมกำลัง ด้วยการใช้ฟังก์ชันฮิวริสติกช่วยแนะนำที่นำเสนอในบทความ Accelerating autonomous learning by using heuristic selection of actions (HAQL) [3] และทำการเพิ่มประสิทธิภาพทางด้านการค้นหาแบบโลคอล (Local Search) จากบทความ Reinforcement Learning Algorithm for the Minimal Consistent Subset Identification [4] และหากพบว่าฟังก์ชันฮิวริสติกที่ใช้ ค้นหาคำตอบได้ไม่ดีเท่าที่ควร เอเจนต์จะสามารถรู้จำคำแนะนำนั้นๆอย่างไร โดยใช้ปัญหาการเลือกโปรโตไทป์และปัญหา MCSI [5] เป็นกรณีศึกษา

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

เพื่อเพิ่มประสิทธิภาพในการเรียนรู้ของอัลกอริธึมการเรียนรู้แบบเสริมกำลัง โดยใช้คำแนะนำจากฟังก์ชันฮิวริสติก ที่เอเจนต์สามารถรู้จำตัวเลือกที่ดีเก็บไว้ อีกทั้งยังลดจำนวนขั้นของการเรียนรู้ที่เป็นการลองผิดลองถูก ผนวกกับอัลกอริธึมที่มีประสิทธิภาพในการค้นหาแบบโลคอล เพื่อให้ได้คำตอบของปัญหาที่ดีที่สุดภายใต้เงื่อนไขที่จำกัด

1.3 สมมติฐานของการศึกษา

การเพิ่มประสิทธิภาพให้กับการเรียนรู้แบบเสริมกำลังด้วยการแนะนำของฟังก์ชันฮิวริสติก จะช่วยเพิ่มความเร็วในการเรียนรู้ให้ลู่เข้าสู่คำตอบที่ดีได้ และหากผสมผสานการค้นหาแบบโลคอลจะทำให้เอเจนต์ค้นหาคำตอบในทิศทางของฮิวริสติกอย่างมีเหตุผล อีกทั้งหากเอเจนต์มีความสามารถในการเลือกไม่เชื่อถือการแนะนำที่ผิดพลาดของฮิวริสติกจะสามารถหลีกเลี่ยงคำตอบที่ดีแบบโลคอลได้

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

แนวคิดหลักที่ใช้ในวิทยานิพนธ์นี้ได้แก่ กระบวนการเรียนรู้แบบเสริมกำลัง(RL)[1-2, 7-8], เทคนิคการเพิ่มความเร็วให้กับการเรียนรู้แบบเสริมกำลัง ด้วยการแนะนำฟังก์ชันฮิวริสติก (HAQL)[3] และเทคนิคการกลับไปยังสถานะที่สามารถเรียนรู้ซ้ำได้ (RL-RCS)[4, 15] โดยเลือกใช้ฟังก์ชันฮิวริสติกที่สร้างจากวิธีแก้ปัญหาการเลือกโปรโตไทป์เช่น เทคนิคอัตราการค้นคืน และเทคนิคการแก้ปัญหา MCSI [5, 9-13] ซึ่งได้ทำการปรับปรุงการคำนวณภายในของเอเจนต์ให้สามารถเลือกเชื่อถือหรือปฏิเสธการแนะนำของฟังก์ชันฮิวริสติกได้

1.5 ขอบเขตการวิจัย

วิทยานิพนธ์เล่มนี้นำเสนอผลการศึกษาและพัฒนากระบวนการเรียนรู้แบบเสริมกำลัง โดยมีขอบเขตงานวิจัยดังนี้

1. การพัฒนาอัลกอริธึมในงานวิจัยนี้ ใช้พื้นฐานของกระบวนการเรียนรู้แบบเสริมกำลัง โดยเลือกใช้อัลกอริธึมที่มีการพัฒนาปรับปรุงมาก่อนหน้านี้ นำมาประยุกต์เพื่อเพิ่มประสิทธิภาพในการค้นหาคำตอบ
2. อัลกอริธึมของการสร้างฟังก์ชันฮิวริสติกในงานวิจัยนี้ ได้ใช้แนวคิดจากผลของอัตราการค้นคืนหรือกระบวนการแก้ปัญหา MCSI เนื่องจากเป็นอัลกอริธึมที่สอดคล้องกับจุดมุ่งหมายในการเรียนรู้เพื่อแก้ปัญหาของกรณีศึกษาที่นำมาใช้ทดสอบ
3. ชุดข้อมูลนำมาทดสอบในกรณีศึกษาจะมีค่าคุณลักษณะเฉพาะที่เป็นจำนวนจริงและในแต่ละชนิดของชุดข้อมูล สมาชิกทุกตัวมีมิติของคุณลักษณะเฉพาะจำนวนเท่าๆกัน
4. ชุดข้อมูลที่นำมาทดสอบเป็นชุดข้อมูลที่นำมาจากเว็บไซต์ของมหาวิทยาลัยแคลิฟอร์เนีย เออร์ไวน์ (University of California, Irvine) [6] ซึ่งเป็นแหล่งรวบรวมชุดข้อมูลที่เหมาะสำหรับใช้ในการทดลองเกี่ยวกับการเรียนรู้ของเครื่องคำนวณ
5. ชุดข้อมูลที่เลือกนำมาทดสอบเป็นกลุ่มข้อมูลขนาดเล็ก เนื่องจากสามารถอ้างอิงได้จากงานวิจัยอื่นๆ ที่มีผู้ค้นพบคำตอบที่ดีที่สุดมาก่อนหน้านี้ และเพื่อหลีกเลี่ยงปัญหาทางด้าน ทรัพยากรหน่วยความจำ ให้ได้คำตอบที่ดีที่สุดก่อนที่ระบบการเรียนรู้จะไม่สามารถเรียนรู้ต่อได้

1.6 ขั้นตอนการศึกษา

1. ศึกษาทฤษฎีและความรู้พื้นฐานที่เกี่ยวข้องกับการเรียนรู้ของระบบการเรียนรู้แบบเสริมกำลัง
2. ศึกษากระบวนการเรียนรู้ของอัลกอริธึมการเรียนรู้แบบเสริมกำลังที่มีผู้นำเสนอมาก่อนหน้านี้
3. ศึกษาทฤษฎีและความรู้พื้นฐานที่เกี่ยวข้องกับการเลือกโปรโตไทป์
4. ศึกษาทฤษฎีและความรู้พื้นฐานที่เกี่ยวข้องกับการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด
5. ศึกษาการเลือกโปรโตไทป์ที่ใช้หลักความครอบคลุม[5], และอัตราการค้นคืน[13]
6. ประยุกต์ปัญหาการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดเข้าสู่ระบบการเรียนรู้แบบเสริมกำลัง
7. พัฒนาอัลกอริธึมการเรียนรู้แบบเสริมกำลังและเทคนิคที่ช่วยในการเพิ่มประสิทธิผลของอัลกอริธึม
8. ทดสอบอัลกอริธึมที่พัฒนาขึ้นและวิเคราะห์ผลที่ได้
9. สรุปผลการทดลองและจัดทำเอกสารประกอบวิทยานิพนธ์

1.7 เครื่องมือและอุปกรณ์ที่ใช้ในงานวิจัย

1. เครื่องคอมพิวเตอร์เซิร์ฟเวอร์
   - หน่วยประมวลผลกลาง Intel i7 x64 3.5 GHz
   - หน่วยความจำหลักขนาด 32 GB
2. ระบบปฏิบัติการ Windows 7 x64
3. โปรแกรม Microsoft Visual C++ Express 2010

1.8 โครงสร้างของวิทยานิพนธ์

วิทยานิพนธ์เล่มนี้ได้แบ่งเนื้อหาออกเป็น 6 บท ดังนี้

บทที่ 1 กล่าวถึงความเป็นมาของงานวิจัย ความมุ่งหมาย วัตถุประสงค์ สมมติฐาน แนวคิดที่ใช้ในการวิจัย ขอบเขตของการวิจัย ขั้นตอนการศึกษา ตลอดจนเครื่องมือและอุปกรณ์ที่ใช้ในงานวิจัย

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่เกี่ยวข้องกับการเรียนรู้แบบเสริมกำลัง

บทที่ 3 กล่าวถึงงานวิจัยที่เกี่ยวข้อง

บทที่ 4 การเพิ่มประสิทธิภาพของการเรียนรู้แบบเสริมกำลังโดยใช้การเรียนรู้จากทางเลือกของฟังก์ชันฮิวริสติก

บทที่ 5 กล่าวถึงการทดลอง และผลการเปรียบเทียบแต่ละอัลกอริธึม

บทที่ 6 กล่าวถึงบทสรุปผลการวิจัยและข้อเสนอแนะ

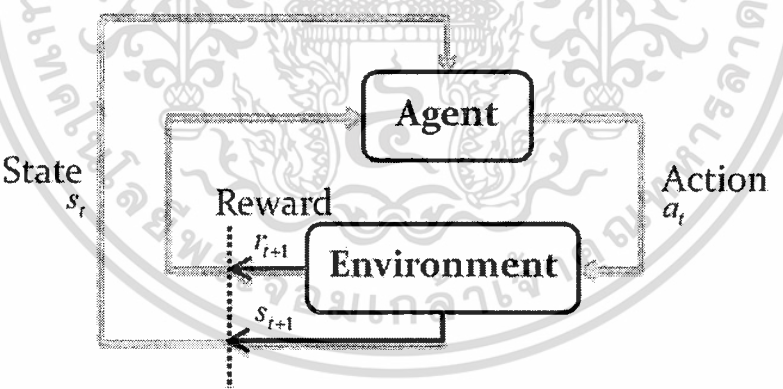
บทที่ 2
ทฤษฎีพื้นฐานที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีพื้นฐานที่ใช้ในงานวิจัยของวิทยานิพนธ์ที่นำเสนอ โดยแบ่งเนื้อหาออกเป็น 3 ส่วน ส่วนแรกอธิบายถึงทฤษฎีพื้นฐานของกระบวนการเรียนรู้แบบเสริมกำลัง ส่วนที่สองคือ แนวคิดในการใช้งานฟังก์ชันฮิวริสติกที่ป้อนเข้าสู่ระบบการเรียนรู้แบบเสริมกำลัง และส่วนที่สาม คือ การนำเสนอวิธีประยุกต์ปัญหาการระบุเซตย่อยสอดคล้องเล็กสุด(Minimal Consistent Set Identification, MCSI) เข้ากับกระบวนการเรียนรู้แบบเสริมกำลัง

2.1 กระบวนการเรียนรู้แบบเสริมกำลัง

2.1.1 การสื่อสารระหว่างเอเจนต์กับสภาพแวดล้อม(Agent-Environment Interface)

กระบวนการเรียนรู้แบบเสริมกำลัง ใช้การโต้ตอบระหว่างสภาพแวดล้อม (environment) และผู้เรียนรู้หรือเอเจนต์ (agent) โดยมีวัตถุประสงค์เพื่อให้เอเจนต์เรียนรู้การแก้ปัญหาไปจนพบคำตอบหรือเป้าหมาย(Goal) ของงาน เริ่มจากสภาพแวดล้อมแจ้งสถานะปัจจุบันที่เป็นปัญหาของงานให้กับเอเจนต์ จากนั้นเอเจนต์สุ่มเลือกการกระทำหนึ่งๆ ตอบกลับไปยังสภาพแวดล้อมเพื่อคำนวณหาสถานะถัดไปและรางวัล(Reward)จากการกระทำ และใช้การโต้ตอบนี้ไปจนกว่าจะค้นพบคำตอบ



รูปที่ 2.1 การสื่อสารระหว่างเอเจนต์และสภาพแวดล้อม

จากรูปที่ 2.1 การสื่อสารระหว่างเอเจนต์และสภาพแวดล้อมเป็นการโต้ตอบไปมาเป็นลำดับของช่วงเวลา $(t = 0, 1, 2, \dots)$ โดยในแต่ละช่วงเวลา t เอเจนต์จะรับทราบปัญหาในรูปแบบของ เซตสถานะที่เป็นไปได้ (Set of possible state) $s_t \in S$ จากนั้นเอเจนต์เลือกกระทำ $a_t \in A(s)$ มาหนึ่งค่า โดยที่ $A(s)$ คือเซตของการกระทำที่สามารถเลือกได้บนสถานะ s_t และเมื่อเวลาผ่านไป $(t + 1)$ เอเจนต์จะรับทราบผลของการกระทำ a_t ผ่านสัญญาณที่เรียกว่ารางวัล (Reward) ที่ $r_{t+1} \in R$ พร้อมกันนั้นจะรับทราบสถานะถัดไป s_{t+1}ที่สภาพแวดล้อมได้

คำนวณให้ และทำการเลื่อนไปยังสถานะถัดไป และวนดำเนินการเช่นเดิมจนกว่าจะค้นพบสถานะที่เป็นคำตอบ

ในขณะที่ช่วงเวลา t ดำเนินไป เอเจนต์จะทำการสร้างแผนผังจับคู่ระหว่างสถานะกับความน่าจะเป็นของการเลือกแต่ละการกระทำที่ได้เลือกมา แผนผังดังกล่าวจะเรียกว่า โพลิซีของเอเจนต์ (Agent's policy) ใช้สัญลักษณ์ π โดยที่ $\pi(s, a)$ หมายถึง ความน่าจะเป็นในการเลือกกระทำ a ใดๆ บนสถานะ s โดยมีจุดมุ่งหมายเพื่อ หาโพลิซี π ใดๆ ที่เป็นคำตอบที่ดีที่สุดของปัญหา โดยมุ่งเน้นให้เอเจนต์เลือกการกระทำ a_t ที่ส่งผลให้ได้รับค่าของรางวัลรวมที่มีค่าสูงสุดในแต่ละรอบของการเรียนรู้ปัญหา

2.1.2 เป้าหมาย, รางวัล และผลตอบแทน (Goals , Rewards and Returns)

สิ่งที่เป็นเป้าหมาย(goal) ของระบบการเรียนรู้แบบเสริมกำลัง คือ ให้เอเจนต์กระทำการใดๆ เพื่อสะสมค่า รางวัล(reward) ที่สภาพแวดล้อมทำการคำนวณจากการกระทำบนแต่ละสถานะ โดยมุ่งหวังให้เอเจนต์เลือกกระทำการใดๆที่ส่งผลให้ได้รับรางวัลสะสมสูงที่สุดที่เรียกว่า ผลตอบแทนที่คาดว่าจะได้รับ(expected return) ดังสมการ 2.1

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_T \qquad (2.1)$$

โดยที่ R_t คือ ฟังก์ชันลำดับของรางวัล
T คือ ลำดับขั้นของเวลาท้ายสุด

โดยทั่วไปแล้วช่วงเวลาของการสะสมรางวัลจากการแก้ปัญหาด้วยระบบการเรียนรู้แบบเสริมกำลังจะถูกแบ่งออกเป็นรอบ (episode) คล้ายกับการเล่นเกม ลำดับขั้นของเวลาท้ายสุดมีความหมายเทียบเคียงกับสถานะสิ้นสุด การแก้ปัญหาจะเริ่มขึ้นจากสถานะเริ่มต้นและดำเนินต่อไป จนพบคำตอบของปัญหา และเริ่มต้นใหม่อีกครั้ง ส่วนงานที่ไม่สามารถระบุขอบเขตสถานะสิ้นสุดได้ลำดับขั้นของเวลาท้ายสุด T จะมีค่าเป็นอนันต์และเรียกงานประเภทนี้ว่างานแบบต่อเนื่อง(continue tasks) ดังนั้นจึงใช้การลดทอน (discounting) เพื่อให้เอเจนต์สามารถเลือกคำนวณรางวัลสะสมที่ผ่านการลดทอนแล้วเพื่อเลี่ยงการคำนวณค่าเวลา T ที่เป็นอนันต์ ดังสมการ 2.2

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \qquad (2.2)$$

โดยที่ γ คือ อัตราการลดทอนค่าตั้งแต่ 0 ถึง 1

2.1.3 กระบวนการตัดสินใจมาร์คอฟ (Markov Decision Process)[2]

การเรียนรู้แบบเสริมกำลังใช้การนำเสนอด้วยหลักการ คุณสมบัติมาร์คอฟที่เรียกว่า กระบวนการตัดสินใจมาร์คอฟ เรียกโดยย่อว่า MDP ซึ่งกระบวนการตัดสินใจมาร์คอฟแบบจำกัดเขต (finite MDP) คือ คู่ลำดับของสถานะและการกระทำสามารถระบุขอบเขตได้

กระบวนการตัดสินใจมาร์คอฟ นิยามให้คู่ลำดับของสถานะและการกระทำ นำเสนอในรูปแบบพลวัต 1 ขั้น (1-step dynamic) คือ การเลือกกระทำ a บนสถานะ s จะเกิดความน่าจะเป็นค่าหนึ่งๆ ที่สามารถเคลื่อนไปยังสถานะถัดไป ดังสมการ 2.3

$$\mathcal{P}^{a}_{ss'} = Pr\{s_{t+1} = s' \mid s_t = s,\ a_t = a\} \tag{2.3}$$

รางวัลที่เกิดจากการเลือกการกระทำ a บนสถานะ s จากนั้นจึงเคลื่อนไปยังสถานะถัดไป s' สามารถนำเสนอในรูปแบบของค่ารางวัลที่คาดว่าจะได้รับ ดังสมการ 2.4

$$\mathcal{R}^{a}_{ss'} = E\{r_{t+1} \mid s_t = s,\ a_t = a, s_{t+1} = s'\} \tag{2.4}$$

2.1.4 ฟังก์ชันมูลค่า (Value Function)

อัลกอริธึมของการเรียนรู้แบบเสริมกำลังใช้พื้นฐานการประมาณฟังก์ชันมูลค่า หมายถึงค่าประมาณที่ให้เอเจนต์ ตัดสินใจกระทำเช่นไรจึงจะได้ผลดี บนสถานะที่ได้รับมา ซึ่งมูลค่าดังกล่าวจะถูกมอบให้แก่เอเจนต์ เป็นรางวัลที่คาดว่าจะได้รับทุกๆสถานะถัดไป เพื่อนำไปสะสมในรูปแบบของผลตอบแทน(Return) ที่จะได้รับ รางวัลที่มอบให้แก่เอเจนต์จะมากหรือน้อยขึ้นอยู่กับการกระทำที่ได้กระทำออกไป และขึ้นกับแต่ละฟังก์ชันมูลค่าที่ใช้กับการระบุโพลิซี (ฟังก์ชันมูลค่าของสถานะ, ฟังก์ชันมูลค่า V หรือ ฟังก์ชันมูลค่าของคู่สถานะ-การกระทำ, ฟังก์ชันมูลค่า Q) โดยมีโพลิซี π เป็นสิ่งกำหนดให้เกิดการเลือกกระทำ $a \in A(s)$ บนสถานะ $s \in S$ นำเสนอด้วยสัญลักษณ์ $\pi(s,a)$

กรณีที่พิจารณาเพียงเฉพาะสถานะ(ไม่ขึ้นกับการกระทำใดๆ) มูลค่าดังกล่าวเรียกว่าฟังก์ชันมูลค่าของสถานะ(ฟังก์ชันมูลค่า V) ใช้สัญลักษณ์ $V^{\pi}(s)$ หมายถึงผลตอบแทนที่คาดว่าจะได้รับบนสถานะ s จากการใช้โพลิซี π ดังสมการ 2.5

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{R_t \mid s_t = s\} \\ &= E_{\pi}\{\textstyle\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\} \end{aligned} \tag{2.5}$$

โดยที่ $E_{\pi}\{\dots\}$ หมายถึง มูลค่าประมาณที่คาดว่าเอเจนต์ได้รับหลังจากที่เคลื่อนไปยังสถานะถัดไปจากการใช้โพลิซีπ

t หมายถึง ณ เวลาหนึ่งที่กำลังสนใจ

สถานะสิ้นสุด(Terminate State) จะมีฟังก์ชันมูลค่า Vเป็นศูนย์

และในกรณีที่พิจาณาฟังก์ชันมูลค่าที่เกิดจากคู่การกระทำ a บนสถานะ s หรือฟังก์ชันมูลค่าของคู่สถานะ-การกระทำ(ฟังก์ชันมูลค่า Q) ใช้สัญลักษณ์ $Q^{\pi}(s,a)$ หมายถึงผลตอบแทนที่คาดว่าจะได้รับบนสถานะ s ที่ได้กระทำ a จากการใช้โพลิซี π ดังสมการ 2.6

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R_t \mid s_t = s, a_t = a\} \\ &= E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right\} \end{aligned} \tag{2.6}$$

2.1.5 ฟังก์ชันมูลค่าที่เหมาะสม (Optimal value function)

กระบวนการเรียนรู้แบบเสริมกำลังมีจุดประสงค์การเรียนรู้ เพื่อหาโพลิซีที่ดีที่สุดให้กับเอเจนต์ตลอดการแก้ปัญหา ซึ่งหากอธิบายตามหลักของกระบวนการตัดสินใจมาร์คอฟแบบจำกัดเขต (หัวข้อ 2.1.3) ได้ให้ความหมายของโพลิซีที่เหมาะสมไว้ว่า จะต้องมีโพลิซีπ ใดๆที่ให้ผลตอบแทนที่คาดว่าจะได้รับดีกว่าหรือเทียบเท่ากับโพลิซี π' ให้ถือว่าโพลิซีนั้นคือ โพลิซีที่เหมาะสม(optimal policy) ใช้สัญลักษณ์ π^*

ดังนั้นฟังก์ชันมูลค่า V ที่เหมาะสมจึงคำนวณจากการเปรียบเทียบฟังก์ชันมูลค่า V ที่ดีที่สุดจากการใช้โพลิซีที่ต่างกัน ใช้สัญลักษณ์ V^* ดังสมการ 2.7

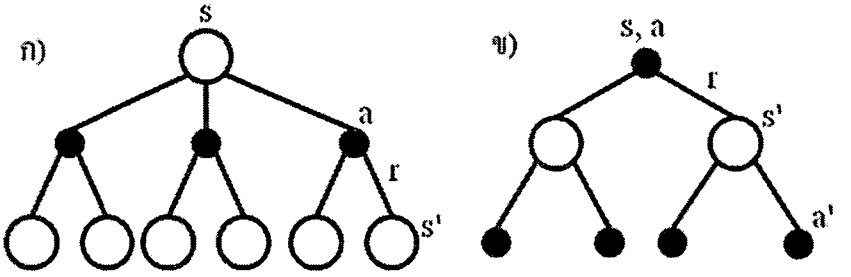
$$V^*(s) = \max_{\pi} V^{\pi}(s) \text{ ในทุกๆ } s \in S \tag{2.7}$$

เช่นเดียวกันฟังก์ชันมูลค่า Q ที่เหมาะสมจึงเกิดจากการเปรียบเทียบการกระทำที่ดีที่สุดบนสถานะเดียวกัน จากการใช้โพลิซีที่ต่างกัน ใช้สัญลักษณ์ Q^* ดังสมการ 2.8

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \text{ ในทุกๆคู่ } (s \in S, a \in A(s)) \tag{2.8}$$

2.1.6 แผนภาพแบคอัพ(Backup Diagram)

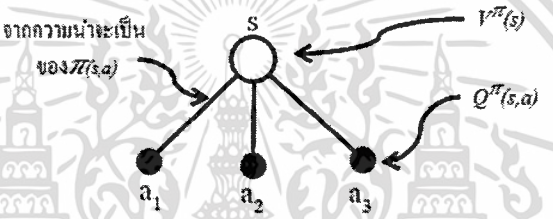
แผนภาพแบคอัพเป็นแผนภาพแสดงความสัมพันธ์ของการปรับปรุงฟังก์ชันมูลค่าที่เกิดจากการเรียนรู้ โดยเรียกว่ากระบวนการแบคอัพ ถือเป็นหัวใจของกระบวนการเรียนรู้แบบเสริมกำลังก็ว่าได้ โดยแสดงถึงการแปลงข้อมูลของฟังก์ชันมูลค่าลงสู่กราฟของสถานะ-การกระทำในลักษณะเริ่มต้นเรียนรู้จากโหนดเริ่มต้นไปสู่โหนดลูกหลานดังรูปที่ 2.2



รูปที่ 2.2 แผนภาพแบคอัพ ก) ฟังก์ชันมูลค่า V, ข) ฟังก์ชันมูลค่า Q

2.1.6.1 แผนภาพแบคอัพของฟังก์ชันมูลค่า V

มูลค่าของสถานะขึ้นอยู่กับมูลค่าของการกระทำที่เป็นไปได้บนสถานะนั้นๆ ซึ่งในแต่ละการกระทำนั้นจะสอดคลองและเป็นไปตามโพลิซีที่ใช้ โดยในแต่ละโหนดของแผนภาพนั้นสามารถอธิบายโดยละเอียดดังรูปที่ 2.3

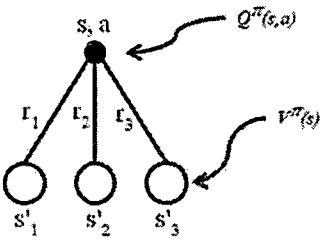


รูปที่ 2.3 แผนภาพแบคอัพฟังก์ชันมูลค่า V

ตามแผนภาพ โหนดเริ่มต้นคือฟังก์ชันมูลค่า $V^{\pi}(s)$ และแต่ละโหนดลูกหลาน คือ ฟังก์ชันมูลค่า $Q^{\pi}(s,a)$ เมื่อสถานะ $s_t = s$ ที่ใช้โพลิซี π

2.1.6.2 แผนภาพแบคอัพของฟังก์ชันมูลค่า Q

มูลค่าที่คาดว่าจะได้รับจากฟังก์ชันมูลค่า $Q^{\pi}(s,a)$ ที่เกิดจากโพลิซี π จะแบ่งออกเป็น 2 ส่วน โดยส่วนแรกเป็น รางวัลที่คาดว่าจะได้รับ ผลของรางวัลไม่ขึ้นกับการกระทำที่ถูกเลือก โดยโพลิซี π ใดๆ ในส่วนถัดมาคือ ผลรวมของค่าตอบแทนที่คาดว่าจะได้รับ ผลของมูลค่าจะสอดคล้องกับสถานะถัดไปและโพลิซีที่เลือกใช้ สามารถแสดงเป็นแผนภาพแบคอัพได้รูปที่ 2.4



รูปที่ 2.4 แผนภาพแบคอัพฟังก์ชันมูลค่า Q

ตามแผนภาพ โหนดเริ่ม คือฟังก์ชันมูลค่า Q บนสถานะ s $(\mathrm{Q}^{\pi}(\mathrm{s},a))$ ที่ใช้โพลิซี π และโหนดลูกหลาน คือ สถานะถัดไปที่เป็นไปได้ประกอบไปด้วยมูลค่าของรางวัลที่คาดว่าจะได้รับ r_{t+1}และมูลค่าของฟังก์ชันมูลค่า V ที่คาดว่าจะได้รับของสถานะถัดไป $\mathrm{V}^{\pi}(\mathrm{s}')$ ที่เกิดจากการเลือกกระทำ a ใดๆ บนสถานะ $s_t = \mathrm{s}$

2.1.7 วิธีการเรียนรู้ (Learning methods)

วิธีการหาคำตอบในการแก้ปัญหาของการเรียนรู้แบบเสริมกำลังแต่ละวิธีการนั้นมีข้อดีและข้อด้อยแตกต่างกันซึ่ง มีทั้งวิธีการที่ใช้การสร้างแบบจำลองที่เกิดจากการคำนวณทางคณิตศาสตร์ แต่ต้องการความสมบูรณ์หรือความแม่นยำของแบบจำลองของปัญหาที่นำมาแก้ หากเป็นแบบจำลองที่ง่ายเกินไปจะไม่ครอบคลุมปัญหาเฉพาะกิจ(Ad-Hoc)บางประการที่ต้องการความซับซ้อนทางการคำนวณ และวิธีการที่มีการสร้างแบบจำลองที่ซับซ้อนมากเกินไปก็ส่งผลกระทบต่อความเร็วของการเรียนรู้หรือคำตอบของปัญหาที่ไม่ใช่คำตอบที่เหมาะสม

2.1.7.1 วิธีไดนามิคโปรแกรมมิ่ง (dynamic Programming, DP)

เป็นอัลกอริธึมที่สามารถคำนวณหาโพลิซีที่เหมาะสมจากการคำนวณจากแบบจำลองที่สมบูรณ์บนสภาพแวดล้อมด้วยกระบวนการตัดสินใจมาร์คอฟ แต่อัลกอริธึมนี้สิ้นเปลืองทรัพยากรทางการคำนวณเพื่อสร้างแบบจำลองมากหากเปรียบเทียบกับวิธีการอื่นๆ

2.1.7.2 วิธีมอนเทคาร์โล (Monte Carlo, MC)[1]

อัลกอริธึมนี้ต้องการเพียงประสบการณ์ในการเรียนรู้จากลำดับของสถานะ, การกระทำ และรางวัลจากการโต้ตอบจริงกับสิ่งแวดล้อม โดยใช้การเฉลี่ยจากผลตอบแทนจริงที่กำหนดไว้ในแต่ละเอพิโสด ซึ่งเมื่อได้คำตอบของปัญหา ค่าประมาณของฟังก์ชันมูลค่า และโพลิซีก็จะถูกปรับปรุงให้ดีขึ้น แต่อาจไม่ดีขึ้นอย่างต่อเนื่องเพราะคุณลักษณะของการค้นหาคำตอบด้วยการสุ่มการกระทำด้วยโพลิซีแบบ ε-greedy

โพลิซีแบบ ε-greedy เป็นโพลิซีที่นิยมใช้อย่างแพร่หลายเนื่องจาก ตลอดช่วงเวลาของการเรียนรู้เอเจนต์มีโอกาสบ่อยครั้งที่สามารถเลือกการกระทำที่ได้รับฟังก์ชันมูลค่ามีค่ามาก(greedy) และมีโอกาสส่วนหนึ่ง ε ที่สามารถเลือกการกระทำอื่นๆที่เป็นการสุ่ม (random) เพื่อค้นหาฟังก์ชันมูลค่าที่ดีกว่าเดิม โดยความน่าจะเป็นของการเลือกการกระทำตามค่ามากมีอัตรา $(1-\varepsilon)+\frac{\varepsilon}{|A(s)|}$ และความน่าจะเป็นของสุ่มการกระทำอื่นมีอัตรา $\frac{\varepsilon}{|A(s)|}$ จะเห็นว่า การใช้งานโพลิซีข้างต้นจะยังคงความสามารถในการประมาณค่าโพลิซีได้ ในขณะที่สามารถปรับปรุงโพลิซีใหม่ที่เกิดจากการสุ่มการสำรวจการกระทำอื่นๆที่ดีกว่าภายในเอพิโสด ซึ่งแสดงเป็นซูโดโค๊ดได้ดังรูปที่ 2.5

กำหนด$Q(s,a)$เป็นค่าใดๆและ$Return(s,a)$เป็นค่าว่างทุกๆคู่สถานะ-การกระทำ, โดยที่โพลิซีπ เป็นไปตามโพลิซีแบบ ε-greedy

ดำเนินการเรียนรู้ซ้ำในแต่ละเอพิโสดด้วยโพลิซีπ

ในแต่ละคู่สถานะ-การกระทำ(s,a)ที่ผ่านไปในเอพิโสด

$R \leftarrow$ผลตอบแทนที่ได้รับจากการผ่านไปยังคู่สถานะ-การกระทำ s,a

$$Return(s,a) \leftarrow Return(s,a) + R$$

$$Q(s,a) \leftarrow avg(Return(s,a))$$

ในแต่ละสถานะ s ในเอพิโสด

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

ในทุกๆ $a \in A(s)$

$$\pi(s,a) \leftarrow \begin{cases} (1-\varepsilon) + \frac{\varepsilon}{|A(s)|}, & \text{กรณี a} = a^* \\ \frac{\varepsilon}{|A(s)|}, & \text{กรณี a} \neq a^* \end{cases}$$

รูปที่ 2.5 ซูโดโค้ดของอัลกอริธึมวิธีมอนเทคาร์โลที่ใช้โพลิซี $\boldsymbol{\pi}$ แบบกรีดี $\boldsymbol{\varepsilon}$

2.1.7.3 วิธีเรียนรู้ผลต่างระหว่างเวลา (Temporal Difference Learning, TD) [7]

กระบวนการเรียนรู้ผลต่างระหว่างเวลาเป็นการเรียนรู้ที่ผสมผสานแนวความคิดระหว่างวิธีมอนเทคาร์โลและไดนามิคโปรแกรมมิ่ง ในส่วนที่เหมือนกับวิธีมอนเทคาร์โลคือเรียนรู้จากการแก้ปัญหาจริงปราศจากแบบจำลองของปัญหาที่อยู่บนสภาพแวดล้อมและส่วนที่เหมือนกับวิธีไดนามิคโปรแกรมมิ่งคือ วิธีการปรับปรุงการเรียนรู้ที่ใช้พื้นฐานการประมาณมูลค่าเรียนรู้ที่ไม่ต้องรอผลของผลตอบแทน ณ ตอนท้ายเอพิโสด

การเรียนรู้ Q (Q-Learning) [2] หรือเรียกว่า การเรียนรู้ Q แบบขั้นเดียว (One-Step Q-learning) เป็นการเรียนรู้ด้วยการอ้างอิงฟังก์ชันมูลค่า Q^* ดังสมการ 2.9

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)] \qquad (2.9)$$

โดยที่ α หมายถึง การปรับขนาด (Step size) ของผลรางวัลที่คาดว่าจะได้รับที่ด้วยเทคนิคค่าเฉลี่ยแบบเคลื่อนที่

$\max_a Q(s_{t+1},a)$ หมายถึง ฟังก์ชันมูลค่าQ ที่เหมาะสมของสถานะถัดไป s_{t+1}

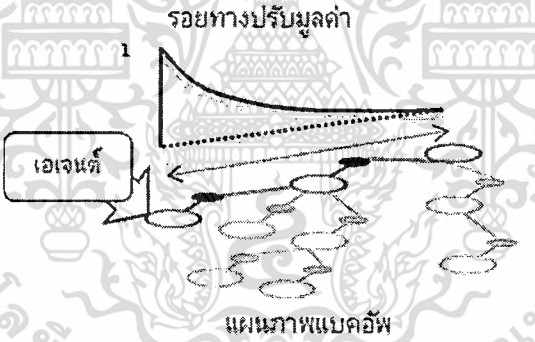
เป็นการเลือกใช้โพลิซีที่มีการคำนวณฟังก์ชันมูลค่า Q ที่อ้างอิงจากผลต่างของฟังก์ชันมูลค่า Q ที่กำลังสนใจกับฟังก์ชันมูลค่า Q^* บนสถานะเดียวกันภายในเอพิโสดหนึ่งๆ เพื่อเป็นอัตราอ้างอิงในการปรับปรุงฟังก์ชันมูลค่าที่ได้จากการเรียนรู้ด้วยการกระทำล่าสุด สามารถแสดงเป็นซูโดโค๊ดดังรูปที่ 2.6

กำหนดค่าเริ่มต้น $Q(s,a)$ เป็นค่าใดๆ
ดำเนินการเรียนรู้ซ้ำในแต่ละเอพิโสด
    กำหนดสถานะ s
    ดำเนินการวนซ้ำในแต่ละขั้นตอนภายในเอพิโสดจนกว่าสถานะ s เป็นสถานะสิ้นสุด
        เลือกการกระทำ a ณ สถานะ s ภายใต้โพลิซีที่คำนวณจากฟังก์ชันมูลค่า Q
        กระทำ a เพื่อให้ทราบรางวัล r และสถานะถัดไป s'
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma\, max_a, Q(s',a') - Q(s,a)]$
        $s \leftarrow s'$

รูปที่ 2.6 ซูโดโค๊ดของอัลกอริธึมวิธีการเรียนรู้ Q

2.1.7.4 วิธีรอยทางปรับมูลค่า

รอยทางปรับมูลค่า หมายถึง ตัวแปรที่ใช้จดจำสถานะต่างที่เอเจนต์เคยเคลื่อนผ่าน คล้ายกับการสร้างเส้นทางจากสถานะปัจจุบันกลับไปยังสถานะเริ่มต้นดังรูปที่ 2.7 โดยมีจุดประสงค์เพื่อให้การปรับปรุงฟังก์ชันมูลค่าบนสถานะ s_t หนึ่งๆ ให้สามารถส่งผลการปรับปรุงฟังก์ชันมูลค่าไปยังสถานะก่อนหน้าได้ $(s_{t-1}, s_{t-2}, \dots, s_{t-n})$ ซึ่งทำให้มูลค่าของการเรียนรู้ลู่เข้าสู่ค่าที่เหมาะสมได้เร็วกว่าเดิม



รูปที่ 2.7 การบันทึกรอยทางปรับมูลค่าจากการค้นหาของเอเจนต์

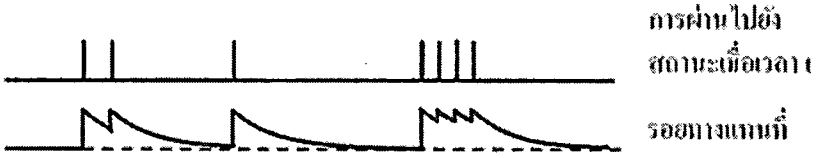
รอยทางปรับมูลค่า บนสถานะ s ที่เวลา t ใดๆ ใช้สัญลักษณ์ $e_t(s) \in R^+$ มูลค่าของรอยทางดังกล่าวจะเกิดการลดทอนของน้ำหนักรอยทาง ดังสมการ

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s), & \text{กรณี } s \neq s_t \\ 1, & \text{กรณี } s = s_t \end{cases} \qquad (2.10)$$

ในทุกๆสถานะ s ที่มิใช่สถานะสิ้นสุด

โดยที่ γ คือ อัตราปรับลด (discount rate)
λ คือ ตัวแปรของการปรับลดเส้นทาง (trace-decay parameter)

รอยทางปรับมูลค่าดังกล่าวมีชื่อเรียกว่า รอยทางแทนที่ (replacing trace) เนื่องจากค่าของรอยทางจะถูกปรับน้ำหนักใหม่เมื่อเอเจนต์ได้ผ่านกลับไปยังสถานะเดิม (ภายในเอพิโสดเดียวกัน) อีกครั้งและมูลค่าสะสมที่ปรับปรุงไปยังสถานะอื่นๆ



รูปที่ 2.8 รอยทางปรับมูลค่าแบบแทนที่ ขณะที่เอเจนต์เคลื่อนผ่านไปยังสถานะต่างๆ

วิธีรอยทางปรับมูลค่าเป็นหนึ่งในกลไกพื้นฐานของการเรียนรู้แบบเสริมกำลัง ตัวอย่างเช่นการประยุกต์ใช้กับอัลกอริธึมการเรียนรู้Q (Q-learning) เพื่อเพิ่มประสิทธิภาพในการเรียนรู้ จึงเป็นอัลกอริธึม$Watkins\ Q(\lambda)$[8] ที่ใช้ความสามารถของรอยทางปรับมูลค่า

อัลกอริธึม$Watkins\ Q(\lambda)$ กำหนดให้การกระทำที่เกิดจากการสำรวจ (nongreedy) ไม่มีมูลค่ารอยทาง และกรณีที่การกระทำถูกเลือกจากการเลือกค่ามากมีมูลค่ารอยทางเท่ากับ 1 และลดทอนด้วยอัตราเท่ากับ $\gamma\lambda$ ไปยังสถานะก่อนหน้า ดังสมการ 2.11

$$e_t(s,a) = \begin{cases} 1, & \text{กรณี } s = s_t \text{ และ } a = a_t; \\ 0, & \text{กรณี } s = s_t \text{ และ } a \neq a_t; \text{ในทุกๆคู่} (s,a) \\ \gamma\lambda e_{t-1}(s,a), & \text{กรณี } s \neq s_t; \end{cases} \quad (2.11)$$

การกระทำที่ถูกเลือก จะมีการประยุกต์ใช้รอยทางปรับมูลค่าให้ผลของฟังก์ชันมูลค่าสามารถปรับปรุงค่าส่งผลไปยังสถานะก่อนหน้าได้ ส่วนการกระทำที่ไม่ได้เลือกนั้นจะไม่ได้รับผลของการกระจายฟังก์ชันมูลค่าออกทางด้านข้าง(ตัวเลือกการกระทำอื่นๆบนสถานะเดียวกัน) ดังนั้นฟังก์ชันมูลค่าQ บนสถานะหนึ่งๆจึงมีมูลค่าของการกระทำ a ใดๆ ที่โดดเด่นหากตัวเลือกเหล่านั้นเอเจนต์ได้เคยถูกเลือกเรียนรู้มากก่อนหน้า ในการคำนวณฟังก์ชันมูลค่า Q สามารถคำนวณจากการแทนค่าพจน์ $e_t(s,a)$ ลงในสมการ 2.12

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a) \quad (2.12)$$

โดยที่ค่าการเรียนรู้ผลต่างระหว่างเวลา δ_tคำนวณได้จากสมการ 2.13

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \quad (2.13)$$

โดยที่ δ_t หมายถึง รางวัลที่คาดว่าจะได้รับของวิธีเรียนรู้ผลต่างระหว่างเวลา (หัวข้อ 2.1.7.3)

อัลกอริธึม $Watkins\ Q(\lambda)$ ที่ประยุกต์อัลกอริธึมการเรียนรู้ Q, การเรียนรู้ผลต่างระหว่างเวลา และรอยทางปรับมูลค่าเข้าด้วยกัน มีกลไกการคำนวณดังซูโดโค๊ดในรูปที่ 2.9

กำหนดค่าเริ่มต้น $Q(s,a)$ เป็นค่าใดๆ, $e(s)$ เป็นศูนย์
ดำเนินการเรียนรู้ซ้ำในแต่ละเอพิโสด
กำหนดสถานะ s, a
วนซ้ำในแต่ละขั้นตอนภายในเอพิโสดจนกว่าสถานะ s เป็นสถานะสิ้นสุด
กระทำ a เพื่อทราบมูลค่าของรางวัล r และสถานะถัดไป s'
เลือกกระทำ a'ณ สถานะs'ด้วยโพลิซีที่คำนวณจากฟังก์ชันมูลค่าQ
$a^* \leftarrow arg\max_b Q(s',b)$ (หากการกระทำ a'มีค่าสูงสุด $a^* \leftarrow a'$)
$\delta \leftarrow r + \gamma Q(s',a^*) - Q(s,a)$
$e(s,a) \leftarrow 1$
ในทุกๆคู่สถานะ-การกระทำ s, a
$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
ปรับปรุงฟังก์ชันมูลค่าตามเงื่อนไขรอยทางมูลค่าสมการ 2.11
$s \leftarrow s', a \leftarrow a'$

รูปที่ 2.9 ซูโดโค๊ดของอัลกอริธึม$\boldsymbol{Watkins\ Q(\lambda)}$

2.2 การใช้งานฟังก์ชันฮิวริสติก[3]

การแนะนำแนวคิดด้วยฟังก์ชันฮิวริสติกจะส่งผลต่อการเรียนรู้ของเอเจนต์โดยตรง หากเลือกใช้ฟังก์ชันฮิวริสติกที่ไม่ดีจะส่งผลให้เอเจนต์เลือกการกระทำที่ผิด คล้อยตามการค้นหาบนปริภูมิของฟังก์ชันฮิวริสติก คำตอบที่ได้จากการเรียนรู้แบบเสริมกำลังจะไม่ไกลจากปริภูมิดังกล่าวมากนัก แต่หากฟังก์ชันฮิวริสติกนั้นเป็นฟังก์ชันที่ดี ก็จะส่งผลให้เอเจนต์ทำการเรียนรู้คล้อยตามคำตอบที่ดีโดยไม่เสียเวลาหรือขั้นตอนในการลองผิดลองถูกบนปริภูมิด้วยตัวของเอเจนต์เอง

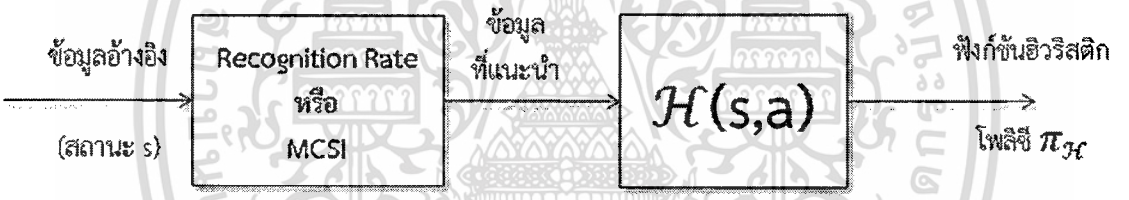
ในการกำหนดฟังก์ชันฮิวริสติกให้กับเอเจนต์ สามารถสร้างขึ้นได้จากหลากหลายวิธีการ ยกตัวอย่างเช่น

- **การใช้แนวคิดภายนอก**มาสกัดเป็นฟังก์ชัน หมายถึง การแก้ปัญหาแบบเฉพาะกิจ (Ad-Hoc) โดยเลือกใช้แนวคิดที่เคยมีผู้หาคำตอบได้ด้วยกลไกอื่นๆ หรือมีวิธีการคำนวณเพื่อแก้ปัญหาอย่างเป็นขั้นตอน ผลของการแนะนำจะมีประสิทธิภาพหรือไม่ ขึ้นอยู่กับตรรกะภายนอกที่นำมาพิจารณา
- **การใช้แนวคิดภายใน**มาสร้างเป็นฟังก์ชันแนะนำการเรียนรู้ ณ เวลาเรียนรู้จริง หมายถึง นำผลของรางวัล, ฟังก์ชันมูลค่า, โพลิซี, ปริภูมิคำตอบที่ได้เรียนรู้ผ่านมา ทำการสร้างฟังก์ชันฮิวริสติก ผลของการแนะนำจากฟังก์ชันดังกล่าวนี้จะมีแนวโน้มใกล้เคียงหรือคล้อยตามผลของการค้นหาคำตอบที่เคยเอเจนต์เคยได้เรียนรู้มา

ดังนั้นฟังก์ชันฮิวริสติกที่ประยุกต์ให้สามารถแนะนำระบบของการเรียนรู้แบบเสริมกำลัง จึงไม่สามารถกำหนดวิธีการสร้างฟังก์ชันฮิวริสติกของระบบได้อย่างชัดเจน ทั้งนี้ขึ้นกับวัตถุประสงค์ เพราะฟังก์ชันฮิวริสติกทั้งสองวิธีข้างต้นมีระบบการแนะนำที่แตกต่าง เช่น คุณสมบัติในการแนะนำให้เลือกเส้นทางการค้นหาที่เอเจนต์ไม่เคยได้เรียนรู้ (จำเป็นจะต้องใช้แนวคิดภายนอก) แทนที่จะกำหนดให้เอเจนต์ลองผิดลองถูกกับตัวเลือกด้วยตนเอง ส่วนกรณีที่ต้องการให้เอเจนต์ยังคงเลือกเส้นทางการค้นหาที่เคยเรียนรู้ว่ามีผลของรางวัลที่ดีมาก่อน (จำเป็นจะต้องใช้แนวคิดภายใน) เพื่อรักษาเส้นทางการค้นหาที่ดีไว้ เป็นต้น

ในวิทยานิพนธ์เล่มนี้จะสร้างฟังก์ชันฮิวริสติกด้วยแนวคิดทั้งสองวิธีข้างต้น ด้วยการประยุกต์กระบวนการเรียนรู้แบบเสริมกำลังเพื่อแก้ปัญหาการเลือกโปรโตไทป์ และใช้ฟังก์ชันฮิวริสติกที่สร้างจากเทคนิคผลรางวัลของอัตราการค้นคืน(Recognition Rate) และอัลกอริธึมการแก้ปัญหา MCSI ที่ใช้คุณสมบัติความครอบคลุม[5] เนื่องจากในงานวิจัยการแก้ปัญหาการเลือกโปรโตไทป์ยังคงมีผู้วิจัยและพัฒนาอย่างต่อเนื่องจนถึงในปัจจุบันและสามารถอ้างอิงได้

การสร้างฟังก์ชันฮิวริสติกให้กับเอเจนต์สามารถแบ่งออกเป็น 2 ขั้นตอน ขั้นแรกเป็นกระบวนการสร้างคำแนะนำ(สกัดปัญหา) จาก สถานะปัจจุบันในที่นี้คือการคำนวณหากลุ่มของข้อมูลที่แนะนำ ขั้นต่อมาเป็นการนำกลุ่มข้อมูลดังกล่าวมาประยุกต์ให้อยู่ในรูปแบบโพลิซีที่แนะนำซึ่งเรียกว่าโพลิซีแบบมีฮิวริสติก โดยกระบวนการดังกล่าวเป็นไปแผนผังการดำเนินงานรูปที่ 2.10



รูปที่ 2.10 แผนผังการดำเนินงานของในการสร้างฟังก์ชันฮิวริสติก

2.3 ปัญหาการระบุเซตย่อยสอดคล้องเล็กที่สุด (Minimal Consistent Set Identification, MCSI) [5]

ลักษณะของอัลกอริธึม MCSI เป็นกระบวนการเลือกโปรโตไทป์ที่แก้ปัญหาจากการกำหนดเซตของโปรโตไทป์ที่มีข้อมูลอ้างอิงทั้งหมดเป็นสมาชิก และทำการลดทอนกลุ่มข้อมูลโดยใช้คุณสมบัติ “ความครอบคลุม”[9] โดยสนใจกลุ่มข้อมูลที่สามารถครอบคลุมข้อมูลอื่นได้สูงสุดเป็นเกณฑ์ จากนั้นนำรัศมีความครอบคลุมที่ได้มากำหนด “ข้อมูลต่างประเภทใกล้เคียงที่สุด” (Nearest Unlike Neighbor, NUN) เพื่อลดทอนข้อมูลที่อยู่ในเซตโปรโตไทป์ด้วยแนวคิดความครอบคลุมสูงสุดไปจนกระทั่งไม่สามารถลดทอนข้อมูลสมาชิกต่อไปได้ เซตของโปรโตไทป์ล่าสุดที่เป็นคำตอบ จะถือเป็นเซตย่อยสอดคล้องเล็กที่สุดที่มีคุณสมบัติความสอดคล้อง(Consistency Property)[10] และความสามารถในการจำแนกข้อมูลไม่ทราบประเภท แทนการใช้ข้อมูลอ้างอิงทั้งชุดได้อย่างถูกต้อง ซึ่งสามารถแสดงเป็นอัลกอริธึมการแก้ปัญหา MCSI ดังรูปที่ 2.11

```
โปรโตไทป์ <-- ข้อมูลอ้างอิง(ทั้งเซต)
สร้างตารางเส้นทางระหว่างชุดข้อมูลอ้างอิงแต่ละตัว
กระบวนการวนซ้ำ
        กำหนดข้อมูลที่มีลักษณะ NUN โดยที่ NUN แต่ละตัวจะต้องถูกเลือกเป็นโปรโตไทป์
        สร้างเซตครอบคลุมให้กับข้อมูลที่จะถูกเลือกเป็นโปรโตไทป์แต่ละตัว
        กระบวนการวนซ้ำ
                เลือกโปรโตไทป์ด้วยหลักเกณฑ์ของคุณสมบัติความครอบคลุมสูงที่สุด
                ทำการลดทอนข้อมูลที่ถูกคลอบคลุมออกจากเซตครอบคลุม
        จนกระทั่ง (ผลรวมในเซตครอบคลุมจะเป็นศูนย์) หรือ
                (เซตของโปรโตไทป์ในรอบก่อนหน้า = เซตของโปรโตไทป์ในรอบปัจจุบัน)
จนกระทั่ง (เซตของโปรโตไทป์ในรอบก่อนหน้า = เซตของโปรโตไทป์ในรอบปัจจุบัน)
```

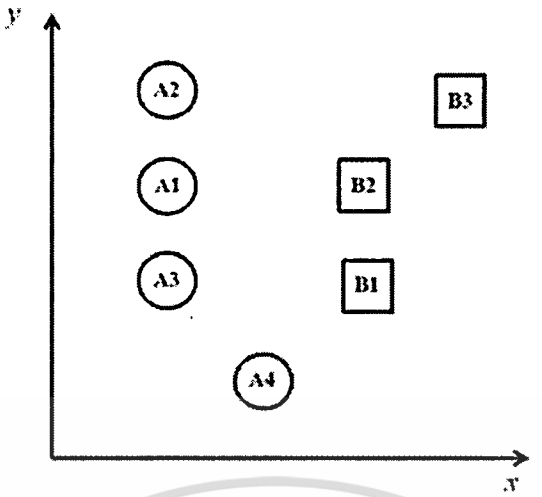
รูปที่ 2.11 อัลกอริธึมการแก้ปัญหา MCSI

2.3.1 กระบวนการเลือกโปรโตไทป์ (Prototype Selection Process) [11]

การเลือกโปรโตไทป์ หมายถึง การเลือกกลุ่มข้อมูลมาเป็นเซตของตัวแทนข้อมูลอ้างอิงโดยที่กลุ่มข้อมูลตัวแทนจะต้องทำการจำแนกข้อมูลอ้างอิงได้อย่างถูกต้องครบถ้วนด้วยกฎการจำแนกจากระยะห่างของข้อมูลอ้างอิงน้อยที่สุด (Nearest Neighbor Rule, NN-Rule)[12] จุดมุ่งหมายของการเลือกโปรโตไทป์ เพื่อลดจำนวนของสมาชิกในเซตข้อมูลอ้างอิงให้เป็นเซตย่อยเล็กสุดที่เป็นไปได้(เซตข้อมูลอ้างอิงใหม่) ที่ยังคงมีความสามารถในการจำแนกข้อมูลอ้างอิงทั้งหมดได้ ซึ่งจะช่วยลดขั้นตอนการคำนวณ และระยะเวลาที่ใช้ในการจำแนกข้อมูล ของข้อมูลไม่ทราบประเภทด้วยNN-Rule แทนการใช้ข้อมูลอ้างอิงทั้งชุด

2.3.2 ชุดข้อมูลตัวอย่างการแก้ปัญหา MCSI

กำหนดให้การแก้ปัญหา MCSI ใช้ชุดข้อมูลตัวอย่างดังแสดงในรูปที่ 2.12 ที่มีจำนวนสมาชิกตัวอย่างทั้งสิ้น 7 ตัว โดยมีค่าคุณลักษณะใน 2 มิติ (พิกัดแกน X, พิกัดแกน Y) และแบ่งข้อมูลออกเป็น 2 ประเภท (A, B) แยกเป็นประเภท A 4 ตัว {A1, A2, A3, A4} และประเภท B 3 ตัว {B1, B2, B3} ซึ่งคุณลักษณะต่างๆ สามารถแสดงเป็นชุดข้อมูลได้ดังในตารางที่ 2.1



รูปที่ 2.12 กราฟแสดงคุณลักษณะ 2 มิติของชุดข้อมูลตัวอย่างที่ 1

ตารางที่ 2.1 คุณลักษณะของข้อมูลในชุดข้อมูลตัวอย่างที่ 1

ข้อมูล	ประเภท	พิกัด (x, y)
A1	A	(1, 3)
A2	A	(1, 4)
A3	A	(1, 2)
A4	A	(2, 1)
B1	B	(3, 2)
B2	B	(3, 3)
B3	B	(4, 4)

จากข้อมูลในตารางที่ 2.1 สามารถสร้างเมตริกระยะทาง (distance matrix) ระหว่างข้อมูลแต่ละคู่ของชุดข้อมูลตัวอย่าง ด้วยการคำนวณระยะทางแบบยูคลิด (Euclidean distance)[13] มีค่าดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 เมตริกระยะทางของข้อมูลในชุดข้อมูลตัวอย่างที่ 1

	A1	A2	A3	A4	B1	B2	B3
A1	0.00	1.00	1.00	2.24	2.24	2.00	3.16
A2	1.00	0.00	2.00	3.16	2.83	2.24	3.00
A3	1.00	2.00	0.00	1.41	2.00	2.24	3.61
A4	2.24	3.16	1.41	0.00	1.41	2.24	3.61
B1	2.24	2.83	2.00	1.41	0.00	1.00	2.24
B2	2.00	2.24	2.24	2.24	1.00	0.00	1.41
B3	3.l6	3.00	3.61	3.61	2.24	1.41	0.00

จากเมตริกระยะทางแบบยูคลิด (ตารางที่ 2.2) ทำการจัดเรียงระยะทางให้เป็นเมตริกระยะทางมาตรฐาน (Normalization Distance Matrix) โดยพิจารณานัยสำคัญจากระยะทาง และประเภทของข้อมูล (ในกรณีระยะทางเท่ากัน ข้อมูลที่ประเภทต่างกันมีนัยสำคัญกว่า) ได้เป็นเมตริกระยะทางมาตรฐานดังตารางที่ 2.3

ตารางที่ 2.3 เมตริกระยะทางมาตรฐานของข้อมูลในชุดข้อมูลตัวอย่างที่ 1

	A1	A2	A3	A4	B1	B2	B3
A1	1	2	3	6	5	4	7
A2	2	1	3	7	5	4	6
A3	2	5	1	3	4	6	7
A4	5	6	3	1	2	4	7
B1	5	7	4	3	1	2	6
B2	4	5	6	7	2	1	3
B3	5	4	6	7	3	2	1

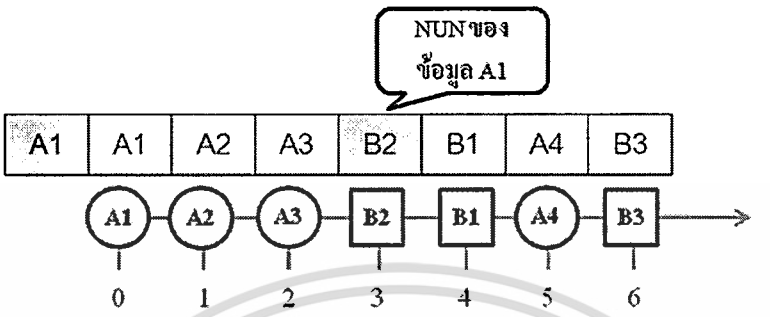
จากนั้นทำเครื่องหมาย (Label) กับข้อมูลจากตารางที่ 2.3 เพื่อสร้างเป็นเมตริกที่ทำการจัดเรียงระยะทางพร้อมทั้งทำเครื่องหมายแล้ว (Sorted and Labeled Matrix) ดังตารางที่ 2.4

ตารางที่ 2.4 เมตริกที่ทำการจัดเรียงระยะทางพร้อมทั้งทำเครื่องหมายแล้วในชุดข้อมูลตัวอย่างที่ 1

A1	A1	A2	A3	B2	B1	A4	B3
A2	A2	A1	A3	B2	B1	B3	A4
A3	A3	A1	A4	B1	A2	B2	B3
A4	A4	B1	A3	B2	A1	A2	B3
B1	B1	B2	A4	A3	A1	B3	A2
B2	B2	B1	B3	A1	A2	A3	A4
B3	B3	B2	B1	A2	A1	A3	A4

2.3.3 ข้อมูลต่างประเภทใกล้เคียงที่สุด (Nearest Unlike Neighbor, NUN)[9]

ข้อมูลต่างประเภทใกล้เคียงที่สุด หมายถึง ข้อมูลมีระยะห่างทางคุณลักษณะน้อยที่สุด และเป็นข้อมูลคนละประเภทเมื่อเทียบกับประเภทของข้อมูลที่กำลังพิจารณา ยกตัวอย่างจากตารางที่ 2.4 ด้วยข้อมูลที่กำลังพิจารณา A1

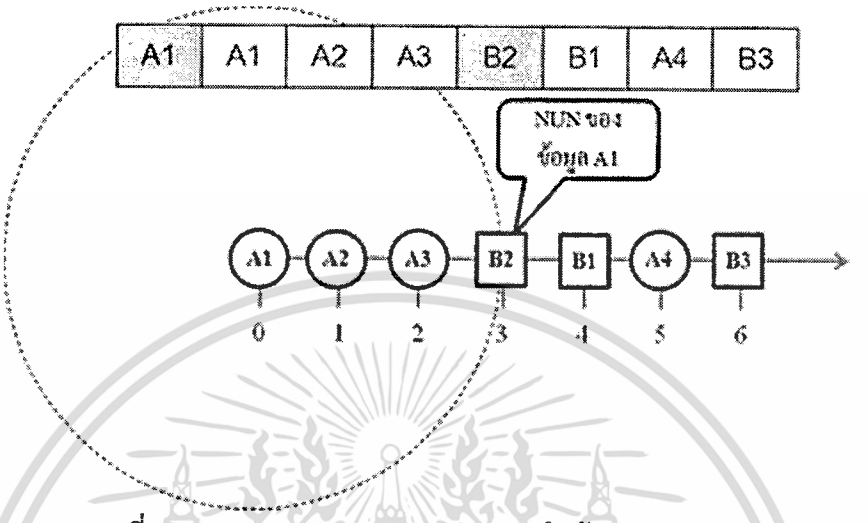


รูปที่ 2.13 ชุดข้อมูลที่กำลังพิจารณา A1

จากรูปที่ 2.13 เมื่อพิจารณาข้อมูล A1 จะพบว่ามีข้อมูลประเภท B อยู่ 3 ตัวคือข้อมูล B2, B1 และ B3 ซึ่งมีระยะห่างทางคุณลักษณะจาก A1 เป็นระยะ 3, 4 และ 6 หน่วยตามลำดับ ดังนั้น ข้อมูล B2 จึงถือว่าเป็นข้อมูลคนละประเภทที่อยู่ใกล้ข้อมูลที่กำลังพิจารณา A1 มากที่สุด หรือกล่าวได้ว่า ข้อมูล B2 เป็น NUN ของข้อมูล A1

2.3.4 เซตของข้อมูลที่ครอบคลุม (Cover List)

การกำหนดข้อมูล NUN จากหัวข้อ2.3.3 จะพบว่าข้อมูลที่กำลังพิจารณาและข้อมูลข้างเคียงที่อยู่ภายในรัศมีของระยะทางจากข้อมูล NUN เป็นข้อมูลประเภทเดียวกันทั้งสิ้น ดังรูปที่ 2.14



รูปที่ 2.14 ขอบเขตของความครอบคลุมในข้อมูล A1

ข้อมูล A1 จึงมีความครอบคลุม(Cover) กับข้อมูล A1(ตนเอง), A2 และ A3 เพราะมีข้อมูล B2 เป็นขอบเขตของความครอบคลุมเนื่องจากเป็นข้อมูลประเภท B ซึ่งต่างจากข้อมูลประเภท A ดังนั้นเซตของข้อมูลที่ถูกครอบคลุมจากข้อมูล A1 จึงมีสมาชิกเป็น {A1,A2, A3}

จากเซตของความครอบคลุมนี้หากทำการจำแนกข้อมูลด้วย NN-Rule จะพบว่าหากกำหนดให้ A1 เป็นตัวแทนในการจำแนกข้อมูลจะสามารถจำแนกทั้งข้อมูล A1, A2 และ A3 เป็นข้อมูลประเภท A ได้ถูกต้อง จากนั้นการคำนวณข้อมูล NUN กับข้อมูลสมาชิกทุกตัวเพื่อหาผลของการลงคะแนน (Vote) ครอบคลุมเพื่อมาสร้างเป็นเซตของข้อมูลที่ครอบคลุม(Cover List) ดังตารางที่ 2.5 และพิจารณาเฉพาะข้อมูลที่อยู่ในรัศมี NUN เท่านั้น

ตารางที่ 2.5 เซตของข้อมูลที่ถูกครอบคลุม (Cover List)

ข้อมูลที่สนใจ	ข้อมูลที่ครอบคลุม			NUN	ข้อมูลไม่ครอบคลุม			Cove List
A1	A1	A2	A3	B2	B1	A4	B3	A1 = {A1, A2, A3}
A2	A2	A1	A3	B2	B1	B3	A4	A2 = {A1, A2, A3}
A3	A3	A1	A4	B1	A2	B2	B3	A3 = {A1, A3, A4}
A4	A4	B1	A3	B2	A1	A2	B3	A4 = {A4}
B1	B1	B2	A4	A3	A1	B3	A2	B1 = {B1, B2}
B2	B2	B1	B3	A1	A2	A3	A4	B2 = {B1, B2, B3}
B3	B3	B2	B1	A2	A1	A3	A4	B3 = {B1, B2, B3}

2.3.5 ค่าความสามารถในการครอบคลุม (Cover Values, CV)

ค่าความสามารถในการครอบคลุม คือ จำนวนของข้อมูลที่สามารถครอบคลุมได้จากข้อมูลที่กำลังพิจารณา (จำนวนกลุ่มข้อมูลที่ใช้ข้อมูลที่กำลังพิจารณาเป็นข้อมูลตัวแทนได้) โดยนำตารางที่ 2.6 มาคำนวณหาผลรวมหากเลือกข้อมูลที่กำลังพิจาณาเป็นโปรโตไทป์แล้ว จะสามารถลดทอนจำนวนข้อมูลสมาชิกที่อยู่ภายในรัศมีความครอบคลุมลงได้

ตารางที่ 2.6 ตารางความครอบคลุม (Cover Table)

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : {Ø}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	1	1	1	0	0	0	0	A1 = {A1, A2, A3}
A2	1	1	1	0	0	0	0	A2 = {A1, A2, A3}
A3	1	0	1	1	0	0	0	A3 = {A1, A3, A4}
A4	0	0	0	1	0	0	0	A4 = {A4}
B1	0	0	0	0	1	1	0	B1 = {B1, B2}
B2	0	0	0	0	1	1	1	B2 = {B1, B2, B3}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	3	2	3	2	3	3	2	

2.3.6 การเลือกโปรโตไทป์ด้วยคุณสมบัติความครอบคลุมสูงที่สุด

จากอัลกอริธึมการแก้ปัญหา MCSI รูปที่ 2.11 ในรอบการคำนวณที่ 1 เมื่อพิจารณาค่าความสามารถในการครอบคลุม ในตารางที่ 2.6 จะพบว่าข้อมูล A1, A3, B1 และ B2 มีค่าความสามารถในการครอบคลุมข้อมูลอื่น 3 หน่วย นั่นหมายถึงหากเลือกข้อมูลตัวใดตัวหนึ่งก็จะสามารถใช้ข้อมูลที่เลือก เป็นข้อมูลอ้างอิงในการจำแนกข้อมูลแทนข้อมูลสมาชิกที่อยู่ภายในรัศมีความครอบคลุมของตัวเองด้วย NN-Rule ได้อย่างถูกต้อง แต่เนื่องจากอัลกอริธึมการแก้ปัญหา MCSI ไม่ได้ระบุถึงเกณฑ์การพิจาณาเลือกข้อมูลที่มีค่าความสามารถในการครอบคลุมสูงที่สุดเท่ากันไว้ ดังนั้นความเป็นไปได้ในการเลือกข้อมูลโปรโตไทป์จะสามารถเลือกใช้วิธีการสุ่ม และการเลือกตามลำดับก่อน-หลังก็ได้ จึงสมมุติให้ทำการเลือกข้อมูล A1 เป็นโปรโตไทป์เพื่อทำการลดทอนข้อมูล A2, A3 ออกจากเซตของโปรโตไทป์และทำการปรับปรุงค่าความสามารถความครอบคลุมดังตารางที่ 2.7

ตารางที่ 2.7 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลักจากเลือกข้อมูล A1 เป็นสมาชิกของเซตโปรโตไทป์

Data Set : { -, -, -, A4, B1, B2, B3}

Prototype Set : {A1}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	x	x	x	0	0	0	0	A1 = {Ø}
A2	x	x	x	0	0	0	0	A2 = {Ø}
A3	x	0	x	x	0	0	0	A3 = {Ø}
A4	0	0	0	1	0	0	0	A4 = {A4}
B1	0	0	0	0	1	1	0	B1 = {B1, B2}
B2	0	0	0	0	1	1	1	B2 = {B1, B2, B3}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	0	0	0	1	3	3	2	

พิจารณาตารางที่ 2.7 หลังจากเลือกข้อมูล A1 เป็นสมาชิกของเซตโปรโตไทป์จะสามารถลดข้อมูล A2 และ A3 ออกได้ ทำให้ค่าความครอบคลุมของข้อมูล A2, A3 ถูกลบออกไป ดังนั้นเมื่อพิจารณาค่าความสามารถในการครอบคลุมสูงสุดอีกครั้ง จะเหลือข้อมูลที่มีโอกาสถูกเลือกคือข้อมูล B1 และ B2 เป็นสมาชิกของเซตโปรโตไทป์ในรอบการคำนวณถัดมา จากนั้นสมมุติให้ทำการเลือกข้อมูล B2 เป็นสมาชิกของเซตโปรโตไทป์ จากนั้นทำการปรับปรุงค่าความสามารถความครอบคลุมอีกครั้ง ดังตารางที่ 2.8

ตารางที่ 2.8 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลักจากเลือกข้อมูล B2 เป็นสมาชิกของเซตโปรโตไทป์

Data Set : { -, -, -, A4, -, -, -}

Prototype Set : {A1, B2}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	x	x	x	0	0	0	0	A1 = {Ø}
A2	x	x	x	0	0	0	0	A2 = {Ø}
A3	x	0	x	x	0	0	0	A3 = {Ø}
A4	0	0	0	1	0	0	0	A4 = {A4}
B1	0	0	0	0	x	x	0	B1 = {Ø}
B2	0	0	0	0	x	x	x	B2 = {Ø}
B3	0	0	0	0	x	x	x	B3 = {Ø}
CV	0	0	0	1	0	0	0	

เมื่อพิจารณาตารางที่ 2.8 จะเหลือเพียงข้อมูล A4 เพียงข้อมูลเดียวที่สามารถเลือกเป็นสมาชิกของเซตโปรโตไปท์ ซึ่งมีค่าความสามารถในการครอบคลุมเพียงตนเองเท่ากัน จึงทำการเลือกข้อมูล A4 เป็นสมาชิกของเซตโปรโตไทป์และทำการปรับปรุงค่าความสามารถในการครอบคลุมใหม่ได้เป็นตารางที่ 2.9

ตารางที่ 2.9 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลักจากเลือกข้อมูล A4 เป็นสมาชิกของเซตโปรโตไทป์

Data Set : { -, -, -, -, -, -, -}

Prototype Set : {A1, A4, B2}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	x	x	x	0	0	0	0	A1 = {Ø}
A2	x	x	x	0	0	0	0	A2 = {Ø}
A3	x	0	x	x	0	0	0	A3 = {Ø}
A4	0	0	0	x	0	0	0	A4 = {Ø}
B1	0	0	0	0	x	x	0	B1 = {Ø}
B2	0	0	0	0	x	x	x	B2 = {Ø}
B3	0	0	0	0	x	x	x	B3 = {Ø}
CV	0	0	0	0	0	0	0	

ดังนั้นในการคำนวณหาเซตโปรโตไทป์ในรอบการคำนวณที่ 1 จะมีสมาชิกเซตย่อยสอดคล้องเล็กที่สุดจำนวน 3 ตัวประกอบด้วย {A1, A4, B2} ที่มีคุณสมบัติความสอดคล้องและสามารถจำแนกข้อมูลอ้างอิงตัวอย่างจำนวน 7 ตัวได้อย่างถูกต้อง

ในรอบการคำนวณที่ 2 จะใช้สมาชิกของเซตคำตอบในรอบการคำนวณก่อนหน้า สร้างเป็นตัวแทนในการกำหนดตำแหน่งของข้อมูล NUN ส่งผลให้ระยะของรัศมีความครอบคลุมจะเปลี่ยนไปจากตารางที่ 2.5 ดังนั้น จำเป็นต้องสร้างเซตของข้อมูลที่ครอบคลุมขึ้นใหม่ ด้วยเงื่อนไขของรัศมีความครอบคลุมที่กำหนดโดย {A1, A4, B2} ดังตารางที่ 2.10 และตารางความครอบคลุมดังตารางที่ 2.11

ตารางที่ 2.10 เซตของข้อมูลที่ถูกครอบคลุม(Cover List) ในรอบการคำนวณที่ 2

ข้อมูลที่สนใจ	ข้อมูลที่ครอบคลุม			NUN	Out Bound		
A1	A1	A2	A3	B2	B1	A4	B3
A2	A2	A1	A3	B2	B1	B3	A4
A3	A3	A1	A4	~~B1~~	A2	B2	B3
A4	A4	~~B1~~	A3	B2	A1	A2	B3
B1	B1	B2	A4	A3	A1	B3	A2
B2	B2	B1	B3	A1	A2	A3	A4
B3	B3	B2	B1	~~A2~~	A1	A3	A4

Cove List

A1 = {A1, A2, A3}
A2 = {A1, A2, A3}
A3 = {A1, A2, A3, A4}
A4 = {A3, A4}
B1 = {B1, B2}
B2 = {B1, B2, B3}
B3 = {B1, B2, B3}

ตารางที่ 2.11 ตารางความครอบคลุม(Cover Table) ในรอบการคำนวณที่ 2

Data Set : { A1, -, -, A4, -, B2, -}

Prototype Set : {Ø}

	A1	A2	A3	A4	B1	B2	B3
A1	1	1	1	0	0	0	0
A2	1	1	1	0	0	0	0
A3	1	1	1	1	0	0	0
A4	0	0	1	1	0	0	0
B1	0	0	0	0	1	1	0
B2	0	0	0	0	1	1	1
B3	0	0	0	0	1	1	1
CV	3	~~3~~	~~4~~	2	~~3~~	3	~~2~~

Cove List

A1 = {A1, A2, A3}
A2 = {A1, A2, A3}
A3 = {A1, A2, A3, A4}
A4 = {A3, A4}
B1 = {B1, B2}
B2 = {B1, B2, B3}
B3 = {B1, B2, B3}

จากตารางที่ 2.11 เป็นการคำนวณในรอบการคำนวณที่ 2 แต่ด้วยเงื่อนไขของข้อมูล NUN ทุกตัวจะต้องถูกเลือกเป็นโปรโตไทป์ ทำให้คำตอบในรอบปัจจุบันต้องเป็นสมาชิกในเซตของโปรโตไทป์ที่ถูกคำนวณก่อนหน้านี้ จึงจะสามารถถูกเลือกได้ ดังนั้นจากเซตคำตอบของรอบการคำนวณที่ 1 คือ {A1, A4, B2} จึงพิจารณาค่า CV เพียง 3 ตัว สมมุติให้ข้อมูล A1 ถูกเลือกเป็นสมาชิกของเซตโปรโตไทป์อีกครั้ง และทำการปรับปรุงค่าความสามารถในการครอบคลุม ดังตารางที่ 2.12

ตารางที่ 2.12 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลักจากเลือกข้อมูล A1 เป็นสมาชิกของเซตโปรโตไทป์ในรอบการคำนวณที่ 2

Data Set : { -, -, -, A4, -, B2, -}

Prototype Set : {A1}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	x	x	x	0	0	0	0	A1 = {Ø}
A2	x	x	x	0	0	0	0	A2 = {Ø}
A3	x	x	x	x	0	0	0	A3 = {Ø}
A4	0	0	x	1	0	0	0	A4 = { A4}
B1	0	0	0	0	1	1	0	B1 = {B1, B2}
B2	0	0	0	0	1	1	1	B2 = {B1, B2, B3}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	0	~~0~~	~~0~~	1	~~3~~	3	~~2~~	

เมื่อพิจารณาข้อมูลที่ยังคงสามารถเลือกได้ จะพบว่ามีเพียงข้อมูล B2 ที่มีค่า CV สูงสุด จึงทำการเลือกข้อมูล B2 เป็นสมาชิกของเซตโปรโตไทป์ พร้อมปรับปรุงค่าความสามารถในการครอบคลุมอีกครั้ง ดังตารางที่ 2.13

ตารางที่ 2.13 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลักจากเลือกข้อมูล B2 เป็นสมาชิกของเซตโปรโตไทป์ในรอบการคำนวณที่ 2

Data Set : { -, -, -, A4, -, -, -}

Prototype Set : {A1, B2}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	x	x	x	0	0	0	0	A1 = {Ø}
A2	x	x	x	0	0	0	0	A2 = {Ø}
A3	x	x	x	x	0	0	0	A3 = {Ø}
A4	0	0	x	1	0	0	0	A4 = { A4}
B1	0	0	0	0	x	x	0	B1 = {Ø}
B2	0	0	0	0	x	x	x	B2 = {Ø}
B3	0	0	0	0	x	x	x	B3 = {Ø}
CV	0	~~0~~	~~0~~	1	~~0~~	0	~~0~~	

ในลำดับถัดมา ทำการเลือกข้อมูล A4 เป็นสมาชิกของเซตโปรโตไทป์และปรับปรุงค่าความสามารถในการครอบคลุมอีกครั้ง ครั้งนี้เป็นการคำนวณในขั้นตอนสุดท้ายในรอบการคำนวณที่ 2 เนื่องจากผลรวมของค่า CV ของข้อมูลทุกตัวมีค่าเป็น 0 ทั้งหมด ดังตารางที่ 2.14

ตารางที่ 2.14 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลักจากเลือกข้อมูล A4 เป็นสมาชิกของเซตโปรโตไทป์ในรอบการคำนวณที่ 2

Data Set : { -, -, -, -, -, -, -}

Prototype Set : {A1, A4, B2}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	x	x	x	0	0	0	0	A1 = {Ø}
A2	x	x	x	0	0	0	0	A2 = {Ø}
A3	x	x	x	x	0	0	0	A3 = {Ø}
A4	0	0	x	x	0	0	0	A4 = {Ø}
B1	0	0	0	0	x	x	0	B1 = {Ø}
B2	0	0	0	0	x	x	x	B2 = {Ø}
B3	0	0	0	0	x	x	x	B3 = {Ø}
CV	0	~~0~~	~~0~~	0	~~0~~	0	~~0~~	

ในการคำนวณรอบที่ 2 เซตย่อยสอดคล้องเล็กสุดที่ได้ยังคงเป็นเซต {A1, A4, B2} ที่มีจำนวนสมาชิกเท่ากับเซตคำตอบในรอบการคำนวณที่ 1 และสมาชิกในเซตยังคงเป็นเซตคำตอบเดียวกัน การคำนวณหาเซตโปรโตไทป์ของปัญหา MCSI ในกลุ่มข้อมูลตัวอย่างจึงสิ้นสุดลงด้วยเงื่อนไขของสมาชิกในเซตคำตอบที่ไม่เปลี่ยนแปลง มีจำนวนสมาชิก 3 ตัวประกอบไปด้วย {A1, A4, B2}

บทที่ 3
งานวิจัยที่เกี่ยวข้อง

เนื้อหาในบทนี้จะกล่าวถึง แนวทางในการปรับปรุงประสิทธิภาพของกระบวนการเรียนรู้แบบเสริมกำลัง ที่มีการนำเสนอในบทความวิจัยต่างๆ ซึ่งได้พัฒนามาก่อนหน้านี้ โดยเริ่มจาก ลักษณะของการเรียนรู้ด้วยโพลิซีกรีดี ε, การเพิ่มประสิทธิภาพให้กับอัลกอริธึม RL ด้วยการกลับมายังสถานะที่สามารถเรียนรู้ซ้ำได้ (Reinforcement Learning with returning to the last known consistent state, RL-RCS), การเพิ่มประสิทธิภาพให้กับอัลกอริธึมการเรียนรู้ Q ด้วยการแนะนำจากฟังก์ชันฮิวริสติก (Heuristically accelerated Q-learning, HAQL) และการประยุกต์ใช้การเรียนรู้แบบเสริมกำลังในการแก้ปัญหาการเลือกโปรโตไทป์ ตามลำดับ

3.1 ลักษณะการเรียนรู้ด้วยโพลิซีกรีดี ε [1]

ในการเรียนรู้กับปัญหาขนาดใหญ่ ที่มีสถานะในการเรียนรู้จำนวนมาก การใช้โพลิซีที่มีลักษณะแบบกรีดี ε (ε-greedy) เพียงอย่างเดียว ไม่สามารถควบคุมการเรียนรู้ได้ดีเท่าที่ควร เนื่องจากอัตราการสำรวจ ε เป็นการสุ่มความน่าจะเป็นด้วยอัตราร้อยละ ซึ่งมีผลในด้านความลึกของเส้นทางการค้นหาปริภูมิคำตอบ

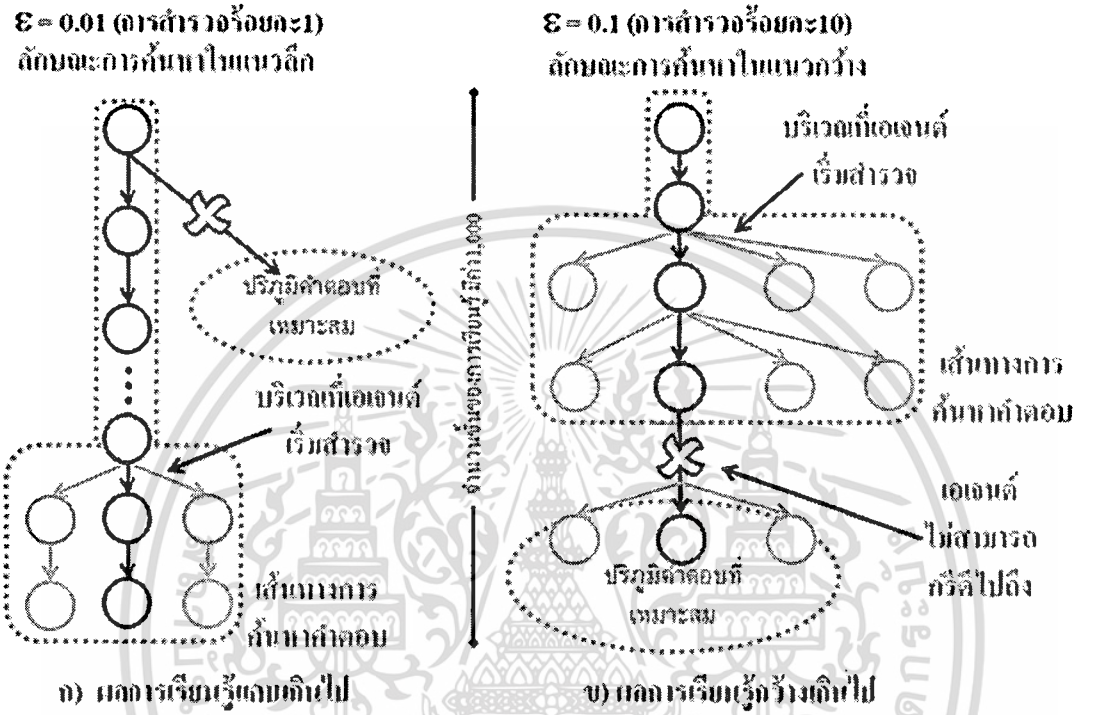
- หากค่า ε มีค่าน้อยเกินไป (อัตราการสำรวจต่ำ)

  เส้นทางการค้นหาของเอเจนต์จะมีลักษณะการค้นหาในแนวลึก (depth-first search) หากคำตอบของปัญหาอยู่ในปริภูมิระดับไม่ลึกมาก มีโอกาสค่อนข้างน้อยที่เอเจนต์จะค้นพบคำตอบจากกรณีเลือกสำรวจ เนื่องจากอัตราส่วน ε จะควบคุมให้เอเจนต์เลือกการกระทำแบบกรีดีเป็นหลัก และในกรณีของการแก้ปัญหาบนสถานะขนาดใหญ่ที่ความยาวของเส้นทางการค้นหาอยู่ลึกพอสมควร อีกทั้งมีโอกาสที่เอเจนต์เองจะเลือกสำรวจก่อนเวลาอันควร ยกตัวอย่างกรณี ความลึกในการค้นหา 1000 ขั้น (step) มีอัตราส่วนการสำรวจกำหนดที่ 0.01 (ร้อยละ 1) โอกาสที่เอเจนต์จะสำรวจเส้นทางการค้นหาที่ไม่เคยเรียนรู้จะเกิดขึ้นทุกๆ 100 ขั้นของการเรียนรู้ทำให้เอเจนต์ไม่สามารถรักษาเส้นทางการค้นหาเดิมที่ดีได้ในเอพิโสดถัดไป ดังรูปที่ 3.1 ก

- หากค่าของ ε มากเกินไป (อัตราการสำรวจสูง)

  เส้นทางการค้นหาของเอเจนต์จะมีลักษณะการค้นหาในแนวกว้าง (breadth-first search) หากคำตอบของปัญหาอยู่ในปริภูมิคำตอบระดับลึก โอกาสส่วนมากของเอเจนต์จะเลือกทำการสำรวจปริภูมิคำตอบที่ไม่เคยเรียนรู้เป็นหลัก ทำให้เอเจนต์ไม่สามารถสำรวจเส้นทางการค้นหาเดิมที่ดีและมีโอกาสน้อยที่จะใช้ความรู้ที่จดจำไว้(ฟังก์ชันมูลค่า) ยกตัวอย่างกรณี ความลึกในการค้นหา 1000 ขั้น (step) มีอัตราส่วนการสำรวจกำหนด

ที่ 0.1 (ร้อยละ 10) โอกาสที่เอเจนต์จะสำรวจเส้นทางการค้นหาที่ไม่เคยเรียนรู้จะเกิดขึ้นทุกๆ 10 ขั้นของการเรียนรู้ หากคำตอบของปัญหาอยู่ในช่วงความลึกที่ประมาณ 700 ผลการเรียนรู้ในครั้งก่อน ส่งผลให้ทำให้เอเจนต์ไม่สามารถรักษาเส้นทางการค้นหาเดิมที่ดีได้ในเอพิโสดถัดไปเช่นกัน ดังรูปที่ 3.1 ข



รูปที่ 3.1 ผลการเรียนรู้ด้วยโพลิซีแบบกรีดี ε

ดังนั้นอัตราส่วนร้อยละในการสำรวจจึงไม่เหมาะสมความลึกของสถานะเพราะไม่สามารถทราบได้ว่าความลึกเท่าไรจึงจะเพียงพอให้เริ่มทำการสำรวจ และเพื่อเป็นการปรับปรุงโพลิซีการเรียนรู้จึงมีแนวคิดในการพัฒนาออกเป็น 2 แนวทาง ในแนวทางแรกเป็นการปรับปรุงกลไกการเรียนรู้ในส่วนของสภาพแวดล้อมให้สามารถนำเอเจนต์กลับมายังสถานะที่สนใจเพื่อทำการเรียนรู้ซ้ำ (retry state) [4] เพื่อให้เอเจนต์มีโอกาสเลือกเส้นทางการค้นหาอื่นๆ ได้โดยไม่ต้องทำการเรียนรู้ซ้ำในการกลับมายังสถานะที่สนใจนี้อีกครั้งในรอบการเรียนรู้(เอพิโสด)ถัดไป เป็นการปรับปรุงประสิทธิภาพทางด้านการค้นหาบริเวณใกล้เคียง (Local Search) เช่น ในบางสถานะ เอเจนต์ได้เลือกการกระทำที่ผิด โดยสังเกตได้จากค่ารางวัลที่ได้รับจากสภาพแวดล้อม แทนที่จะกำหนดเอเจนต์ทำการค้นหาต่อไป ถ้า เอเจนต์สามารถย้อนกลับมาเลือกการกระทำอื่นจนกว่าจะได้ค่าของผลรางวัลที่อยู่ในเกณฑ์ที่กำหนด จะมีผลต่อการเรียนรู้เร็วขึ้น เนื่องจากในสถานะนี้เอเจนต์อาจมีโอกาสในการกลับมาน้อยมากเนื่องจากสถานะดังกล่าวอาจอยู่ในตำแหน่งที่ไม่ตรงกับค่า ε-greedy ที่ได้กำหนดไว้

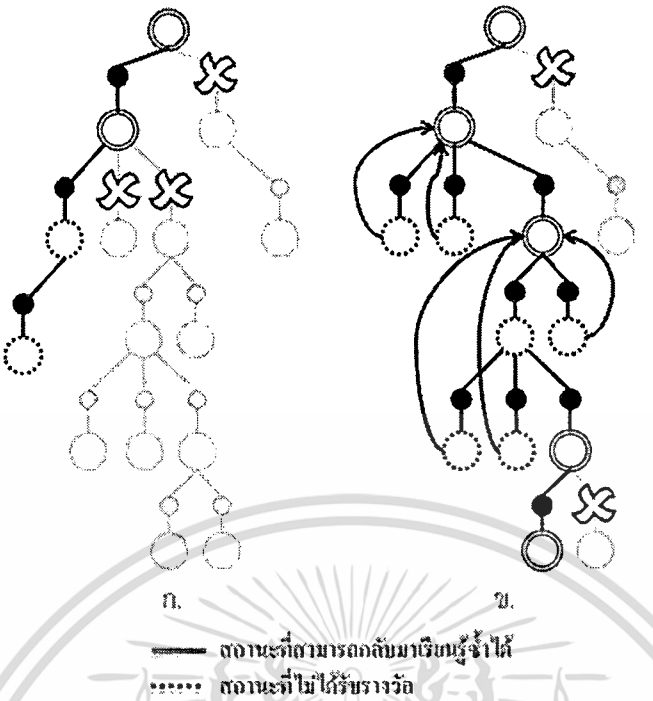
ในแนวทางที่สองเป็นการประยุกต์แนวคิดในการแก้ปัญหาสร้างเป็นฟังก์ชันฮิวริสติก [3] เพื่อช่วยเพิ่มหรือลดทางเลือกในการตัดสินใจให้เอเจนต์ด้วยตัวเลือกการกระทำอันเกิดจากการแนะนำจาก

ฟังก์ชันฮิวริสติก โดยมุ่งหวังให้เอเจนต์สามารถกรองตัวเลือกการกระทำที่ไม่ดีออก เพื่อให้ขนาดของทางเลือกที่ดีที่เหลืออยู่มีขนาดเล็กลง เพื่อเป็นการควบคุมเส้นทางการค้นหาบนปริภูมิคำตอบให้มีทิศทางที่ถูกต้องเหมาะสม ลดจำนวนครั้งในการลองผิดลองถูกของเอเจนต์ ซึ่งฟังก์ชันฮิวริสติกที่เลือกใช้จะต้องมีความสามารถในการคัดกรองเพื่อลดทอนน้ำหนักตัวเลือกการกระทำที่ไม่ดีได้มาก และอาจจำเป็นต้องมีความสามารถในด้านการแนะนำคำตอบที่ดีได้บ้าง

ทั้งสองแนวทางข้างต้นมีจุดประสงค์เพื่อเพิ่มความเร็วในการเรียนรู้ให้สามารถลู่เข้าสู่คำตอบที่ดีได้มากกว่าการเรียนรู้ด้วยกลไกของการเรียนรู้แบบเสริมกำลังทั่วไป ภายใต้เงื่อนไขทางด้านทรัพยากรหน่วยความจำและระยะเวลาที่ใช้ในการค้นหา

3.2 การเพิ่มประสิทธิภาพให้กับอัลกอริธึม RL ด้วยการกลับมายังสถานะที่สามารถเรียนรู้ซ้ำได้ (Reinforcement Learning with returning to the last known consistent state, RL-RCS) [4]

ในการแก้ปัญหาด้วยการเรียนรู้แบบเสริมกำลัง เป็นการเรียนรู้การเลือกการกระทำที่เกิดจากการได้รับรางวัล ณ สถานะหนึ่งมาเก็บสะสมในรูปแบบของฟังก์ชันมูลค่า ในการกระทำบนสถานะใดๆ จำเป็นจะต้องได้รับรางวัลจากการกระทำนั้นเสมอ เพราะหากการแก้ปัญหาบนปริภูมิคำตอบที่ก่อให้เกิดรางวัลเป็นค่าลบ หรือไม่ได้รับรางวัล (รางวัลมีค่าเป็น 0) เมื่อทำการปรับฟังก์ชันมูลค่าแล้วค่าลบหรือค่า 0 จะถูกลดความสำคัญจากความน่าจะเป็นในการเลือกการกระทำแบบ ε-greedy ลงไปและด้วยกลไกของอัลกอริธึม RLปกติ จะมีโอกาสน้อยมากที่เอเจนต์จะสามารถเลือกการกระทำเดิมเพื่อกลับมาแก้ไขการเรียนรู้ให้ดีขึ้น ณ สถานะนี้ในเอพิโสดถัดไปได้ ซึ่งเป็นสาเหตุให้เอเจนต์ไม่สามารถค้นหาคำตอบที่ดีได้



รูปที่ 3.2 เส้นทางการค้นหาโดยใช้ ก).อัลกอริธึม RL ทั่วไป ข). อัลกอริธึม RL-RCS

อัลกอริธึม RL-RCS เป็นอัลกอริธึมที่สามารถทำให้เอเจนต์กลับไปเรียนรู้ซ้ำในสถานะที่เคยได้เลือกการกระทำแล้วแต่ตัวเลือกดังกล่าวเป็นตัวเลือกที่ผิด เอเจนต์จึงมีโอกาสได้กลับไปแก้ไขตัวเลือกการกระทำใหม่ภายในเอพิโสดเดียวกันนี้ได้ (รูปที่ 3.2 ข.) โดยไม่ต้องเสียเวลาเพื่อเรียนรู้ในเอพิโสดถัดไปเพียงเพื่อให้เอเจนต์กลับมายังสถานะที่เคยได้เลือกการกระทำที่ผิด เช่นตัวอย่างในรูปที่ 3.2 ก. ดังนั้น เอเจนต์จึงมีโอกาสกลับไปยังสถานะที่เรียนรู้ซ้ำได้ (ในสถานะที่มีรางวัลเป็นบวกครั้งล่าสุดที่พบ) อย่างไม่จำกัดซึ่งหมายถึงเอเจนต์สามารถปรับปรุงค้นหาให้มีลักษณะวนซ้ำได้

ด้วยเหตุนี้ เอเจนต์จึงสามารถย้อนกลับมาเลือกการกระทำ ณ สถานะที่กำลังสนใจเพื่อเป็นการแก้ไขการค้นหาคราวก่อนได้หลายครั้งเพื่อให้พบคำตอบที่มีฟังก์ชันมูลค่าที่ดี ซึ่งหากเป็นอัลกอริธึม RLปกติ เอเจนต์จะทำการเรียนรู้ต่อไปจนจบเอพิโสด ถึงแม้จะทราบว่าเคยได้รับค่ารางวัลที่ไม่ดี(รูปที่ 3.2 ก. โหนดสถานะเส้นประ) ก็ตาม และในเอพิโสดถัดไปก็ไม่สามารถยืนยันได้ว่าเอเจนต์จะสามารถกลับมาทดลองเลือกทางเลือกอื่นในสถานะที่ดีที่ได้พบในเอพิโสดก่อนหน้านี้ได้

สถานะที่สามารถย้อนกลับไปเรียนรู้ซ้ำได้นี้ จะอ้างอิงจากสถานะล่าสุดที่ได้รับค่ารางวัลเป็นบวก(รูปที่ 3.2 ข. โหนดสถานะเส้นคู่) หากเอเจนต์ได้ทำการกลับมาเรียนรู้ยังสถานะนี้แล้วสามารถเรียนรู้จนพบกับสถานะที่ได้รับรางวัลเป็นบวกต่อไปได้ สถานะที่สามารถย้อนกลับไปเรียนรู้ซ้ำได้จะถูกเปลี่ยนเป็นสถานะใหม่นี้ แต่เนื่องจากเอเจนต์อาจย้อนกลับไปเรียนรู้ในสถานะเดิมในเอพิโสดเดียวกันหลายครั้ง ดังนั้นความยาวของลำดับขั้นของการเรียนรู้ในหนึ่งเอพิโสดจะมีจำนวนครั้งของการค้นหามากกว่าอัลกอริธึม RLปกติ จึงควรกำหนดจำนวนขั้นของการลองผิดลองถูกให้ไม่เกินค่าๆหนึ่ง เพื่อหลีกเลี่ยงการวนกลับไปเรียนรู้ยังสถานะที่เคยผ่านมาแล้วไม่รู้จบ

ในกรณีที่ลำดับของการกระทำที่เอเจนต์ได้เลือกทดลองผ่านไปยังสถานะต่างๆ หากผลของลำดับรางวัลที่ได้รับไม่ใช่ค่าบวกต่อเนื่องกันหลายครั้ง เอเจนต์สามารถย้อนกลับไปยังสถานะที่เป็นบวกครั้งหลังสุด(สถานะที่สามารถเรียนรู้ซ้ำได้)ด้วยค่าความน่าจะเป็นค่าหนึ่งๆ ซึ่งโอกาสที่จะย้อนกลับจะมีค่าแปรผันตามความลึกของลำดับการกระทำที่ไม่ได้รับรางวัลค่าบวกที่ต่อเนื่องข้างต้น และในสถานะสุดท้ายที่ก่อนที่เอเจนต์จะย้อนกกลับไปสถานะที่เป็นบวกล่าสุด ให้กำหนดค่าของรางวัลมีค่าลบ เพื่อทำเครื่องหมายให้เอเจนต์หลีกเลี่ยงตัวเลือกนั้นๆ ส่วนสถานะในลำดับก่อนหน้านี้รางวัลมีค่าเป็นตามปกติ ค่ารางวัลดังกล่าวจะมีน้ำหนักเทียบเท่ากับการกระทำอื่นที่เอเจนต์ยังไม่เคยได้ทำการสำรวจ

อัลกอริธึมนี้จะไม่กระทบกับกระบวนการปรับฟังก์ชันมูลค่า จึงสามารถเลือกใช้อัลกอริธึมวิธีการเรียนรู้ เช่น SARSA[1], Q-Learning[2], SARSA(λ)[14] หรือ Watkin's Q(λ)[8] ได้ทั้งหมด ดังแสดงในรูปที่ 3.3

กำหนดเริ่มต้นให้กับค่า $Q(s,a)$

พิจารณาสถานะปัจจุบัน s

กำหนดสถานะที่สามารถเรียนรู้ซ้ำ $s_{retry} \leftarrow s$

ดำเนินการเรียนรู้ซ้ำ (ในแต่ละเอพิโสด)

ดำเนินการซ้ำ (ในขั้นตอนการเรียนรู้)

เลือกการกระทำ a จากฟังก์ชันมูลค่า $Q(s,a)$ ตามปกติ ด้วยโพลิซีตามสมการ

$$\pi(s_t) = \begin{cases} argmax_{a_t} Q(s_t, a_t), & \varepsilon^0 \leq \varepsilon \\ a_{random}, & \varepsilon^0 > \varepsilon \end{cases}$$

กระทำ a

รับสถานะถัดไป s' และรางวัล $r(s,a)$

กรณี $r(s,a) > 0$

กำหนดสถานะที่สามารถเรียนรู้ซ้ำด้วยสถานะถัดไป $(s_{retry} \leftarrow s')$

กรณี $r(s,a) = 0$

สุ่มความน่าจะเป็น

ถ้าความน่าจะเป็นมากว่าค่าที่กำหนด

ย้อนกลับไปยังสถานะ s_{retry} $(s' \leftarrow s_{retry})$

$$r(s,a) \leftarrow -1$$

ปรับค่าฟังก์ชันมูลค่า $Q(s,a)$ ตามสมการ Watkin's Q(λ)

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$$

เคลื่อนไปยังสถานะถัดไป $(s \leftarrow s')$

จนกว่าสถานะ s เป็นสถานะสิ้นสุด

รูปที่ 3.3 อัลกอริธึม RL-RCS

3.3 การเพิ่มประสิทธิภาพให้กับอัลกอริธึมการเรียนรู้ Q ด้วยการแนะนำจากฟังก์ชันฮิวริสติก (Heuristically accelerated Q-learning, HAQL)[3]

การเพิ่มประสิทธิภาพของการเรียนรู้ด้วยการใช้ฟังก์ชันฮิวริสติกแนะนำ จะช่วยให้เอเจนต์มีโพลิซีที่สามารถตัดสินใจเลือกกระทำเพื่อการสำรวจคำตอบบนปริภูมิได้อย่างมีเหตุผล อีกทั้งยังลดระยะเวลาในการเรียนรู้แบบสุ่ม โดยสิ่งที่แนะนำจากฟังก์ชันฮิวริสติก จะช่วยชี้ทิศทางในการเรียนรู้ที่มีคำตอบอยู่ในบริเวณของปริภูมิคำตอบที่ดีตั้งแต่เริ่มใช้งาน ส่งผลให้ฟังก์ชันมูลค่า Q ถูกปรับปรุงด้วยคำตอบที่มาจากฟังก์ชันฮิวริสติกตั้งแต่เริ่มการเรียนรู้ ซึ่งแตกต่างจากอัลกอริธึมการเรียนรู้แบบเสริมกำลังที่มีการพัฒนามาก่อนหน้านี้ ที่เป็นการปรับปรุงให้เอเจนต์เลือกกรีดีในทิศทางของ

ฟังก์ชันมูลค่าQ สูงสุด ที่เกิดจากการปรับปรุงฟังก์ชันมูลค่าQ จากนั้นเอเจนต์จึงเลือกการกระทำจากการกรีดีในภายหลัง ดังนั้นตัวเลือกการกระทำที่มีค่ามากที่สุดบนสถานะหนึ่งๆ ซึ่งไม่ใช่คำแนะนำด้วยแนวคิดใดๆ

อัลกอริธึมHAQL[3] เป็นการแก้ปัญหาด้วยกระบวนการตัดสินใจมาร์คอฟ[2] ที่ใช้ฟังก์ชันฮิวริสติก $\mathcal{H} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ เป็นมูลค่าที่จะแนะนำการกระทำให้กับเอเจนต์ ใช้สัญลักษณ์ $\mathcal{H}(s_t, a_t)$ คือ กลุ่มของการกระทำที่ดีที่ฟังก์ชันฮิวริสแนะนำให้เอเจนต์เลือกการกระทำ a_t เมื่ออยู่บนสถานะ s_t

คุณลักษณะเด่นของอัลกอริธึม HAQL อยู่ตรงที่ฟังก์ชันฮิวริสติกสามารถปรับปรุงหรือแก้ไขได้แบบออนไลน์ ทำให้กระบวนการเรียนรู้และข้อมูลการแนะนำถูกปรับปรุงตรงกับสถานะโจทย์ตลอดเวลา ดังนั้นการสร้างคำแนะนำจะสามารถใช้แนวคิดฟังก์ชันฮิวริสติกคำนวณไปควบคู่กับแต่ละสถานะที่เอเจนต์ผ่านไปได้ทันที

3.3.1 ฟังก์ชันฮิวริสติก $\mathcal{H}$ ของอัลกอริธึม HAQL

ฟังก์ชันฮิวริสติกเป็นฟังก์ชันที่ช่วยปรับปรุงกระบวนการเลือกการกระทำที่ไม่ส่งผลกระทบต่อกระบวนการปรับปรุงฟังก์ชันมูลค่าของอัลกอริธึมการเรียนรู้แบบเสริมกำลังแต่อย่างใด ยกตัวอย่างกรณี วิธีการเรียนรู้Q (Q-learning) ที่ใช้โพลิซีแบบกรีดี ε (ε -Greedy) โพลิซีนี้จะเลือกการกระทำที่เป็นการกรีดีจากผลรวมระหว่างฟังก์ชันมูลค่าQ $Q(s_t, a_t)$ กับฟังก์ชันฮิวริสติก $\mathcal{H}_t(s_t, a_t)$ และเลือกการสำรวจด้วยการสุ่ม ดังสมการที่ 3.1

$$\pi(s_t) = \begin{cases} argmax_{a_t}\,[Q(s_t, a_t) + \mathcal{H}_t(s_t, a_t)], & \varepsilon^0 \leq \varepsilon \\ a_{random}, & \varepsilon^0 > \varepsilon \end{cases} \qquad (3.1)$$

โดยที่

$Q(s_t, a_t)$ คือ ฟังก์ชันมูลค่าQ ของอัลกอริธึม Q-Learning

$\mathcal{H}_t(s_t, a_t)$ คือ ฟังก์ชันฮิวริสติกที่แนะนำให้เลือกการกระทำ a_t บนสถานะ s_t

ε^0 คือ ค่าของการสุมชนิดไม่มีรูปแบบ มีค่าบนช่วง [0, 1]

ε คือ ตัวแปรที่เป็นตัวกำหนดการสำรวจ

a_{random} คือ การกระทำที่สุ่มจากการกระทำที่เป็นไปได้บนสถานะ s_t

โดยปกติค่าของฟังก์ชัน $\mathcal{H}_t(s_t, a_t)$ จะเป็นค่าที่บวกเพิ่มให้กับฟังก์ชันมูลค่าQ ในทุกๆ ตัวเลือกที่แนะนำ ที่ทำให้ผลรวมมีค่าสูงกว่าฟังก์ชันมูลค่าสูงสุด $(Q^*(s_t, a_t))$ เพื่อให้มีน้ำหนักที่

มากพอจะเอาชนะตัวเลือกการกระทำจากการกรีดีปกติ แต่ก็ไม่ควรกำหนดให้มากจนเกินไป เพราะตัวเลือกที่ฟังก์ชันฮิวริสติกแนะนำอาจไม่ใช่ตัวเลือกที่ถูกต้อง มูลค่าของฟังก์ชัน $\mathcal{H}_t(s_t, a_t)$ สามารถคำนวณได้ดังสมการ 3.2

$$\mathcal{H}_t(s_t, a_t) = \begin{cases} Q^*(s_t, a) - Q(s_t, a_t) + \eta, & \text{กรณี } a_t = \pi^{\mathcal{H}}(s_t) \\ 0, & \text{กรณี } a_t \neq \pi^{\mathcal{H}}(s_t) \end{cases} \quad (3.2)$$

โดยที่

η คือ ค่าน้อยๆ เพื่อกำหนดการแนะนำของฟังก์ชัน $\mathcal{H}_t(s_t, a_t)$

$\pi^{\mathcal{H}}(s_t)$ คือ โพลิซีที่เกิดจากการใช้ฟังก์ชันฮิวริสติก

ผลรวมระหว่างฟังก์ชันมูลค่าQและฟังก์ชันฮิวริสติก$\mathcal{H}_t(s_t, a_t)$ (สมการ 3.1) ที่คำนวณจากทุกๆการกระทำที่แนะนำด้วยฟังก์ชันฮิวริสติกจึงถูกกำหนดค่าให้เท่ากับผลต่างของฟังก์ชันมูลค่าสูงสุดและฟังก์ชันมูลค่า $Q(s_t, a_t)$ บวกเพิ่มด้วยค่าน้อยๆอีก η หน่วย ดังนั้นเมื่อทำการแทนค่าทุกๆตัวเลือกของการกระทำที่แนะนำโดยฮิวริสติก โพลิซี $\pi(s_t)$ จะให้ผลรวมของ $Q(s_t, a_t) + \mathcal{H}_t(s_t, a_t)$ ในแต่ละค่าที่ถูกแนะนำมีค่าเป็น $Q^*(s_t, a_t) + \eta$ โดยที่ตัวเลือกเหล่านี้ไม่ขึ้นกับฟังก์ชันมูลค่าQ ที่ถูกปรับปรุงไว้จริง ส่วนการกระทำที่ไม่ถูกฟังก์ชันฮิวริสติกแนะนำจะมีค่าผลรวมของ $Q(s_t, a_t) + \mathcal{H}_t(s_t, a_t)$ เท่ากับฟังก์ชันมูลค่าQ ปรับปรุงไว้จริงๆ สามารถสรุปเป็นอัลกอริธึมการทำงาน ดังรูปที่ 3.4

กำหนดเริ่มต้นให้กับค่า $Q(s,a)$

คำนวณฟังก์ชันฮิวริสติก $\mathcal{H}(s,a)$ ด้วยวิธีที่เหมาะสม

พิจารณาสถานะปัจจุบัน s

ดำเนินการเรียนรู้ซ้ำ (ในแต่ละเอพิโสด)

ดำเนินการซ้ำ (ในขั้นตอนการเรียนรู้)

เลือกการกระทำ a โดยโพลิซี π ตามสมการ

$$\pi(s_t) = \begin{cases} argmax_{a_t}\,[Q(s_t,a_t) + \mathcal{H}_t(s_t,a_t)], & \varepsilon^0 \leq \varepsilon \\ a_{random}, & \varepsilon^0 > \varepsilon \end{cases}$$

กระทำ a

พิจารณาค่า $r(s,a)$ ที่ได้รับ พร้อมรับสถานะถัดไป s'

คำนวณฟังก์ชันฮิวริสติก $\mathcal{H}(s,a)$ ด้วยวิธีที่เหมาะสม

ปรับค่าฟังก์ชันมูลค่า $Q(s,a)$ ตามสมการ Q-Learning

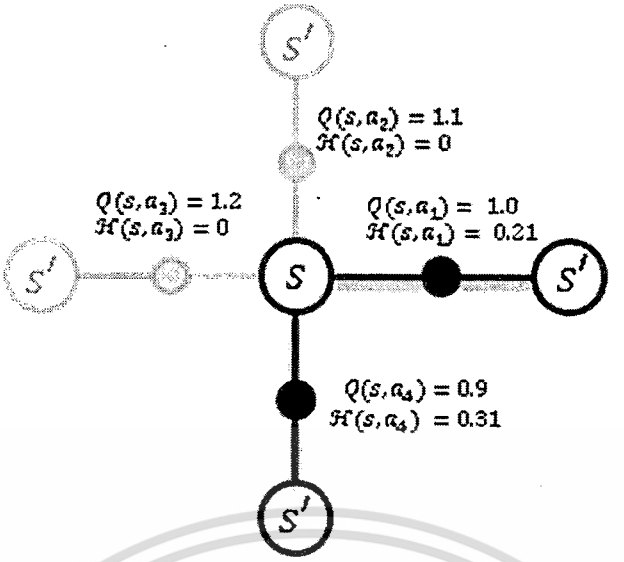
$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a_t) - Q(s_t,a_t)]$$

เคลื่อนไปยังสถานะถัดไป $(s \leftarrow s')$

จนกว่าสถานะ s เป็นสถานะสิ้นสุด

รูปที่ 3.4 อัลกอริธึม HAQL

ยกตัวอย่างเช่น สมมุติให้ สถานะปัจจุบันS (รูปที่ 3.5) มีการกระทำที่เป็นไปได้ $\{a_1, a_2, a_3, a_4\}$ และมีฟังก์ชันมูลค่า $Q(s_t,a_t)$ เท่ากับ {1.0, 1.1, 1.2, 0.9} ตามลำดับ และสมมุติให้ η มีค่า 0.01 ถ้าหากการกระทำ a_1และ a_4 ที่เป็นค่าที่ฟังก์ชันฮิวริสติกแนะนำ ดังนั้น ค่าของฟังก์ชัน $\mathcal{H}(s_t,a_1) = 0.21$ และ $\mathcal{H}(s_t,a_4) = 0.31$ ซึ่งตัวเลือกการกระทำอื่นๆ ค่าแนะนำของฟังก์ชันฮิวริสติกเท่ากับ 0 จะได้ว่า ผลรวม $Q(s_t,a_t) + \mathcal{H}_t(s_t,a_t)$ ของการกระทำ a_1และ a_4 มีค่าเท่ากันที่ 1.21 หน่วย และเป็นค่าผลรวมสูงสุดที่มีโอกาสถูกเลือกด้วยโพลิซีแบบ ε-greedy ด้วยความน่าจะเป็นที่เท่ากัน



รูปที่ 3.5 ตัวอย่างคู่สถานs และการกระทำ a ใดๆ ที่สงผลให้เกิดสถานะถัดไปs' ที่เกิดจากการคำนวณโพลิซีด้วยอัลกอริธึม HAQL

3.4 การประยุกต์ใช้การเรียนรู้แบบเสริมกำลังในการแก้ปัญหาการเลือกโปรโตไทป์

ในการแก้ปัญหาด้วยการเรียนรู้แบบเสริมกำลังจำเป็นจะต้องทำการแปลง (Mapping) ปัญหาที่สนใจให้อยู่ในรูปสถานะ เพื่อให้เอเจนต์หาทางเลือกที่ดีที่สุด โดยเริ่มแก้ปัญหาจากสถานะเริ่มต้นและเปลี่ยนสถานะไปเรื่อยๆ จนถึงสถานะสุดท้าย ซึ่งในแต่ละสถานะที่ผ่านไปจำเป็นต้องกำหนดผลตอบแทนสำหรับแต่ละการกระทำที่เอเจนต์ได้เลือกอย่างมีเหตุผลและสอดคล้องกับปัญหา รวมถึงการเลือกใช้ทฤษฎีการปรับปรุงฟังก์ชันมูลค่า ต้องเลือกใช้เหมาะสมกับปัญหา ในแต่ละสถานะที่ผ่านไปเอเจนต์จะพิจารณาตัวเลือกด้วยโพลิซีแบบ ε-greedy ดังนั้นเอเจนต์จะเลือกการกระทำที่ได้ผลตอบแทนมากที่สุดเป็นหลัก หรืออาจใช้การกระทำสุ่มเพื่อการสำรวจด้วยความน่าจะเป็นที่น้อยกว่า

ดังนั้นหากทำการประยุกต์ใช้การเรียนรู้แบบเสริมกำลังในการแก้ปัญหาโปรโตไทป์ (หัวข้อ 2.3.1) จำเป็นจะต้องแปลงปัญหาให้อยู่ในรูปแบบของสถานะที่เอเจนต์สามารถติดต่อสื่อสารผ่านสภาพแวดล้อมได้

เริ่มต้นปัญหาด้วยการใช้สถานะs นำเสนอตัวแทนของข้อมูลอ้างอิง n ตัว หรืออีกนัยหนึ่งสถานะคือการนำเสนอคำตอบของปัญหาการเลือกโปรโตไทป์ด้วยเซตของข้อมูล จากนั้นจึงให้เอเจนต์เรียนรู้ตัวเลือกการกระทำa ซึ่งหมายถึงให้ทำการเลือกข้อมูลโปรโตไทป์หนึ่งค่าจากเซตข้อมูลอ้างอิง (สถานะปัจจุบัน) เพื่อให้ได้รับรางวัลr มาสะสมในรูปแบบของฟังก์ชันมูลค่าQ เอเจนต์และสภาพแวดล้อมจะโต้ตอบกันด้วยสถานะ, การกระทำ และรางวัลที่คาดว่าจะได้รับอยู่ตลอดเวลา ไปจนกระทั่งพบคำตอบของปัญหานั่นคือ สถานะที่มีจำนวนข้อมูลน้อยที่สุดที่ยังคงสามารถจำแนกข้อมูลอ้างอิงจำนวน n ตัวได้อย่างถูกต้อง ดังนั้น เส้นทางการค้นหาคำตอบที่ดีที่สุดจะเกิดขึ้นได้ก็ต่อเมื่อเอเจนต์ทำการเลือกการกระทำในทุกๆ สถานะตลอดเส้นทางการเรียนรู้ในหนึ่งครั้ง ที่ก่อให้เกิดมูลค่าของผลตอบแทนสูงที่สุด (ผลรวมของรางวัลตลอดเส้นทางการค้นหา)

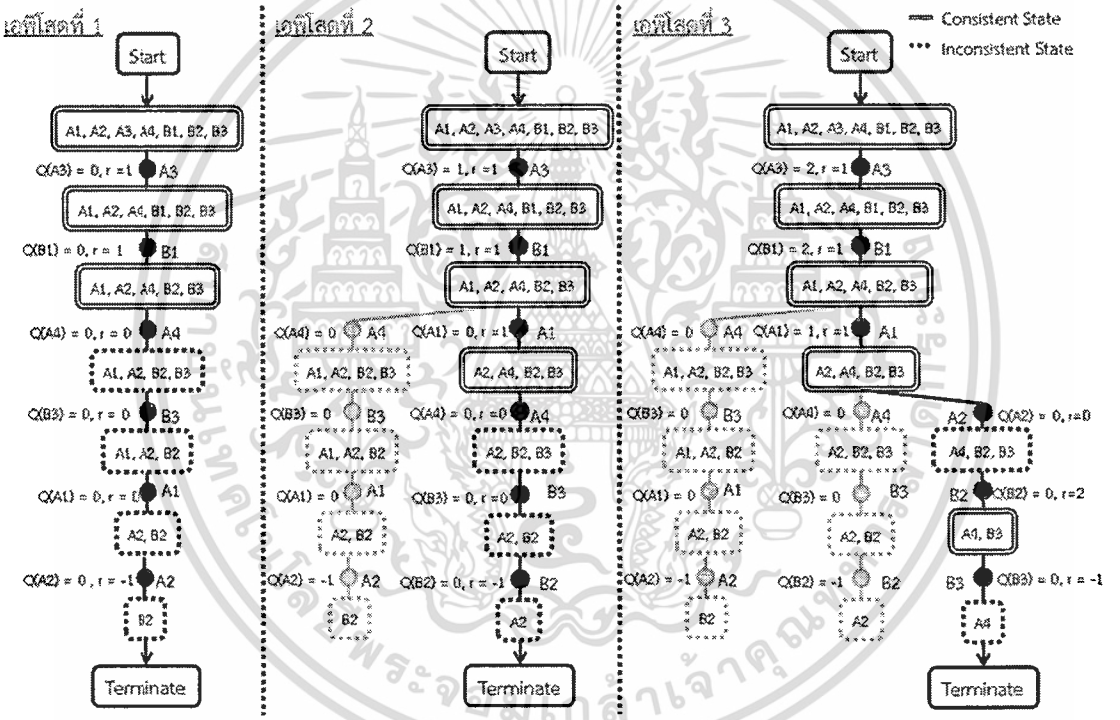
ในการเลือกโปรโตไทป์ด้วยกระบวนการเรียนรู้แบบเสริมกำลังแบ่งออกเป็น 2 แนวทาง แนวทางแรกคือกระบวนการเลือกโปรโตไทป์ด้วยการลดข้อมูลอ้างอิงออกจากเซต เริ่มจากการใช้ข้อมูลอ้างอิงเป็นเซตของโปรโตไทป์จากนั้นทำการลดข้อมูลออกทีละตัว โดยตลอดช่วงของการแก้ปัญหา เซตคำตอบใหม่ที่ได้จะต้องมีคุณสมบัติในการจำแนกชุดข้อมูลอ้างอิงขนาด n ตัวได้อย่างถูกต้อง หากเซตใดที่ไม่สามารถจำแนกข้อมูลอ้างอิงได้ครบจะถือว่าเซตคำตอบที่พบครั้งล่าสุดคือคำตอบของรอบการแก้ปัญหานั้นๆ ในแนวทางนี้จะพบว่าเอเจนต์จะสามารถเลือกข้อมูลอ้างอิงออกถูกต้องเป็นส่วนใหญ่ เนื่องจากเซตคำตอบที่เหมาะสมจะมีขนาดเล็ก ดังนั้นช่วงต้นของการค้นหา เอเจนต์สามารถเลือกข้อมูลตัวใดที่ไม่ใช่สมาชิกในเป็นเซตคำตอบออกก็ได้ เนื่องความสามารถในการจำแจกข้อมูลอ้างอิงยังคงถูกต้อง หรืออาจคลาดเคลื่อนไม่มากนัก สังเกตได้จากผลรางวัลที่ได้รับ หากมีค่าบวกหมายถึงเอเจนต์ทำการเลือกลดจำนวนข้อมูลได้อย่างถูกต้อง แต่การแก้ปัญหาด้วยการลดจำนวนข้อมูลจะมีเส้นทางการค้นหาคำตอบที่ค่อนข้างลึก เอเจนต์อาจไม่สามารถค้นหาคำตอบที่เหมาะสมได้เนื่องจากทรัพยากรหน่วยความจำในกรณีที่สถานะมีขนาดใหญ่มากๆ

ส่วนในแนวทางที่สองคือกระบวนการเลือกข้อมูลอ้างอิงเพื่อสร้างเป็นเซตโปรโตไทป์ เมื่อเริ่มต้นการแก้ปัญหาเซตคำตอบจะเป็นเซตว่าง จึงไม่สามารถทำการจำแนกชุดข้อมูลอ้างอิงได้ถูกต้องตั้งแต่แรกและเมื่อเอเจนต์ทำการเลือกข้อมูลอ้างอิงเพิ่มเข้ามา จะไม่มีทางทราบได้ว่าเป็นเส้นทางการค้นหาที่ถูกต้องหรือไม่ เพราะหากเซตคำตอบที่ได้ไม่ยังสามารถจำแนกข้อมูลอ้างอิงได้จะได้รับรางวัลเป็นศูนย์ตลอดเวลา เอเจนต์อาจทำการเพิ่มข้อมูลที่ใช่สมาชิกของเซตคำตอบที่เหมาะสมเข้ามาเมื่อใดก็ได้ ทำให้เซตคำตอบอาจไม่ใช่เซตขนาดเล็กสุดจริง ข้อดีของแนวทางนี้คือ เส้นทางการค้นหาคำตอบที่เอเจนต์เรียนรู้จะไม่ลึกมากหากเทียบกับวิธีแรก แต่อาจใช้เวลาในการลองผิดลองถูกค่อนข้างมาก เนื่องจากเอเจนต์จะพบรางวัลก็ต่อเมื่อเจอเซตคำตอบที่สามารถจำแนกข้อมูลได้ถูกต้อง(คำตอบของปัญหา) เท่านั้น

ดังนั้นในการยกตัวอย่างเพื่ออธิบายขั้นตอนการทำงานของกระบวนการเรียนรู้แบบเสริมกำลังในการแก้ปัญหาการเลือกโปรโตไทป์ จึงเลือกแนวทางการเลือกโปรโตไทป์ด้วยการลดข้อมูลอ้างอิงออกจากเซต เนื่องจากในทุกๆขั้นของการเรียนรู้ แนวโน้มของคำตอบจะมีทิศทางที่ดีขึ้นพร้อมทั้งสามารถนำเสนอขั้นตอนการแก้ไขปัญหาได้โดยง่าย โดยสมมุติข้อมูลตัวอย่างในหัวข้อ 2.3.2 เพื่อแก้ปัญหาด้วยอัลกอริธึมRLแบบปกติ, อัลกอริธึมRL-RCS และอัลกอริธึม HAQL ตามลำดับ

3.4.1 อัลกอริธึม RL ปกติ

เอเจนต์เริ่มต้นการเรียนรู้ในเอพิโสดที่ 1 สถานะเริ่มต้นประกอบด้วยข้อมูลอ้างอิง 7 ตัวเป็นสมาชิกในเซตโปรโตไทป์(รูปที่ 3.6 ซ้าย) จากนั้นให้เอเจนต์ทำการเรียนรู้ด้วยการสุ่มสำรวจข้อมูล A3, และ B1 สถานะของเซตคำตอบ {A1, A2, A4, B2, B3} ยังคงสามารถจำแนกข้อมูลอ้างอิงได้อย่างถูกต้อง และเมื่อเอเจนต์สุ่มสำรวจข้อมูล A4, B3, A1 และ B2 พบว่า สถานะที่ผ่านมาสามารถทำจำแนกข้อมูลอ้างอิงได้อย่างถูกต้อง จนกระทั่งเอเจนต์เคลื่อนไปถึงสถานะ {B2} เอพิโสดจะสิ้นสุดลง เนื่องจากเซตคำตอบสูญเสียความสามารถในการจำแนกข้อมูลชนิด A ดังนั้นเซตโปรโตไทป์ที่เป็นเซตย่อยสอดคล้องขนาดเล็กที่สุดที่พบในเอพิโสด 1 คือเซต {A1, A2, A4, B2, B3} ประกอบด้วยสมาชิกในเซตจำนวน 5 ตัว



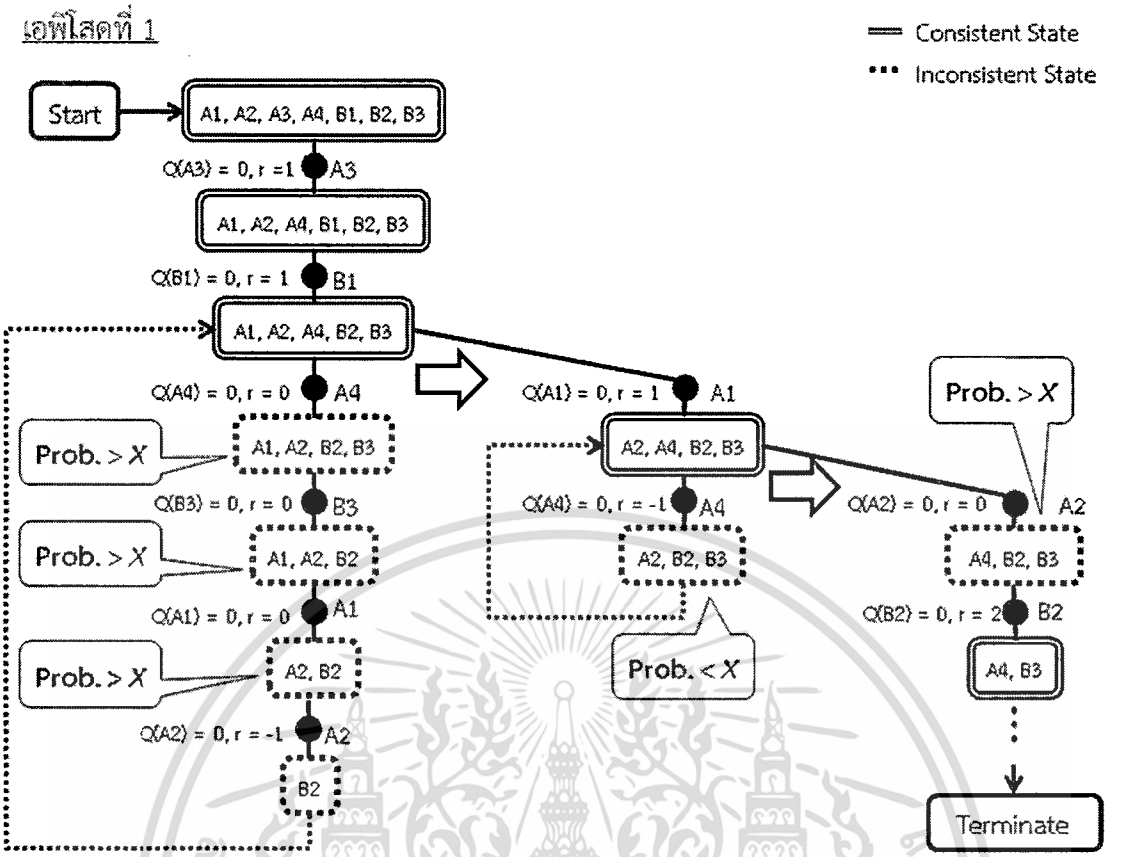
รูปที่ 3.6 แผนภาพแบคอัพแสดงลำดับการตัดสินใจของเอเจนต์ด้วยอัลกอริธึมRLปกติ

เมื่อเริ่มต้นเอพิโสดที่ 2 สถานะเริ่มต้นประกอบด้วยข้อมูลอ้างอิง 7 ตัวเป็นสมาชิกในเซตโปรโตไทป์ดังเดิม (รูปที่ 3.6 กลาง) เอเจนต์เริ่มต้นการเรียนรู้ด้วยการเลือกกรีดข้อมูล A3, B1 จากนั้นสุ่มสำรวจข้อมูล A1 ณ สถานะนี้เอเจนต์ได้พบเซตคำตอบ {A2, A4, B2, B3} ที่เป็นเซตย่อยสอดคล้องเล็กที่สุดที่มีสมาชิกน้อยกว่าเซตคำตอบที่ได้จากเอพิโสดที่ 1 ซึ่งหลังจากสถานะนี้ เอเจนต์ทำการสุ่มสำรวจข้อมูล A4, B3 และ B2 แต่ก็ไม่พบสถานะที่นำมาจำแนกข้อมูลอ้างอิงได้อย่างถูกต้อง เอพิโสดที่ 2 จึงสิ้นสุด ณ สถานะ {A2} นี้โดยมีเซตคำตอบ {A2, A4, B2, B3} ประกอบด้วยสมาชิกในเซตจำนวน 4 ตัว

เอพิโสดที่ 3 สถานะเริ่มต้นยังคงใช้ข้อมูลอ้างอิง 7 ตัวเป็นสมาชิกในเซตโปรโตไทป์ (รูปที่ 3.6 ขวา) เอเจนต์เริ่มเรียนรู้ด้วยการกรีดีเช่นเดิมโดยเลือกข้อมูล A3, B1 และ A1 ตามลำดับ และได้เคลื่อนสู่สถานะ {A2, A4, B2, B3} จากนั้นเอเจนต์ได้สุ่มสำรวจข้อมูล A2 และได้เคลื่อนสู่สถานะ {A4, B2, B3} แต่เนื่องจากสถานะนี้เป็นสถานะที่ไม่สามารถนำมาใช้จำแนกข้อมูลอ้างอิงได้เนื่องจากสูญเสียความสามารถในการจำแนกข้อมูลชนิด A ไปบางส่วน และเนื่องจากการสำรวจสถานะคำตอบบางครั้งเอเจนต์จำเป็นต้องผ่านไปมากกว่าหนึ่งสถานะ สภาพแวดล้อมจึงอนุญาตให้ เอเจนต์ได้ทำการเรียนรู้ต่อไป เพื่อให้เอเจนต์ทำการเลือกข้อมูลที่ทำให้เซตคำตอบกลับมาทำการจำแนกข้อมูลได้อีกครั้ง เอเจนต์จึงทำการสุ่มเลือกข้อมูล B2 และเคลื่อนสู่สถานะ {A4, B3} ซึ่งเป็นเซตย่อยสอดคล้องเล็กที่สุดประกอบด้วยสมาชิกในเซตจำนวน 2 ตัว โดยที่เซตคำตอบ {A4, B3} เป็นเซตคำตอบที่ดีที่สุดตั้งแต่เอเจนต์เริ่มทำการเรียนรู้และเป็นเซตคำตอบที่ดีที่สุดสำหรับชุดข้อมูลตัวอย่าง

3.4.2 อัลกอริธึม RL-RCS

เริ่มต้นเอพิโสดจากสถานะเริ่มต้นที่มีข้อมูลอ้างอิงทุกตัวเป็นสมาชิกในเซตโปรโตไทป์(รูปที่ 3.7) จากนั้นให้เอเจนต์ทำการเรียนรู้ด้วยการเลือกโปรโตไทป์ A3, B1, A4, B3, A1 และ B2 ตามลำดับ ออกจากการเป็นสมาชิกของเซตโปรโตไทป์ จนกระทั่งเอเจนต์เคลื่อนไปถึงสถานะ {A2, B2} ไปสู่สถานะ {B2} (ปกติอัลกอริธึม RL จะถือว่าสถานะนี้เป็นสถานะสิ้นสุดของเอพิโสด) ณ สถานะนี้ เซตโปรโตไทป์ที่เป็นเซตย่อยสอดคล้องขนาดเล็กที่สุดที่พบคือเซต {A1, A2, A4, B2, B3} ดังนั้นด้วยอัลกอริธึม RL-RCS นี้เอเจนต์จึงถูกย้ายจากสถานะ {A2, B2} กลับไปสู่สถานะ {A1, A2, A4, B2, B3} แทนที่จะเคลื่อนยังสถานะ {B2} เพื่อให้โอกาสในการสำรวจต่อจากสถานะ {A1, A2, A4, B2, B3} ใหม่อีกครั้ง แทนที่การสิ้นสุดเอพิโสดตามปกติ



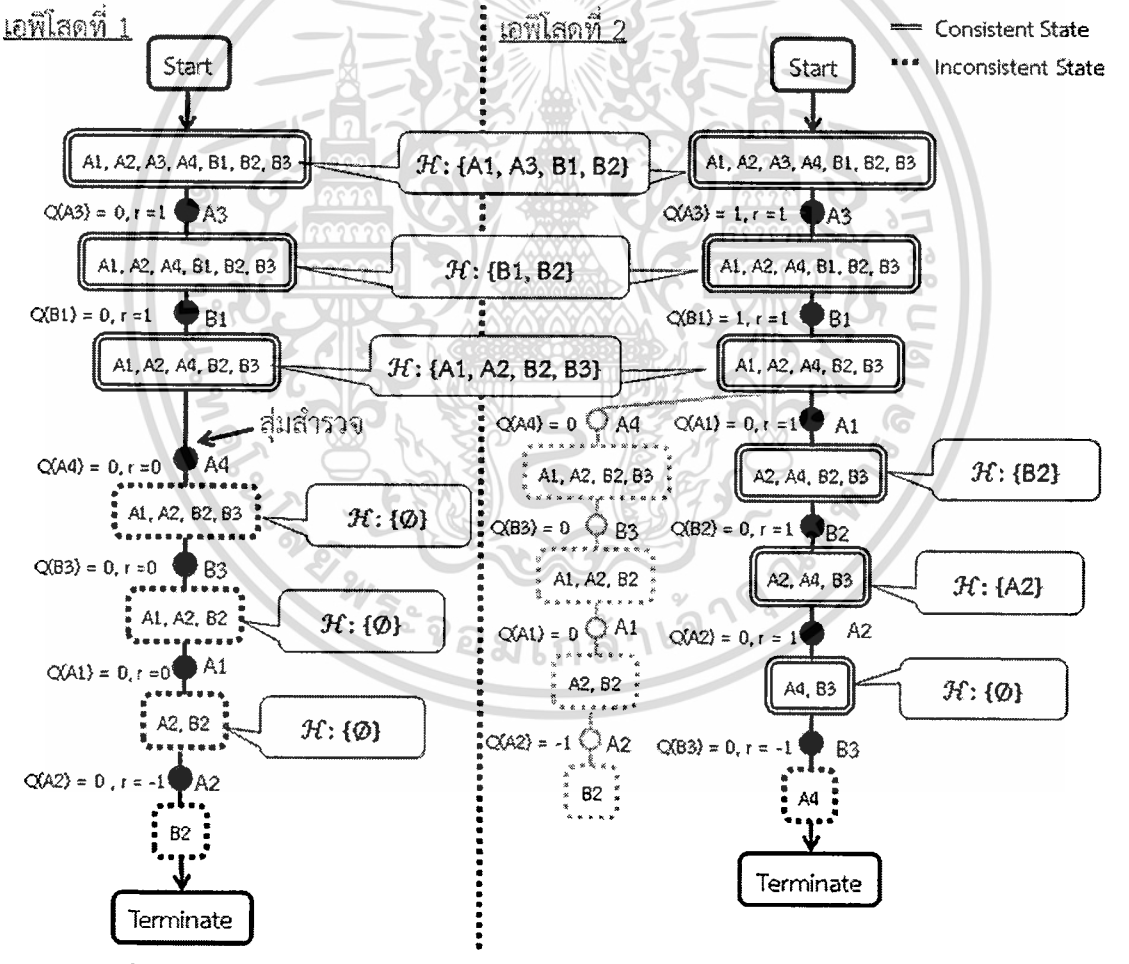
รูปที่ 3.7 แผนภาพแบคอัพแสดงลำดับการตัดสินใจของเอเจนต์ด้วยอัลกอริธึมRL-RCS

เมื่อเอเจนต์ทำการเลือกข้อมูลออกจากเซตโปรโตไทป์ ณ สถานะ {A1, A2, A4, B2, B3} ซ้ำอีกครั้ง ในระหว่างเส้นทางการค้นหาที่ไม่ยังพบเซตสอดคล้องใหม่ อาจมีโอกาสที่จะเกิดการย้อนกลับไปที่สถานะที่สามารถเรียนรู้ซ้ำได้ล่าสุด ภายในเอพิโสดนี้อีกครั้งหนึ่ง เช่น ในรูปจากการย้ายสถานะตามปกติจากสถานะ {A2, A4, B2, B3} ไปยังสถานะ {A2, B2, B3} ซึ่งไม่ใช่สถานะที่เซตโปรโตไทป์มีความสอดคล้อง ดังนั้นด้วยความน่าจะเป็น X จึงอาจจะบังคับให้เอเจนต์ย้อนกลับไปที่สถานะ {A2, A4, B2, B3} อีกครั้งหนึ่ง (ดังแสดงในขั้นตอนที่สองของรูปที่ 3.7) เอเจนต์จึงมีโอกาสได้ลองคัดโปรโตไทป์ออกจากสถานะ {A2, A4, B2, B3} ได้อีกหลายครั้งจนกระทั่งพบเซตย่อยสอดคล้อง {A4, B3} ในที่สุด

เมื่อเอเจนต์พบเซต {A4, B3} ซึ่งสถานะนี้พบคุณสมบัติความสอดคล้องด้วย ภายในเอพิโสดนี้เอเจนต์จึงไม่ย้อนกลับไปยังสถานะ {A2, A4, B2, B3} อีก ดังนั้นเอเจนต์ทำการคัดเลือกโปรโตไทป์ A4 หรือ B3 ออกจากเซตโปรโตไทป์ {A4, B3} เรื่อยไปและถูกย้ายกลับมายังสถานะ {A4, B3} ในภายหลัง ซึ่งสถานะถัดไปเป็นเซตที่ขาดคุณสมบัติสอดคล้อง (ทั้งเซต {A4} หรือเซต {B3}) จนจบเอพิโสด เซตคำตอบที่ได้จึงเป็นเซต {A4, B3} ซึ่งเป็นเซตคำตอบที่ดีที่สุดสำหรับชุดข้อมูลตัวอย่าง

3.4.3 อัลกอริธึม HAQL

สมมุติให้ใช้งานการแนะนำของฟังก์ชันฮิวริสติกในเอพิโสดที่ 1-2 เมื่อเริ่มต้นเอพิโสดที่ 1 มีสถานะเริ่มต้นเป็นข้อมูลอ้างอิงจำนวน 7 ตัวอยู่ในเซตของโปรโตไทป์ (รูปที่ 3.8 ซ้าย) เอเจนต์จะทำการเรียนรู้ด้วยการเลือกการกระทำตามที่ฟังก์ชันฮิวริสติกแนะนำข้อมูล A3, B1 ตามลำดับ ในสถานะ {A1, A2, A4, B2, B3} เอเจนต์เกิดความน่าจะเป็นที่เป็นการเลือกสำรวจข้อมูล A4 ทำให้สถานะภายหลัง ไม่ว่าเอเจนต์จะเลือกข้อมูล B3 หรือ A1 ความสามารถในการจำแนกข้อมูลชนิด A ก็ยังคงเสียไปบางส่วนทำให้ไม่ได้รับผลรางวัล จนกระทั่งเอเจนต์ได้ผ่านสู่สถานะ {B2} ความสามารถในการจำแนกข้อมูลชนิด A จึงสูญเสียไปอย่างถาวร เอพิโสดนี้จึงสิ้นสุดลงด้วยเงื่อนไขที่สถานะปัจจุบันไม่สามารถจำแนกข้อมูลอ้างอิงได้ถูกต้อง ดังนั้นในเอพิโสดที่ 1 นี้เอเจนต์สามารถค้นพบเซตย่อยสอดคล้องเล็กสุดบนสถานะ {A1, A2, A4, B2, B3} ที่มีจำนวนสมาชิกทั้งสิ้น 5 ตัว



รูปที่ 3.8 แผนภาพแบคอัพแสดงลำดับการตัดสินใจของเอเจนต์ด้วยอัลกอริธึมHAQL

จากนั้นจึงทำการเริ่มต้นเอพิโสดที่ 2 สถานะเริ่มต้นคือข้อมูลอ้างอิงจำนวน 7 ตัวที่เป็นสมาชิกในเซตโปรโตไทป์(รูปที่ 3.8 ขวา) เอเจนต์จะทำการกรีดีจากสมการ3.1 ในทุกๆ สถานะตลอดเส้นทางการค้นหาเอเจนต์จึงทำการเลือกข้อมูล A3, B1, A1, B2 และ A2 ในแต่ละสถานะที่ผ่านไปเซตคำตอบ

ที่ได้สามารถทำการจำแนกข้อมูลอ้างอิงได้อย่างถูกต้อง และมีขนาดเล็กลงเรื่อยๆ จนกระทั่งเอเจนต์เข้าสู่สถานะ {A4} ซึ่งสูญเสียความสามารถในการจำแนกข้อมูลชนิด B ทำให้เอพิโสดจบลงด้วยสถานะ {A4,B3} ซึ่งเป็นเซตย่อยสอดคล้องเล็กที่สุดที่มีขนาดสมาชิก 2 ตัว

จากตัวอย่างการเรียนรู้ข้างต้นทั้ง 3 กรณีจะพบว่ากระบวนการเรียนรู้แบบเสริมกำลังปกติ จะทำการเรียนรู้ไปไม่ถึงเซตคำตอบที่เหมาะสมได้ภายในเอพิโสดเดียว ดังนั้นการลองผิดลองถูกจะไม่เกิดประโยชน์หากฟังก์ชันมูลค่าQ นั้นไม่ถูกใช้งาน (ไม่พบฟังก์ชันมูลค่าQ ที่มีค่า) เอเจนต์จะไม่สามารถเลือกการกระทำแบบกรีดีได้เลย หากเอเจนต์ไม่มีแนวคิดภายนอกมาช่วยแนะนำ และหากใช้งานฟังก์ชันฮิวริสติก(อัลกอริธึม HAQL) เมื่อไรก็ตามที่เอเจนต์ได้เลือกการกระทำแบบสุ่มสำรวจ มีโอกาสค่อนข้างมากที่เอเจนต์จะเคลื่อนเข้าสู่บริเวณสถานะที่การแนะนำของฟังก์ชันฮิวริสติกไม่สามารถคำนวณต่อไปได้ จึงต้องอาศัยรอบของการเรียนรู้ถัดไปเพื่อค้นหาเส้นทางของคำตอบในทิศทางอื่นหรืออาจเพิ่มแนวคิดในการกลับมาเรียนรู้ซ้ำ ให้เอเจนต์ทำการเลือกการกระทำที่มีทิศทางการเรียนรู้ตามที่ฟังก์ชันฮิวริสติกแนะนำซึ่งช่วยขั้นตอนการลองผิดลองถูกของกระบวนการเรียนรู้ด้วยอัลกอริธึม RL-RCS ส่วนหัวข้อการเพิ่มประสิทธิภาพของกระบวนการเรียนรู้แบบเสริมกำลังจะกล่าวถึงในบทถัดไป

บทที่ 4
การเพิ่มประสิทธิภาพของการเรียนรู้แบบเสริมกำลัง โดยใช้การเรียนรู้จากทางเลือกของฮิวริสติก

เมื่อทำการวิเคราะห์อัลกอริธึม RL-RCS และอัลกอริธึม HAQL ในการแก้ปัญหาการเลือกโปรโตไทป์ที่นำเสนอในบทที่ 3 นั้นพบว่าทั้งสองอัลกอริธึมได้นำเสนอแนวคิดในการแก้ปัญหาโดยการสำรวจการกระทำที่ให้ฟังก์ชันมูลค่าที่ดีหรือคาดว่าจะได้มูลค่าที่ดี แต่หากปัญหาที่นำมาแก้ไขมีปริภูมิการค้นหาขนาดใหญ่ การสุ่มเลือกการกระทำโดยใช้โพลิซีกรีดี ε จะใช้เวลาในการค้นหาค่อนข้างนาน และด้วยกลไกของอัลกอริธึม RL-RCS จะใช้การเรียนเพื่อลองผิดลองถูกให้เอเจนต์สามารถผ่านไปยังสถานะถัดไป ดังนั้นการเรียนรู้ที่ได้จะเป็นการสุ่มสำรวจไปในสถานะที่ไม่ทราบฟังก์ชันมูลค่าอยู่เสมอ และหากในเอพิโสดถัดไป ถ้าเอเจนต์เลือกการกระทำที่เป็นการสำรวจเกิดขึ้น ผลการเรียนรู้ที่เคยสะสมมาจะไม่สามารถนำมาใช้เป็นน้ำหนักในการตัดสินใจเลือกตัวเลือกได้เลย ในส่วนของการใช้งานอัลกอริธึม HAQL นั้น จากโพลิซีที่ใช้เลือกการกระทำโดยปกติ หากการแนะนำของฮิวริสติกไม่ได้มีการปรับปรุงให้ทันสมัยตลอดเวลาหรือมีประสิทธิภาพการแนะนำไม่ดีเท่าที่ควร ตัวเลือกการกระทำที่เคยได้เลือกกระทำไปแล้วได้รับรางวัลที่ไม่ดีจะไม่ถูกกรองออกจากชุดของคำแนะนำ ก่อให้เกิดการเลือกการกระทำที่ไม่ดีซ้ำไปซ้ำมาได้ จึงเป็นที่มาของการปรับปรุงประสิทธิภาพของอัลกอริธึม RL ด้วยวิธีการเรียนรู้ทางเลือกของฮิวริสติก

ในบทนี้จะนำแสนอแนวทางในการแก้ปัญหาเพื่อเพิ่มประสิทธิภาพให้กับอัลกอริธึม RL ในการค้นหาคำตอบของปัญหาการเลือกโปรโตไทป์สำหรับชุดข้อมูลที่มีขนาดใหญ่ จะพิจารณาถึงการเพิ่มประสิทธิภาพด้วย การปรับน้ำหนักของการแนะนำด้วยฟังก์ชันฮิวริสติก และประยุกต์เข้ากับเทคนิคการกลับยังสถานะที่สามารถเรียนรู้ซ้ำได้โดย เลือกทดลองกับปัญหาการเลือกโปรโตไทป์ในกรณีของการลดจำนวนโปรโตไทป์ เนื่องจากกระบวนการเลือกข้อมูลออกที่มีโอกาสในการเลือกการกระทำที่ถูกต้องเสียเป็นส่วนใหญ่ อีกทั้งยังสามารถกำหนดรางวัลให้กับเอเจนต์ได้ง่ายจากการคำนวณสถานะถัดไปที่ค้นพบคุณสมบัติความสอดคล้องอยู่เสมอ หากเทียบกับกระบวนการเลือกข้อมูลเข้าที่ไม่ทราบเส้นทางการค้นหาคำตอบได้จนกว่าเอเจนต์จะค้นพบเซตที่มีความสอดคล้องโดยสมบูรณ์

4.1 แนวทางการปรับปรุงอัลกอริธึมการเรียนรู้แบบเสริมกำลังที่กลับมายังสถานะที่สามารถเรียนรู้ซ้ำได้(RL-RCS)[4]

กระบวนการเรียนรู้แบบเสริมกำลังที่ใช้อัลกอริธึม RL-RCS มีจุดเด่นตรงที่สามารถสำรวจเส้นทางการค้นหาที่เคยผ่านมาแล้วภายในเอพิโสดเดียวกันได้ เพื่อให้เอเจนต์กลับมาทำการค้นหาตัวเลือกอื่นๆที่ดี และหลีกเลี่ยงปริภูมิคำตอบแบบโลคอล ในทันทีที่เอเจนต์ได้เลือกการกระทำที่ได้รับผลรางวัลไม่พึงประสงค์ เช่น รางวัลมีค่าเป็นลบ ให้ทำการย้ายเอเจนต์ย้อนกลับไปยังสถานะที่ดีล่าสุด เพื่อให้สามารถเลือกการกระทำในตัวเลือกอื่นได้ต่อไป แทนที่จะโยนความรู้ที่ได้ทดลองเรียนรู้จนถึงสถานะปัจจุบันทิ้ง เพื่อให้โอกาสเอเจนต์ได้แก้ไขทางเลือกที่ไม่ดี ผลการเรียนรู้ที่ได้ก็จะลดขั้นตอนการเรียนรู้ในเส้นทางการค้นหาหลักของเอพิโสดถัดไป ส่งผลให้การเรียนรู้ลู่เข้าสู่ค่าที่ดีขึ้น

หากเทียบกับอัลกอริธึมRLปกติในการแก้ปัญหาการเลือกโปรโตไทป์ เมื่อคำตอบเกิดความผิดพลาด สภาพแวดล้อมจะกำหนดให้เอเจนต์ทำการเรียนรู้ในเอพิโสดถัดไปทันที เพราะเมื่อเอเจนต์ได้รับผลรางวัลที่เป็นลบแล้ว ไม่ว่าจะเคลื่อนไปสู่สถานะใดในภายหลัง ผลของรางวัลก็ยังคงติดลบเพิ่มขึ้นเรื่อยๆ จนกว่าเอพิโสดปัจจุบันจะสิ้นสุดลง ซึ่งถือว่าเส้นทางการค้นหาถูกผ่านไปยังสถานะถัดไปก่อนเวลาอันควร โดยที่อาจจะไม่ได้ทำการค้นหาปริภูมิโดยรอบก่อนก็เป็นได้ ทั้งนี้เอเจนต์อาจทำการเลือกตัวเลือกที่ผิดลำดับในบางสถานะเท่านั้น อีกทั้งยังเสียเวลาส่วนหนึ่งในการค้นหาในเอพิโสดก่อนหน้า แต่กลับไม่สามารถใช้ประโยชน์จากฟังก์ชันมูลค่าQ ที่ได้เรียนรู้มาได้

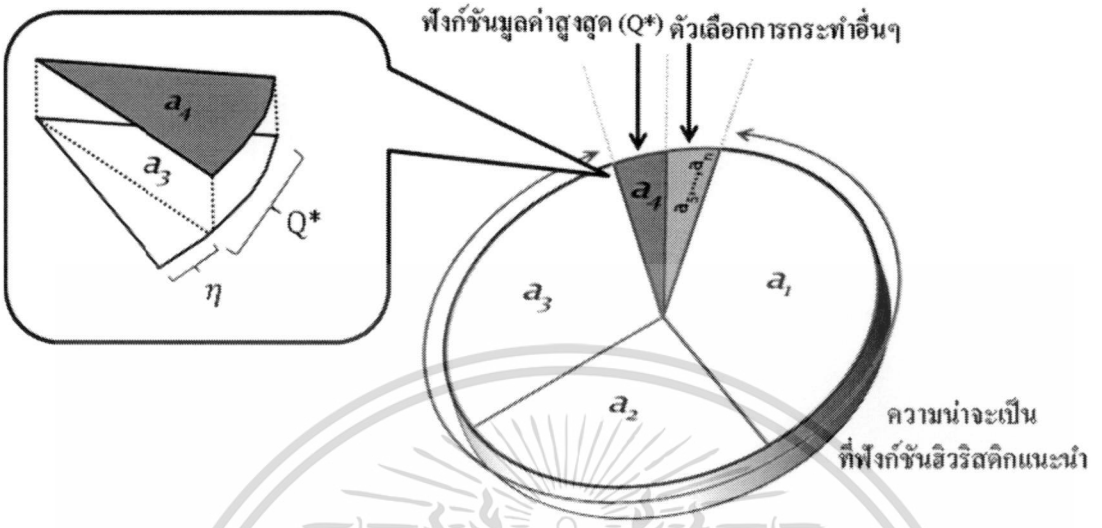
การประยุกต์เทคนิคที่อาศัยคำแนะนำจากการคำนวณด้วยฟังก์ชันฮิวริสติก จะสามารถแนะนำเอเจนต์ให้มีทิศทางการเรียนรู้คำตอบได้ดีขึ้น หากประยุกต์ใช้อย่างเหมาะสม ผลการเรียนรู้ของเอเจนต์จะเข้าสู่คำตอบตั้งแต่เริ่มใช้งาน หากเปรียบเทียบกับการเรียนรู้จากการสุ่มกระทำโดยไม่ใช้พื้นฐานการคำนวณ

ผลของการเรียนรู้ที่ได้จากกระประยุกต์คำแนะนำของฟังก์ชันฮิวริสติกนั้น ประสิทธิภาพในการค้นหาคำตอบของเอเจนต์จะแปรผันกับความสามารถในการแนะนำของฟังก์ชันฮิวริสติกที่ใช้ในการแก้ปัญหาโดยตรง ถึงแม้การเลือกเชื่อคำแนะนำทุกการกระทำอาจไม่ใช่คำตอบที่ถูกต้องเสียทีเดียว เนื่องจากเอเจนต์สามารถเรียนรู้การกระทำจากรางวัลที่ได้สะสมในรูปฟังก์ชันมูลค่าQ ดังนั้นในกรณีที่คำแนะนำไม่ดี หลังจากที่เอเจนต์ได้เรียนรู้การกระทำที่ไม่ดีไปแล้ว ในการเรียนรู้ครั้งถัดไปหากเอเจนต์ได้กลับมายังสถานะนี้อีกครั้ง เอเจนต์จะเลี่ยงตัวเลือกที่เกิดจากแนะนำดังกล่าวโดยอัตโนมัติ และทางตรงกันข้ามหากฟังก์ชันฮิวริสติกให้คำแนะนำในตัวเลือกที่ดี รางวัลที่ได้รับย่อมมีมูลค่าที่ดีตามไปด้วย ดังนั้นเมื่อเอเจนต์กลับมายังสถานะนี้อีกครั้ง จะมีโอกาสในการเลือกกรีดการกระทำที่ดีนี้อีกครั้งด้วย และในส่วนของการคำนวณคำแนะนำ ไม่ว่าจะนำมาใช้แนะนำหรือไม่ใช้แนะนำให้กับเอเจนต์ก็ตาม หากเป็นการคำนวณมาจากฟังก์ชันฮิวริสติก เอเจนต์ก็ควรจะสำรวจการกระทำนั้นเพื่อการเรียนรู้ผลของรางวัลด้วยลำดับความสำคัญสูงสุด ซึ่งดีกว่าการสุ่มสำรวจโดยปราศจากความรู้ใดๆ ซึ่งคำแนะนำที่เลือกมาช่วยคำนวณยังสามารถปรับเปลี่ยนคำแนะนำให้กับอัลกอริธึม RL ได้ตามความเหมาะสมโดยที่ไม่กระทบกับกระบวนการปรับปรุงฟังก์ชันมูลค่าV และฟังก์ชันมูลค่าQ แต่อย่างใด

4.2 ข้อจำกัดของการใช้งานอัลกอริธึมการเรียนรู้ Q ด้วยการแนะนำจากฟังก์ชันฮิวริสติก (HAQL) [3]

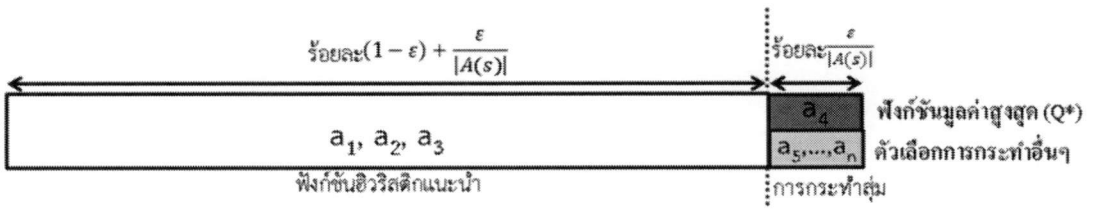
กระบวนการเรียนรู้แบบเสริมกำลังที่ใช้อัลกอริธึม HAQL มีจุดประสงค์เพื่อแนะนำตัวเลือกการกระทำที่คาดว่าให้ผลคำตอบที่ดีจากการคำนวณปัญหาโจทย์ด้วยฟังก์ชันฮิวริสติกนำมาสร้างเป็นชุดของการกระทำที่แนะนำบนสถานะหนึ่งๆ ดังนั้นในเลือกใช้ฟังก์ชันฮิวริสติกส่งผลโดยตรงต่อคำตอบ หากประสิทธิภาพในการคำนวณสามารถแนะนำการกระทำที่ลู่เข้าสู่คำตอบที่ดีที่สุด (Optimal Solution) ได้ การเรียนรู้ที่ใช้การแนะนำนั้นจะส่งผลให้การเรียนรู้ลู่เข้าสู่ค่าที่เหมาะสมได้เร็วขึ้น แต่หากประสิทธิภาพในการคำนวณที่แนะนำให้เอเจนต์เลือกการกระทำที่ดีเฉพาะจุด (Local Optima) เอเจนต์จะมีโอกาสน้อยมากที่จะค้นหาพบคำตอบที่เหมาะสมเนื่องจาก เนื่องจากเอเจนต์จะกรีดฟังก์ชันมูลค่าเข้าสู่คำตอบที่อยู่ในปริภูมิเฉพาะที่ตามที่ฟังก์ชันฮิวริสติกแนะนำ ในกรณีของ HAQL

จากในสมการ 3.1 สามารถนำเสนอพจน์ของโพลิซี $\pi(s_t)$ เป็นหลักการรูเล็ทวีล (Roulette Wheel Principle) ได้ดังรูปที่ 4.1



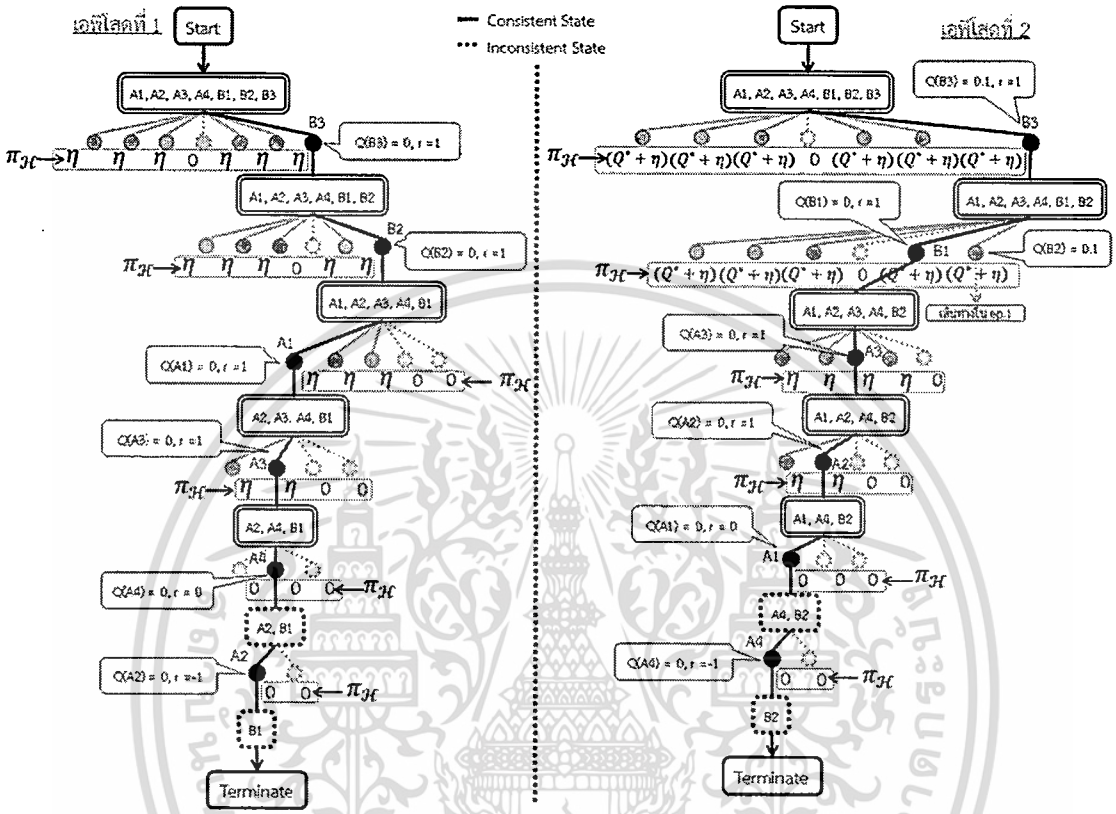
รูปที่ 4.1 หลักการรูเล็ทวีลของอัลกอริธึม HAQL

เมื่อพิจารณาจากตัวเลือกการกระทำ ณ สถานะหนึ่ง ตัวเลือกที่เป็นการแนะนำหากคำนวณด้วยสมการ 3.1 นั้น จะเกิดค่าของความน่าจะเป็นในการเลือกตัวเลือกที่แนะนำสูงกว่าค่าที่เป็นฟังก์ชันมูลค่าสูงสุด Q^* อยู่ η หน่วย ซึ่งหมายถึงมีโอกาสสูงที่เอเจนต์จะเลือกสำรวจการกระทำที่ฮิวริสติกแนะนำ และมีโอกาสถูกเลือกมากกว่าการกระทำที่มีฟังก์ชันมูลค่าสะสมสูงสุดที่ไม่ได้รับการแนะนำเสมอ ทำให้ในทุกๆ ครั้งที่ใช้ฟังก์ชันฮิวริสติกที่แนะนำหากมีตัวเลือกการกระทำที่ไม่ดีอยู่ในชุดของคำแนะนำ จะมีโอกาสค่อนข้างมากที่เอเจนต์จะเลือกการกระทำที่ไม่ดีนี้อีกครั้ง เพราะโพลิซีแบบกรีดีจะทำให้เอเจนต์สนใจตัวเลือกที่มีค่า $Q^*+\eta$ โดยไม่ได้พิจารณาถึงน้ำหนักของฟังก์ชันมูลค่า Q ที่ไม่ดี ถือว่าเป็นการบังคับให้กระทำโดยไม่ได้เรียนรู้ และหากนำเสนอโพลิซีการเลือกการกระทำในลักษณะข้างต้น(จากรูปที่ 4.1) ด้วยโพลิซีแบบ ε-greedy จะได้ว่า ความน่าจะเป็นร้อยละ $(1-\varepsilon)+\frac{\varepsilon}{|A(s)|}$ จะเป็นการกรีดีการกระทำที่ฟังก์ชันฮิวริสติกแนะนำ และความน่าจะเป็นร้อยละ $\frac{\varepsilon}{|A(s)|}$ เป็นการเลือกสำรวจการกระทำอื่นๆที่สามารถเลือกได้ ซึ่งรวมถึงอาจมีการกระทำที่มีฟังก์ชันมูลค่าสูงสุดปะปนอยู่ด้วย ดังรูปที่ 4.2



รูปที่ 4.2 แสดงโพลิซีกรีดี εของอัลกอริธึมHAQL

หากเอเจนต์กลับมายังสถานะเดิมอีกครั้ง จะพบว่าตัวเลือกที่ไม่ถูกแนะนำจากฟังก์ชันฮิวริสติกมีโอกาสน้อยมากที่จะถูกเลือก ตลอดจนตัวเลือกที่ผลรางวัลที่ไม่ดีแต่ถูกแนะนำจะไม่ถูกลดน้ำหนักในการพิจารณาให้น้อยลงแต่อย่างใด ดังนั้นน้ำหนักของฟังก์ชันมูลค่าที่เอเจนต์ได้จดจำมาจะไม่ถูกนำพิจารณาในการเลือกการกระทำ



รูปที่ 4.3 แผนภาพแบคอัพของการเรียนรู้ปัญหาการเลือกโปรโตไทป์ของข้อมูลตัวอย่างด้วยอัลกอริธึม HAQL

หากยกตัวอย่างโพลิซีของอัลกอริธึม HAQL ในการแก้ปัญหาการเลือกโปรโตไทป์กับชุดข้อมูลตัวอย่าง และใช้โพลิซี $\pi_{\mathcal{H}}$ เป็นไปตามสมการ 3.1 ปริภูมิการค้นหาที่เก็บอยู่ในรูปของฟังก์ชันมูลค่าจะถูกสร้างเป็นแผนภาพแบคอัพ จะพบว่า หากการเรียนรู้ของเอเจนต์บนสถานะใดๆ ฟังก์ชันฮิวริสติกจะใช้การแนะนำที่มีผลรวมเท่ากับ $Q^* + \eta$ ไม่ว่าเอเจนต์จะเคย หรือไม่เคยได้เรียนรู้การกระทำนั้นๆก็ตาม เมื่อเริ่มต้นการเรียนรู้ในเอพิโสดที่ 1 (รูปที่ 4.3 ซ้าย) หากเอเจนต์ยังไม่เคยได้เรียนรู้บนสถานะหนึ่งๆ ฟังก์ชันมูลค่าสูงสุดบนสถานะนั้นจะมีค่าเท่ากับศูนย์ $(Q^* = 0)$ ส่งผลให้ตัวเลือกการกระทำ $a_{\mathcal{H}}$ ใดๆ จะมีค่าโพลิซี $\pi_{\mathcal{H}}$ เท่ากับ η และจะเป็นเช่นนี้ตลอดการเรียนรู้ที่เป็นการเลือกสำรวจไปบนสถานะที่ไม่ทราบค่ามาก่อน หากเอเจนต์ทำการกรีดมูลค่าสูงสุดไม่ว่าจะตัวเลือกใดจะพบว่า น้ำหนักของตัวเลือกที่แนะนำจะมีค่าสูงสุดที่สูงกว่าQ^* นั่นก็คือเป็นการสุ่มเลือกการกระทำจากกลุ่มการกระทำที่แนะนำโดยฟังก์ชันฮิวริสติก(การกระทำที่ไม่ได้แนะนำจะไม่โดนเลือกเลย) เมื่อจบการเรียนรู้ในเอพิโสดที่ 1 จะสังเกตว่าเมื่อเอเจนต์ทำการพิจาณาตัวเลือกในสถานะเริ่มต้นของเอพิโสดที่ 2 (รูปที่ 4.3 ขวา) เมื่อพิจารณาน้ำหนักของการกระทำที่แนะนำด้วยฟังก์ชันฮิวริสติก จะเกิดตัวเลือกที่แนะนำด้วยอัตราส่วนความน่าจะเป็นของน้ำหนักที่เท่ากับน้ำหนักของตัวเลือกในเอพิโสดที่

1 เช่นเดิม แตกต่างกันตรงที่ฟังก์ชันมูลค่าสูงสุดเริ่มมีค่ามากกว่าศูนย์เท่านั้น $(Q^* > 0)$ แต่การกระทำที่ฟังก์ชันฮิวริสติกแนะนำจะมีค่าเพิ่มเป็น $Q^* + \eta$ ทุกตัวเลือก ดังนั้นจึงมีโอกาสค่อนข้างมากที่จะสุ่มเลือกการกระทำอื่นๆ ที่ฮิวริสติกแนะนำ และมีโอกาสที่เอเจนต์จะกลับมาเลือกตัวเลือกการกระทำในเส้นทางการค้นหาครั้งก่อนได้น้อยมาก เช่นดังรูปที่ 4.3 ขวา ในสถานะเริ่มต้นเอเจนต์อาจจะสามารถเลือกการกระทำเดิมได้อยู่แต่เมื่อเปลี่ยนสถานะเข้าสู่สถานะที่ 2 ด้วยน้ำหนักการแนะนำจะทำให้เอเจนต์เลือกเส้นการค้นหาในทิศทางอื่นและจะพบว่าคำตอบทั้งสองเอพิโสดนั้นไม่ใช่คำตอบเดียวกันและไม่ได้ใช้เส้นทางการค้นหาคำตอบจากความรู้ที่มีอยู่ก่อนแต่อย่างใด เป็นเพียงการค้นหาด้วยการสุ่มสำรวจปริภูมิของฟังก์ชันฮิวริสติกเท่านั้น

ข้อจำกัดอีกประการหนึ่งของอัลกอริธึม HAQL คือ การนำฟังก์ชันฮิวริสติกมาใช้ ณ เอพิโสดที่ได้กำหนดไว้ เช่น ในทุกๆ 10 หรือ 100 เอพิโสด ดังนั้นผลลัพธ์จากการแนะนำหากพิจารณาปัจจัยตามประสิทธิภาพการแนะนำจะขึ้นกับความถี่ในการใช้งานคำแนะนำของฟังก์ชันฮิวริสติกที่ได้กำหนดคงที่เอาไว้ ซึ่งมีแนวทางของคำตอบในกรณี

- การใช้การแนะนำจากฟังก์ชันฮิวริสติกที่ประสิทธิภาพ “ไม่ดี” บ่อยครั้ง สามารถทำให้ผลของคำตอบในปริภูมิคำตอบที่ดีเฉพาะจุด เนื่องจากฟังก์ชันฮิวริสติกจะแนะนำลู่เข้าสู่ปริภูมิคำตอบดังกล่าวอยู่เสมอ ส่วนการใช้คำแนะนำจากฟังก์ชันฮิวริสติกที่ประสิทธิภาพ “ไม่ดี” น้อยครั้ง เอเจนต์จะยังคงใช้ความสามารถในการค้นหาหลักจากการสุ่มสำรวจของอัลกอริธึมRL ปกติ
- การใช้การแนะนำจากฟังก์ชันฮิวริสติกที่ประสิทธิภาพ “ดี” น้อยครั้ง ผลการเรียนรู้ก็อาจเป็นคำตอบที่ในปริภูมิคำตอบที่ดีเฉพาะจุดได้ เนื่องจากความถี่ในการใช้งานน้อยครั้งเกินไปและใช้งานไม่ต่อเนื่องเท่าที่ควร เอเจนต์จึงไม่สามารถพิจารณาตัวเลือกการกระทำที่ถูกแนะนำนี้ให้สะสมอยู่ในรูปของฟังก์ชันมูลค่าได้ เนื่องจากการปรับปรุงฟังก์ชันมูลค่าจะถูกสะสมครั้งละน้อยๆ ไม่สูงขึ้นอย่างทันทีทันใดจนสังเกตได้ ซึ่งเป็นข้อจำกัดภายในกลไกของกระบวนการปรับปรุงฟังก์ชันมูลค่าของอัลกอริธึมRLปกติ[15] และหากการใช้การแนะนำจากฟังก์ชันฮิวริสติกที่ประสิทธิภาพ “ดี” บ่อยครั้งความสามารถในการค้นหาของเอเจนต์มีลักษณะของการค้นหาแบบสุ่มบนปริภูมิที่ฮิวริสติกแนะนำ ทำให้น้ำหนักในการเรียนรู้ที่เคยสะสมไว้ไม่ถูกนำมาคิดคำนวณ

4.3 กระบวนการปรับปรุงประสิทธิภาพในการเรียนรู้แบบเสริมกำลังที่นำเสนอ

ในการเพิ่มประสิทธิภาพให้กับอัลกอริธึม RL หากทำการประยุกต์ความสามารถในการค้นหาแบบโลคอลของอัลกอริธึม RL-RCS ที่สามารถค้นหาคำตอบที่เหมาะสมได้ [4] รวมเข้ากับความสามารถในการแนะนำกลุ่มของการกระทำที่ดีที่เกิดจากคำแนะนำของฟังก์ชันฮิวริสติกจากอัลกอริธึม HAQL [3] จะสามารถเพิ่มประสิทธิภาพของการเรียนรู้คำตอบในปริภูมิแบบโลคอลในเส้นทางการค้นหาที่ถูกแนะนำด้วยฟังก์ชันฮิวริสติก ซึ่งการนำส่วนที่มีประสิทธิภาพของทั้ง 2 อัลกอริธึมมารวมเข้าด้วยกัน มีจุดมุ่งหมายเพื่อให้เอเจนต์เรียนรู้ในการเลือกการกระทำอย่างมีเหตุมีผลตามที่ฟังก์ชันฮิวริสติกแนะนำ เพื่อลดขั้นตอนในการลองผิดลองถูกด้วยความน่าจะเป็นสุ่ม อีกทั้งเมื่อคำตอบที่ทำการแก้ปัญหาเกิดความผิดพลาดตามเกณฑ์ที่กำหนดไว้(คาดว่าอยู่ในปริภูมิคำตอบที่ดี

เฉพาะจุด) เอเจนต์ยังสามารถกลับมายังสถานะที่ทำการบันทึกไว้เพื่อให้โอกาสในการเรียนรู้ตัวเลือกการกระทำอื่นๆ ที่เอเจนต์ไม่เคยผ่านไปแต่เป็นตัวเลือกที่ฟังก์ชันฮิวริสติกแนะนำไว้

ปัญหาที่พบจากการทดสอบอัลกอริธึม HAQL-RCS คือ หลักเกณฑ์ในการใช้งานการแนะนำจากฟังก์ชันฮิวริสติกที่ถูกกำหนดเป็นความถี่การใช้การแนะนำเป็นหน่วยเอพิโสด หากมีค่าน้อยเกินไปคำตอบที่เหมาะสมจะถูกค้นพบจากประสิทธิภาพของอัลกอริธึม RL-RCS เพียงอย่างเดียว และหากทำการปรับปรุงความถี่ในการแนะนำด้วยหน่วยของเอพิโสดให้บ่อยครั้งขึ้น ประสิทธิภาพในการค้นหาคำด้วยอัลกอริธึม RL จะลดประสิทธิภาพลง เนื่องจากอัลกอริธึม HAQL จะก่อให้เกิดความน่าจะเป็นของการสำรวจปริภูมิการค้นหาที่ไม่ใช่เส้นทางการค้นหาเดิมตลอดเวลา ทำให้คำตอบที่ได้เป็นคำตอบที่เกิดจากการค้นหาแบบสุ่ม (บนปริภูมิของฟังก์ชันฮิวริสติก) ซึ่งไม่ใช่ลักษณะการค้นหาของการเรียนรู้แบบเสริมกำลัง เท่ากับว่าน้ำหนักของการเรียนรู้ที่ผ่านมาไม่สามารถนำกลับมาช่วยคำนวณได้เลย

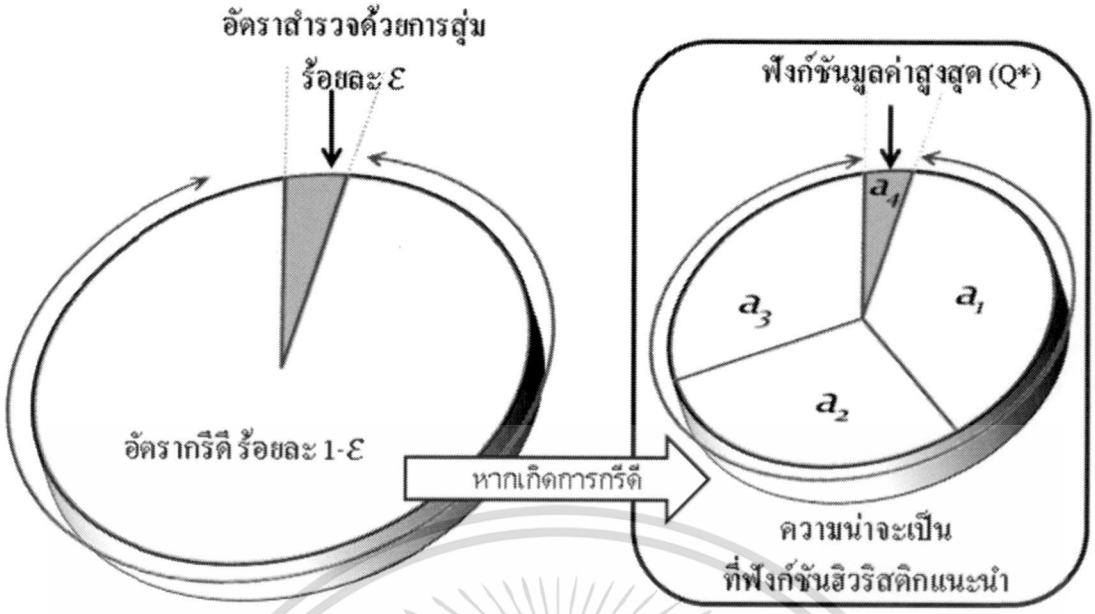
ดังนั้นเพื่อเพิ่มประสิทธิภาพอัลกอริธึมอย่างแท้จริงจำเป็นต้องแก้ไขผลกระทบของการประยุกต์อัลกอริธึมทั้งสองข้างต้น ด้วยปัจจัยต่างๆตามลักษณะของการเรียนรู้ ซึ่งสามารถแบ่งออกเป็นส่วนต่างๆดังเช่น ตัวเลือกที่เคยเรียนรู้มาก่อนจะต้องมีน้ำหนักที่มีนัยสำคัญสูงกว่า และคำแนะนำที่ต้องสามารถปรับค่าน้ำหนักได้เพื่อให้เอเจนต์สามารถรู้จำตัวเลือกที่เคยเรียนรู้มาก่อนหน้าแล้วได้

4.3.1 โพลิซีแบบซอร์ฟแมกซ์ (Softmax Policy)

เนื่องจากปัญหาขนาดใหญ่ที่มีสถานะจำนวนมาก พบว่าลักษณะของการกรีดฟังก์ชันมูลค่าสูงสุดเพียงค่าเดียว (ดังที่กล่าวในหัวข้อ 3.1) ด้วยโพลิซีแบบกรีดี ε จะส่งผลให้กลุ่มของการกระทำที่มีฟังก์ชันมูลค่าสูงสุดค่าเดียวนั้นจะถูกกระจายด้วยความน่าจะเป็นที่ด้วยโอกาสเท่าๆกัน และปัญหาดังกล่าวได้ส่งผลต่ออัลกอริธึม HAQL เช่นกัน หากพิจารณารูปที่ 4.1 และรูปที่ 4.2 พบว่าน้ำหนักของตัวเลือกอื่นๆ ที่ให้ผลรางวัลที่ดีเช่นกันแต่ฟังก์ชันมูลค่าที่สะสมไว้มีค่าน้อยกว่า จะไม่ถูกนำมาพิจารณาในการลงคะแนนสูงสุด(Vote) จำเป็นต้องอาศัยการสุ่มสำรวจของเอเจนต์ด้วยอัตราร้อยละ ε เพื่อให้เส้นทางที่คาดว่าจะได้รับผลรางวัลที่ดีนี้กลับมามีน้ำหนักเพิ่มขึ้น ซึ่งในความเป็นจริงเกือบจะไม่มีโอกาสเลยที่ตัวเลือกดังกล่าวจะถูกเลือกอีกครั้ง เนื่องจากช่วงของอัตราสุ่มร้อยละ ε ย่อมมีตัวเลือกอื่นๆ ที่ยังไม่เคยถูกสำรวจหรือตัวเลือกที่ให้ผลรางวัลที่ไม่ดี ที่ถูกแบ่งเป็นอยู่ในช่วงดังกล่าวด้วยอัตราสัดส่วนเท่าๆกัน ดังนั้นจึงประยุกต์ใช้โพลิซีแบบซอร์ฟแมกซ์[15] เพื่อกระจายโอกาสในการถูกเลือกด้วยความน่าจะเป็นให้กับการกระทำที่ดีในอันดับรองลงมาตามสมการ 4.1

$$\frac{Q(s,a)^{1.4}}{\sum_{b=1}^{n} Q(s,b)^{1.4}} \text{ ; สำหรับทุกๆ ค่า } b \text{ ที่มีค่า } Q(s,b) > 0 \qquad (4.1)$$

โพลิซีแบบซอร์ฟแมกซ์จากสมการ 4.1 นี้จะแตกต่างจากการใช้การแจกแจงแบบกิบส์เนื่องจากใช้การสำรวจร้อยละ ε เพื่อควบคุมการเลือกการกระทำเช่นเดียวกับโพลิซีแบบกรีดี ε แต่จะพิจาณาร่วมกับตัวเลือกอื่นๆที่มีฟังก์ชันมูลค่า Q ในอันดับรองลงมาจากค่าที่ดีที่สุดตามสัดส่วน สามารถนำเสนอด้วยหลักการรูเล็ทวีลได้ดังรูปที่ 4.4



รูปที่ 4.4 หลักการรูเล็ทวีลด้วยโพลิซีแบบซอร์ฟแมกซ์

4.3.2 วิธีการปรับน้ำหนักของคำแนะนำสำหรับฟังก์ชันฮิวริสติก

เพื่อเป็นการหลีกเลี่ยงปัญหาการเลือกการกระทำที่ไม่ดีซ้ำๆ เนื่องจากคำแนะนำของฟังก์ชันฮิวริสติกที่ลู่เข้าสู่ปริภูมิคำตอบที่ดีเฉพาะจุด หรือการสำรวจเส้นทางใหม่ของฟังก์ชันฮิวริสติกตลอดเวลา สามารถปรับปรุงได้ด้วยการประยุกต์ใช้การปรับน้ำหนักคำแนะนำของฟังก์ชันฮิวริสติกวัตถุประสงค์เพื่อให้เอเจนต์นำผลของฟังก์ชันมูลค่าQ จากการกระทำที่เคยได้เรียนรู้มาก่อนหน้ารวมเข้ากับค่าน้ำหนักของคำแนะนำจากฟังก์ชันฮิวริสติก$\mathcal{H}$ ดังนั้นค่าของความน่าจะเป็นในการถูกเลือกในแต่ละการกระทำจะมีค่ามากน้อยไม่เท่ากันขึ้นกับการกระทำนั้นๆว่าเอเจนต์เคยเรียนรู้มาก่อนหรือไม่ หากเรียนรู้มาก่อนแล้วผลของการเรียนรู้ดีหรือไม่ดีอย่างไร ผลรวมข้างต้นมีผลทำให้เอเจนต์มีโอกาสที่จะเลือกการกระทำเดิมที่ดีได้อีกครั้งเพื่อปรับปรุงเส้นทางการค้นหาคำตอบที่ดีเพื่อให้ฟังก์ชันมูลค่าถูกสะสมจนเด่นชัดขึ้น และในส่วนของการกระทำที่แนะนำแต่ให้ผลของรางวัลที่ไม่ดีซึ่งเอเจนต์ได้เคยเรียนรู้ผ่านมาแล้ว จะถูกลดทอนน้ำหนักคำแนะนำลงด้วยฟังก์ชันมูลค่าQที่ไม่ดี(มีค่าเป็นลบ) ดังนั้นตัวเลือกที่ไม่ดีจึงมีโอกาสในการถูกเลือกน้อยกว่ากลุ่มคำแนะนำที่ดี

ดังนั้นการปรับน้ำหนักของการแนะนำด้วยฟังก์ชันฮิวริสติก ในพจน์ของฟังก์ชัน$\mathcal{H}(s_t, a_t)$ จากสมการ 3.2 ซึ่งจากเดิมที่ใช้ผลลบของพจน์ $Q(s, a)$ เพื่อนำไปหักล้างกับพจน์ $Q(s, a)$ ในสมการที่ 3.1 จึงมีการปรับกลไกในการเพิ่มลดน้ำหนักด้วยเงื่อนไขว่า หากเป็นตัวเลือกการกระทำที่เกิดจากคำแนะนำและถ้าเอเจนต์ไม่เคยสำรวจมาก่อนให้กำหนดเป็นค่าน้อยๆ เพื่อให้เอเจนต์สามารถสังเกตน้ำหนักในการตัดสินใจเลือกได้บ้าง แต่หากตัวเลือกที่แนะนำเหล่านั้นเคยถูกสำรวจตัวแล้ว ให้ปรับน้ำหนักด้วยอัตราส่วนของฟังก์ชันมูลค่าที่เหมาะสม ซึ่งเกิดจากการคำนวณน้ำหนักด้วยหลักการลูทเล็ทวีลบนชุดของคำแนะนำ ผลของการปรับปรุงเป็นดังสมการที่ 4.2

$$\mathcal{H}(s_t, a_t) = \begin{cases} \eta, & \text{กรณี } a_t = \pi^{\mathcal{H}}(s_t) \text{ และ } Q(s_t, a_t) = 0 \\ \frac{W_{a_t}}{\sum_{i=1}^{\pi^{\mathcal{H}}} W_i} \times |\pi^{\mathcal{H}}| \times Q^*(s_t, a), & \text{กรณี } a_t = \pi^{\mathcal{H}}(s_t) \text{ และ } Q(s_t, a_t) \neq 0 \\ 0, & \text{กรณี } a_t \neq \pi^{\mathcal{H}}(s_t) \end{cases} \quad (4.2)$$

โดยที่ $\frac{W_{a_t}}{\sum_{i=1}^{\pi^{\mathcal{H}}} W_i}$ หมายถึง อัตราส่วนของคำแนะนำให้เลือกการกระทำ a_t ต่อคำแนะนำให้เลือกการกระทำอื่นๆ

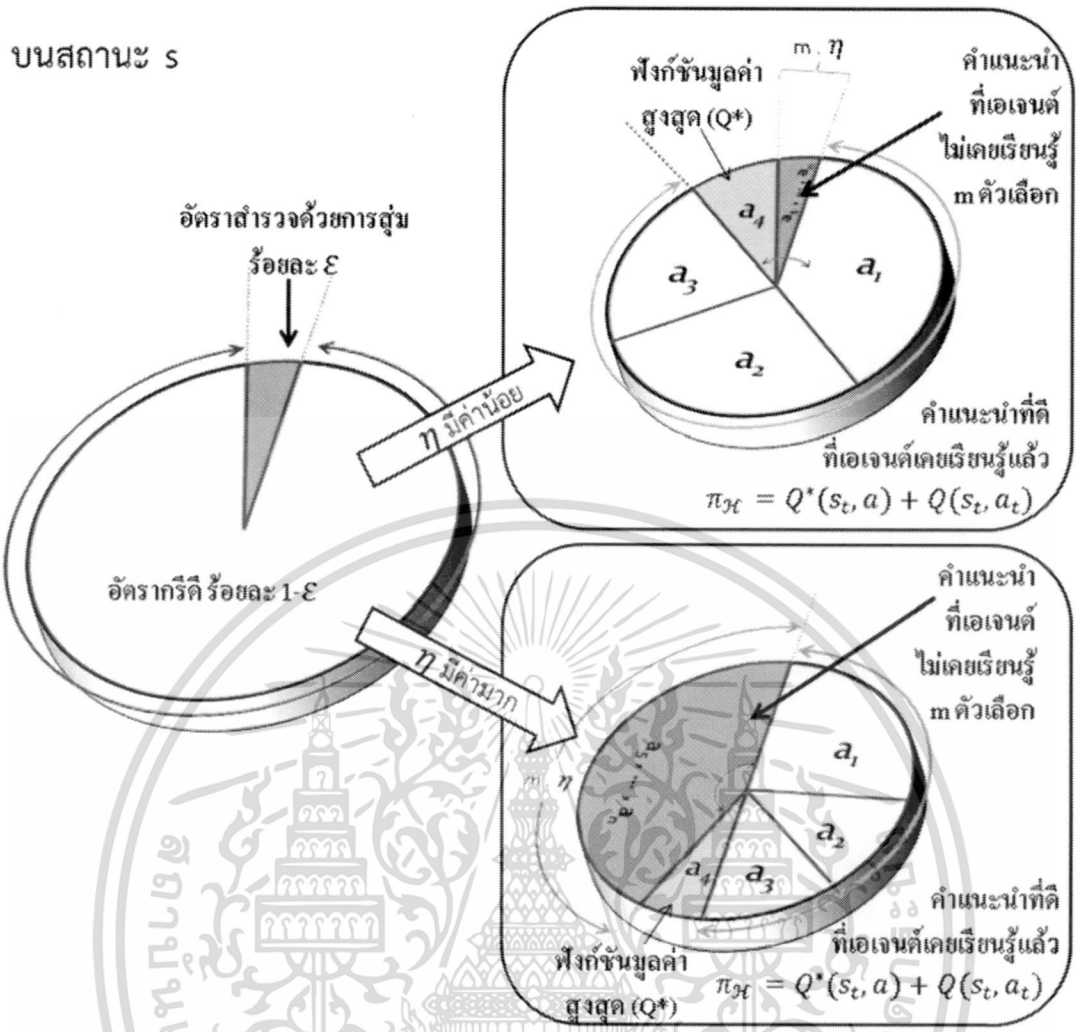
$|\pi^{\mathcal{H}}|$ หมายถึง จำนวนของคำแนะนำที่มีบนสถานะปัจจุบัน s_t

ฟังก์ชัน $\mathcal{H}(s_t, a_t)$ จะมีค่าก็ต่อเมื่อในแต่ละตัวเลือกนั้นฟังก์ชันฮิวริสติกสามารถคำนวณคำแนะนำได้ และในกรณีที่เอเจนต์ยังไม่เคยเรียนรู้ตัวเลือกที่กำลังแนะนำ(ฟังก์ชันมูลค่า Q เป็นศูนย์) ตัวเลือกดังกล่าวจะถูกกำหนดน้ำหนักของคำแนะนำด้วยค่าเริ่มต้นเท่ากับ η ซึ่งหมายถึงการเพิ่มโอกาสในการถูกเลือกของตัวเลือกที่แนะนำแต่เอเจนต์ยังไม่ทราบฟังก์ชันมูลค่าQ โดยโอกาสเล็กน้อย

ในกรณีที่เอเจนต์เคยเรียนฟังก์ชันมูลค่าQของตัวเลือกนั้นๆมาแล้ว(ฟังก์ชันมูลค่าQ มีค่าไม่เท่ากับศูนย์) จะกำหนดน้ำหนักของคำแนะนำด้วยขนาดของฟังก์ชันมูลค่าQที่เหมาะสม $\mathrm{Q}^*(s_t, \mathrm{a})$

ดังนั้นเมื่อพิจารณาการกรีดผลรวมของฟังก์ชันมูลค่าQ และคำแนะนำจากฟังก์ชันฮิวริสติก $\mathcal{H}$ (โพลิซี $\pi_{\mathcal{H}}$) จากสมการ 3.1 พบว่าน้ำหนักที่นำมาคำนวณด้วยหลักการรูเล็ทวีล มีผลรวมของการกระทำสามารถถูกเลือกได้มีค่าเท่ากับ $Q(s,a) + \mathcal{H}(s,a)$ ส่วนการกระทำที่ไม่ถูกแนะนำผลรวมที่อยู่ในรูเล็ทวีลจะถูกคำนวณด้วยฟังก์ชันมูลค่าQ เพียวอย่างเดียว ส่งผลให้โพลิซี $\pi_{\mathcal{H}}$ ที่ไม่ดี (คำแนะนำที่ผลรวม $Q(s,a) + \mathcal{H}(s,a)$ และฟังก์ชันมูลค่าQ ของกระทำที่ไม่ถูกแนะนำมีค่าเป็นลบ) จะไม่อยู่ในกลุ่มที่จะถูกเลือกด้วยความน่าจะเป็น $(1-\varepsilon) + \frac{\varepsilon}{|A(s)|}$ ดังนั้นฟังก์ชันมูลค่าQ ที่เคยเรียนรู้มาก่อนจะนำอัตราส่วนของฟังก์ชันมูลค่าสูงสุด Q^* มาเพิ่มเป็นน้ำหนักคำแนะนำ จึงเพิ่มโอกาสที่จะเลือกถูกให้มากขึ้น ส่วนการกระทำที่แนะนำแต่เอเจนต์ไม่เรียนรู้จะถูกแนะนำด้วยค่า η เนื่องจากยังไม่มีผลของฟังก์ชันมูลค่าQของการกระทำนี้ได้บันทึกไว้ ส่วนกรณีที่เป็นการกระทำที่ไม่แนะนำก็ควรกำหนดให้ผลรวมของโพลลิซี $\pi_{\mathcal{H}}$ มีมูลค่าเพียงแค่ฟังก์ชันมูลค่า Q ส่วนค่าของ η จะถูกกำหนดให้มีค่ามากหรือน้อยนั้นจะขึ้นกับจุดประสงค์ว่าจะให้เอเจนต์เลือกเส้นทางการค้นหาเดิมบ่อยครั้งเพียงใด

หากต้องการให้สัดส่วนในการเลือกการกระทำเดิมที่ดีเป็นหลักควรกำหนดให้ η มีมูลค่าน้อยๆ (ในที่นี้กำหนดให้มีค่าที่เป็นบวกน้อยที่สุดที่ชนิดของข้อมูลสามารถคำนวณได้) ส่วนการกำหนดค่า η ที่สูงจะส่งผลให้เอเจนต์เลือกทำการสำรวจตัวเลือกที่ฟังก์ชันฮิวริสติกแนะนำแต่เอเจนต์ยังไม่เคยได้เรียนรู้ด้วยอัตราส่วนที่สูงขึ้น ทั้งนี้ควรกำหนดมีค่าไม่สูงกว่าฟังก์ชันมูลค่าสูงสุด ณ ขณะนั้น เนื่องจากเอเจนต์อาจใช้พฤติกรรมการค้นหาแบบสุ่มไปโดยปริยาย ซึ่งเอเจนต์จะไม่ใช่น้ำหนักของการเรียนรู้ที่ผ่านมาช่วยตัดสินใจ (ประสิทธิภาพเทียบเท่ากับอัลกอริธึม HAQL) จึงควรเลือกปรับค่าตามความเหมาะสม ดังรูปที่ 4.5



รูปที่ 4.5 หลักการกรีดด้วยโพลิซีแบบซอร์ฟแมกซ์

เมื่อไรก็ตามที่เอเจนต์กลับมายังสถานะที่เคยผ่านมาแล้วตัวเลือกการกระทำหนึ่งๆ มีค่าของฟังก์ชันมูลค่าQเกิดขึ้น เอเจนต์จะทำการปรับน้ำหนักมูลค่าของฟังก์ชันฮิวริสติก $\mathcal{H}(s_t, a_t)$ ในแต่ละตัวเลือกด้วยน้ำหนักของการแนะนำในตามหลักการรูเล็ทวีล(รูปที่ 4.5) และเมื่อพิจารณาพจน์ของโพลิซี $\pi(s_t)$ สมการ 3.1 ผลลัพธ์จากการคำนวณจะได้ค่าน้ำหนักของการแนะนำมีค่าเป็นสัดส่วนของฟังก์ชันมูลค่าสูงสุดQ^*

4.3.3 อัลกอริธึมการเรียนรู้แบบเสริมกำลังที่ใช้การเรียนรู้จากทางเลือกของฮิวริสติก

เมื่อประยุกต์วิธีการแก้ปัญหาด้วยโพลิซีแบบซอร์ฟแมกซ์และการปรับน้ำหนักของการแนะนำสำหรับฟังก์ชันฮิวริสติก เข้ากับอัลกอริธึม HAQL-RCS จะทำให้อัลกอริธึมที่ได้มีลักษณะของการเรียนรู้ตัวเลือกการกระทำที่ฟังก์ชันฮิวริสติกแนะนำ จากตัวเลือกที่เอเจนต์ได้ทำการเรียนรู้ตามคำแนะนำ อีกทั้งยังสามารถรู้จำตัวเลือกที่เคยให้ผลของรางวัลที่ดีหรือไม่ดี อีกทั้งยังมีความสามารถในการย้อนกลับมาเรียนรู้ในสถานะที่ให้ความสนใจได้จนกว่าเอเจนต์จะพบสถานะของคำตอบได้ดีกว่าเกณฑ์ที่กำหนด เป็นการลดจำนวนขั้นของการเริ่มต้นเรียนรู้จากสถานะเริ่มต้นลงไปได้ จึงถือว่าเป็นการปรับปรุงเพื่อเพิ่มประสิทธิภาพให้กับ อัลกอริธึมการเรียนรู้แบบเสริมกำลังโดยใช้การเรียนรู้จาก

ทางเลือกของฮิวริสติก (Learning Choice generated from Heuristic guides, LCH) ที่มีความสามารถในการกลับมายังสถานะที่สามารถเรียนรู้ซ้ำได้ สำหรับซูโดโค๊ดสามารถนำเสนอได้ดังรูปที่ 4.6

กำหนดเริ่มต้นให้กับค่า $Q(s,a) = 0$
คำนวณฟังก์ชันฮิวริสติก $\mathcal{H}(s,a)$ ด้วยวิธีที่เหมาะสม
พิจารณาสถานะปัจจุบัน s
กำหนดสถานะที่สามารถกลับไปเรียนรู้ซ้ำ $s_{retry} \leftarrow s$
ดำเนินการเรียนรู้ซ้ำ (ในแต่ละเอพิโสด)
ดำเนินการซ้ำ (ในขั้นตอนการเรียนรู้)
เลือกการกระทำ a จากคำแนะนำของฟังก์ชันฮิวริสติก$\mathcal{H}(s,a)$ (หัวข้อ 4.3.2)
ด้วยโพลิซีแบบซอร์ฟแมกซ์ (หัวข้อ 4.3.1) ตามสมการ

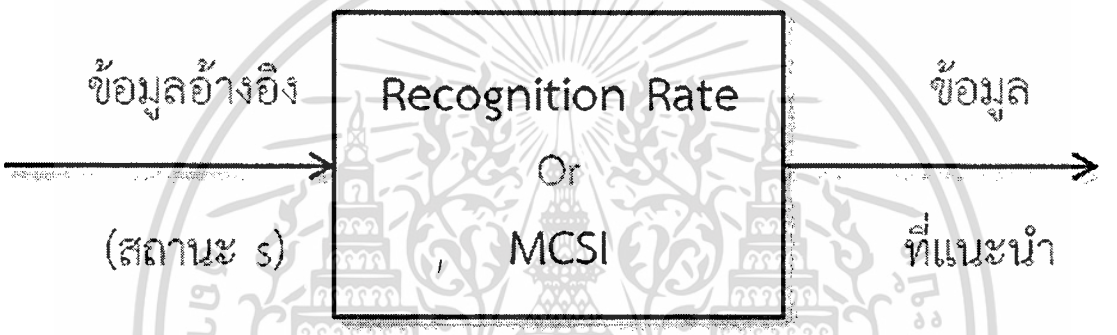
$$\pi(s_t) = \begin{cases} argmax_{a_t}\,[Q(s_t,a_t) + \mathcal{H}_t(s_t,a_t)], & \varepsilon^0 \leq \varepsilon \\ a_{random}, & \varepsilon^0 > \varepsilon \end{cases}$$

กระทำ a พร้อมรับสถานะถัดไป s'
พิจารณาค่า s' กรณีสถานะนั้นสามารถเรียนรู้ซ้ำได้
$r(s,a) = 1$
กำหนดสถานะที่สามารถเรียนรู้ซ้ำด้วยสถานะถัดไป $(s_{retry} \leftarrow s')$
กรณีอื่นๆ :
ถ้า s' เป็นสถานะที่ไม่สามารถอ้างอิงเพื่อเรียนรู้ซ้ำได้หรือสุ่มด้วยความน่าจะเป็นที่น้อยกว่าค่า x
$(s' \leftarrow s_{retry})$
$r(s,a) = -1$
คำนวณฟังก์ชันฮิวริสติก $\mathcal{H}(s,a)$ ด้วยวิธีที่เหมาะสม
ปรับค่าฟังก์ชันมูลค่า $Q(s,a)$ ตามสมการ Watkin's Q(λ)
$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
เคลื่อนไปยังสถานะถัดไป $(s \leftarrow s')$
จนกว่าสถานะ s เป็นสถานะสิ้นสุด
จนกว่าสถานะ s เป็นสถานะสิ้นสุด

รูปที่ 4.6 ซูโดโค๊ดของอัลกอริธึมการเรียนรู้แบบเสริมกำลังโดยใช้การเรียนรู้จากทางเลือกของฮิวริสติก

4.4 การคำนวณเพื่อสร้างฟังก์ชันฮิวริสติกในการแก้ปัญหาการเลือกโปรโตไทป์

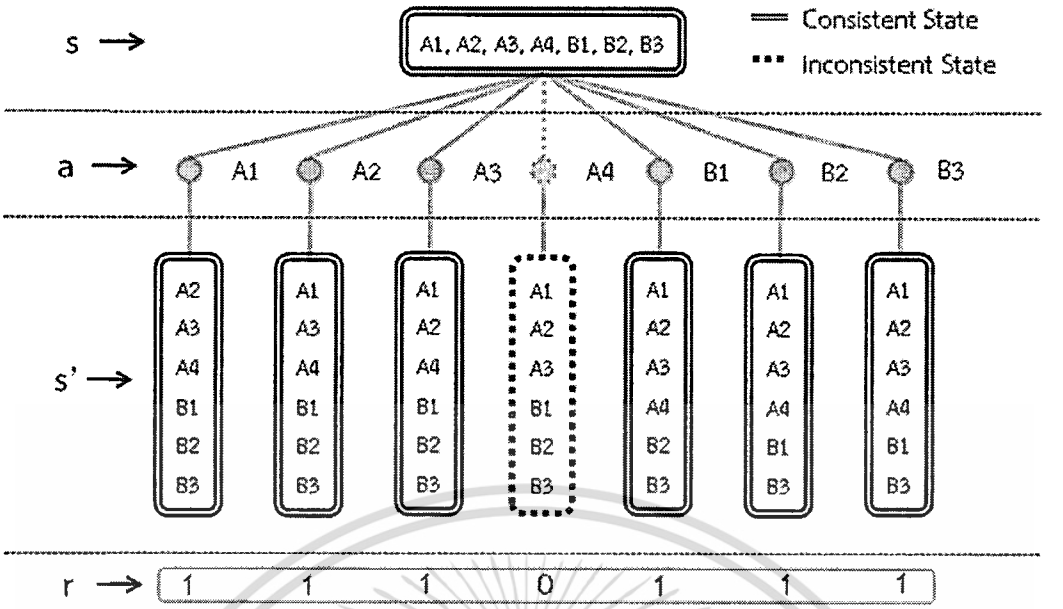
ในการสร้างฟังก์ชันฮิวริสติกเพื่อแนะนำให้กับกระบวนการเรียนรู้แบบเสริมกำลังสามารถประยุกต์แนวคิดต่างๆได้หลายรูปแบบ แต่จำเป็นต้องประยุกต์แนวคิดทั้งหลายเหล่านั้นให้อยู่ในรูปของการกระทำที่แนะนำบนสถานะโจทย์ที่ป้อนเข้าไป จากนั้นจึงนำคำแนะนำที่ได้สร้างเป็นฟังก์ชันฮิวริสติก $\mathcal{H}(s,a)$ เพื่อนำไปคำนวณรวมกับโพลิซีของเอเจนต์ จึงจะได้เป็นโพลิซีแบบมีฮิวริสติก ซึ่งหัวข้อนี้จะเป็นคำอธิบายขั้นตอนในการสร้างคำแนะนำของฟังก์ชันฮิวริสติกดังที่กล่าวในหัวข้อ 2.2 โดยเริ่มจากส่วนของการคำนวณการกระทำที่แนะนำจากฟังก์ชันฮิวริสติกดังรูป 4.7 โดยเลือกใช้แนวคิดในการแก้ปัญหาการเลือกโปรโตไทป์ด้วย 2 เทคนิค คือ เทคนิคที่สร้างจากผลของรางวัลที่เอเจนต์จะได้รับจากการเลือกกระทำหนึ่งๆบนสถานะปัจจุบัน และเทคนิคที่สร้างจากแนวคิดของคุณสมบัติความครอบคลุม



รูปที่ 4.7 แผนผังการดำเนินงานในส่วนของการคำนวณการกระทำที่แนะนำ

4.4.1 เทคนิคที่สร้างจากรางวัลที่เอเจนต์ได้รับจากการเลือกกระทำหนึ่งๆบนสถานะปัจจุบัน

เริ่มต้นจากการนำข้อมูลสถานะปัจจุบัน s ที่ได้มาคำนวณหาการกระทำที่เป็นไปได้ทั้งหมดเพื่อคำนวณหาผลรางวัลของการกระทำ a ด้วยการคำนวณอัตราการค้นคืน(Recognition Rate)ของสถานะถัดไป s' ของแต่ละการกระทำที่เอเจนต์สามารถเลือกได้ ดังรูปที่ 4.8

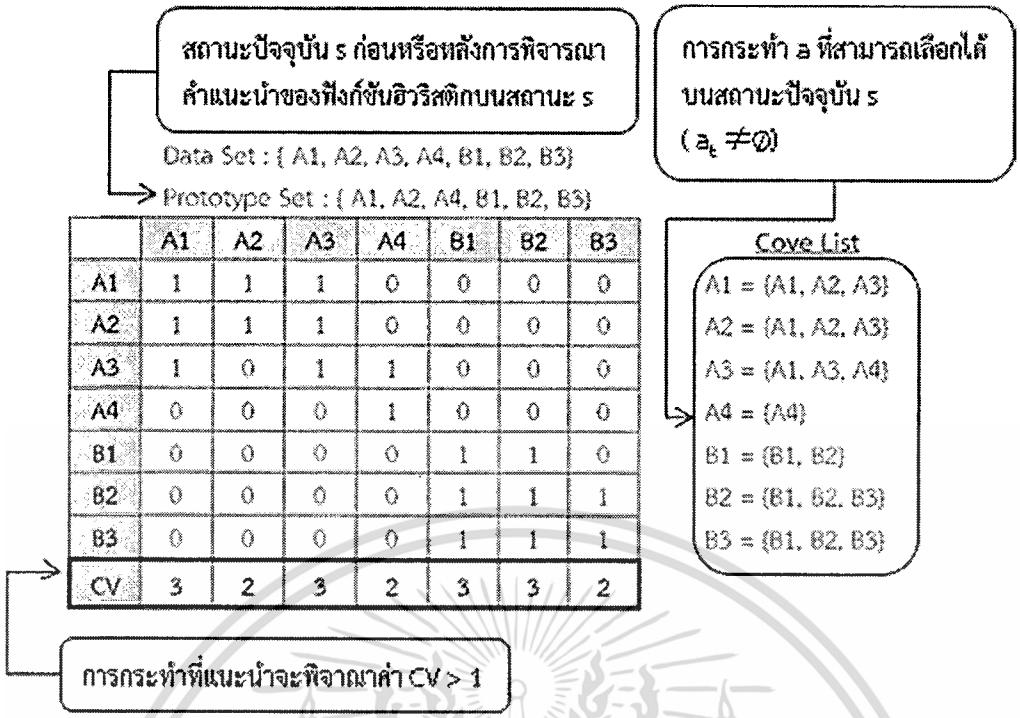


รูปที่ 4.8 การคำนวณหาผลรางวัลในสถานะถัดไป s' ใดๆ

จากตัวอย่างรูปที่ 4.8 จะพบว่ามีเพียงการกระทำ A4 เท่านั้นที่ไม่ได้รับรางวัลหากเอเจนต์จะทำการเลือกเนื่องจากสถานะถัดไป {A1, A2, A3, B1, B2, B3} จะมีผลการค้นคืนที่ไม่สามารถจำแนกข้อมูลตัวอย่างได้สมบูรณ์ (Recognition Rate < 100%) จึงถือว่าบนสถานะ {A1, A2, A3, A4, B1, B2, B3} มีการกระทำที่ฟังก์ชันฮิวริสติกแนะนำประกอบด้วยการกระทำ A1, A2, A3, B1, B2, B3 และหากเอเจนต์ได้เปลี่ยนสถานะไปยังสถานะถัดไปก็จะใช้เทคนิคเช่นเดิมเพื่อคำนวณหารางวัลที่คาดว่าน่าจะได้รับบนสถานะ s'' ต่อไปจนกว่าสถานะที่เปลี่ยนไปเป็นสถานะสิ้นสุด

4.4.2 เทคนิคที่สร้างจากแนวคิดของคุณสมบัติความครอบคลุม

คุณสมบัติความครอบคลุมจะใช้แนวคิดเดียวกับหลักการแก้ปัญหา MCSI ในหัวข้อ 2.3.6 แต่เนื่องจากเป็นการประยุกต์ใช้อัลกอริธึมการแก้ปัญหาการเลือกโปรโตไทป์บนกระบวนการเรียนรู้แบบเสริมกำลังจึงจำเป็นต้องปรับแนวคิดให้สอดคล้องกัน เช่น การกระทำ a ที่เป็นการเลือกข้อมูลออกจะเป็นการกระทำดึงข้อมูล a นั้นๆ ออกทีละหนึ่งข้อมูล (ปกติอัลกอริธึม MCSI จะทำการเลือกข้อมูลออกเป็นกลุ่ม) หรือ กระบวนการเรียนรู้แบบเสริมกำลังจะทำการตรวจสอบอัตราการค้นคืนข้อมูลในทุกๆครั้งที่เปลี่ยนสถานะถัดไปเสมอ ทำให้ตารางความครอบคลุมจะปรับปรุงในทุกๆครั้งที่เอเจนต์ได้เคลื่อนผ่านสถานะ s ไป เป็นต้น



	A1	A2	A3	A4	B1	B2	B3
A1	1	1	1	0	0	0	0
A2	1	1	1	0	0	0	0
A3	1	0	1	1	0	0	0
A4	0	0	0	1	0	0	0
B1	0	0	0	0	1	1	0
B2	0	0	0	0	1	1	1
B3	0	0	0	0	1	1	1
CV	3	2	3	2	3	3	2

รูปที่ 4.9 แสดงวิธีอนุมานอัลกอริธึมการแก้ปัญหา MCSI เป็นกระบวนการเรียนรู้แบบเสริมกำลัง

ดังนั้นสำหรับการอธิบายจะขอยกตัวอย่างตารางความครอบคลุมที่ประยุกต์ให้เข้ากับการอนุมานเป็นคำแนะนำในกระบวนการเรียนรู้แบบเสริมกำลัง ด้วยการใช้วิธีการดำเนินการแก้ปัญหา MCSI ใน 1 รอบการเรียนรู้ด้วยอัลกอริธึมการแก้ปัญหาแบบ MCSI กำหนดให้ทำการเลือกข้อมูลตัวอย่างออกครั้งละ 1 ตัว และทำการคำนวณคุณสมบัติความครอบคลุมใหม่ทุกครั้งที่ขนาดของเซตโปรโตไทป์มีขนาดลดลง เช่นดังรูปที่ 4.9 เซตโปรโตไทป์ที่นำเสนอในตารางความครอบคลุมคือสถานะปัจจุบัน s ของการเรียนรู้แบบเสริมกำลัง ส่วนเซตที่ถูกครอบคลุมด้านขวาคือกลุ่มของการกระทำ a ที่สามารถเลือกได้บนสถานะปัจจุบัน s โดยที่ ข้อมูลที่สามารถเลือกได้จะมีค่า (a ใดๆที่ไม่ใช่เซตว่าง) และค่าของการกระทำที่อัลกอริธึมการแก้ปัญหา MCSI แนะนำ ($a_{\mathcal{H}}$) คือ กลุ่มของการกระทำที่สามารถเลือกได้และให้ค่า CV มากกว่า 1

เริ่มต้นจากการใช้ข้อมูลอ้างอิงทั้งหมดเป็นเซตโปรโตไทป์ ทำการเลือกข้อมูลที่มีผลรวมของค่าความสามารถในการครอบคลุมสูงที่สุด ออกจากเซตโปรโตไทป์ ด้วยเงื่อนไขว่า ข้อมูลสมาชิกตัวที่ถูกเลือกออกนั้นจะต้องไม่ทำให้ค่า CV ของข้อมูลอื่นใดเกิดค่า 0 เนื่องจากการมีค่า 0 เกิดขึ้นในตารางความครอบคลุมแบบเลือกออก นั่นหมายถึง ข้อมูลตัวนั้นจะไม่สามารถทำการจำแนกด้วยข้อมูลตัวแทนใดๆ ได้ถูกต้อง ดังนั้นกระบวนการเลือกออกจะต้องให้ความระมัดระวังในกระบวนการคำนวณเป็นพิเศษและยังใช้การคำนวณเพื่อหามุมมองล่วงหน้า 1 ขั้นกับทุกความเป็นไปได้ จากตารางที่ 2.6

ในขั้นตอนแรกของรอบการคำนวณที่ 1 พบว่าข้อมูลที่มีค่า CV สูงสุดทุกตัวสามารถถูกเลือกออกได้โดยไม่ทำให้เกิดค่า 0 เกิดขึ้นดังนั้นจึงสมมติให้ดึงข้อมูล A3 ออกจากเซตของโปรโตไทป์และทำการปรับปรุงค่าความสามารถในการครอบคลุมในกรรมวิธีของการเลือกออก คือ กรองความสามารถในการครอบคลุมของตนเองออกจากกลุ่มเพียงชุดเดียว ดังตารางที่ 4.1 ที่สามารถลดทอนสมาชิกลงไปได้ 1 ตัว

ตารางที่ 4.1 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลังจากเลือกข้อมูล A3 ออกจากเซตโปรโตไทป์

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : { A1, A2, A4, B1, B2, B3}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	1	1	1	0	0	0	0	A1 = {A1, A2, A3}
A2	1	1	1	0	0	0	0	A2 = {A1, A2, A3}
A3	x	0	x	x	0	0	0	A3 = {Ø}
A4	0	0	0	1	0	0	0	A4 = {A4}
B1	0	0	0	0	1	1	0	B1 = {B1, B2}
B2	0	0	0	0	1	1	1	B2 = {B1, B2, B3}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	2	2	2	1	3	3	2	

ในขั้นตอนถัดมาจำเป็นต้องทำการคำนวณเซตของข้อมูลที่ถูกครอบคลุมใหม่ด้วย NUN ที่เป็นสมาชิกของเซตโปรโตไทป์ที่ได้จากการคำนวณมาสร้างเป็นเซตของข้อมูลที่ถูกครอบคลุมใหม่ ดังตารางที่ 4.2 และคำนวณตารางความครอบคลุมใหม่อีกครั้ง ดังตารางที่ 4.3

ตารางที่ 4.2 เซตของข้อมูลที่ถูกครอบคลุม (Cover List) หลังจากเลือกข้อมูล A3 ออกจากเซตโปรโตไทป์

ข้อมูลที่สนใจ	ข้อมูลที่ครอบคลุม			NUN	ข้อมูลไม่ครอบคลุม			Cove List
A1	A1	A2	A3	B2	B1	A4	B3	A1 = {A1, A2, A3}
A2	A2	A1	A3	B2	B1	B3	A4	A2 = {A1, A2, A3}
A3	-	-	-	-	-	-	-	A3 = {Ø}
A4	A4	B1	A3	B2	A1	A2	B3	A4 = {A4}
B1	B1	B2	A4	A3	A1	B3	A2	B1 = {B1, B2}
B2	B2	B1	B3	A1	A2	A3	A4	B2 = {B1, B2, B3}
B3	B3	B2	B1	A2	A1	A3	A4	B3 = {B1, B2, B3}

ตารางที่ 4.3 ตารางความครอบคลุม (Cover Table)

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : { A1, A2, A4, B1, B2, B3}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	1	1	1	0	0	0	0	A1 = {A1, A2, A3}
A2	1	1	1	0	0	0	0	A2 = {A1, A2, A3}
A3	0	0	0	0	0	0	0	A3 = {Ø}
A4	0	0	0	1	0	0	0	A4 = {A4}
B1	0	0	0	0	1	1	0	B1 = {B1, B2}
B2	0	0	0	0	1	1	1	B2 = {B1, B2, B3}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	2	2	2	~~1~~	3	3	2	

จากตารางที่ 4.3 สังเกตพบว่าข้อมูล A4 หลังจากนี้ไม่สามารถเลือกออกได้เนื่องจากจะทำให้ค่า CV เป็นค่า 0 จึงต้องพิจารณาข้อมูลอื่นต่อไปที่มีผลรวมของค่า CV สูงสุดซึ่งประกอบด้วยข้อมูล {B1, B2} จึงทำการเลือกข้อมูล B1 ออกจากเซตของโปรโตไทป์ และปรับปรุงตารางความครอบคลุมได้เป็นตารางที่ 4.4

ตารางที่ 4.4 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลังจากเลือกข้อมูล B1 ออกจากเซตโปรโตไทป์

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : { A1, A2, A4, B2, B3}

	A1	A2	A3	A4	B1	B2	B3
A1	1	1	1	0	0	0	0
A2	1	1	1	0	0	0	0
A3	0	0	0	0	0	0	0
A4	0	0	0	1	0	0	0
B1	0	0	0	0	x	x	0
B2	0	0	0	0	1	1	1
B3	0	0	0	0	1	1	1
CV	2	2	2	~~1~~	2	2	2

Cove List

A1 = {A1, A2, A3}

A2 = {A1, A2, A3}

A3 = {Ø}

A4 = {A4}

B1 = {Ø}

B2 = {B1, B2, B3}

B3 = {B1, B2, B3}

ทำการคำนวณเซตของข้อมูลที่ถูกครอบคลุมใหม่อีกครั้งด้วย NUN ที่เป็นสมาชิกของเซตโปรโตไทป์ล่าสุด ดังตารางที่ 4.5

ตารางที่ 4.5 เซตของข้อมูลที่ถูกครอบคลุม (Cover List) หลังจากเลือกข้อมูล B1 ออกจากเซตโปรโตไทป์

ข้อมูลที่สนใจ	ข้อมูลที่ครอบคลุม			NUN	ข้อมูลไม่ครอบคลุม			Cove List
A1	A1	A2	A3	B2	B1	A4	B3	A1 = {A1, A2, A3}
A2	A2	A1	A3	B2	B1	B3	A4	A2 = {A1, A2, A3}
A3	-	-	-	-	-	-	-	A3 = {Ø}
A4	A4	~~B1~~	A3	B2	A1	A2	B3	A4 = {A3, A4}
B1	-	-	-	-	-	-	-	B1 = {Ø}
B2	B2	B1	B3	A1	A2	A3	A4	B2 = {B1, B2, B3}
B3	B3	B2	B1	A2	A1	A3	A4	B3 = {B1, B2, B3}

พบว่าสมาชิกของ NUN เปลี่ยนไปเนื่องจากข้อมูล A4 เมื่อเลื่อนขอบเขตของ NUN จากข้อมูล B1 ไปเป็นขอมูล B2 จะสามารถครอบคลุมข้อมูล A3 ได้ด้วย ด้วยข้อมูลจากตารางที่ **4.5** ทำการคำนวณตารางความครอบคลุมใหม่ ดังตารางที่ 4.6

ตารางที่ 4.6 ตารางความครอบคลุม(Cover Table)

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : { A1, A2, A4, B2, B3}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	1	1	1	0	0	0	0	A1 = {A1, A2, A3}
A2	1	1	1	0	0	0	0	A2 = {A1, A2, A3}
A3	0	0	0	0	0	0	0	A3 = {Ø}
A4	0	0	1	1	0	0	0	A4 = {A3, A4}
B1	0	0	0	0	0	0	0	B1 = {Ø}
B2	0	0	0	0	1	1	1	B2 = {B1, B2, B3}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	2	2	~~3~~	~~1~~	~~2~~	2	2	

จากตารางที่ 4.6 พบว่าผลรวมค่า CV ของข้อมูล A3 มีการเปลี่ยนแปลงแต่เนื่องจาก A3 ไม่เป็นสมาชิกของเซตโปรโตไทป์จึงต้องพิจารณาข้อมูลที่มีค่า CV สูงรองลงมานั่นคือผลรวมเท่ากับ 2 มีสมาชิกที่สามารถถูกเลือกออกได้ 4 ตัวได้แก่ {A1, A2, B2, B3} สมมุติให้ทำการเลือกข้อมูล A1 ออก หลังจากปรับปรุงตารางความครอบคลุมจะได้เป็น ตารางที่ 4.7

ตารางที่ 4.7 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลังจากเลือกข้อมูล A1 ออกจากเซตโปรโตไทป์

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : { A2, A4, B2, B3}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	x	x	x	0	0	0	0	A1 = {Ø}
A2	1	1	1	0	0	0	0	A2 = {A1, A2, A3}
A3	0	0	0	0	0	0	0	A3 = {Ø}
A4	0	0	1	1	0	0	0	A4 = {A3, A4}
B1	0	0	0	0	0	0	0	B1 = {Ø}
B2	0	0	0	0	1	1	1	B2 = {B1, B2, B3}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	~~1~~	1	~~2~~	~~1~~	~~2~~	2	2	

จากนั้น ทำการคำนวณเซตของข้อมูลที่ถูกครอบคลุมใหม่ ด้วย NUN ที่เป็นสมาชิกของเซตโปรโตไทป์ { A2, A4, B2, B3} ดังตารางที่ 4.8

ตารางที่ 4.8 เซตของข้อมูลที่ถูกครอบคลุม (Cover List) หลังจากเลือกข้อมูล A1 ออกจากเซตโปรโตไทป์

ข้อมูลที่สนใจ	ข้อมูลที่ครอบคลุม			NUN	ข้อมูลไม่ครอบคลุม			Cove List
A1	-	-	-	-	-	-	-	A1 = {Ø}
A2	A2	A1	A3	B2	B1	B3	A4	A2 = {A1, A2, A3}
A3	-	-	-	-	-	-	-	A3 = {Ø}
A4	A4	~~B1~~	A3	B2	A1	A2	B3	A4 = {A3, A4}
B1	-	-	-	-	-	-	-	B1 = {Ø}
B2	B2	B1	B3	~~A1~~	A2	A3	A4	B2 = {B1, B2, B3}
B3	B3	B2	B1	A2	A1	A3	A4	B3 = {B1, B2, B3}

เซตของข้อมูลที่ถูกครอบคลุมใหม่ล่าสุด สามารถสร้างเป็นตารางความครอบคลุมที่ให้ผลดังตารางที่ 4.7 เช่นเดิม ทำการเลือกข้อมูลออกจากเซตโปรโตไทป์ในลำดับถัดไปซึ่งสามารถเลือกสมาชิกออกได้ 2 ตัวได้แก่ {B2, B3} สมมุติให้ทำการเลือกข้อมูล B2 ออกจากเซตของโปรโตไทป์ และทำการปรับปรุงตารางความครอบคลุมได้ดังตารางที่ 4.9

ตารางที่ 4.9 ตารางความครอบคลุม(Cover Table) ที่ปรับปรุงหลังจากเลือกข้อมูล B2 ออกจากเซตโปรโตไทป์

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : { A2, A4, B3}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	0	0	0	0	0	0	0	A1 = {Ø}
A2	1	1	1	0	0	0	0	A2 = {A1, A2, A3}
A3	0	0	0	0	0	0	0	A3 = {Ø}
A4	0	0	1	1	0	0	0	A4 = {A3, A4}
B1	0	0	0	0	0	0	0	B1 = {Ø}
B2	0	0	0	0	x	x	x	B2 = {Ø}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	~~1~~	1	~~2~~	~~1~~	~~1~~	~~1~~	1	

ทำการคำนวณเซตของข้อมูลที่ถูกครอบคลุมใหม่ ด้วย NUN ที่เป็นสมาชิกของเซตโปรโตไทป์ล่าสุด ดังตารางที่ 4.10

ตารางที่ 4.10 เซตของข้อมูลที่ถูกครอบคลุม (Cover List) หลังจากเลือกข้อมูล B2 ออกจากเซตโปรโตไทป์

ข้อมูลที่สนใจ	ข้อมูลที่ครอบคลุม			NUN	ข้อมูลไม่ครอบคลุม			Cove List
A1	-	-	-	-	-	-	-	A1 = {Ø}
A2	A2	A1	A3	~~B2~~	~~B1~~	B3	A4	A2 = {A1, A2, A3}
A3	-	-	-	-	-	-	-	A3 = {Ø}
A4	A4	~~B1~~	A3	~~B2~~	A1	A2	B3	A4 = {A1,A2, A3, A4}
B1	-	-	-	-	-	-	-	B1 = {Ø}
B2	-	-	-	-	-	-	-	B2 = {Ø}
B3	B3	B2	B1	A2	A1	A3	A4	B3 = {B1, B2, B3}

เมื่อทำการสร้างตารางความครอบคลุมด้วยข้อมูลจากตารางที่ 4.10 ผลรวมของค่า CV ที่ได้ค่อนข้างเปลี่ยนไปจากขั้นตอนการคำนวณที่ผ่านมาพอสมควร เนื่องจาก NUN ที่ถูกเลือกเป็นโปรโตไทป์มีลักษณะชัดเจนขึ้นและมีขอบเขตความครอบคลุมข้อมูลตัวที่สนใจได้กว้างขึ้น หลังจากทำการลดจำนวนข้อมูลจากชุดข้อมูลอ้างอิง ดังนั้นตารางความครอบคลุมที่ได้จึงเป็นดังตารางที่ 4.11

ตารางที่ 4.11 ตารางความครอบคลุม(Cover Table)

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : { A2, A4, B3}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	0	0	0	0	0	0	0	A1 = {Ø}
A2	1	1	1	0	0	0	0	A2 = {A1, A2, A3}
A3	0	0	0	0	0	0	0	A3 = {Ø}
A4	1	1	1	1	0	0	0	A4 = {A1,A2, A3, A4}
B1	0	0	0	0	0	0	0	B1 = {Ø}
B2	0	0	0	0	0	0	0	B2 = {Ø}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	~~2~~	2	~~2~~	~~1~~	~~1~~	~~1~~	~~1~~	

ณ ขั้นตอนนี้จะเหลือข้อมูล A2 เพียงตัวเดียวที่สามารถเลือกออกแล้วไม่ทำให้ค่า CV เป็น 0 เกิดขึ้นในตาราง จึงทำการเลือกข้อมูล A2 ออกและคำนวณเซตข้อมูลที่ถูกครอบคลุมด้วย NUN {A4, B3} ได้ดังตารางที่ 4.12 และคำนวณตารางความครอบคลุมใหม่อีกครั้ง ดังตารางที่ 4.13

ตารางที่ 4.12 เซตของข้อมูลที่ถูกครอบคลุม (Cover List) หลังจากเลือกข้อมูล A2 ออกจากเซตโปรโตไทป์

ข้อมูลที่สนใจ	ข้อมูลที่ครอบคลุม			NUN	ข้อมูลไม่ครอบคลุม			Cove List
A1	-	-	-	-	-	-	-	A1 = {Ø}
A2	-	-	-	-	-	-	-	A2 = {Ø}
A3	-	-	-	-	-	-	-	A3 = {Ø}
A4	A4	~~B1~~	A3	~~B2~~	A1	A2	B3	A4 = {A1,A2, A3, A4}
B1	-	-	-	-	-	-	-	B1 = {Ø}
B2	-	-	-	-	-	-	-	B2 = {Ø}
B3	B3	B2	B1	~~A2~~	~~A1~~	~~A3~~	A4	B3 = {B1, B2, B3}

ตารางที่ 4.13 ตารางความครอบคลุม(Cover Table)

Data Set : { A1, A2, A3, A4, B1, B2, B3}

Prototype Set : { A4, B3}

	A1	A2	A3	A4	B1	B2	B3	Cove List
A1	0	0	0	0	0	0	0	A1 = {Ø}
A2	0	0	0	0	0	0	0	A2 = {Ø}
A3	0	0	0	0	0	0	0	A3 = {Ø}
A4	1	1	1	1	0	0	0	A4 = {A1,A2, A3, A4}
B1	0	0	0	0	0	0	0	B1 = {Ø}
B2	0	0	0	0	0	0	0	B2 = {Ø}
B3	0	0	0	0	1	1	1	B3 = {B1, B2, B3}
CV	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	

จะพบว่าผลรวมของค่า CV จะไม่มีค่าใดมากที่สุดเพราะมีค่าเป็น 1 ทุกข้อมูลและไม่สามารถเลือกโปรโตไทป์ A4, B3 ออกจากเซตได้อีกต่อไป ทำให้การคำนวณหาเซตย่อยสอดคล้องเล็กที่สุดด้วยวิธีเลือกออกสิ้นสุดลงที่เซตคำตอบขนาด 2 ตัวมีคำตอบเป็น {A4,B3}

บทที่ 5
การทดลองและผลการทดลอง

บทนี้กล่าวถึง การทดลองเพื่อเปรียบเทียบผลลัพธ์ของการเรียนรู้แบบเสริมกำลังด้วยอัลกอริธึมRLปกติ, อัลกอริธึม HAQL, อัลกอริธึม RL-RCS, และอัลกอริธึมที่นำเสนอ (LCH-RCS) ในการแก้ปัญหาการเลือกโปรโตไทป์ด้วยวิธีการคัดข้อมูลสมาชิกออก

5.1 ชุดข้อมูลและข้อกำหนดที่ใช้ในการทดลอง

5.1.1 ชุดข้อมูลมาตรฐาน

ชุดข้อมูลที่ใช้ทดลองเป็นข้อมูลจาก ฐานข้อมูลมาตรฐาน “UCI Machine Learning Repository”[6] ซึ่งเป็นเว็บไซต์ที่ให้บริการดาวน์โหลดชุดข้อมูลจริงเพื่อใช้ทดสอบความสามารถทางด้านการค้นคืน ในการเรียนรู้ของเครื่อง(คอมพิวเตอร์) ชุดข้อมูลที่นำมาทดสอบมีรายละเอียด ดังนี้

ตารางที่ 5.1 แสดงรายละเอียดของชุดข้อมูลมาตรฐานที่นำมาทดสอบ

ชุดข้อมูล	จำนวนข้อมูล	คุณลักษณะเฉพาะ	ชนิด	ขนาด
IRIS	150	4	3	เล็ก
GLASS	214	9	7	เล็ก
ECOLI	336	7	8	เล็ก
CREDIT	1000	24	2	กลาง
YEAST	1484	8	10	กลาง

เซตข้อมูลไอริส(IRIS) แสดงกลุ่มข้อมูลของดอกไอริส เป็นข้อมูลบ่งชี้ถึงคุณลักษณะของกลีบเลี้ยง

เซตข้อมูลแก้ว(GLASS) แสดงกลุ่มข้อมูลของแก้ว เป็นข้อมูลบ่งชี้ถึงคุณลักษณะของสารประกอบออกไซด์

เซตข้อมูลอีโคไล(ECOLI) แสดงกลุ่มชุดข้อมูลของแบคทีเรียอีโคไล เป็นข้อมูลบ่งชี้ถึงคุณลักษณะของชนิดแบคทีเรียอีโคไล

เซตข้อมูลสินเชื่อ(CREDIT) แสดงกลุ่มชุดข้อมูลของการพิจาณาอนุมัติสินเชื่อ

เซตข้อมูลยีส (YEAST) แสดงกลุ่มข้อมูลชนิดของยีส

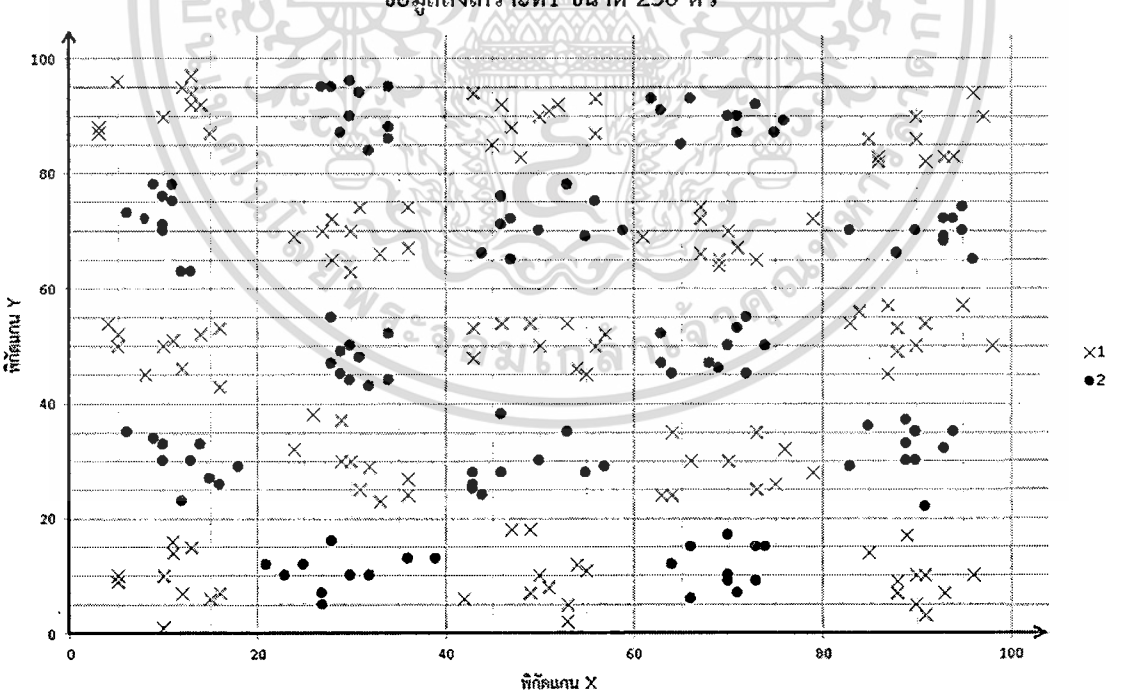
5.1.2 ชุดข้อมูลสังเคราะห์

ชุดข้อมูลสังเคราะห์มีจุดประสงค์เพื่อใช้ทดสอบประสิทธิภาพในการแก้ปัญหาการเลือกโปรโตไทป์ของกระบวนการเรียนรู้แบบเสริมกำลังโดยทดสอบกับอัลกอริธึมRLปกติ, อัลกอริธึม HAQL, อัลกอริธึม RL-RCS, และอัลกอริธึม LCH-RCS โดยข้อมูลที่สร้างขึ้น ได้มีการกำหนดกลุ่มข้อมูลที่เป็นเซตย่อยสอดคล้องเล็กที่สุดไว้จำนวนหนึ่ง และถือเป็นคำตอบที่เหมาะสม (Optimal Solution) โดยมีพิกัดแต่ละจุดอยู่บนศูนย์กลางของกลุ่มข้อมูลแต่ละกลุ่ม จากนั้นทำการสุ่มพิกัดเพื่อสร้างข้อมูลข้างเคียงเพื่อสร้างเป็นกลุ่มของชุดข้อมูลที่มีขนาดใหญ่ขึ้น โดยมีรายละเอียดดังตารางที่ 5.2 และคุณลักษณะเฉพาะของชุดข้อมูลสังเคราะห์ทั้ง 2 ชุด ระบุไว้ในภาคผนวก ก.

ตารางที่ 5.2 แสดงรายละเอียดของชุดข้อมูลสังเคราะห์ที่นำมาทดสอบ

ชุดข้อมูล	จำนวนข้อมูล	คุณลักษณะเฉพาะ	ชนิด	ขนาด
สังเคราะห์1	250	2	2	เล็ก
สังเคราะห์2	225	2	2	เล็ก

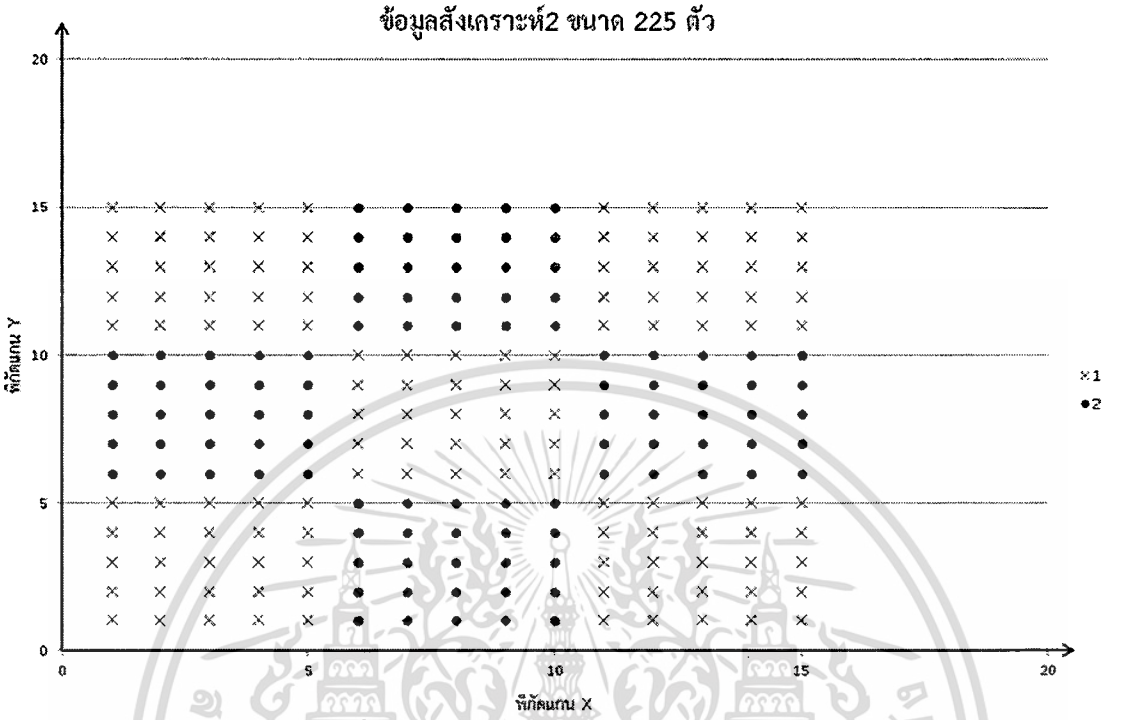
เซตข้อมูลสังเคราะห์ แสดงกลุ่มข้อมูลสังเคราะห์ ที่มีการกำหนดเซตย่อยสอดคล้องที่เหมาะสมและข้อมูลแบบสุ่มพิกัด เพื่อทดสอบประสิทธิภาพในการค้นหาของแต่ละอัลกอริธึมที่นำมาทดสอบ



รูปที่ 5.1 ข้อมูลสังเคราะห์1 ขนาด 250 ตัว

จากรูปที่ 5.1 ข้อมูลสังเคราะห์1 เป็นกลุ่มข้อมูลกำหนดให้เป็นเซตย่อยสอดคล้องเล็กที่สุดจำนวน 25 ตัวประกอบไปด้วย {(10, 10), (30, 10), (50, 10), (70, 10), (90, 10), (10, 30), (30, 30), (50, 30), (70, 30), (90, 30), (10, 50), (30, 50), (50, 50), (70, 50), (90, 50), (10, 70),

(30, 70), (50, 70), (70, 70), (90, 70), (10, 90), (30, 90), (50, 90), (70, 90), (90, 90)} ส่วนข้อมูลตัวอื่นๆ สร้างจากการสุ่มพิกัดในบริเวณใกล้เคียงโดยมีจำนวนรวมทั้งสิ้น 250 ตัว



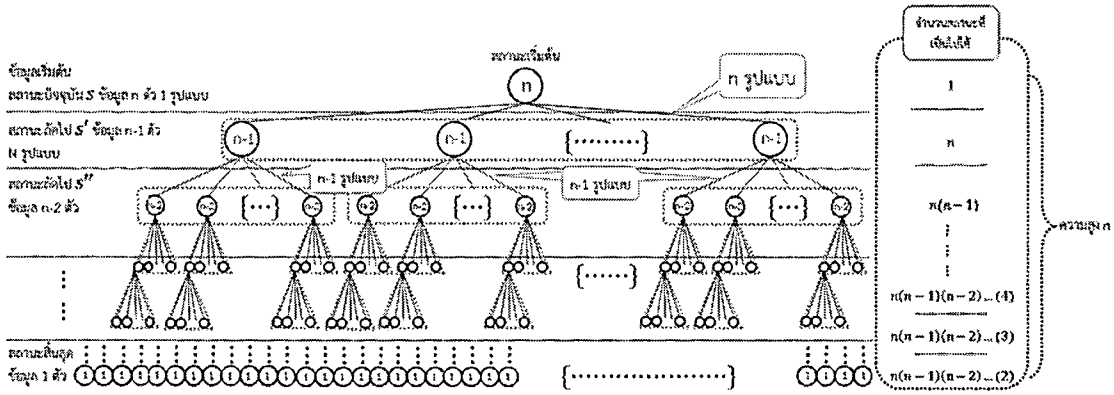
รูปที่ 5.2 ข้อมูลสังเคราะห์2 ขนาด 225 ตัว

จากรูปที่ 5.2 ข้อมูลสังเคราะห์2 เป็นกลุ่มข้อมูลกำหนดให้เป็นเซตย่อยสอดคล้องเล็กที่สุด จำนวน 9 ตัวประกอบไปด้วย {(3, 3), (8, 3), (13, 3), (3, 8), (8, 8), (13, 8), (3, 13), (8, 13), (13, 13)} ส่วนข้อมูลตัวอื่นๆ สร้างจากการการระบุพิกัดภายในรัศมี 2 หน่วย โดยมีจำนวนรวมทั้งสิ้น 225 ตัว

ดังนั้นลักษณะของกลุ่มคำตอบจากการทดลอง ควรจะพบเซตย่อยสอดคล้องเล็กที่สุดที่มีขนาดเทียบเท่าหรือใกล้เคียงกับเซตคำตอบที่กำหนดไว้ แต่อาจแตกต่างในด้านความหลากหลายของข้อมูลสมาชิกในเซตคำตอบจากการสุ่มสร้างข้อมูลอื่นๆ ที่ปะปนมาด้วย

5.1.3 ลักษณะของข้อมูลที่นำมาทดลองและการบริโภคหน่วยความจำ

ในการทดลองเพื่อทดสอบสมมุติฐาน ทางด้านประสิทธิภาพในการค้นหาด้วยโพลิซีแบบกรีดี ε เนื่องจากโพลิซีดังกล่าวมีลักษณะของการสำรวจเป็นอัตราสุ่มแบบร้อยละ ดังนั้นลักษณะของการแก้ปัญหาที่มีความลึกในการค้นหามากกว่า 100 ขั้น(step) จะค่อนข้างสะท้อนให้เห็นถึงข้อจำกัดในการคงเส้นทางการค้นหาที่ดีไว้ได้ค่อนข้างลำบาก แต่หากพิจารณาถึง ความหลากหลายของเซตคำตอบ หรือเส้นทางการค้นหาเพื่อสร้างเป็นต้นไม้ค้นหา(Search Tree) ในรูปที่ 5.2



รูปที่ 5.3 ต้นไม้ค้นหาของข้อมูลสมมุติขนาด n ตัว

ซึ่งแต่ละระดับความลึก เอเจนต์จะต้องจดจำเส้นทางที่ผ่านไปด้วยจำนวนในแต่ละดับมีค่า

$$n, (n-1), (n-2), \dots, 1 \text{ ตามลำดับ}$$

ดังนั้น ณ ระดับความลึก n จะพบสถานะที่มีขนาดเล็กสุดที่หลากหลายและแตกต่างกันได้ทั้งสิ้น

$$n \times (n-1) \times (n-2) \times \dots \times 2 \text{ หรือ } n! \text{ รูปแบบ}$$

ดังนั้น หากคำนวณหาจำนวนสถานะที่เป็นได้ ของต้นไม้ทั้งต้นจะได้ว่า

$$\begin{aligned}\text{จำนวนสถานะที่เป็นไปได้} &= [n] + [n(n-1)] + [n(n-1)(n-2)] \\ &\quad + \dots + [n(n-1)(n-2)\dots(2)] \\ &= \left[\frac{n(n-1)!}{(n-1)!}\right] + \left[\frac{n(n-1)(n-2)!}{(n-2)!}\right] + \left[\frac{n(n-1)(n-2)(n-3)!}{(n-3)!}\right] \\ &\quad + \dots + [n!] \\ &= \sum_{i=1}^{n-1} \frac{n!}{i!}\end{aligned}$$

โดยที่ n หมายถึง จำนวนเส้นทางที่เลือกได้ ณ สถานะเริ่มต้น

i หมายถึง ความสูงของต้นไม้ค้นหาโดยที่ i เริ่มนับในชั้นลึกที่สุดมีค่าเท่ากับ 1

และจากสูตรการประมาณค่าของสเตอร์ลิง [16] ดังสมการที่ 5.1

$$n! \approx n^n e^{-n} \sqrt{2\pi n} \tag{5.1}$$

ดังนั้น ค่าโดยประมาณของ $n!$ ในผลรวมของจำนวนสถานะที่เป็นไปได้จะพบว่า

$$\begin{aligned}\text{จำนวนสถานะที่เป็นไปได้} = \ & O([n] + [n^2 - n] + [n^3 - 3n^2 + 2n] \\ & + \cdots + [n^n e^{-n}\sqrt{2\pi n}]) \\ & \approx O(n^n) \\ & \approx [n^n]\end{aligned}$$

หากสมมุติ สถานะเริ่มต้นมีขนาดเท่ากับ 100 จะมีจำนวนขั้นต่ำของสถานะที่เป็นไปได้โดยประมาณ 100^{100} สถานะ หรือประมาณ 1×10^{200} สถานะ หากแต่ละเส้นทางมีฟังก์ชันมูลค่าQ สะสมอยู่ด้วย ข้อมูลประเภททศนิยม (double) ซึ่งใช้หน่วยความจำขนาด 6 ไบต์เบื่อบรรจุ ดังนั้น ต้นไม้ค้นหาทั้งต้นอาจบริโภคทรัพยากรหน่วยความจำไม่ต่ำกว่า 6×10^{200} ไบต์ และหากสถานะเริ่มต้นมีขนาดใหญ่กว่า 100 เส้นทาง จะทำให้อัตราการบริโภคหน่วยความจำเติบโตขึ้นในเชิงของเลขยกกำลัง (exponent)

ในแง่ของอัลกอริธึมการเรียนรู้แบบเสริมกำลังจึง ถูกกำหนดการค้นหาเพื่อหลีกเลี่ยงปัจจัยข้างต้น ให้มีลักษณะของการค้นพบคำตอบที่เหมาะสม หรือคำตอบที่ดีที่สุด ก่อนที่เงื่อนไขทางด้านหน่วยความจำหรือเวลาจะสิ้นสุดลง ดังนั้นการค้นหาเพียงเส้นทางเดียวหรือค้นหาผ่านสถานะหนึ่งๆไปเพียง 1 ครั้งบนปริภูมิที่มีสถานะจำนวนมาก (มากกว่า 100 สถานะขึ้นไป) จะไม่สามารถการันตีผลของการเรียนรู้ได้ว่าคำตอบที่พบเหล่านั้น เป็นคำตอบที่ดีที่สุดหรือคำตอบที่เหมาะสมได้ เพราะหากฟังก์ชันมูลค่าQ ไม่สามารถลู่เข้าสู่ค่าที่เหมาะสม หรือไม่ลู่เข้าสู่ค่าใดค่าหนึ่ง ก็อาจทำให้เอเจนต์ไม่สามารถค้นหาเส้นทางด้วย การอ้างอิงจากผลของการเรียนรู้อย่างแท้จริงได้ ดังนั้นผู้วิจัยจึงพิจารณากลุ่มของข้อมูลที่นำมาทดสอบทั้งในหัวข้อ 5.1.1 และ 5.1.2 ด้วยเกณฑ์ของขนาดข้อมูลที่มากกว่า 100 ตัวขึ้นไป แต่ไม่ควรมีขนาดใหญ่กว่านี้มากนัก ซึ่งกลุ่มข้อมูลขนาด “กลาง” จากตารางที่ 5.1 ในการทดลองจริงพบว่า ระบบปฏิบัติการไม่สามารถจัดสรรทรัพยากรหน่วยความจำจนจบเงื่อนไขของจำนวนเอพิโสดได้ ดังนั้นจึงถือเอาคำตอบที่ดีที่สุดก่อนสิ้นสุดการทดลองในแต่ละครั้งมาพิจารณา

5.1.4 การกำหนดค่าตัวแปรคงที่สำหรับการทดสอบอัลกอริธึมการเรียนรู้แบบเสริมกำลัง

ความสามารถในการเรียนรู้ด้วยอัลกอริธึม RL จำเป็นจะต้องปรับค่าของตัวแปรต่างๆที่ใช้ในระบบให้เหมาะสมกับปัญหา สำหรับอัลกอริธึม RL ปกติ, อัลกอริธึม HAQL, อัลกอริธึม RL-RCS และอัลกอริธึมLCH-RCS ที่นำมาทดสอบ กำหนดให้มีค่าเท่าๆกันเพื่อให้ผลการทดสอบสมมุติฐานมีความสอดคล้องกับทุกผลการทดลองที่กำลังทำการเปรียบเทียบ

การทดลองเพื่อเปรียบเทียบประสิทธิภาพของอัลกอริธึม RL ในวิทยานิพนธ์เล่มนี้จะกำหนดตัวแปรคงที่ ในสมการที่ 2.11

$$e_t(s,a) = \begin{cases} 1, & \text{กรณี } s = s_t \text{ และ } a = a_t; \\ 0, & \text{กรณี } s = s_t \text{ และ } a \neq a_t; \text{ในทุกๆคู่} (s,a) \\ \gamma\lambda e_{t-1}(s,a), & \text{กรณี } s \neq s_t; \end{cases}$$

สมการที่ 2.12

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a)$$

และสมการที่ 2.13

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t)$$

โดยกำหนดค่าดังต่อไปนี้

1. γ (Discount Rate) = 1.0 อัตราการลดทอน ไม่ถูกปรับลดด้วยอัตรานี้
2. λ (Decay Factor) = 0.1 การปรับปรุงรอยทางปรับมูลค่ามีอัตราเสื่อมสลายของรอยทางร้อยละ 10
3. α (Step size) = 0.1 การปรับฟังก์ชันมูลค่าใช้เทคนิคค่าเฉลี่ยแบบเคลื่อนที่ด้วยอัตราร้อยละ 10
4. ε = 0.01 โพลิซีที่ใช้ทดลองมีความน่าจะเป็นในการเลือกสำรวจด้วยอัตราร้อยละ 1
5. (r) Reward =
   - 1 กรณีเซตคำตอบที่ได้สามารถจำแนกข้อมูลอ้างอิงได้ครบ
   - 0 กรณีเซตคำตอบที่ได้ไม่สามารถจำแนกข้อมูลอ้างอิงได้ครบ
   - -1 กรณีเซตคำตอบที่ได้ทำให้ข้อมูลอ้างอิงบางชนิดสูญหายจนไม่สามารถจำแนกข้อมูลอ้างอิงได้ครบหรือมีโอกาสน้อยที่จะเจอคำตอบจากทางเลือกนี้
6. จำนวนเอพิโสดสูงสุด = 5,000 รอบและ/หรือ จนกว่าเอเจนต์ไม่สามารถใช้ทรัพยากรหน่วยความจำต่อไปได้
7. จำนวนครั้งของการทดลองเพื่อคำนวณค่าเฉลี่ย = 5 รอบ/ชุดข้อมูล
8. ความถี่ในการแนะนำจากฟังก์ชันฮิวริสติกกำหนดไว้ที่ทุกๆ 1 และ 10 เอพิโสด (สำหรับการทดลองหัวข้อ 5.2)
9. ความถี่ในการแนะนำจากฟังก์ชันฮิวริสติกกำหนดใช้งานในทุกๆ 1 เอพิโสด (สำหรับการทดลองหัวข้อ 5.3)

5.2 การเปรียบเทียบผลลัพธ์ระหว่างอัลกอริธึม RL ปกติ, อัลกอริธึม HAQL และอัลกอริธึม RL-RCS

ในหัวข้อนี้จะพิจารณาถึงความสามารถในการหาเซตผลลัพธ์ของปัญหาการเลือกโปรโตไทป์ที่ค้นหาด้วยอัลกอริธึมการเรียนรู้แบบเสริมกำลังปกติ (อัลกอริธึม RL)[1], อัลกอริธึมการเรียนรู้แบบเสริมกำลังที่ใช้ฟังก์ชันฮิวริสติกแนะนำ (อัลกอริธึม HAQL)[3] ด้วยความถี่คำแนะนำในทุก 1 และ 10 เอพิโสด และกระบวนการเรียนรู้แบบเสริมกำลังที่สามารถกลับมายังสถานะเรียนรู้ซ้ำได้ (อัลกอริธึม RL-RCS)[4] โดยการนำมาทดสอบกับชุดข้อมูลจากหัวข้อ 5.1.1

5.2.1 การพิจาณากลุ่มของเซตคำตอบที่พบโดยเฉลี่ย

หากพิจารณาถึงกลุ่มคำตอบที่ดีที่สุดที่พบ การเรียนรู้ด้วย 4 อัลกอริธึมข้างต้นทดสอบกับข้อมูลจริง โดยนำเสนอจำนวนสมาชิกของเซตสอดคล้องเล็กที่สุดที่พบโดยเฉลี่ย จำนวน 5 รอบการทดลอง และจำนวนสมาชิกขนาดเล็กสุดที่พบ(ค่าต่ำสุด) ดังในตารางที่ 5.3

ตารางที่ 5.3 ผลการทดสอบอัลกอริธึม RL, อัลกอริธึม HAQL และอัลกอริธึม RL-RCS

ชุดข้อมูล	RL		HAQL (ทุกๆ เอพิโสด)		HAQL (10 เอพิโสด)		RL-RCS	
	เฉลี่ย	(ต่ำสุด)	เฉลี่ย	(ต่ำสุด)	เฉลี่ย	(ต่ำสุด)	เฉลี่ย	(ต่ำสุด)
ไอริส (150)	13.8	(12)	10	(10)	11.2	(10)	10	(10)
แก้ว (214)	85.4	(82)	80.6	(80)	80.8	(80)	82.4	(81)
อีโคไล (336)	109.8	(100)	94	(93)	95.4	(95)	97.2	(95)
สินเชื่อ (1,000)	771.4	(754)	460	(459)	467	(464)	470	(461)
ยีส (1,484)	1,336.8	(1,316)	883.4	(876)	921.2	(882)	883	(876)
สังเคราะห์1 (250)	33	(32)	28.2	(27)	29.2	(28)	28	(27)
สังเคราะห์2 (225)	89.2	(80)	15.6	(14)	16	(16)	16	(16)

จากตารางที่ 5.3 ผลการเรียนรู้ที่ดีควรให้ผลของจำนวนสมาชิกในเซตคำตอบโดยเฉลี่ยที่มีขนาดเล็ก(มีค่าน้อยๆ) และผลของคำตอบเล็กที่สุดที่พบ (ค่าในวงเล็บ) ไม่ควรแตกต่างจากค่าเฉลี่ยมากนัก ซึ่งในการทดสอบพบว่าประสิทธิภาพของการเรียนรู้ด้วยอัลกอริธึม RL จะให้ผลของคำตอบโดยเฉลี่ย และคำตอบเล็กสุดที่พบจะมีขนาดใหญ่ที่สุดทั้ง 7 ชุดข้อมูล หากเปรียบเทียบกับผลคำตอบจากการเรียนรู้ด้วยอัลกอริธึมอื่นๆ ถือว่ามีประสิทธิภาพของการค้นหาคำตอบจะต่ำสุดจาก 4 อัลกอริธึมที่นำมาทดสอบ

การเรียนรู้ด้วยอัลกอริธึมของ HAQL กรณีใช้คำแนะนำในทุกๆ เอพิโสด จะสามารถค้นพบคำตอบทั้ง 7 ชุดข้อมูลได้ดีที่สุดทั้งในด้านค่าเฉลี่ยและคำตอบเล็กสุดที่พบ ถือว่ามีประสิทธิภาพของการค้นหาคำตอบสูงสุด

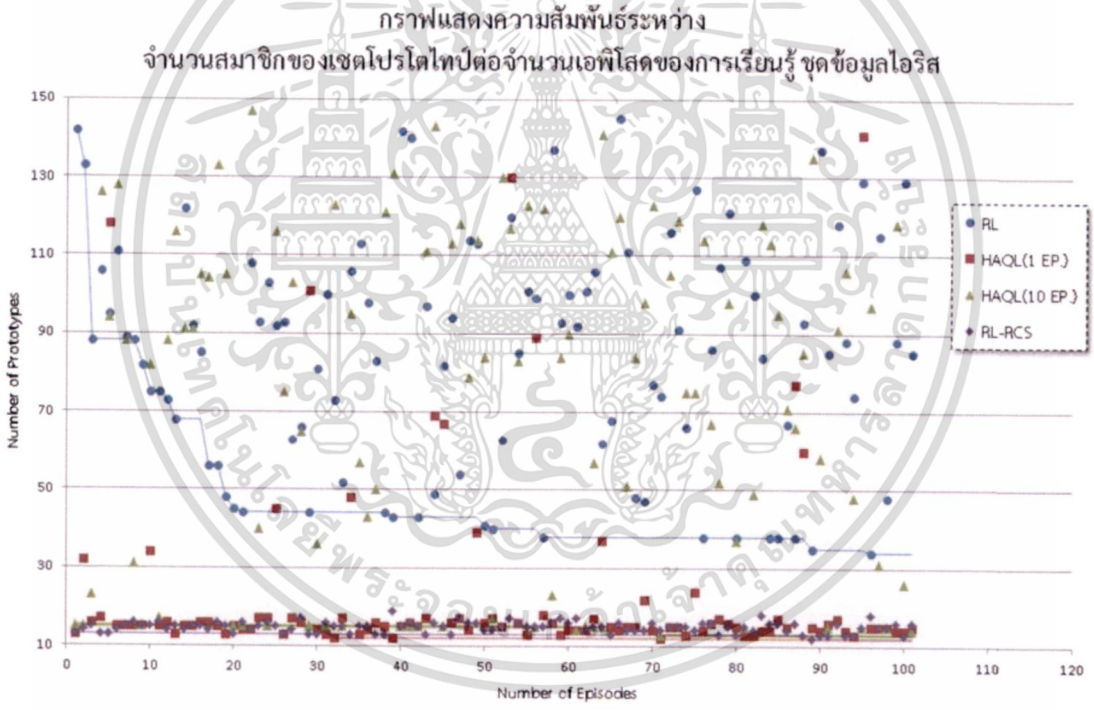
การเรียนรู้ด้วยอัลกอริธึม HAQL ในกรณีที่ใช้คำแนะนำทุกๆ 10 เอพิโสดให้คำตอบใกล้เคียงกับอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆเอพิโสด แต่จะแตกต่างกันตรงที่ เมื่อไรก็ตามที่ไม่ได้ใช้งานฟังก์ชันฮิวริสติก พฤติกรรมการค้นหาของเอเจนต์จะยังคงเป็นอัลกอริธึม RL แบบปกติดังนั้นทำให้

ค่าเฉลี่ยในการเรียนรู้จึงมีค่าเฉลี่ยที่สูงตามปัจจัยของความถี่ในการใช้การค้นหาด้วยกลไกของอัลกอริธึม RL แบบปกติ ถึงถือว่าที่มีประสิทธิภาพที่ใกล้เคียงกับอัลกอริธึม RL-RCS

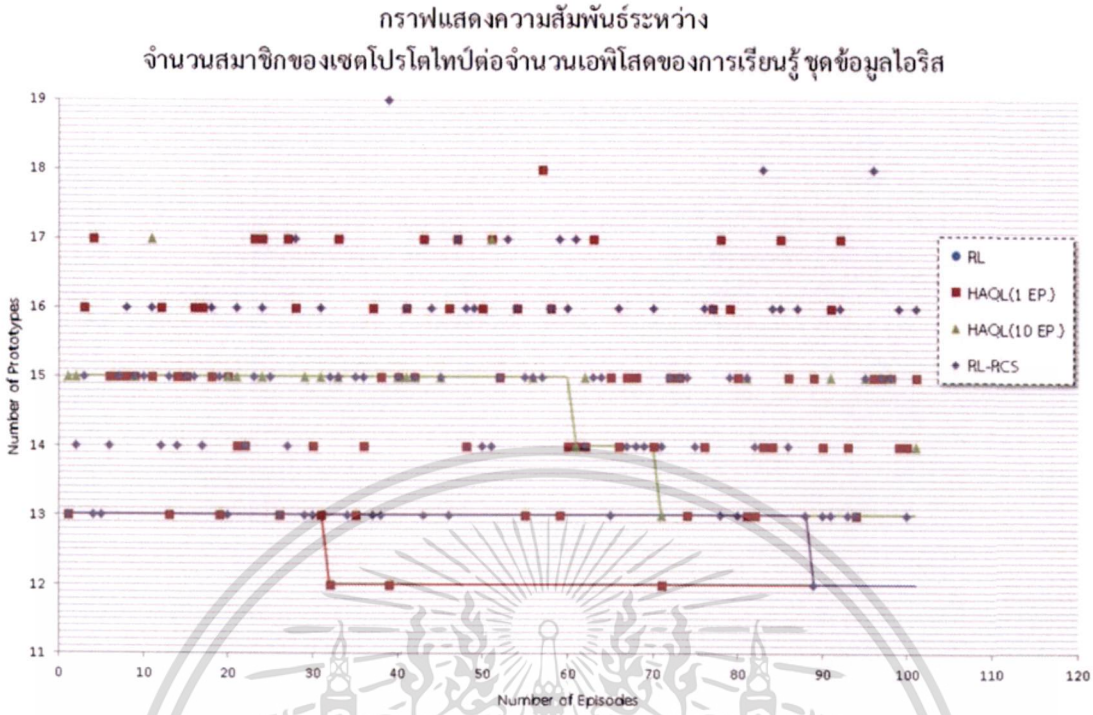
การเรียนรู้ด้วยอัลกอริธึม RL-RCS ที่มีการเพิ่มประสิทธิภาพในการค้นหาแบบโลคอล พบว่าประสิทธิภาพในการค้นหาคำตอบโดยเฉลี่ยและคำตอบเล็กสุดที่พบดีกว่าหากเทียบกับอัลกอริธึม RL แต่มีประสิทธิภาพด้อยกว่าหากเทียบกับอัลกอริธึม HAQL ที่ใช้ความถี่ในการแนะนำโดยตลอด

5.2.2 การพิจาณาแนวโน้มการค้นหาของเอเจนต์

หากพิจารณาพฤติกรรมการค้นหาของการเรียนรู้ด้วย อัลกอริธึมทั้ง 4 ข้างต้น สำหรับชุดข้อมูลไอริส (Iris) ผลการทดสอบแสดงในรูปภาพที่ 5.3 - 5.4 ส่วนผลการทดลองในชุดข้อมูลแก้ว (Glass), แบคทีเรียอีโคไล (Ecoli), สินเชื่อ (Credit), ยีส (Yeast), ข้อมูลสังเคราะห์1 และข้อมูลสังเคราะห์ 2 ผลของคำตอบมีลักษณะในทำนองเดียวกัน และได้รวบรวมไว้ในภาคผนวก ข. และภาคผนวก ค.



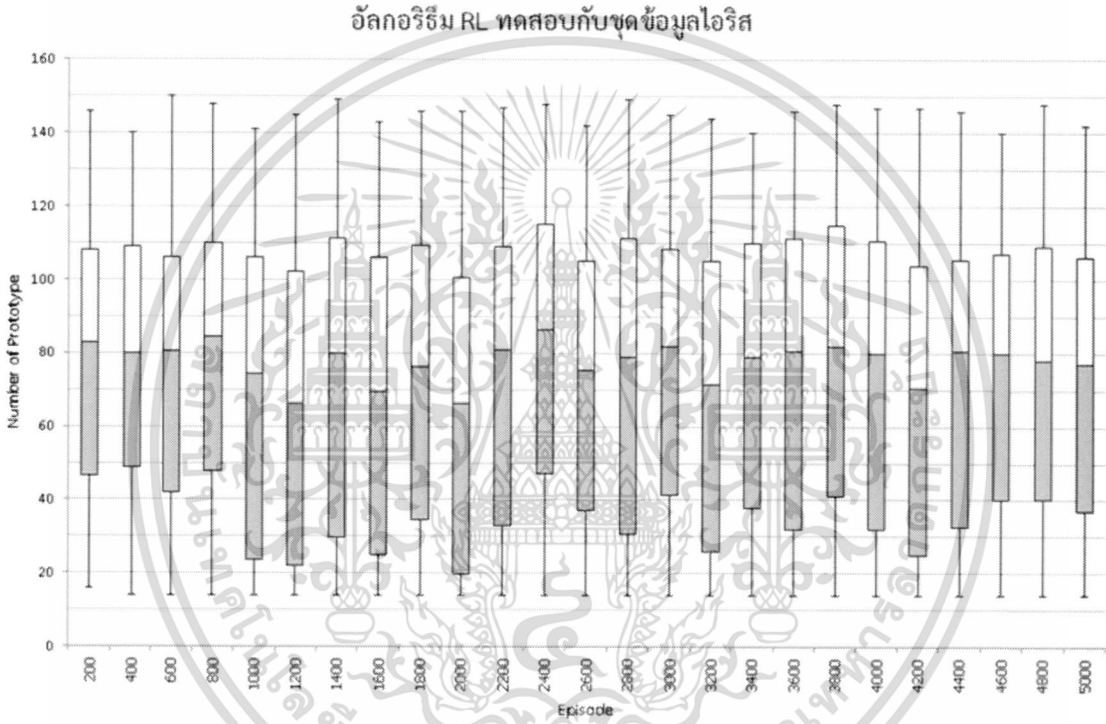
รูปที่ 5.4 กราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลไอริสที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ



รูปที่ 5.5 ภาพขยายกราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลไอริสที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ

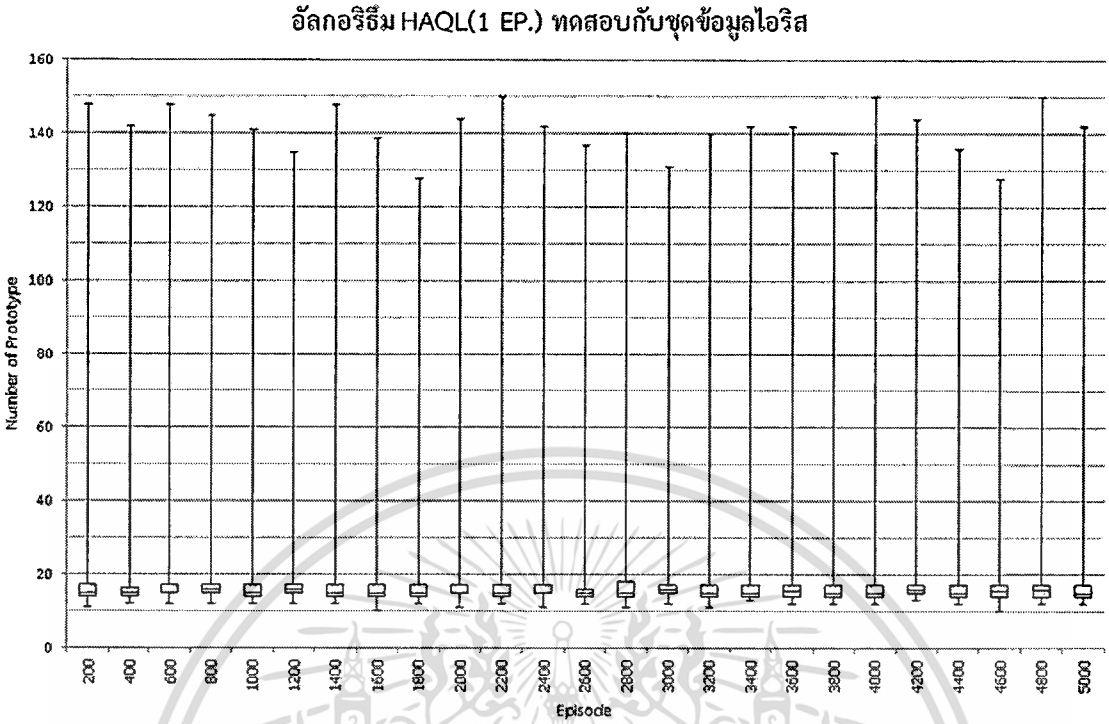
จากกราฟในรูปที่ 5.3 และ 5.4 แสดงผลการทดลองของ 100 เอพิโสดแรกของชุดข้อมูลไอริส สำหรับแกน x นำเสนอจำนวนของเอพิโสด, แกน y นำเสนอจำนวนโปรโตไทป์ที่ค้นพบในเอพิโสดที่ระบุโดยแกน x โดยที่อัลกอริธึม RL (จุดวงกลมสีคราม), อัลกอริธึม HAQL ที่ใช้อัตราการค้นคืนเป็นฟังก์ชันฮิวริสติกแนะนำด้วยความถี่ทุกๆ หนึ่งหน่วยเอพิโสด (จุดสี่เหลี่ยมจัตุรัสสีแดง), อัลกอริธึม HAQL ที่ใช้อัตราการค้นคืนเป็นฟังก์ชันฮิวริสติกแนะนำด้วยความถี่ทุกๆ 10 เอพิโสด (จุดสามเหลี่ยมสีเขียว) และอัลกอริธึม RL-RCS (จุดสี่เหลี่ยมขนมเปียกปูนสีม่วง) โดยแต่ละเส้นในกราฟ แสดงจำนวนเซตคำตอบของสมาชิกในเซตโปรโตไทป์คำตอบดีที่สุดที่พบ ตั้งแต่เริ่มต้นการแก้ปัญหาด้วยอัลกอริธึมแบบต่างๆ

การค้นหาของอัลกอริธึม RL มีลักษณะของคำตอบกระจายอยู่ทั่วปริภูมิคำตอบ เนื่องจากลักษณะของการลองผิดลองถูกโดยปราศจากความรู้ก่อนหน้าว่าทางเลือกใดจะได้รับรางวัลอย่างไร และไม่สามารถย้อนกลับไปเรียนรู้ซ้ำได้ ซึ่งบ่อยครั้งที่เอเจนต์ได้เลือกสำรวจ ที่ไม่ได้รับรางวัลหรือได้รับรางวัลเป็นลบ ทำให้เอเจนต์จะยังคงต้องเลือกตัวเลือกที่เหลืออยู่ในสถานะถัดไปเรื่อยๆ จนกว่าจะจบเอพิโสด และได้คำตอบที่ไม่ดี หากพิจารณาถึงการกระจายตัวของกลุ่มคำตอบที่พบภายในหนึ่งรอบการทดลอง โดยนำเสนอในรูปแบบของ แผนภูมิกล่อง (Box-plot) จะพบว่าการเรียนรู้ด้วยอัลกอริธึม RL มีการเรียนรู้ที่ให้ผลของเซตคำตอบที่กระจายทั่วปริภูมิคำตอบ คือมีลักษณะของเซตคำตอบตั้งแต่ขนาดเล็กจนถึงเซตคำตอบขนาดใหญ่ ดังรูปที่ 5.5 ซึ่งเซตคำตอบจะมีค่ามัธยฐานที่ค่อนข้างสูงและกลุ่มของความหนาแน่นของคำตอบ กระจายในช่วงปริภูมิที่ค่อนข้างกว้างอีกด้วย



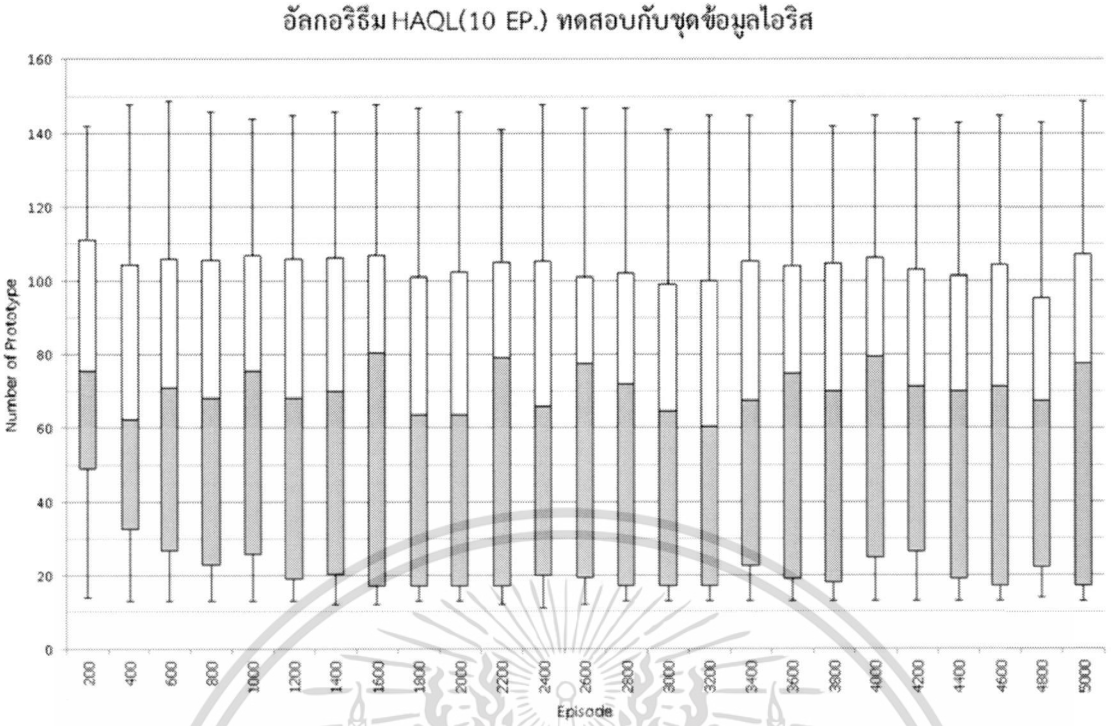
รูปที่ 5.6 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลไอริสด้วยอัลกอริธึมRL

พฤติกรรมการค้นหาของอัลกอริธึม HAQL ในรูปที่ 5.3 และ 5.4 จะให้ผลของคำตอบที่ดีตั้งแต่เริ่มต้นการเรียนรู้ และผลของคำตอบจะแปรผันกับความถี่ในการใช้งาน อัลกอริธึม HAQL ที่ใช้งานคำแนะนำทุกๆเอพิโสดจะให้ผลของคำตอบที่มีขนาดเล็ก สาเหตุเป็นเพราะเมื่อไรก็ตามที่เอเจนต์เลือกสำรวจการกระทำที่ไม่ได้รับรางวัล ก็ยังมีโอกาสจะเลือกการกระทำที่ได้รับคำแนะนำ และกลับมาสู่สถานะที่ดีได้อยู่ตราบเท่าที่ยังสามารถคำนวณคำแนะนำได้อยู่ ทำให้เส้นทางการค้นหาที่เอเจนต์เคลื่อนผ่านในแต่ละเอพิโสดจะค่อนข้างลึก และหากนำเสนอผลการเรียนรู้ในหนึ่งรอบการทดลองบนรูปแบบของ แผนภูมิกล่อง (Box-plot) จะพบว่าเซตคำตอบที่พบมีขนาดเล็ก มีค่ามัธยฐานและค่าต่ำสุด อยู่ในปริภูมิคำตอบที่ค่อนข้างลึกมากซึ่งถือว่าเป็นการค้นหาที่ดี แต่สิ่งที่น่าสังเกตคือค่าสูงสุดของเซตคำตอบยังคงมีขนาดใหญ่ ซึ่งยังมีโอกาสที่จะเกิดการสำรวจด้วยค่าความน่าจะเป็น ε ที่เอเจนต์ได้เลือกการกระทำที่ไม่ดี และฟังก์ชันฮิวริสติกไม่สามารถช่วยแนะนำได้อีก ดังรูปที่ 5.6



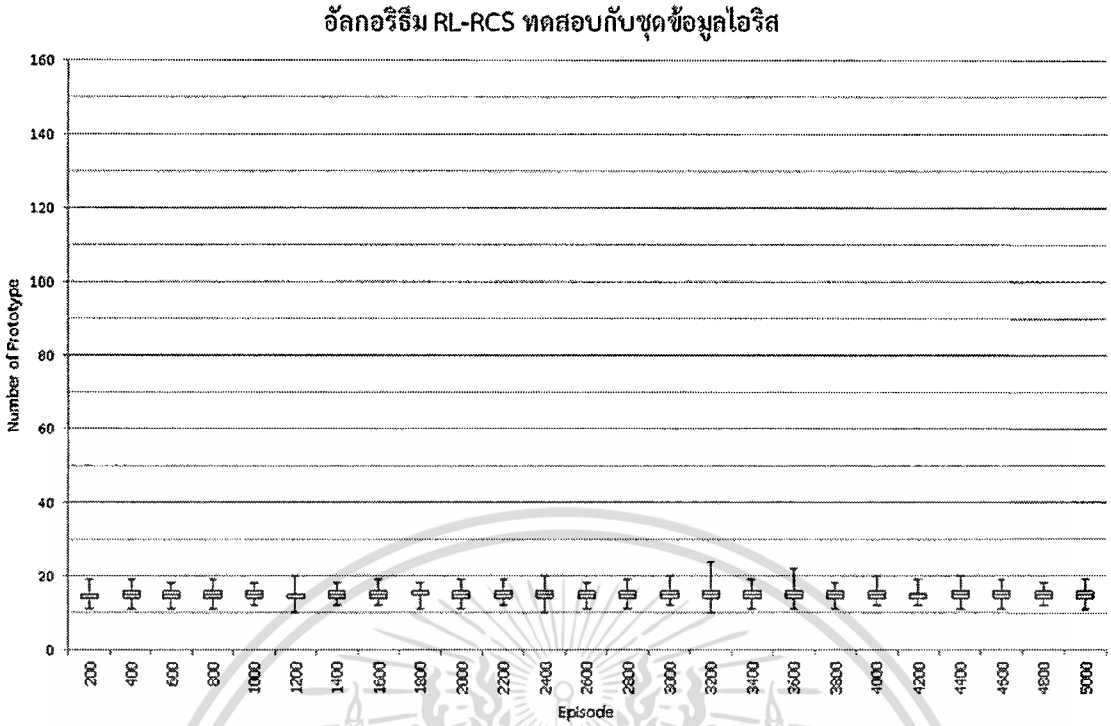
รูปที่ 5.7 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลไอริสด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ เอพิโสด

ส่วนพฤติกรรมการค้นหาของอัลกอริธึม HAQL ที่ใช้งานคำแนะนำทุกๆ 10 เอพิโสด ในรูปที่ 5.3 และ 5.4 ในเอพิโสดใดๆ ที่ไม่ได้ใช้คำแนะนำ การเรียนรู้จะมีพฤติกรรมที่สอดคล้องกับอัลกอริธึม RL ปกติ ผลของคำตอบจึงกระจายอยู่ในปริภูมิที่มีเซตคำตอบค่อนข้างใหญ่เช่นเดิม และผลของเซตคำตอบที่ดีจะถูกพบในเอพิโสดที่ใช้งานคำแนะนำเป็นส่วนใหญ่ และถ้านำเสนอผลการเรียนรู้ในหนึ่งรอบการทดลอง ในรูปแบบของแผนภูมิกล่อง (Box-plot) จะพบว่า ความหนาแน่นของกลุ่มคำตอบจะกระจายอยู่ในบริเวณที่เซตคำตอบมีขนาดเล็กกว่ากลุ่มคำตอบของอัลกอริธึม RL เล็กน้อย เนื่องจากเอเจนต์พยายามใช้หลักการกรีดีในทิศที่ฟังก์ชันฮิวริสติกเคยได้แนะนำ ทำให้กลุ่มของคำตอบมีความหนาแน่นค่อนไปทางด้านล่าง ดังรูปที่ 5.7



รูปที่ 5.8 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลไอริสด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ 10 เอพิโสด

พฤติกรรมการค้นหาของอัลกอริธึม RL-RCS ในรูปที่ 5.3 และ 5.4 กลุ่มของคำตอบตลอดการทดลองจะมีจำนวนสมาชิกของเซตคำตอบขนาดเล็ก เนื่องจากความสามารถในการกลับยังสถานะเรียนรู้ซ้ำ เพื่อให้เอเจนต์กลับมาแก้ไขการกระทำที่ไม่ดีเกิดขึ้นได้ตลอดเวลา ทำให้เอเจนต์สามารถเคลื่อนผ่านไปยังสถานะที่ดีขึ้นเรื่อยๆภายในเอพิโสดเดียวกัน ผลของคำตอบจึงอยู่ในกลุ่มของปริภูมิคำตอบที่ดีมีขนาดเซตคำตอบมีขนาดเล็ก และการกระจายอยู่ในช่วงแคบๆ ซึ่งสามารถนำเสนอผลการเรียนรู้ในหนึ่งรอบการทดลอง ในรูปแบบของแผนภูมิกล่อง (Box-plot) ดังรูปที่ 5.8



รูปที่ 5.9 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลไอริสด้วยอัลกอริธึม RL-RCS

จะสังเกตได้ว่า อัลกอริธึม HAQL จะมีประสิทธิภาพการเรียนรู้ที่ดีที่สุดก็ต่อเมื่อเปิดการใช้งานคำแนะนำในทุกๆ หนึ่งหน่วยเอพิโสด เพราะเนื่องจากทันทีที่เอเจนต์ไม่เชื่อฟังคำแนะนำ (ไม่เลือกตามคำแนะนำ) ก็จะมีโอกาสบ่อยครั้งที่ผลของคำตอบจะแย่ลงในทันที หากสังเกตจากแผนภูมิกล่องในรูปที่ 5.4 เนื่องจากประสิทธิภาพในการเรียนรู้ที่ไม่ใช้คำแนะนำจะเป็นอัลกอรุธึม RL ที่มีประสิทธิภาพในการเรียนรู้ที่ค่อนข้างจะลู่เข้าสู่ค่าที่เหมาะสมได้ช้ากว่าหากเทียบกับอัลกอริธึม RL-RCS ที่ให้ผลของเซตคำตอบที่อยู่ในปริภูมิที่ค่อนข้างดีตั้งแต่เริ่มต้นการเรียนรู้

จึงเป็นที่มาของการเพิ่มประสิทธิภาพอัลกอริธึม RL-RCS ด้วยอัลกอริธึม HAQL เพื่อมุ่งหวังให้เอเจนต์มีการเรียนรู้ตัวเลือกที่ฮิวริสติกแนะนำในลักษณะของการค้นหาที่มีทิศทางการค้นหาที่สอดคล้องกับคำแนะนำของฟังก์ชันฮิวริสติกในขณะที่เอเจนต์อยู่ในสถานะที่สามารถคำนวณฮิวริสติกได้ แต่ในกรณีที่ฟังก์ชันฮิวริสติกไม่แนะนำทางเลือกใดเลยจะใช้ลักษณะของการค้นหาแบบโลคอล (อัลกอริธึม RL-RCS) ที่สามารถย้อนกลับมายังสถานะเรียนรู้ซ้ำ เพื่อที่จะหาตัวเลือกที่สามารถนำเอเจนต์เคลื่อนไปยังสถานะถัดไปที่ดีกว่าสถานะปัจจุบันได้ ในการเปรียบเทียบผลการเรียนรู้จะกล่าวในหัวข้อถัดไป

5.3 การเพิ่มประสิทธิภาพอัลกอริธึม RL-RCS ด้วยการประยุกต์อัลกอริธึม HAQL ให้สามารถเรียนรู้ทางเลือกได้

ในหัวข้อนี้จะพิจารณาถึงความสามารถในการหาเซตผลลัพธ์ของปัญหาการเลือกโปรโตไทป์ที่ค้นหาด้วยกระบวนการเรียนรู้แบบเสริมกำลังที่สามารถกลับมายังสถานะเรียนรู้ซ้ำได้ (อัลกอริธึม RL-RCS) และกระบวนการเรียนรู้แบบเสริมกำลังที่เพิ่มประสิทธิภาพการเรียนรู้ด้วยฟังก์ชันฮิวริสติก (อัลกอริธึม HAQL) เปรียบเทียบกับกระบวนการเรียนรู้แบบเสริมกำลังที่นำเสนอ ที่ทำการปรับปรุงความสามารถในการเรียนรู้จากทางเลือกที่แนะนำโดยฟังก์ชันฮิวริสติก (Learning choice generated from heuristic guides) อีกทั้งยังมีความสามารถในการกลับมายังสถานะเรียนรู้ซ้ำได้ (Returning to the last known consistent state) ดังนั้นเพื่อความสะดวกในการอธิบายจึงขอเสนอชื่อโดยย่อว่า อัลกอริธึม LCH-RCS

5.3.1 การเปรียบเทียบผลการเรียนรู้ระหว่างอัลกอริธึม RL-RCS, อัลกอริธึม HAQL-RCS และอัลกอริธึม LCH-RCS ที่เลือกใช้ฟังก์ชันฮิวริสติกด้วยอัตราการค้นคืน

ในหัวข้อนี้นำเสนอผลการเปรียบเทียบการเรียนรู้ 3 อัลกอริธึม ทดสอบกับชุดข้อมูลจากหัวข้อ 5.1.1 โดยเลือกใช้ฟังก์ชันฮิวริสติกที่คำนวณจากอัตราการค้นคืน(หัวข้อ 4.4.1) ของสมาชิกในเซตโปรโตไทป์ที่ยังคงอยู่บนสถานะถัดไปที่เป็นไปได้ หากคำตอบนั้นสามารถจำแนกข้อมูลอ้างอิงได้ถูกต้องทั้งหมด เอเจนต์ก็จะได้รับรางวัลที่มีค่าบวก ซึ่งเท่ากับว่าตัวเลือกจากคำแนะนำที่คำนวณได้ จะสอดคล้องกับรางวัลที่เอเจนต์ได้รับด้วยหรืออีกนัยหนึ่งคือ สามารถรู้ล่วงหน้าว่าทางเลือกใดจะได้รับผลรางวัล โดย ฮิวริสติกจะให้คำแนะนำเฉพาะทางเลือกที่ได้รับรางวัลเท่านั้น

ผลการเปรียบเทียบนำเสนอด้วยจำนวนสมาชิกของเซตสอดคล้องเล็กที่สุดที่พบโดยเฉลี่ยจำนวน 5 รอบการทดลอง และจำนวนสมาชิกขนาดเล็กสุดที่พบ(ค่าต่ำสุด) ดังตารางที่ 5.4

ตารางที่ 5.4 ผลการทดสอบอัลกอริธึม RL-RCS, อัลกอริธึม HAQL-RCS และอัลกอริธึม LCH-RCS ที่ใช้อัตราการค้นคืนสร้างเป็นฟังก์ชันฮิวริสติก

ชุดข้อมูล	RL-RCS		HAQL-RCS		LCH-RCS	
	เฉลี่ย	(ต่ำสุด)	เฉลี่ย	(ต่ำสุด)	เฉลี่ย	(ต่ำสุด)
ไอริส (150)	10	(10)	10	(10)	10	(10)
แก้ว (214)	82.4	(81)	79.2	(79)	79	(79)
อีโคไล (336)	97.2	(95)	91.2	(90)	91	(90)
สินเชื่อ (1,000)	470	(461)	436	(433)	428	(424)
ยีส (1,484)	883	(876)	855	(855)	851.5	(850)
สังเคราะห์1 (250)	28	(27)	25.8	(25)	26.2	(26)
สังเคราะห์2 (225)	16	(16)	13.2	(12)	13	(12)

ผลการทดลองจากตารางที่ 5.4 พบว่าอัลกอริธึมที่ใช้งานฟังก์ชันฮิวริสติกทั้ง 2 อัลกอริธึมจะให้ผลของการค้นหาที่มีจำนวนสมาชิกในเซตขนาดเล็กกว่าเมื่อเทียบกับ อัลกอริธึม RL-RCS แต่จะให้ผลของเซตคำตอบที่มีจำนวนสมาชิกในเซตที่เท่ากันระหว่างอัลกอริธึม HAQL-RCS และอัลกริธึม LCH-RCS เนื่องจากแนวคิดของการคำนวณคำแนะนำด้วยฟังก์ชันฮิวริสติกเดียวกัน ดังนั้นคำตอบที่เหมาะสม (Optimal Solution) จึงควรจะสอดคล้องกันหรือเป็นคำตอบเดียวกันและเป็นการยากที่จะเปรียบเทียบประสิทธิภาพด้วยจำนวนสมาชิกในเซตผลลัพธ์จากคำตอบที่ได้

ดังนั้นจึงพิจารณา จากทางเลือกการกระทำที่เอเจนต์เลือก ว่าได้ใช้น้ำหนักการตัดสินใจที่มาจากความรู้เดิมมากน้อยเพียงใด โดยจะพิจารณาทุกๆการกระทำที่เอเจนต์ได้เลือกตลอดการทดลอง เพื่อแบ่งประเภทตามฟังก์ชันมูลค่าQ ของทางเลือก ออกเป็น 3 กรณี ดังนี้

1. กรีดีฟังก์ชันมูลค่าที่ฟังก์ชันฮิวริสติก $\mathcal{H}$ แนะนำ โดยที่ทางเลือกนั้นไม่ใช้น้ำหนักของฟังก์ชันมูลค่าQ ($Q = 0$ หรือ $Q^* + \eta$)
2. การกรีดีฟังก์ชันมูลค่า(RL-RCS) หรือ กรีดีฟังก์ชันมูลค่าที่ฟังก์ชันฮิวริสติก $\mathcal{H}$ แนะนำ (HAQL-RCS และ LCH-RCS) โดยทางเลือกที่ได้จะมีฟังก์ชันมูลค่าQ มากกว่าศูนย์ ($Q > 0$) หรือการพิจารณาฮิวริสติกร่วมกับน้ำหนักเดิมของฟังก์ชันมูลค่าQ
3. การสุ่มสำรวจปริภูมิคำตอบ ถือว่าเอเจนต์กำลังทำการเรียนรู้จากการสำรวจ
    - เกิดจากทางเลือกที่เป็นมุมมองการสำรวจที่เกิดจากความน่าจะเป็นร้อยละ ε และมีฟังก์ชันมูลค่าQ ที่ไม่ใช่ค่ามากสุด
    - เกิดจากเงื่อนไขที่เอเจนต์ได้ผ่านไปยังปริภูมิที่ไม่ทราบฟังก์ชันมูลค่ามาก่อนและอยู่บนปริภูมิที่ฟังก์ชันฮิวริสติกไม่แนะนำให้เลือก (ค่าการแนะนำเป็นศูนย์)

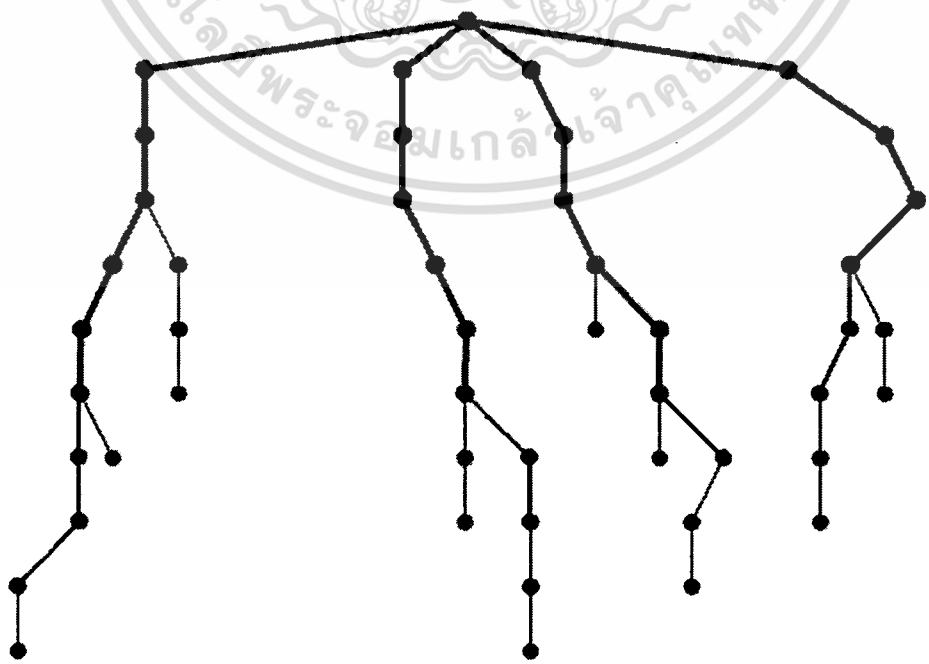
ผลจากการวิเคราะห์เหตุผลในการเลือกการกระทำของเอเจนต์ ด้วยเงื่อนไขทั้ง 3 กรณี กับกลุ่มข้อมูลจริงที่ทดลอง ทำการนับทุกๆการกระทำที่เอเจนต์ได้เลือกตลอดการทดลอง (ใช้ทั้งสิ้น 5,000 เอพิโสด) ทั้งนี้หากในกรณีของกลุ่มข้อมูลที่การทดลองสิ้นสุดก่อนเกณฑ์ที่กำหนดเพราะเงื่อนไขทางด้านทรัพยากรหน่วยความจำ (สิ้นสุดการทดลองขณะที่เอพิโสดน้อยกว่า 5,000 ครั้ง) จะถือเอาการกระทำทั้งหมดนับจากเริ่มต้นการทดลองจนถึงเอพิโสดที่สิ้นสุดนี้เป็นขอบเขตการคำนวณทั้ง 5 ครั้งของชุดข้อมูลจริงที่นำมาทดสอบ

ตารางที่ 5.5 ร้อยละของการเลือกการกระทำตลอดการทดลองที่เรียนรู้ด้วยอัลกอริธึม RL-RCS, อัลกอริธึม HAQL-RCS และอัลกอริธึม LCH-RCS กับชุดข้อมูลทดสอบ

ชุดข้อมูล	RL-RCS		HAQL-RCS			LCH-RCS		
	Q > 0	Q ≤ 0	$\mathcal{H}$	Q > 0	Q ≤ 0	$\mathcal{H}$	Q > 0	Q ≤ 0
ไอริส	12.60%	87.40%	17.74%	0.29%	81.97%	7.56%	10.51%	81.93%
แก้ว	8.66%	91.34%	11.87%	0.18%	87.95%	3.73%	8.39%	87.88%
อีโคไล	6.75%	93.25%	13.90%	0.10%	86.00%	7.11%	6.93%	85.96%
สินเชื่อ	3.46%	96.54%	10.78%	0.07%	89.15%	7.36%	3.50%	89.14%
ยีส	2.66%	97.34%	8.20%	0.06%	91.74%	5.66%	2.58%	91.76%
สังเคราะห์1	9.67%	91.33%	17.05%	0.20%	82.75%	10.13%	7.13%	82.74%
สังเคราะห์2	33.0%	67.00%	12.54%	6.79%	80.67%	5.59%	12.48%	81.93%

จากการทดลองในตารางที่ 5.5 จะเห็นว่าทั้งอัลกอริธึม RL-RCS และอัลกอริธึม LCH-RCS มีจำนวนของอัตราร้อยละในกรณี 2 ค่อนข้างสูงกว่าอัลกอริธึม HAQL-RCS และพบว่าเอเจนต์จะอยู่ในปริภูมิการค้นหาที่ไม่พบเจอคำตอบที่ดี(กรณีที่ 3)กว่าร้อยละ 80 โดยประมาณของการกระทำที่ได้เลือกทั้งหมด ส่วนการกระทำที่ดี(กรณี 1 และ 2) สามารถอธิบายแยกตามอัลกอริธึมดังนี้

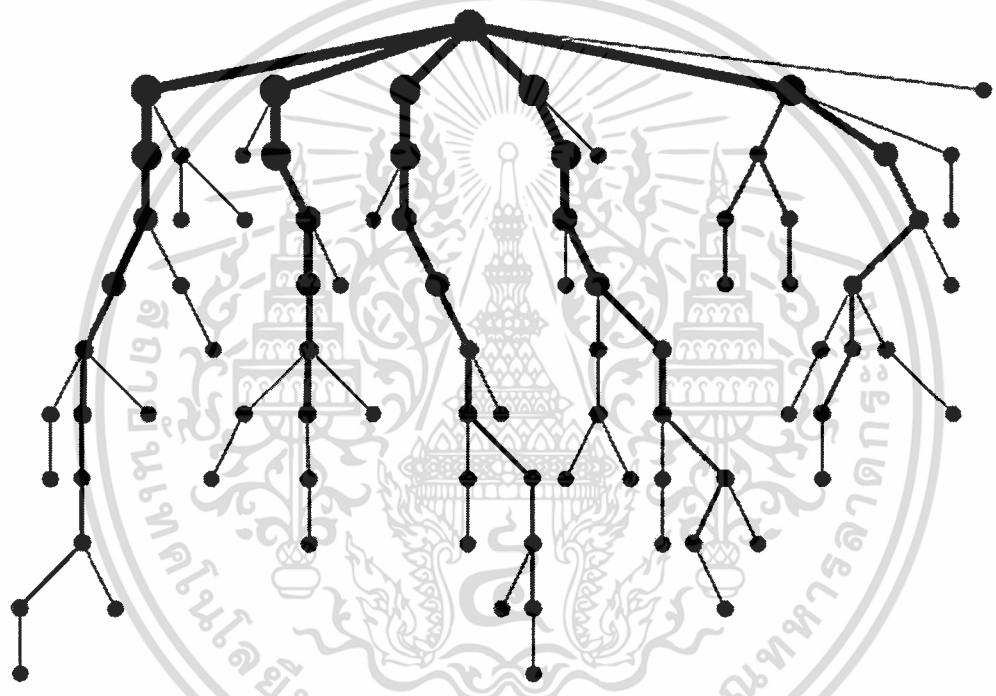
สำหรับกรณีของอัลกอริธึม RL-RCS เส้นทางที่เอเจนต์ได้เลือกเรียนรู้เป็นการเลือกตามฟังก์ชันมูลค่าQที่ดี (กรณี 2) หมายถึง เอเจนต์จะเลือกกรีดตามการกระทำใดๆที่พบว่ามีฟังก์ชันมูลค่าQ สูงสุด(แบบซอร์ฟแมกซ์)อยู่เสมอหรือเป็นการเลือกตามเส้นทางที่เคยเรียนรู้มาก่อน และจะเกิดกรณีการเบี่ยงเส้นทางการค้นหาออกจากเส้นทางเดิมเพียงกรณีเดียว คือเกิดการสำรวจของโพลิซีแบบกรีดี ε ด้วยอัตราร้อยละ 1 เท่านั้น ทำให้เอเจนต์เคลื่อนเข้าไปเรียนรู้ในปริภูมิที่ไม่ทราบค่า (กรณี 3) ด้วยเหตุผลดังกล่าว ในการสุ่มการกระทำบนปริภูมิที่ฟังก์ชันมูลค่าเป็นศูนย์ จะไม่ถือเป็นการกรีดีด้วยโพลิซีซอร์ฟแมกซ์



รูปที่ 5.10 เส้นทางการค้นหาคำตอบที่ใช้อัลกอริธึม RL-RCS

จากรูปที่ 5.9 เส้นทางการค้นหาคำตอบของอัลกอริธึม RL-RCS ที่เส้นทางหลัก (กรีดีค่ามาก) จะสะสมฟังก์ชันมูลค่าอยู่ปริมาณหนึ่ง (เส้นทึบ) เส้นทางการค้นหามีลักษณะมุ่งค้นหาลงไปในแนวลึก (แนวดิ่ง) การสุ่มสำรวจบนปริภูมิหนึ่งๆจะไม่กระจัดกระจายมากนัก (โหนดลูกหลานจะไม่ค่อยแตกแขนง) ทิศทางการค้นหาคำตอบจะเป็นแบบสรุปไม่ได้ ขึ้นอยู่กับปัจจัยการกรีดีและการสุ่มขณะนั้น (เนื่องจากไม่ได้แนวคิดของคำแนะนำจากภายนอก)

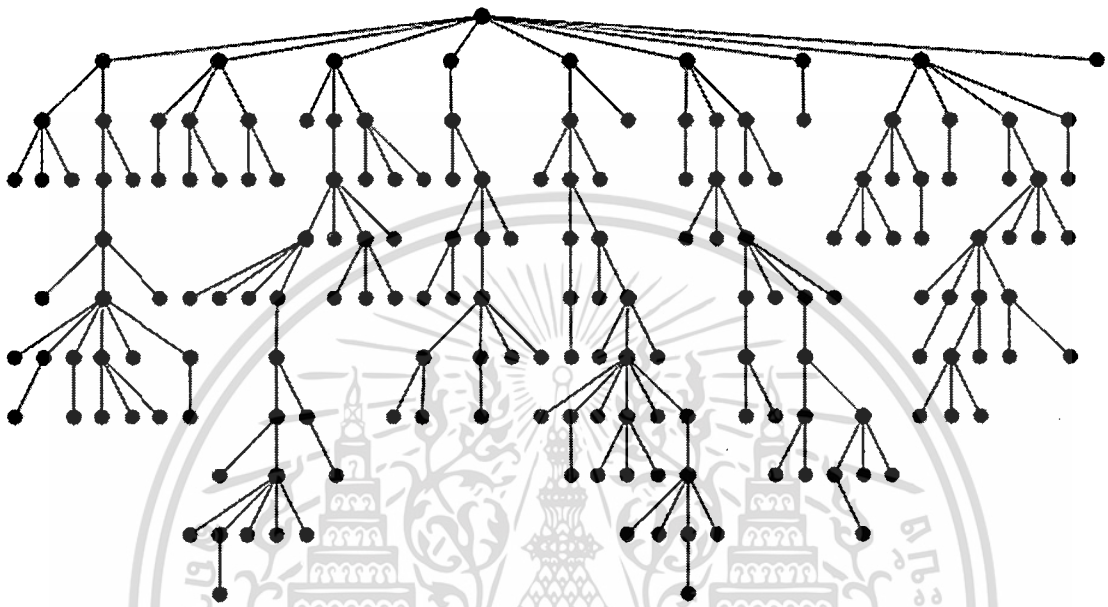
สำหรับในกรณีของอัลกอริธึม LCH-RCS ทางเลือกที่มีฟังก์ชันมูลค่าเป็นศูนย์จะมีโอกาสถูกเลือกมากขึ้น หากตัวเลือกนั้นถูกแนะนำด้วยฟังก์ชันฮิวริสติก(กรณี 1) ทั้งนี้หากพบว่าทางเลือกที่ถูกแนะนำมีฟังก์ชันค่า Q เป็นศูนย์ก็จะมีโอกาสถูกเลือกน้อยกว่า ตัวเลือกที่ถูกแนะนำที่เอเอจนต์เคยเรียนรู้ผ่านมาเนื่องจากการปรับปรุงในสมการ 4.2 (ทำให้เพิ่มอัตราร้อยละในกรณี 2 และลดอัตราร้อยละในกรณี 1) ส่วนกรณีที่ฟังก์ชันฮิวริสติกไม่แนะนำก็จะมีโอกาสถูกเลือกน้อยมาก



รูปที่ 5.11 เส้นทางการค้นหาคำตอบที่ใช้อัลกอริธึม LCH-RCS

จากรูปที่ 5.10 เส้นทางการค้นหาคำตอบของอัลกอริธึม LCH-RCS จะใช้การกรีดีบนสถานะเดิมที่พิจารณาฮิวริสติกที่เคยได้เลือกมาก่อนอีกครั้ง ทำให้เส้นทางการค้นหาถูกสะสมเพิ่มมากขึ้นในทิศทางเดิมเสียเป็นส่วนใหญ่ เส้นทางการค้นหาหลักจะเริ่มมีการสะสมค่าที่เพิ่มขึ้นอย่างสังเกตได้(เส้นทึบ) มีการนำน้ำหนักในการเรียนรู้มาก่อนหน้าช่วยตัดสินใจบ่อยครั้งขึ้น กิ่งก้านของเส้นทางการค้นหาแตกแขนงอยู่ในระดับหนึ่ง แต่อาจจะน้อยกว่าหากเทียบกับอัลกอริธึม HAQL-RCS เนื่องจากทิศทางของการค้นหาถูกกำหนดให้ค้นหาจากเส้นทางหลักเสียเป็นส่วนมากและไม่กระจายอยู่ทั่วปริภูมิคำตอบ

ส่วนในกรณี HAQL-RCS เนื่องจากฟังก์ชันมูลค่าQ ที่นำมาใช้ มิได้มาจากน้ำหนักของการเรียนรู้ที่สะสมไว้ ดังนั้นทุกทางเลือกที่ฮิวริสติกแนะนำไม่ว่าทางใด จะมีฟังก์ชันมูลค่าเท่ากับ $Q^* + \eta$ เสมอ(กรณี 1) ดังนั้นโอกาสที่เอเจนต์เลือกเรียนรู้ในเส้นทางใหม่เป็นไปได้บ่อยครั้ง โดยเฉพาะอย่างยิ่งในกรณีที่ฮิวริสติกแนะนำตัวเลือกที่มีจำนวนค่อนข้างมาก โอกาสในการเลือกตัวเลือกที่เคยเรียนรู้มาก็จะลดลง



รูปที่ 5.12 เส้นทางการค้นหาคำตอบที่ใช้อัลกอริธึม HAQL-RCS

จากรูปที่ 5.11 เส้นทางการค้นหาคำตอบของอัลกอริธึม HAQL-RCS จะใช้ลักษณะของผลรวมการกรีดีเท่ากับ $Q^* + \eta$ ทำให้เส้นทางของการค้นหาคำตอบเกิดการกระจายตัวไปทั่วบริเวณปริภูมิคำตอบ เนื่องจากฟังก์ชันมูลค่าที่ถูกปรับปรุงจะเริ่มสะสมจากศูนย์ในเส้นทางใหม่ๆเสมอ ทำให้เกิดเพียงมูลค่าน้อยๆ ขึ้นทั่วปริภูมิการค้นหา ดังจะเห็นจากในรูป ว่าเส้นทางการค้นหามีฟังก์ชันมูลค่าสะสมเพียงไม่มาก(เส้นบาง) และแตกแขนงไปยังโหนดลูกหลายอย่างทวีคูณ

ดังนั้นจึงสรุปได้ว่า กรณีของอัลกอริธึม RL-RCS และอัลกอริธึม LCH-RCS จะมีการใช้เส้นทางการเรียนรู้เดิมค่อนข้างมาก ทำให้การปรับปรุงฟังก์ชันมูลค่าสามารถถูกสะสมด้วยรอยทางปรับมูลค่ากลับขึ้นไปสู่โหนดของสถานะเริ่มต้นด้านบนได้บ่อยครั้ง ลักษณะของแผนภาพแบคอัพค่อนข้างแคบและลึก ซึ่งในทางกลับกันอัลกอริธึม HAQL-RCS จะมีลักษณะของแผนภาพแบคอัพที่กว้าง มีการสุ่มทางเลือกกระจายไปทั่วปริภูมิการค้นหา ดังนั้นแผนภาพแบคอัพเกิดน้ำหนักของการเรียนรู้ที่สะสมอย่างกระจัดกระจายตั้งแต่สถานะเริ่มต้น และในสถานะสุดท้ายจะไม่สามารถปรับปรุงฟังก์ชันมูลค่าด้วยรอยทางปรับมูลค่าได้ถึงสถานะเริ่มต้น ดังนั้นฟังก์ชันมูลค่าที่สะสมอยู่อาจจะยังไม่ใช่ฟังก์ชันมูลค่าQ อย่างแท้จริง แต่ก็อาจจะไม่สำคัญ เนื่องจากอัลกอริธึม HAQL-RCS จะสุ่มทางเลือกใหม่ออกไปเรื่อยๆ ทางเลือกที่ได้จะมีความกว้างมากกว่า ดังนั้นข้อดีของอัลกอริธึม HAQL-RCS จึงมีโอกาสติดอยู่ในปริภูมิแบบโลคอล ได้น้อยกว่าและคำตอบใหม่ที่พบมักจะไม่เกิดจากการปรับปรุงทางเลือกเดิม ทั้งนี้ขึ้นกับคุณภาพของฟังก์ชันฮิวริสติกที่นำมาแนะนำ

การเลือกใช้ฟังก์ชันฮิวริสติกที่แนะนำได้ดี ส่งผลให้คำตอบที่ได้ก็จะมีคุณภาพที่ดีไม่ว่าอัลกอริธึมประยุกต์ประเภทใดๆ แต่ในกรณีฮิวริสติกที่ไม่ดี พบว่าอัลกอริธึม HAQL-RCS กลับไม่ได้ใช้น้ำหนักจากฟังก์ชันมูลค่า Q ในการพิจารณาเลย ตัวเลือกที่ไม่ดีก็จะยังคงมีโอกาสถูกเลือกอยู่เสมอ และจะทำการเรียนรู้ทางเลือกของฮิวริสติก ในเอพิโสดที่ไม่ใช้งานคำแนะนำเท่านั้น ในบางครั้งอาจไม่พบคำตอบที่เหมาะสมจากการใช้ประสิทธิภาพของอัลกอริธึม RL-RCS เพียงอย่างเดียว ซึ่งหากเปรียบเทียบกับอัลกอริธึม LCH-RCS ที่พยายามใช้งานคำแนะนำด้วยน้ำหนักของฟังก์ชันมูลค่า Q ที่สะสมมาและน้ำหนักของคำแนะนำจากฟังก์ชันฮิวริสติกตลอดเวลา จึงไม่จำเป็นต้องปิดคำแนะนำจากฟังก์ชันฮิวริวติกเลย และเพื่อการทดลองว่าค่าฮิวริสติกที่แนะนำไม่ดีพอจะส่งผลการเรียนรู้ของทั้งสองอัลกอริธึม(HAQL-RCS และ LCH-RCS)อย่างไร ในเชิงของการค้นหาคำตอบที่เป็นการหลีกเลี่ยงปริภูมิคำตอบแบบโลคอล จึงเลือกใช้ฟังก์ชันฮิวริสติกประเภทอื่น ซึ่งกรณีนี้จะเลือกใช้เทคนิคความครอบคลุม(Covering Concept) ที่คำนวณคำแนะนำจากผลโหวตจำนวนสูงสุดของตัวข้อมูลที่ถูกครอบลุมด้วยข้อมูลตัวอื่นๆ (ในหัวข้อ4.4.2) ซึ่งเป็นเทคนิคที่ค่อนข้างคล้ายกับอัลกอริธึมการแก้ปัญหา MCSI [5] มาทดสอบ

5.3.2 การทดสอบอัลกอริธึมการแก้ปัญหา MCSI[5] กับชุดข้อมูลมาตรฐานสำหรับสร้างฟังก์ชันฮิวริสติก $\mathcal{H}$

เนื่องจากอัลกอริธึมการแก้ปัญหา MCSI ที่นำเสนอนี้เป็นการประยุกต์เพื่อสร้างฟังก์ชันฮิวริสติกที่มีคำแนะนำสำหรับกระบวนการเรียนรู้แบบเสริมกำลัง เป็นการประยุกต์เทคนิคการลงคะแนนสูงสุดของข้อมูลตัวที่ถูกครอบคลุมจากสมาชิกตัวอื่น(หัวข้อ4.4.2) ที่แตกต่างจากเทคนิคการลงคะแนนสูงสุดของข้อมูลที่ครอบคลุมสมาชิกตัวอื่น(หัวข้อ 2.3) ลักษณะการเลือกสุ่มการกระทำจึงเป็นการสุ่มเพื่อเลือกข้อมูลออกจากเซตโปรโตไทป์แทน ทั้งนี้อัลกอริธึม MCSI (สำหรับอัลกอริธึม RL) จำเป็นจะต้องปรับปรุงตัวเลือกคำแนะนำที่ไม่ก่อให้เกิดการจำแนกข้อมูลผิดพลาด (ไม่แนะนำการกระทำที่สถานะถัดไปไม่สามารถค้นคืนชุดข้อมูลได้ถูกต้อง)

วิธีการระบุข้อมูลที่เป็นกลุ่มของโปรโตไทป์ในลักษณะของการลงคะแนนให้กับข้อมูลตัวที่ครอบคลุมสมาชิกตัวอื่นจำนวนมากที่สุดเป็นลำดับแรก หากคะแนนที่ได้เท่ากัน หลักเกณฑ์ในการพิจาณาจะถือว่าสามารถเลือกข้อมูลตัวใดตัวหนึ่งแทนกันได้ ดังนั้นในการประยุกต์ใช้กับอัลกอริธึมการเรียนรู้แบบเสริมกำลังจึงสมมุติฐานการทดสอบออกเป็น 2 แนวทางคือ เรียงตามลำดับของคะแนนจากก่อนไปหลัง และวิธีการสุ่มเลือกข้อมูลที่มีคะแนนสูงที่สุด (ในกรณีที่มีคะแนนสูงสูงเท่ากันมากกว่า 1 ข้อมูล)ผลการทดสอบอัลกอริธึม MCSI ที่ประยุกต์ทั้ง 2 แนวทางเทียบกับเทคนิคของงานวิจัย[5] ที่ทำการทดสอบกับชุดข้อมูลมาตรฐาน ผลการเปรียบเทียบนำเสนอด้วยจำนวนสมาชิกของเซตสอดคล้องเล็กที่สุดที่พบโดยเฉลี่ย จำนวน 5 รอบการทดลอง และจำนวนสมาชิกขนาดเล็กสุดที่พบ(ค่าต่ำสุด) ดังแสดงในตารางที่ 5.6

ตารางที่ 5.6 การทดสอบอัลกอริธึม MCSI (สำหรับอัลกอริธึมการเรียนรู้แบบเสริมกำลัง) กรณีเลือกตามลำดับและเลือกแบบสุ่มเพื่อนำมาสร้างเป็นฟังก์ชันฮิวริสติก

ชุดข้อมูล	จำนวนสมาชิกในเซตโปโตไทป์					
	MCSI ปกติ[5]	MCSIแบบสุ่ม[5]		$\mathcal{H}$ (MCS ปกติ)	$\mathcal{H}$(MCS แบบสุ่ม)	
		Avg.	Min		เฉลี่ย	(ต่ำสุด)
ไอริส (150)	15	15.00	(14)	16	16.80	(16)
แก้ว (214)	84	84.60	(82)	106	107.25	(99)
อีโคไล (336)	101	100.80	(100)	125	128.20	(108)
สินเชื่อ (1,000)	474	474.00	(471)	644	628.65	(560)
ยีส (1,484)	886	879.6	(879)	1,122	1,109.75	(1,044)
สังเคราะห์1 (250)	34	33.6	(33)	63	51.2	(50)
สังเคราะห์2 (225)	113	113	(113)	104	104	(104)

จากผลการทดลองพบว่า ฟังก์ชันฮิวริสติกดังกล่าวจะให้ผลการแนะนำที่ด้อยกว่าผลการทดลองในงานวิจัย[5] เนื่องจากการปรับปรุงขั้นตอนในการคำนวณส่งผลให้การแนะนำไม่สนับสนุนให้เอเจนต์เลือกข้อมูลออกเป็นจำนวนมากกว่าหนึ่งตัวเพื่อให้อัตราการค้นคืนกลับมาถูกต้องทั้งหมดได้ เพราะการแนะนำที่ไม่ก่อให้เกิดผลรางวัลกับเอเจนต์บนช่วงสถานะหนึ่ง ในกระบวนการเรียนรู้แบบเสริมกำลังก็ไม่อาจถือว่าเป็นการแนะนำที่ดีได้ จึงจำเป็นต้องให้เอเจนต์เลือกตามคำแนะนำให้ได้รับรางวัล(มีค่าเป็นบวก) เท่านั้น

5.3.3 การเปรียบเทียบผลการเรียนรู้ระหว่างอัลกอริธึม RL-RCS, อัลกอริธึม HAQL-RCSและอัลกอริธึม LCH-RCS ที่ใช้ฟังก์ชันฮิวริสติกคำนวณจากการแก้ปัญหา MCSI

จากผลการทดลองในหัวข้อ 5.3.2 ทำให้พบว่าขีดความสามารถในการให้คำแนะนำจากฟังก์ชันฮิวริสติกที่คำนวณจากการแก้ปัญหา MCSI มีขีดความสามารถที่จำกัดในระดับหนึ่ง หากทำมาเปรียบเทียบกับผลการทดลองของการเรียนรู้แบบเสริมกำลังปกติเท่ากับว่าการแก้ปัญหา MCSI มีประสิทธิภาพที่ด้อยกว่าอย่างสังเกตได้ หากนำมาประยุกต์ใช้เป็นคำแนะนำของฟังก์ชันฮิวริสติก ก็จะนำมาซึ่งคำตอบที่มีลักษณะอยู่บนปริภูมิคำตอบแบบโลคอลอย่างแน่นอน

ดังนั้นเพื่อเป็นการทดสอบอัลกอริธึมการเรียนรู้ทางเลือกของฮิวริสติก (LCH-RCS) ว่ามีประสิทธิภาพการรู้จำตัวเลือกที่แนะนำจากฟังก์ชันฮิวริสติก เพื่อหลีกเลี่ยงปริภูมิแบบโลคอล โดยเปรียบเทียบกับอัลกอริธึม HAQL-RCS และทำการเปรียบเทียบกับขีดความสามารถของอัลกอริธึม RL-RCS ที่ไม่ได้ใช้คำแนะนำจากฟังก์ชันฮิวริสติกควบคู่ไปด้วย ผลการเปรียบเทียบนำเสนอด้วยจำนวนสมาชิกของเซตสอดคล้องเล็กที่สุดที่พบโดยเฉลี่ย จำนวน 5 รอบการทดลอง และจำนวนสมาชิกขนาดเล็กสุดที่พบ(ค่าต่ำสุด) ดังแสดงในตารางที่ 5.7

ตารางที่ 5.7 ผลการทดสอบอัลกอริธึม RL-RCS, อัลกอริธึม HAQL-RCS และอัลกอริธึม LCH-RCS ที่ใช้การแก้ปัญหา MCSI สร้างเป็นฟังก์ชันฮิวริสติก

Dataset	RL-RCS		HAQL-RCS		LCH-RCS	
	Avg	Min	Avg	Min	Avg	Min
Iris (150)	10	(10)	12	(12)	11.2	(10)
Glass (214)	82.4	(81)	84	(84)	82.8	(82)
Ecoli (336)	97.2	(95)	99.6	(99)	98.6	(97)
สินเชื่อ (1,000)	470	(461)	479.4	(478)	467.6	(462)
ยีส (1,484)	883	(876)	891.2	(889)	884.6	(882)
สังเคราะห์1 (250)	28	(27)	30.6	(30)	27.6	(27)
สังเคราะห์2 (225)	16	(16)	12	(12)	13.2	(12)

จากผลการทดลองในตารางที่ 5.7 พบว่า ทิศทางการค้นหาที่ใช้การแนะนำด้วยฮิวริสติกจะส่งผลให้เอเจนต์เลือกการกระทำตามคำแนะนำโดยตลอด และทำให้เอเจนต์มีเส้นทางในการหาปริภูมิคำตอบที่อยู่บริเวณโลคอล เอเจนต์จึงจำเป็นต้องลองผิดลองถูกกับตัวเลือกจากสถานะที่เหลืออยู่นั้น หรือใช้การสำรวจเส้นทางอื่นที่ไม่ใช่คำแนะนำจากฟังก์ชันฮิวริสติกจึงจะสามารถพบเซตคำตอบที่ดีกว่าคำตอบที่คำนวณจากฟังก์ชันฮิวริสติกได้ แต่เมื่อเปรียบเทียบประสิทธิภาพในการค้นหาคำตอบของทั้ง 2 อัลกอริธึม(HAQL-RCS และ LCH-RCS) จะสังเกตได้ว่า ตัวเลือกที่ทำการรู้จำเป็นประโยชน์ในการเลือกตัวเลือกอย่างไร เพราะหากเป็นตัวเลือกที่แนะนำและได้คำตอบตอบที่ดี เอเจนต์ก็ควรจะเลือกอีกครั้งแบบกรีดีค่ามากเพื่อเป็นการบังคับโดยนัยว่าเอเจนต์ไม่ควรเลือกตัวเลือกอื่นๆเพื่อสำรวจปริภูมิฮิวริสติกอีกแล้ว (กรีดีตัวเลือกอื่นๆแบบSoftmax) เท่ากับเป็นการ “จำกัดทางเลือก” (Limited Choice) ให้เอเจนต์มีทางเลือกที่ดีในจำนวณที่ลดลงและแตกต่างจากทางเลือกอื่นเพื่อการสำรวจทำให้เซตคำตอบที่ได้จากอัลกอริธึม LCH-RCS สามารถเรียนรู้ทางเลือกของฮิวริสติกที่ดีกว่าอัลกอริธึม HAQL-RCS ถึงแม้ความสามารถจะไม่เทียบเท่ากับคำตอบที่ถูกค้นพบจากอัลกอริธึม RL-RCS แต่ก็เป็นเหตุผลอันเนื่องมาจากประสิทธิภาพการคำนวณตัวเลือกที่แนะนำของฟังก์ชันฮิวริสติกซึ่งสามารถปรับเปลี่ยนให้ดีขึ้นได้

บทที่ 6
สรุปผลการทดลองและข้อเสนอแนะ

6.1 สรุปผลการทดลอง

กระบวนการเรียนรู้แบบเสริมกำลังที่นำมาประยุกต์ใช้แก้ปัญหาการเลือกโปรโตไทป์ มีจุดประสงค์เพื่อใช้ทดสอบประสิทธิภาพการเรียนรู้ในการแก้ปัญหาเพื่อสังเกตการลู่เข้าสู่ค่าการเรียนรู้ที่เหมาะสมของอัลกอริธึมการเรียนรู้แบบเสริมกำลัง เพื่อระบุคำตอบของเซตย่อยสอดคล้องเล็กที่สุดที่ค้นพบ จากการเลือกโปรโตไทป์ในวิทยานิพนธ์เล่มนี้ มีลักษณะการแก้ปัญหาด้วยการเลือกข้อมูลออกจากเซตโปรโตไทป์ทีละ 1 ตัวต่อหนึ่งการกระทำ เพื่อให้ได้สถานะถัดไปที่คาดว่าจะเป็นเซตคำตอบ มีขนาดเล็กลงเรื่อยๆ โดยที่ทุกๆสถานะตลอดเส้นทางการค้นหาในปัจจุบันจะต้องสามารถจำแนกข้อมูลจริงด้วยกฎ 1-NN ได้อย่างถูกต้องเสมอ แต่เนื่องจากอัลกอริธึมการเรียนรู้แบบเสริมกำลังโดยปกติจะใช้ลักษณะการแก้ปัญหาบนสถานะหนึ่งเพื่อเคลื่อนไปยังสถานะถัดไป ทำให้ปัญหาเช่นกระบวนการเลือกโปรโตไทป์ที่มีจำนวนของสถานะและตัวเลือกจำนวนมาก เอเจนต์จะไม่สามารถค้นพบคำตอบที่เหมาะสม หรือสำรวจปริภูมิคำตอบอย่างทั่วถึง ภายใต้ระยะเวลาหรือทรัพยากรหน่วยความจำที่จำกัดได้ และหากข้อมูลจริงที่นำมาทดสอบมีขนาดใหญ่ขึ้น จำนวนสถานะที่เอเจนต์ต้องค้นหาผ่านไปจะเติบโตขึ้นอย่างทวีคูณ ความหลากหลายของเซตคำตอบก็จะเพิ่มมากขึ้น ตลอดจนจำนวนสมาชิกในเซตคำตอบก็จะมีขนาดใหญ่มากขึ้น ทรัพยากรที่ใช้ในการคำนวณไม่ว่าจะเป็นทางด้านเวลา, ทรัพยากรหน่วยความจำ ก็จะใช้ในปริมาณที่เพิ่มขึ้นตามไปด้วย

ดังนั้นกระบวนการปรับปรุงเพื่อเพิ่มประสิทธิภาพจึงถูกพัฒนาในสองแนวทางคือ การเพิ่มความสามารถในการค้นหาแบบโลคอล (อัลกอริธึม RL-RCS) ที่สามารถเคลื่อนกลับมาเรียนรู้ซ้ำบนสถานะที่คาดว่าได้เลือกการกระทำที่ผิดไป ได้บ่อยครั้งเท่าที่ต้องการจนกว่าจะพบสถานะที่ดีกว่าเซตคำตอบ ณ ปัจจุบัน ดังจะเห็นได้จากผลการทดลองที่ 5.2 และเส้นทางการค้นหาในหัวข้อ 5.4.1 ลักษณะการกระจายตัวของเซตผลลัพธ์ที่ได้จากการเรียนรู้ด้วยอัลกอริธึม RL-RCS กระจายอยู่ในกลุ่มที่มีจำนวนสมาชิกในเซตขนาดเล็ก และมีเส้นทางการค้นหาลึกลงไปในแนวดิ่งของปริภูมิคำตอบ ในอีกแนวทางหนึ่งเป็นการเพิ่มประสิทธิภาพการเรียนรู้จากคำแนะนำตัวเลือกการกระทำที่คำนวณจากฟังก์ชันฮิวริสติก (อัลกอริธึม HAQL) ด้วยลักษณะการกรีดฟังก์ชันมูลค่าสูงสุดตามที่ฟังก์ชันฮิวริสติกแนะนำให้บนสถานะหนึ่งๆ ทำให้ทิศทางของการแก้ปัญหามีเซตคำตอบที่มีขนาดของสมาชิกลดลงอย่างต่อเนื่องภายในหนึ่งเอพิโสด ดังจะเห็นได้จากผลการทดลองที่ 5.2 พิสูจน์ว่ากลุ่มของเซตคำตอบที่ดีจะมีจำนวนสมาชิกขนาดเล็ก แต่ทว่าการกระจายตัวของกลุ่มคำตอบจะกว้างมากหรือน้อยขึ้นกับความถี่ในการใช้งานคำแนะนำว่าบ่อยครั้งเพียงใด แต่ถ้าหากเป็นการแก้ปัญหาบนสถานะขนาดใหญ่เช่นกระบวนการเลือกโปรโตไทป์นี้ ควรจะใช้งานคำแนะนำให้บ่อยครั้งที่สุดเท่าที่เป็นไปได้เนื่องจากสถานะของคำตอบที่ดีจะอยู่ค่อนข้างลึกบนปริภูมิ ซึ่งหากคำแนะนำไม่สามารถแนะนำให้ผ่านลงไปถึงปริภูมิบริเวณนั้นได้ หรือช่วงของการไม่ใช้คำแนะนำเกิดการสำรวจที่ไม่ดีเกิดขึ้น เซตคำตอบที่พบก็อาจไม่ใช่เซตคำตอบที่เหมาะสมอย่างแท้จริง

ลักษณะเด่นของอัลกอริธึม RL-RCS คือใช้ลักษณะของการย้อนกลับมาลงผิดลองถูกบนสถานะที่ต้องการเรียนรู้ซ้ำ ทำให้เอเจนต์สามารถย้อนกลับมาลองผิดลองถูกได้บ่อยครั้ง โดยที่

ไม่ต้องประยุกต์ใช้แนวคิดภายนอกที่ค่อนข้างจะบริโภคทางด้านทรัพยากรหน่วยความจำหรือทรัพยากรเวลาในการคำนวณ ส่วนลักษณะเด่นของอัลกอริธึม HAQL คือใช้ลักษณะของการผ่านไปยังสถานะที่ดีอยู่ตลอดเวลาทำให้ทิศทางของการค้นหาจะถูกสะสมอยู่ในเส้นทางที่ค่อนข้างเป็นประโยชน์ต่อการเรียนรู้คือสามารถนำฟังก์ชันมูลค่าที่ได้กลับมาพิจารณาน้ำหนักในการกรีดฟังก์ชันมูลค่าสูงสุดได้เมื่อไม่ใช้งานคำแนะนำจากฟังก์ชันฮิวริสติก แต่อาจจะต้องแลกกับเวลาที่สูญเสียไปในการคำนวณคำแนะนำหรือทรัพยากรหน่วยความจำที่ใช้บันทึก ซึ่งถือว่าคุ้มค่าเนื่องจากเส้นทางการค้นหาที่ได้เรียนรู้เป็นการเรียนรู้ในตัวเลือกที่ดี ไปตลอดช่วงสถานะที่ฟังก์ชันฮิวริสติกแนะนำได้ และไม่ต้องสละเวลาเพื่อการลองผิดลองถูกอย่างไร้ทิศทาง

เมื่อนำจุดเด่นของทั้งสองอัลกอริธึมมาประยุกต์เข้าด้วยกัน สิ่งที่พบจากการทดลองคือเส้นทางของการค้นหาจะเป็นไปตามทิศทางที่ฟังก์ชันฮิวริสติกแนะนำและเมื่อไรก็ตามที่ขีดความสามารถของฟังก์ชันฮิวริสติกไม่สามารถแนะนำได้ก็จะใช้ประสิทธิภาพของการค้นหาแบบโลคอลทำให้ผลของคำตอบที่พบ มีจำนวนสมาชิกในเซตคำตอบที่เล็กกว่าเซตคำตอบที่พบจากการทดลองที่ใช้เพียงอัลกอริธึมอย่างใดอย่างหนึ่ง (ในการทดลองที่ 5.3.1)

แต่ทั้งนี้หากพิจาณาถึง ฟังก์ชันมูลค่าที่สะสมตลอดปริภูมิคำตอบ เมื่อเปรียบเทียบระหว่างอัลกอริธึม RL-RCS และอัลกอริธึม HAQL-RCS พบว่าการกรีดฟังก์ชันมูลค่าสูงสุดในอัลกอริธึม (HAQL-RCS) ไม่สอดคล้องกับการสะสมฟังก์ชันมูลค่าจริง เนื่องจากการพิจารณาค่าสูงสุดในสมการ 3.1 ไม่ใช่ค่าสูงสุดแต่เป็นการชักนำตัวเลือกด้วยผลรวมที่ฟังก์ชันฮิวริสติกแนะนำด้วยค่า $Q^* + \eta$ ดังนั้น เอเจนต์จะทำการเลือกตัวเลือกที่แนะนำด้วยอัตราส่วนของความน่าจะเป็นเท่าๆกันเสมอ ทำให้มีโอกาสบ่อยครั้งที่เอเจนต์จะทำการเลือกตัวเลือกอื่นๆที่ไม่ใช่เส้นทางการค้นหาหลัก ฟังก์ชันมูลค่าจึงถูกสะสมอย่างกระจัดกระจายทั่วปริภูมิที่เอเจนต์เคยผ่านไป (ดังเส้นทางการค้นหาในหัวข้อ 5.4.2) และเหตุที่เอเจนต์ไม่สามารถกรีดตัวเลือกจากฟังก์ชันมูลค่าสูงสุดได้จริง จากเหตุผลในการเลือกการกระทำผลลัพธ์ในตารางที่ 5.4 จึงพบว่าอัลกอริธึม HAQL-RCS ใช้ลักษณะจากการกรีดผลรวมของโพลิซี $\pi_{\mathcal{H}}$ เสมอ เหตุผลของการถูกเลือกเพราะฟังก์ชันมูลค่าสูงสุดจริงๆ เกิดขึ้นไม่บ่อยครั้งเท่าที่ควรเมื่อเปรียบเทียบกับอัลกอริธึม RL-RCS และอัลกอริธึม LCH-RCS

เมื่อพิจารณาถึงประสิทธิภาพการเรียนรู้ของอัลกอริธึม LCH-RCS เมื่อเปรียบเทียบกับอัลกอริธึม HAQL-RCS หากใช้ฟังก์ชันฮิวริสติกที่มีประสิทธิภาพผลของเซตคำตอบที่ได้จะไม่แตกต่างกันอันเนื่องจากประสิทธิภาพในการแนะนำเส้นทางการค้นหาคำตอบที่ดีทำให้เอเจนต์รู้จำแต่ตัวเลือกที่ถูกต้องเป็นส่วนมาก แต่หากนำอัลกอริธึมการแก้ปัญหา MCSI มาสร้างเป็นฟังก์ชันฮิวริสติกให้กับระบบการเรียนรู้แบบเสริมกำลังจะพบว่าบ่อยครั้งที่ตัวเลือกของคำแนะนำ ไม่สามารถพาเอเจนต์เข้าสู่ปริภูมิที่เหมาะสมได้ เซตคำตอบที่ได้จึงมีจำนวนสมาชิกขนาดใหญ่กว่า หากเปรียบเทียบกับคำตอบที่ใช้ฟังก์ชันฮิวริสติกแบบอัตราการค้นคืน ด้วยเหตุนี้การปรับปรุงประสิทธิภาพการรู้จำทางเลือกของอัลกอริธึม LCH-RCS จึงได้ทำการจัดเรียงน้ำหนักของการกระทำที่ดีที่เคยเรียนรู้ผ่านมา ส่งผลให้ทิศทางของการเรียนรู้สามารถสะสมฟังก์ชันมูลค่าให้อยู่ในรูปของเส้นทางการค้นหาหลักที่มีทิศทางไม่กระจัดกระจาย (จากเส้นทางการค้นหาในหัวข้อ 5.4.3) เท่ากับว่าเป็นการจำกัดเส้นทางการค้นหาที่ดี (Limited Choice) ให้มีจำนวนตัวเลือกที่ดีลดน้อยลง เอเจนต์จึงมีโอกาสในการค้นหาคำตอบที่เหมาะสมที่อาจจะอยู่ภายใต้เส้นทางการค้นหาที่ฟังก์ชันฮิวริสติกกำลังแนะนำได้พบด้วยคุณสมบัติการค้นหาแบบโลคอลของอัลกอริธึม RL-RCS

และจะเห็นได้ว่าอัลกอริธึม HAQL-RCS ได้ละทิ้งนัยสำคัญของเส้นทางการค้นหาปัจจุบันก่อนเวลาอันควร อันเนื่องมาจากโพลิซีที่มุ่งเน้นการสำรวจปริภูมิที่ฟังก์ชันฮิวริสติกแนะนำตลอดเวลา (จากเส้นทางการค้นหาในหัวข้อ 5.4.2)

การเลือกใช้ฟังก์ชันฮิวริสติกจะส่งผลโดยตรงต่อประสิทธิภาพในการค้นหาคำตอบ หากฟังก์ชันที่ใช้มีความสามารถในการคำนวณตัวเลือกที่แนะนำได้กว้างมากพอที่บรรจุ กลุ่มของคำตอบที่เหมาะสมอยู่ภายใน หรือกรองตัวเลือกที่ไม่สำคัญออก ก็จะช่วยแนะนำให้เอเจนต์สามารถค้นพบคำตอบหรือเข้าใกล้คำตอบที่เหมาะสมได้ยิ่งขึ้น แต่ในทางกลับกันหากคำแนะนำที่ใช้เป็นกลุ่มคำตอบที่อยู่ในปริภูมิคำตอบที่ดีเฉพาะจุดหรือตัดตัวเลือกที่ดีทิ้งไป เอเจนต์ก็จะต้องใช้เวลาประมาณหนึ่งในการเรียนรู้เพื่อแก้ไข หรือหลบเลี่ยงไปยังคำตอบที่ดีกว่า ซึ่งหากเปรียบเทียบกับกระบวนการเรียนรู้แบบเสริมกำลังแบบปกติที่ใช้การค้นหาแบบสุ่มเพียงอย่างเดียว อาจไม่เพียงพอที่จะค้นพบคำตอบภายในเงื่อนไขที่จำกัดได้ ดังนั้นการประยุกต์ใช้คำแนะนำจึงถือว่าคุ้มค่า เพราะกลไกที่ให้คำแนะนำนั้น จะช่วยจำกัดให้ปริภูมิการค้นหามีขนาดลดลงจากปริภูมิทั้งหมด เหลือเพียงปริภูมิที่ฮิวริสติกแนะนำ และด้วยจำนวนตัวเลือกที่มีมาก หากเป็นกลไกการเรียนรู้ตามปกติ เอเจนต์จะเสียเวลาในการเรียนรู้สถานะที่ไม่ดีโดยไม่จำเป็น ดังนั้น ข้อดีของอัลกอริธึมกระบวนการเรียนรู้แบบเสริมกำลังโดยใช้การเรียนรู้จากทางเลือกของฮิวริสติก (อัลกอริธึม LCH-RCS) คือ การปรับปรุงประสิทธิภาพในการจำกัดเส้นทางของการค้นหาด้วยการสำรวจบนปริภูมิของคำแนะนำซึ่งมีขนาดเล็กกว่า อีกทั้งยังสามารถแก้ไขการกระทำที่ผิดพลาด (เทคนิค RCS) ผนวกกับเทคนิคการรู้จำน้ำหนักของตัวเลือกจากผลรวมของการเรียนรู้ (ฟังก์ชันมูลค่าQ) และคำแนะนำ (ฟังก์ชัน$\mathcal{H}$) อย่างแท้จริง ซึ่งหากพิจารณาการทดลองในบทที่ 5 จะพบว่า อัลกอริธึม LCH-RCS จะให้ผลของการค้นหาคำตอบของเซตโปรโตไทป์ที่มีขนาดเล็กที่สุดทั้งในด้านคำตอบโดยเฉลี่ยและคำตอบที่ดีที่สุดเปรียบเทียบจาก 3 อัลกอริธึม แสดงให้เห็นถึงประสิทธิภาพของการค้นหาที่สามารถชี้นำเอเจนต์ให้ค้นหาเส้นทางของคำตอบเข้าใกล้คำตอบที่เหมาะสมได้ยิ่งขึ้น ในกรณีที่เลือกใช้ฟังก์ชันฮิวริสติกที่มีประสิทธิภาพ และมีประสิทธิภาพในการเรียนรู้เส้นทางของการค้นหาคำตอบที่ดีกว่าอัลกอริธึม RL, อัลกอริธึม HAQL และอัลกอริธึม RL-RCS แต่อย่างไรก็ตาม **ไม่สามารการันตี**ได้ว่าเทคนิคของการใช้คำแนะนำ จะเหมาะสมกับการประยุกต์แก้ปัญหาในทุกๆ กรณี

6.2 ข้อเสนอแนะ

การทดสอบสมมติฐานวิธีประยุกต์ใช้การแก้ปัญหาการเลือกโปรโตไทป์ เข้ากับอัลกอริธึมการเรียนรู้แบบเสริมกำลังคาดการณ์ว่า ฟังก์ชันฮิวริสติกน่าจะมีจุดเด่นตรงที่ช่วยแนะนำให้ผลการเรียนรู้ลู่เข้าสู่คำตอบที่เหมาะสมได้เร็วขึ้น แต่อาจเกิดการแนะนำให้คำตอบลู่เข้าสู่ปริภูมิแบบโลคอลได้ทุกขณะ ซึ่งในการแก้ปัญหาด้วยอัลกอริธึม MCSI ที่ทดสอบกับชุดข้อมูลขนาดเล็ก ขนาดของสถานะไม่มาก กลุ่มของข้อมูลมีความซับซ้อนไม่มากนัก ฟังก์ชันฮิวริสติกที่เลือกใช้ก็ยังส่งผลให้เกิดคำตอบที่มีลักษณะเป็นคำตอบที่ดีเฉพาะจุด

ดังนั้นเพื่อแก้ไข การใช้งานฟังก์ชันฮิวริสติกที่มีลักษณะของการแนะนำคำตอบที่มีลักษณะเป็นปริภูมิคำตอบที่ดีเฉพาะจุด จำเป็นจะต้องแบ่งส่วนที่ความสามารถในการแนะนำที่ดีและไม่ดีออกจากกัน เพื่อให้มีการสุ่มเลือกการกระทำในส่วนที่ไม่ดีนี้เสียใหม่ โดยหวังผลให้ทิศทางในการค้นหาคำตอบในส่วนไม่ดีนี้เกิดแนวโน้มการสะสมฟังก์ชันมูลค่าที่แตกต่างจากทิศทางที่ฟังก์ชันฮิวริสติก

แนะนำ หรือส่วนที่ดีก็สามารถช่วยเพิ่มความน่าจะเป็นในการเลือกการกระทำให้เป็นไปตามที่ฟังก์ชันฮิวริสติกได้ชี้นำ เนื่องจากปัญหาที่มีขนาดของสถานะขนาดใหญ่ อัลกอริธึม RL จะทำงานในลักษณะของการมองตัวเลือกว่าสามารถเลือกตัวเลือกใดได้บ้าง แล้วใช้การกรีดีหรือการเลือกสำรวจเป็นหลัก มิได้มีการขอบเขตตัวเลือก(Limit Choice), การวัดค่าความเหมาะสม (Fitness) หรือการตัดตัวเลือกออก(Prune) แต่อย่างใด ทำให้โอกาสที่จะค้นหาไปในทุกๆ สถานะที่ดีเป็นไปได้ยาก แม้แต่การสุ่มการกระทำแบบปกติก็ไม่สามารถค้นเจอคำตอบได้ ดังนั้นการเลือกใช้ฟังก์ชันฮิวริสติกจึงส่งผลโดยตรงต่อคุณภาพของคำตอบที่ต้องการ

การประยุกต์ใช้ในกระบวนการเรียนรู้แบบเสริมกำลังที่นำเสนอในวิทยานิพนธ์เล่มนี้ จึงใช้แนวคิดที่สามารถปรับเปลี่ยนฟังก์ชันการคำนวณคำแนะนำได้ตามความเหมาะสม เนื่องจากเทคนิคการคำนวณคำแนะนำ อาจมีแนวคิดหรือตรรกะที่ดีกว่าเกิดขึ้นได้ในอนาคต และเพื่อให้สามารถกำหนดทิศทางในการปรับปรุงประสิทธิภาพในการพัฒนาต่อยอดแนวคิดออกเป็นสองแนวทางคือ ปรับปรุงประสิทธิภาพของกลไกภายในของกระบวนการเรียนรู้แบบเสริมกำลัง หรือแนวทางการปรับปรุงกลไกของการคำนวณแนวคิดคำแนะนำจากฟังก์ชันฮิวริสติก

เอกสารอ้างอิง

[1] R. S. Sutton and A. G. Barto, **"Reinforcement Learning: An Introduction."** MIT Press, Cambridge, MA, 1998.

[2] C. J. C. H. Watkins, **"Learning with Delayed Rewards."** Ph.D. dissertation, Cambridge University, Cambridge, 1989.

[3] R. A. C. Bianchi, C. H. C. Ribeiro and A. H. R. Costa, **"Accelerating autonomous learning by using heuristic selection of actions."** Journal of Heuristics 14(2), 135–168 (2008)

[4] E. Anantapornkit and B. Kruatrachue, **"Reinforcement Learning Algorithm for the Minimal Consistent Subset Identification."** International Conference on Machine Learning and Data Analysis, 2009, pp. 61-65.

[5] B. V. Dasarathy, **"Minimal Consistent Set (MCS) Identification for Optimal Nearest Neighbor Decision Systems Design."** IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 3, 1994, pp. 511-517.

[6] Department of Information and Computer Science, University of California, Irvine. **"UCI Machine Learning Repository"** [Online]. Available: http://archive.ics.uci.edu/ml/. 1998.

[7] R. S. Sutton, **"Learning to Predict by the Methods of Temporal Differences."** Machine Learning, Vol. 3, 1988, pp. 9-44.

[8] C. J. C. H. Watkins and P. Dayan, **"Q-Learning"**, Machine Learning 8, 1992, pp. 279-292.

[9] B. V. Dasarathy, **"Nearest Neighbor (NN) Nom: NN Pattern Classification Techniques."** Los Alamitos, CA: IEEE Computer Society Press, 1991.

[10] P. E. Hart, **"The Condensed Nearest Neighbor Rule"** IEEE Transactions on Information Theory, Vol. 14, 1968, pp. 515-516.

[11] R. A. Mollineda, F. J. Ferri, and E. Vidal. **"Merged-based prototype selection for nearest neighbor classification"**, Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI2000), Orlando, USA, July 2000, pp. 640-645.

[12] R. O. Duda and P. E. Hart, **"Pattern Classification and Scene Analysis"**, Wiley Interscience Publication, 1973.

[13] P. N. Tan, M. Steinbach and V. Kumar, **"Introduction to Data Mining"**, Addison-Wesley, 2006.

[14] R. S. Sutton, **“Generalization in Reinforcement Learing: Successful Examples Using Sparse Coarse Coding,”** Advances in Neural Information Processing Systems 8, MIT Press, Reading, MA, 1996, pp. 1038-1044.

[15] E. Anantapornkit, **"Minimal consistent subset selection using reinforcement learning"** M.D. dissertation, King mongkut's institute of technology ladkrabang, 2009

[16] N. Batir, **“Very accurate approximations for the factorial function,”** J. Math. Inequal. 4(2010), no.3, pp. 335–344.

ภาคผนวก

ภาคผนวก ก.

ข้อมูลสังเคราะห์

ก.1 ข้อมูลสังเคราะห์1

ข้อมูลสังเคราะห์1 เป็นข้อมูลที่สร้างขึ้นเพื่อทดสอบประสิทธิภาพของกระบวนการเรียนรู้แบบเสริมกำลัง ประกอบไปด้วยข้อมูลทั้งสิ้น 250 ตัว โดยมีรายละเอียดดังตารางที่ ก.1

ตารางที่ ก.1 คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
1	1	3	87
2	1	3	88
3	1	4	54
4	1	5	9
5	1	5	10
6	1	5	50
7	1	5	52
8	1	5	96
9	1	8	45
10	1	10	1
11	1	10	10
12	1	10	50
13	1	10	90
14	1	11	14
15	1	11	16
16	1	11	51
17	1	12	7
18	1	12	46
19	1	12	95
20	1	13	15
21	1	13	92
22	1	13	94
23	1	13	97
24	1	14	52
25	1	14	92
26	1	15	6
27	1	15	87

ตารางที่ ก.1 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
28	1	16	7
29	1	16	43
30	1	16	53
31	1	24	32
32	1	24	69
33	1	26	38
34	1	27	70
35	1	28	65
36	1	28	72
37	1	29	30
38	1	29	37
39	1	30	30
40	1	30	63
41	1	30	70
42	1	31	25
43	1	31	74
44	1	32	29
45	1	33	23
46	1	33	66
47	1	36	24
48	1	36	27
49	1	36	67
50	1	36	74
51	1	42	6
52	1	43	48
53	1	43	53
54	1	43	94
55	1	45	85
56	1	46	54
57	1	46	92
58	1	47	18
59	1	47	88
60	1	48	83
61	1	49	7
62	1	49	18

ตารางที่ ก.1 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
63	1	49	54
64	1	50	10
65	1	50	50
66	1	50	90
67	1	51	8
68	1	51	91
69	1	52	92
70	1	53	2
71	1	53	5
72	1	53	54
73	1	54	12
74	1	54	46
75	1	55	11
76	1	55	45
77	1	56	50
78	1	56	87
79	1	56	93
80	1	57	52
81	1	61	69
82	1	63	24
83	1	64	24
84	1	64	35
85	1	66	30
86	1	67	66
87	1	67	72
88	1	67	74
89	1	69	64
90	1	69	65
91	1	70	30
92	1	70	70
93	1	71	67
94	1	73	25
95	1	73	35
96	1	73	65
97	1	75	26

ตารางที่ ก.1 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
98	1	76	32
99	1	79	28
100	1	79	72
101	1	83	54
102	1	84	56
103	1	85	14
104	1	85	86
105	1	86	82
106	1	86	83
107	1	87	45
108	1	87	57
109	1	88	7
110	1	88	9
111	1	88	49
112	1	88	53
113	1	89	17
114	1	90	5
115	1	90	10
116	1	90	50
117	1	90	86
118	1	90	90
119	1	91	3
120	1	91	10
121	1	91	54
122	1	91	82
123	1	93	7
124	1	93	83
125	1	94	83
126	1	95	57
127	1	96	10
128	1	96	94
129	1	97	90
130	1	98	50
131	2	6	35
132	2	6	73

ตารางที่ ก.1 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
133	2	8	72
134	2	9	34
135	2	9	78
136	2	10	30
137	2	10	33
138	2	10	70
139	2	10	71
140	2	10	76
141	2	11	75
142	2	11	78
143	2	12	23
144	2	12	63
145	2	13	30
146	2	13	63
147	2	14	33
148	2	15	27
149	2	16	26
150	2	18	29
151	2	21	12
152	2	23	10
153	2	25	12
154	2	27	5
155	2	27	7
156	2	27	95
157	2	28	16
158	2	28	47
159	2	28	55
160	2	28	95
161	2	29	45
162	2	29	49
163	2	29	87
164	2	30	10
165	2	30	44
166	2	30	50
167	2	30	90

ตารางที่ ก.1 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
168	2	30	96
169	2	31	48
170	2	31	94
171	2	32	10
172	2	32	43
173	2	32	84
174	2	34	44
175	2	34	52
176	2	34	86
177	2	34	88
178	2	34	95
179	2	36	13
180	2	39	13
181	2	43	25
182	2	43	26
183	2	43	28
184	2	44	24
185	2	44	66
186	2	46	28
187	2	46	38
188	2	46	71
189	2	46	76
190	2	47	65
191	2	47	72
192	2	50	30
193	2	50	70
194	2	53	35
195	2	53	78
196	2	55	28
197	2	55	69
198	2	56	75
199	2	57	29
200	2	59	70
201	2	62	93
202	2	63	47

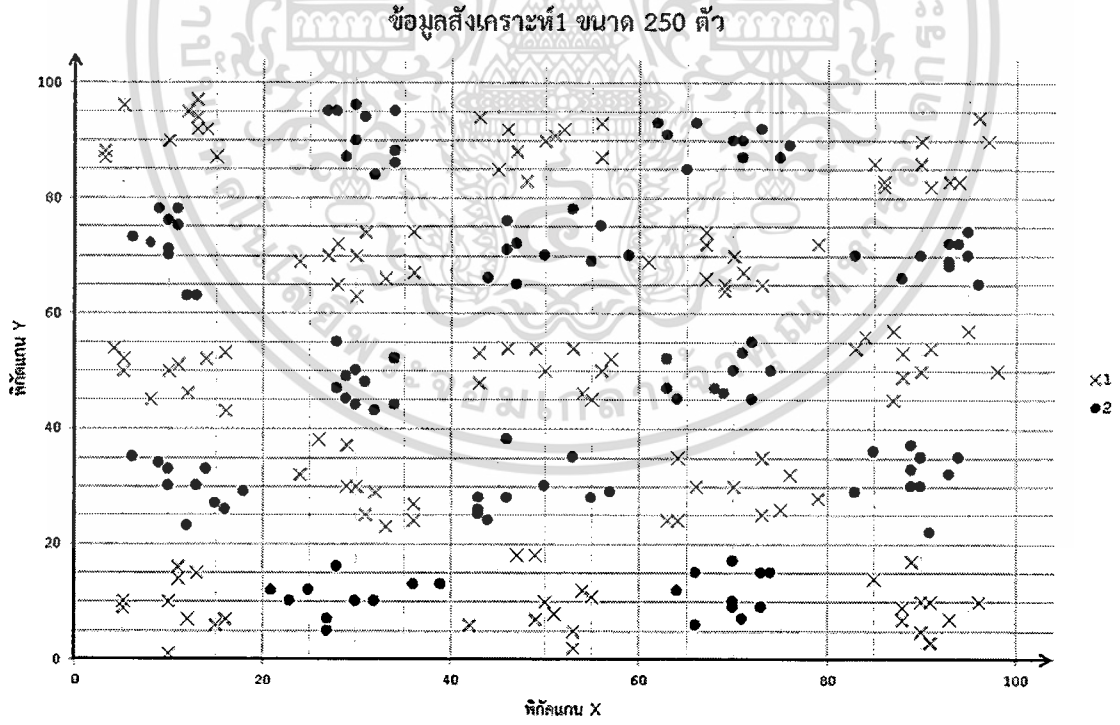
ตารางที่ ก.1 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
203	2	63	52
204	2	63	91
205	2	64	12
206	2	64	45
207	2	65	85
208	2	66	6
209	2	66	15
210	2	66	93
211	2	68	47
212	2	69	46
213	2	70	9
214	2	70	10
215	2	70	17
216	2	70	50
217	2	70	90
218	2	71	7
219	2	71	53
220	2	71	87
221	2	71	90
222	2	72	45
223	2	72	55
224	2	73	9
225	2	73	15
226	2	73	92
227	2	74	15
228	2	74	50
229	2	75	87
230	2	76	89
231	2	83	29
232	2	83	70
233	2	85	36
234	2	88	66
235	2	89	30
236	2	89	33
237	2	89	37

ตารางที่ ก.1 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
238	2	90	30
239	2	90	35
240	2	90	70
241	2	91	22
242	2	93	32
243	2	93	68
244	2	93	69
245	2	93	72
246	2	94	35
247	2	94	72
248	2	95	70
249	2	95	74
250	2	96	65

ลักษณะเฉพาะของข้อมูลสังเคราะห์1 สามารถนำเสนอบนปริภูมิ 2 มิติ ดังรูปที่ ก.1



รูปที่ ก.1 คุณลักษณะเฉพาะของข้อมูลสังเคราะห์1

ก.2 ข้อมูลสังเคราะห์2

ข้อมูลสังเคราะห์2 เป็นข้อมูลที่สร้างขึ้นเพื่อทดสอบประสิทธิภาพของกระบวนการเรียนรู้แบบเสริมกำลัง ประกอบไปด้วยข้อมูลทั้งสิ้น 225 ตัว โดยมีรายละเอียดดังตารางที่ ก.2

ตารางที่ ก.2 คุณลักษณะเฉพาะของข้อมูลสังเคราะห์2

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
1	1	1	1
2	1	1	2
3	1	1	3
4	1	1	4
5	1	1	5
6	1	1	11
7	1	1	12
8	1	1	13
9	1	1	14
10	1	1	15
11	1	2	1
12	1	2	2
13	1	2	3
14	1	2	4
15	1	2	5
16	1	2	11
17	1	2	12
18	1	2	13
19	1	2	14
20	1	2	15
21	1	3	1
22	1	3	2
23	1	3	3
24	1	3	4
25	1	3	5
26	1	3	11
27	1	3	12
28	1	3	13
29	1	3	14
30	1	3	15

ตารางที่ ก.2 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์2

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
31	1	4	1
32	1	4	2
33	1	4	3
34	1	4	4
35	1	4	5
36	1	4	11
37	1	4	12
38	1	4	13
39	1	4	14
40	1	4	15
41	1	5	1
42	1	5	2
43	1	5	3
44	1	5	4
45	1	5	5
46	1	5	11
47	1	5	12
48	1	5	13
49	1	5	14
50	1	5	15
51	1	6	6
52	1	6	7
53	1	6	8
54	1	6	9
55	1	6	10
56	1	7	6
57	1	7	7
58	1	7	8
59	1	7	9
60	1	7	10
61	1	8	6
62	1	8	7
63	1	8	8
64	1	8	9
65	1	8	10

ตารางที่ ก.2 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์2

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
66	1	9	6
67	1	9	7
68	1	9	8
69	1	9	9
70	1	9	10
71	1	10	6
72	1	10	7
73	1	10	8
74	1	10	9
75	1	10	10
76	1	11	1
77	1	11	2
78	1	11	3
79	1	11	4
80	1	11	5
81	1	11	11
82	1	11	12
83	1	11	13
84	1	11	14
85	1	11	15
86	1	12	1
87	1	12	2
88	1	12	3
89	1	12	4
90	1	12	5
91	1	12	11
92	1	12	12
93	1	12	13
94	1	12	14
95	1	12	15
96	1	13	1
97	1	13	2
98	1	13	3
99	1	13	4
100	1	13	5

ตารางที่ ก.2 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์2

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
101	1	13	11
102	1	13	12
103	1	13	13
104	1	13	14
105	1	13	15
106	1	14	1
107	1	14	2
108	1	14	3
109	1	14	4
110	1	14	5
111	1	14	11
112	1	14	12
113	1	14	13
114	1	14	14
115	1	14	15
116	1	15	1
117	1	15	2
118	1	15	3
119	1	15	4
120	1	15	5
121	1	15	11
122	1	15	12
123	1	15	13
124	1	15	14
125	1	15	15
126	2	1	6
127	2	1	7
128	2	1	8
129	2	1	9
130	2	1	10
131	2	2	6
132	2	2	7
133	2	2	8
134	2	2	9
135	2	2	10

ตารางที่ ก.2 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์2

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
136	2	3	6
137	2	3	7
138	2	3	8
139	2	3	9
140	2	3	10
141	2	4	6
142	2	4	7
143	2	4	8
144	2	4	9
145	2	4	10
146	2	5	6
147	2	5	7
148	2	5	8
149	2	5	9
150	2	5	10
151	2	6	1
152	2	6	2
153	2	6	3
154	2	6	4
155	2	6	5
156	2	6	11
157	2	6	12
158	2	6	13
159	2	6	14
160	2	6	15
161	2	7	1
162	2	7	2
163	2	7	3
164	2	7	4
165	2	7	5
166	2	7	11
167	2	7	12
168	2	7	13
169	2	7	14
170	2	7	15

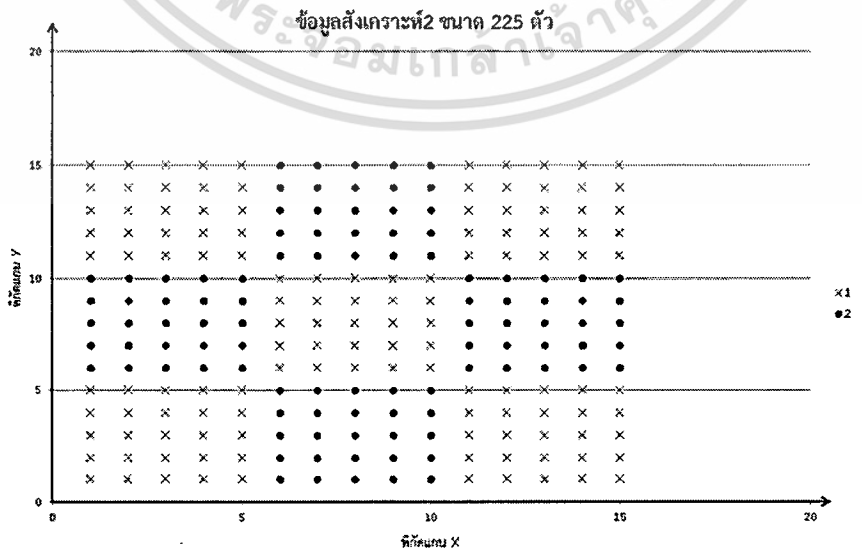
ตารางที่ ก.2 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์2

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
171	2	8	1
172	2	8	2
173	2	8	3
174	2	8	4
175	2	8	5
176	2	8	11
177	2	8	12
178	2	8	13
179	2	8	14
180	2	8	15
181	2	9	1
182	2	9	2
183	2	9	3
184	2	9	4
185	2	9	5
186	2	9	11
187	2	9	12
188	2	9	13
189	2	9	14
190	2	9	15
191	2	10	1
192	2	10	2
193	2	10	3
194	2	10	4
195	2	10	5
196	2	10	11
197	2	10	12
198	2	10	13
199	2	10	14
200	2	10	15
201	2	11	6
202	2	11	7
203	2	11	8
204	2	11	9
205	2	11	10

ตารางที่ ก.2 (ต่อ) คุณลักษณะเฉพาะของข้อมูลสังเคราะห์2

ลำดับที่	ชนิด	พิกัด X	พิกัด Y
206	2	12	6
207	2	12	7
208	2	12	8
209	2	12	9
210	2	12	10
211	2	13	6
212	2	13	7
213	2	13	8
214	2	13	9
215	2	13	10
216	2	14	6
217	2	14	7
218	2	14	8
219	2	14	9
220	2	14	10
221	2	15	6
222	2	15	7
223	2	15	8
224	2	15	9
225	2	15	10

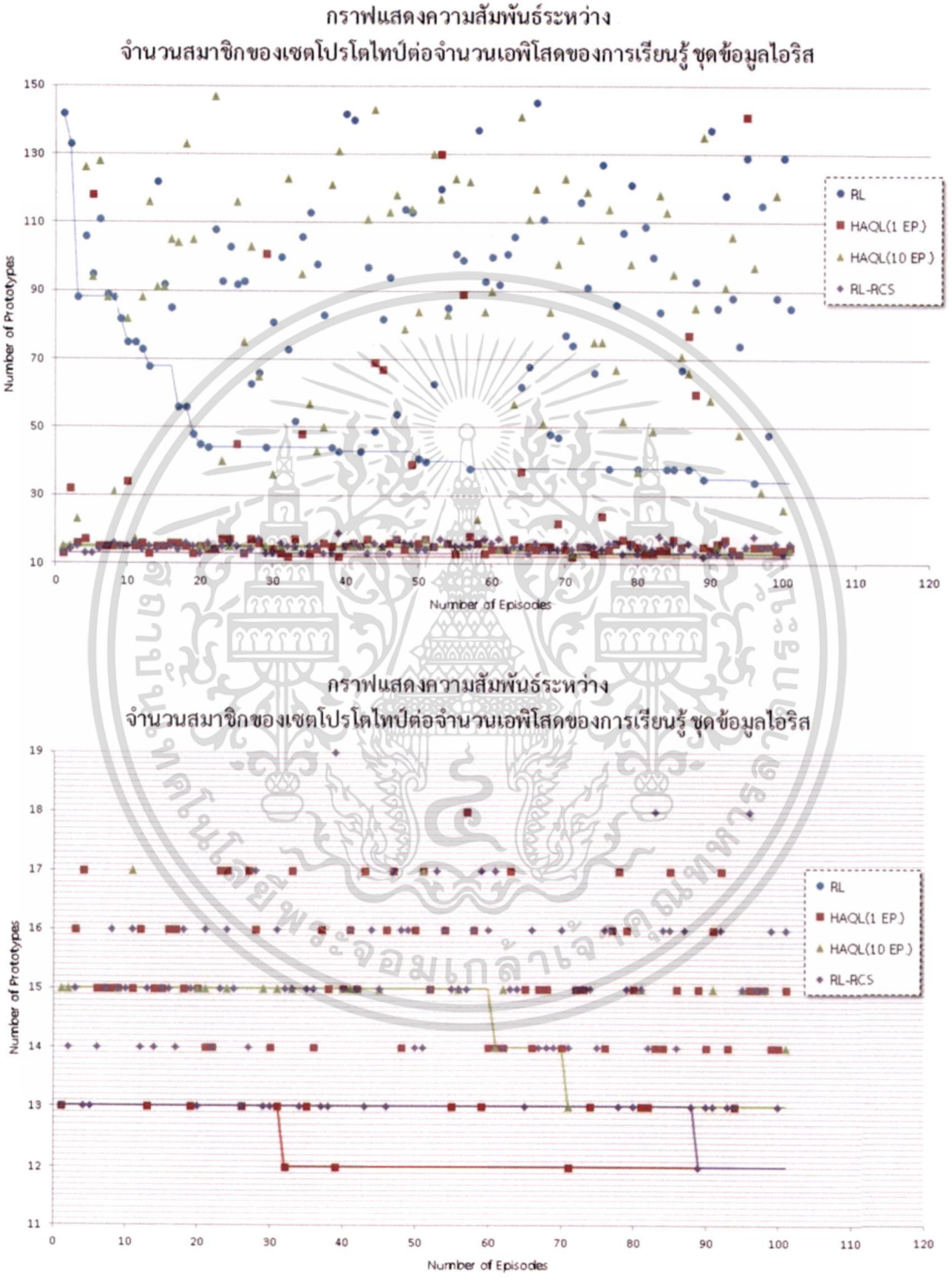
ลักษณะเฉพาะของข้อมูลสังเคราะห์2 สามารถนำเสนอบนปริภูมิ 2 มิติ ดังรูปที่ ก.2



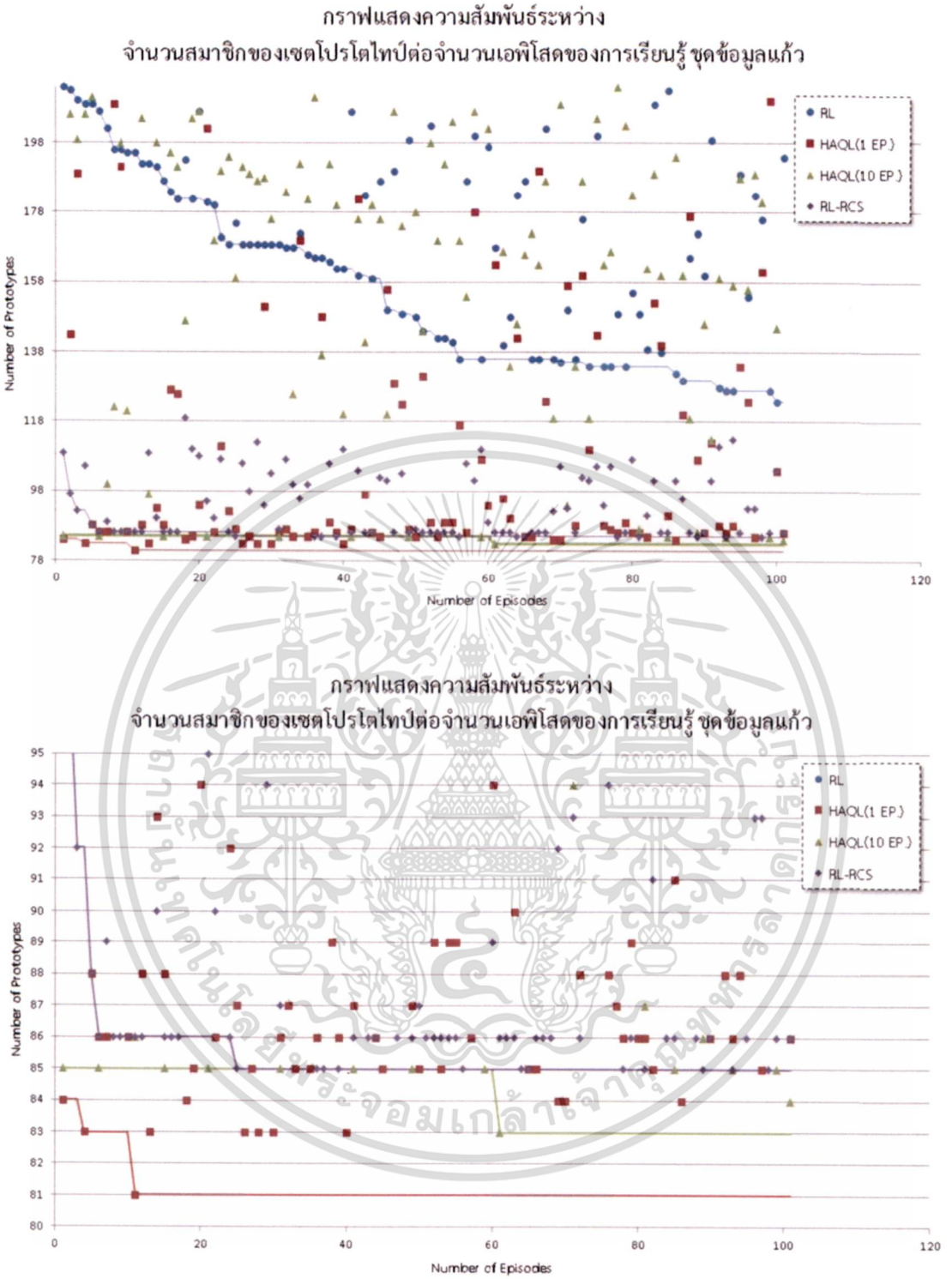
รูปที่ ก.2 คุณลักษณะเฉพาะของข้อมูลสังเคราะห์2

ผนวก ข.

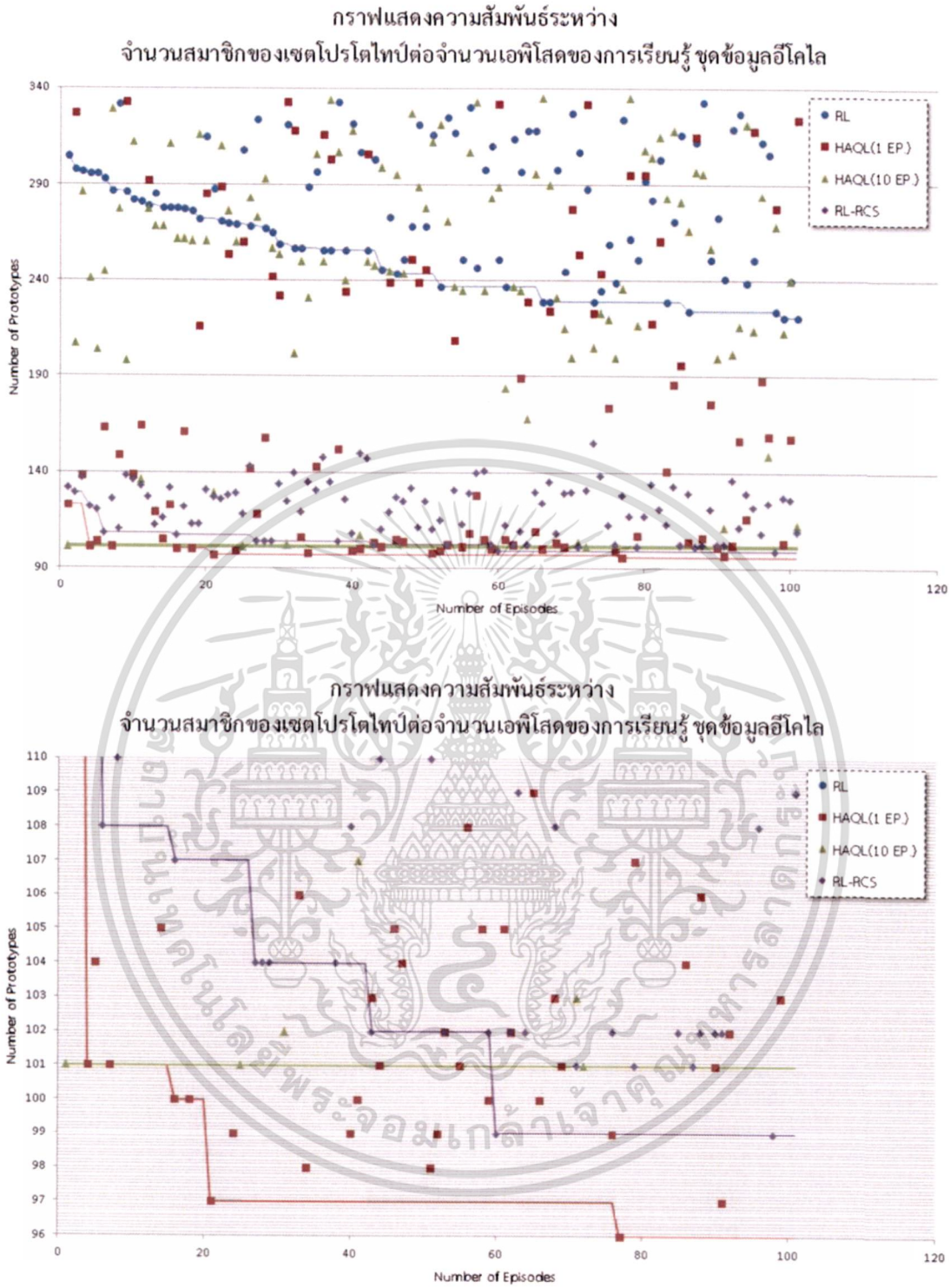
กราฟ 100 เอพิโสดแรกของผลการทดลองที่ 5.2



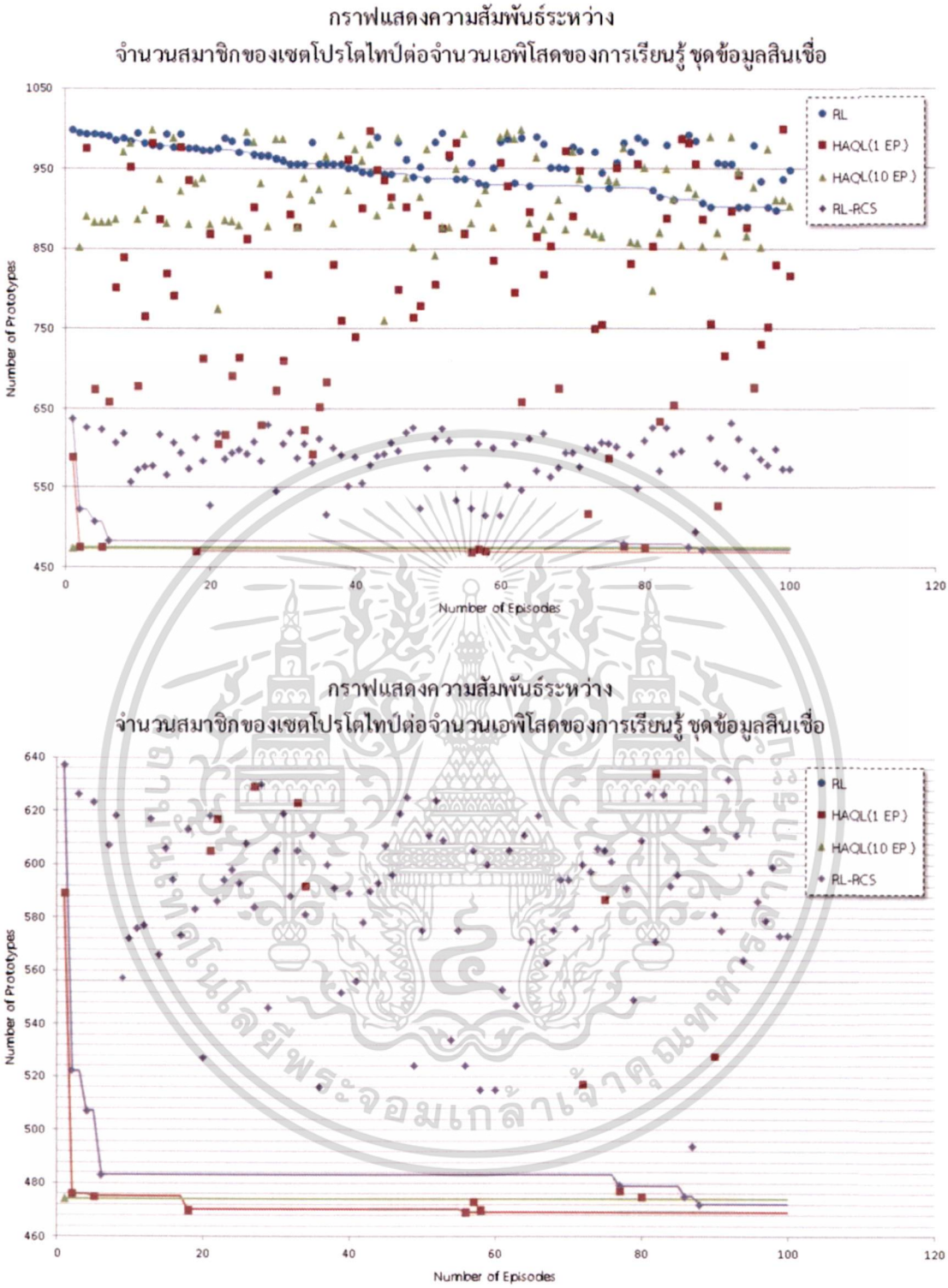
รูปที่ ข.1 กราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลไอริสที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ (บน) และภาพขยาย (ล่าง)



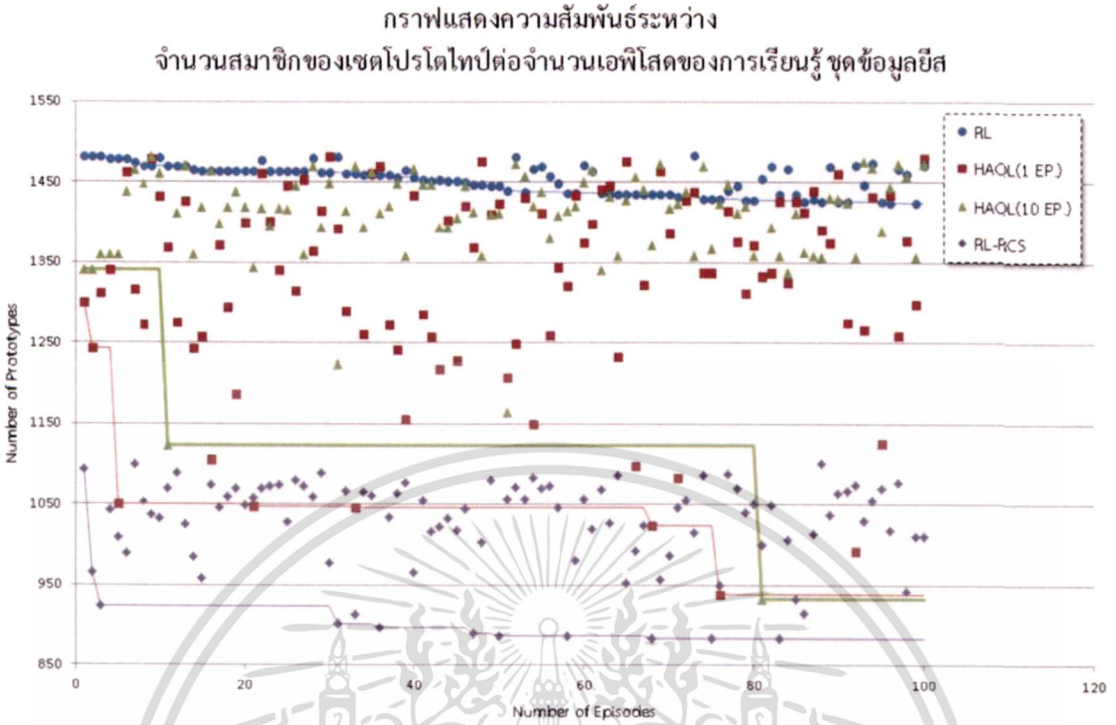
รูปที่ ข.2 กราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลแก้วที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ (บน) และภาพขยาย (ล่าง)



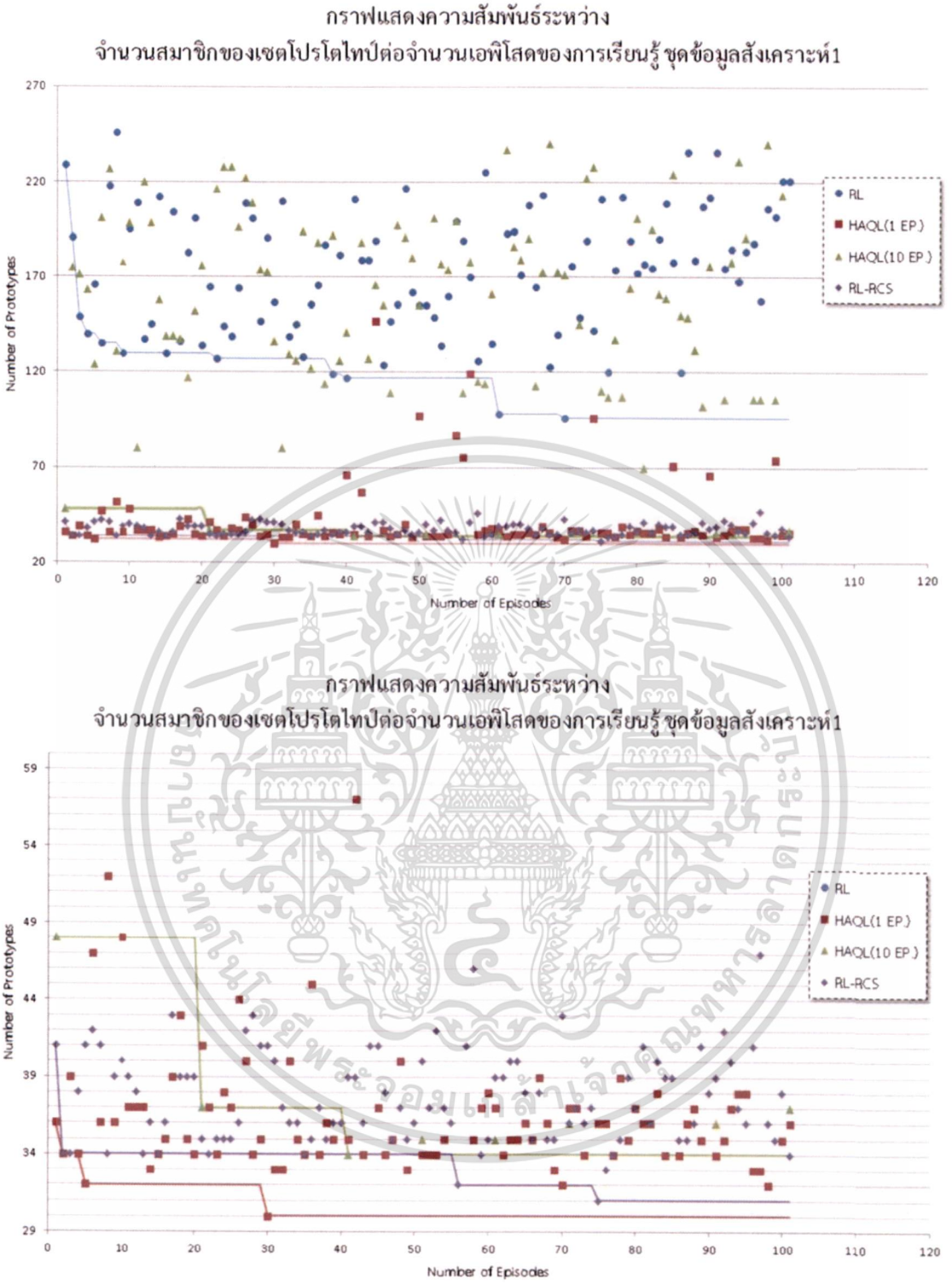
รูปที่ ข.3 กราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลอีโคไลที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ (บน) และภาพขยาย (ล่าง)



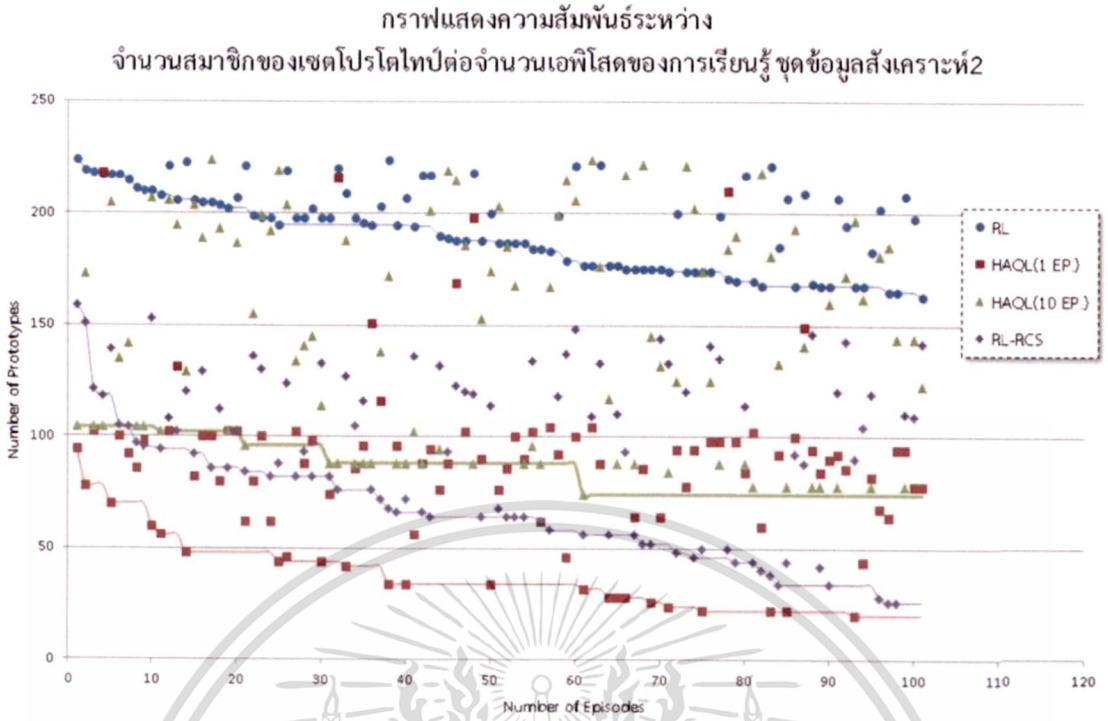
รูปที่ ข.4 กราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลสินเชื่อที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ (บน) และภาพขยาย (ล่าง)



รูปที่ ข.5 กราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลยีสที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ



รูปที่ ข.6 กราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลสังเคราะห์1 ที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ (บน) และภาพขยาย (ล่าง)



รูปที่ ข.7 กราฟเปรียบเทียบจำนวนของโปรโตไทป์ในเซตสอดคล้องเล็กที่สุดของชุดข้อมูลสังเคราะห์2 ที่พบในแต่ละเอพิโสดด้วยอัลกอริธึม RL แบบต่างๆ

ผนวก ค.
แผนภูมิกล่องและกราฟผลการทดลองที่ 5.2

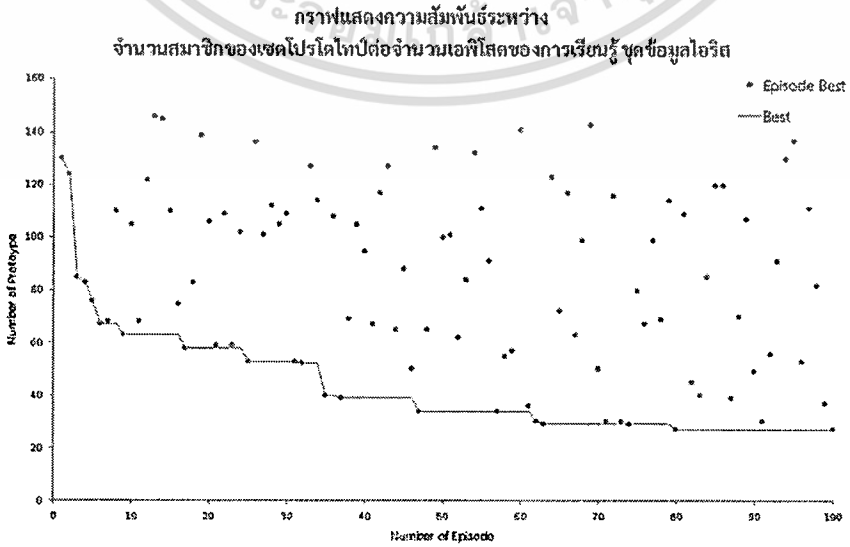
ค.1 แผนภูมิกล่อง (Box plot)

แผนภูมิกล่อง (Boxplot, Box Plot and Box-and-Whisker Diagram) เป็นวิธีการนำเสนอกลุ่มของประชากรที่เป็นข้อมูลเชิงตัวเลข ด้วยข้อมูลสรุปทั้ง 5 ที่เกิดจากการสำรวจ อันได้แก่

- ข้อมูลตัวอย่างที่มีขนาดเล็กสุด (Sampler Minimum, MIN)
- ควอไทล์ที่ 1 (Q1)
- มัธยฐาน (Median, MED) หรือ ควอไทล์ที่ 2 (Q2)
- ควอไทล์ที่ 3 (Q3)
- ข้อมูลตัวอย่างที่มีขนาดใหญ่สุด (Sampler Maximum, MAX)
- และ/หรือ อาจพิจารณากรณี พบค่าผิดปกติ (Outlier) ด้วย

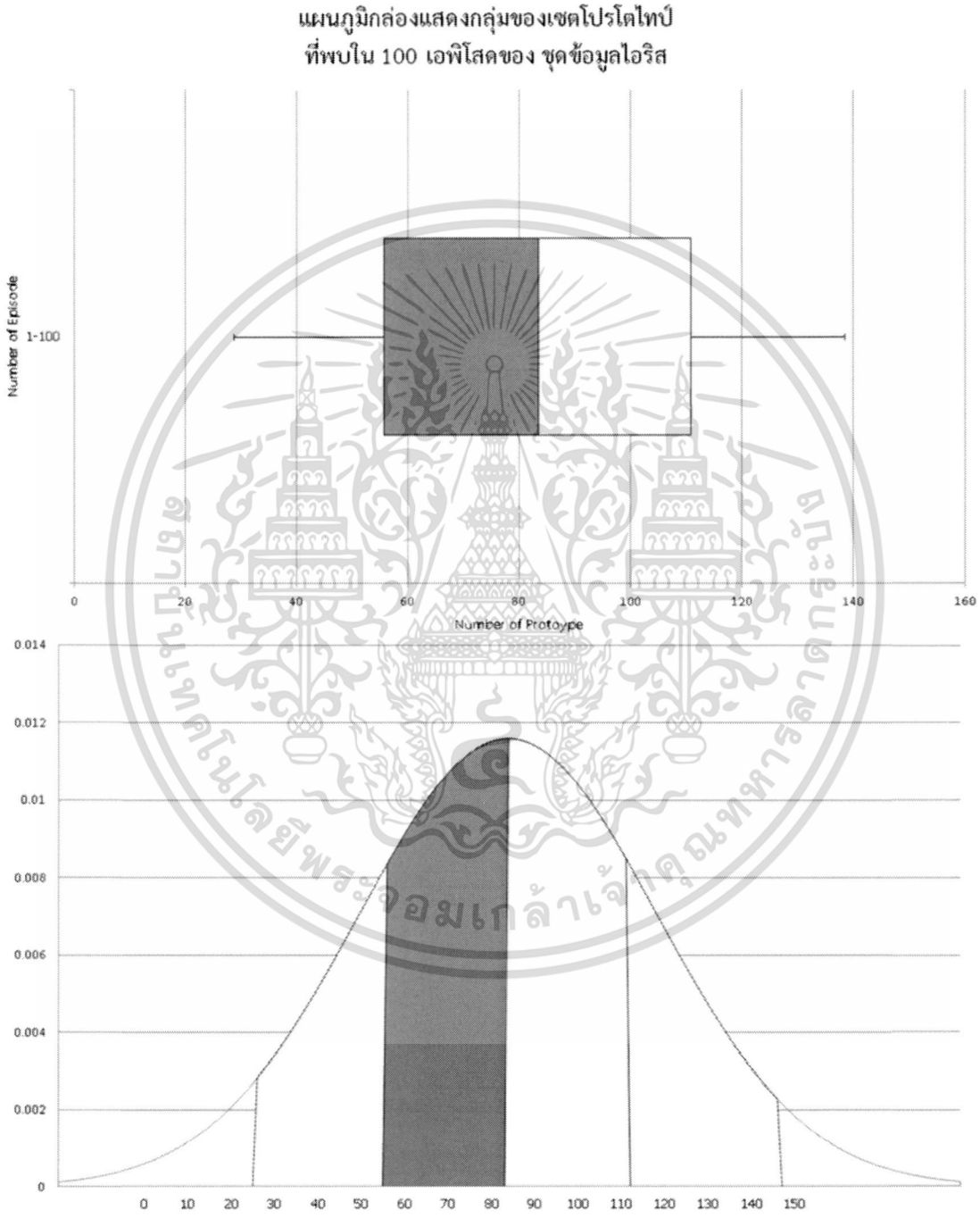
แผนภูมิกล่อง เป็นการนำเสนอความแต่ต่างของประชากร ที่ไม่ได้ใช้สมมุติฐานการกระจายตัวทางสถิติ(Statistical Distribution) เพียงแต่เป็นการแสดงช่วงของความแตกต่างที่พบจากกลุ่มประชากรที่สามารถแสดงด้วยระดับของการกระจายตัว (Degree of Dispersion) และความเบ้ (Skewness) และการกระบุค่าผิดปกติ โดยที่แผนภูมิกล่องสามารถสำเสนอได้ทั้ง 2 แกน ทั้งแกนตั้งและแกนนอน

หากนำเสนอกลุ่มของข้อมูลจากผลการทดลองการเลือกโปรโตไทป์ด้วยอัลกอริธึมการเรียนรู้แบบเสริมกำลังปกติ (อัลกอริธึม RL) ในชุดข้อมูลไอริส โดยทดลองทั้งสิ้น 100 เอพิโสด ผลการเรียนรู้ดังรูป ค.1



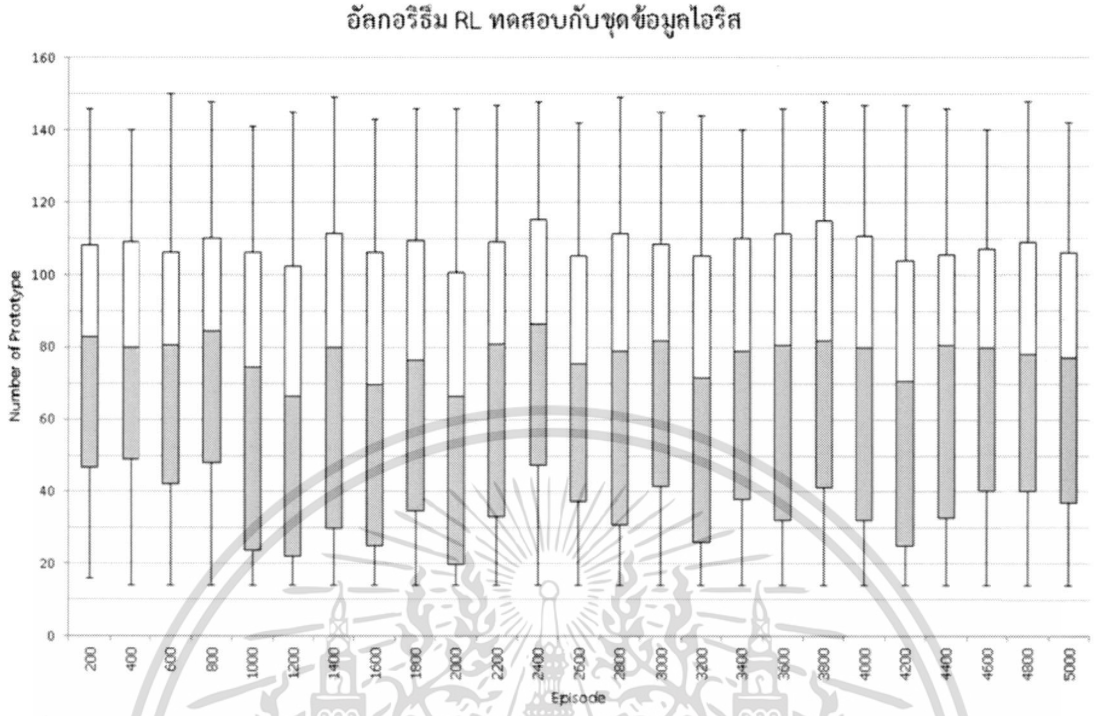
รูปที่ ค.1 ตัวอย่างผลการเรียนรู้ที่ใช้อัลกอริธึม RL

เพื่อนำผลของเซตคำตอบที่ได้ สมมุติให้เป็นกลุ่มประชากร และใช้หลักการทางสถิติ สร้างเป็นแผนภูมิกล่องและแผนภูมิการกระจายตัวแบบโค้งระฆังคว่ำ ได้ดังรูปที่ ค.2 โดยขั้นตอนการคำนวณทั้ง 2 วิธีการจะไม่กล่าวถึงในที่นี้ ผู้อ่านที่มีความสนใจสามารถศึกษาเพิ่มเติมได้จาก หลักการสถิติเชิงพรรณนา (Descriptive Statistics)

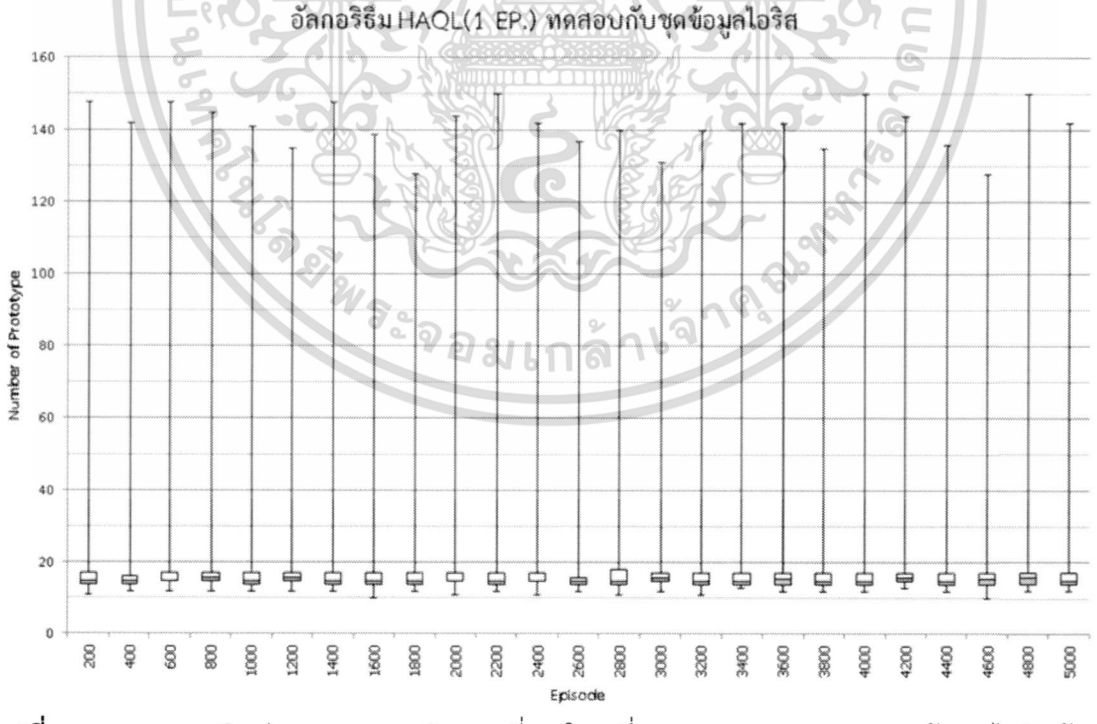


รูปที่ ค.2 **(บน)** แผนภูมิกล่อง (ล่าง) กราฟการแจกแจงปกติ ที่คำนวณจากตัวอย่างผลการเรียนรู้ที่ใช้อัลกอริธึม RL

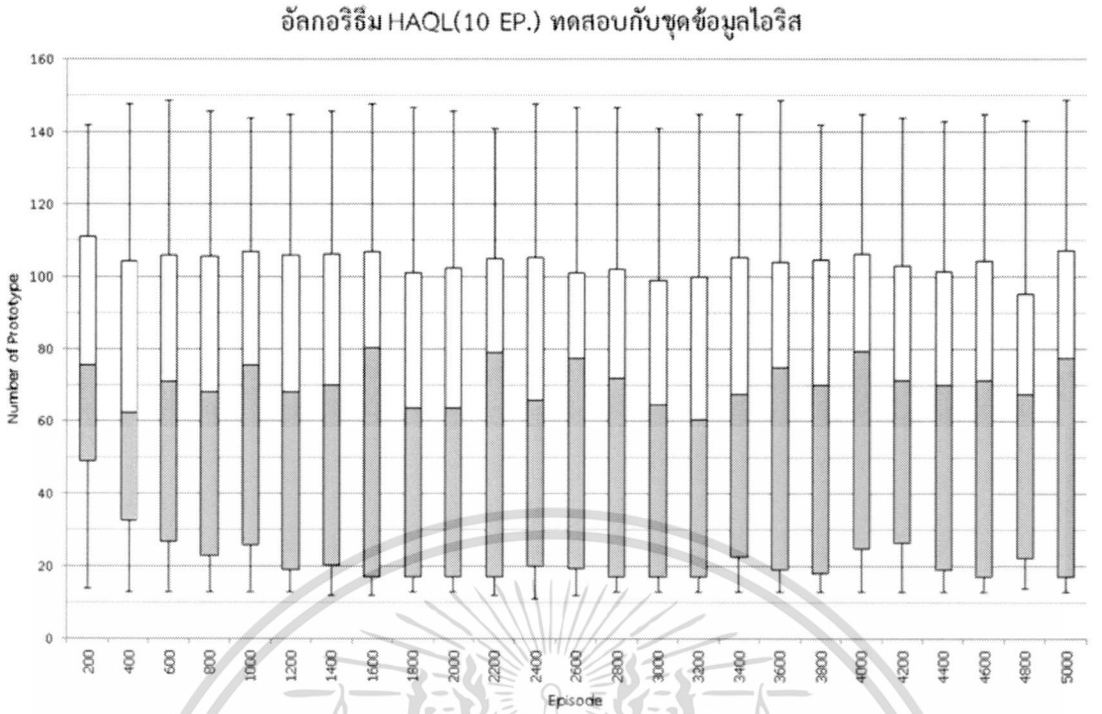
ค.2 ผลการทดลองจากหัวข้อ 5.2



รูปที่ ค.3 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลไอริสด้วยอัลกอริธึม RL

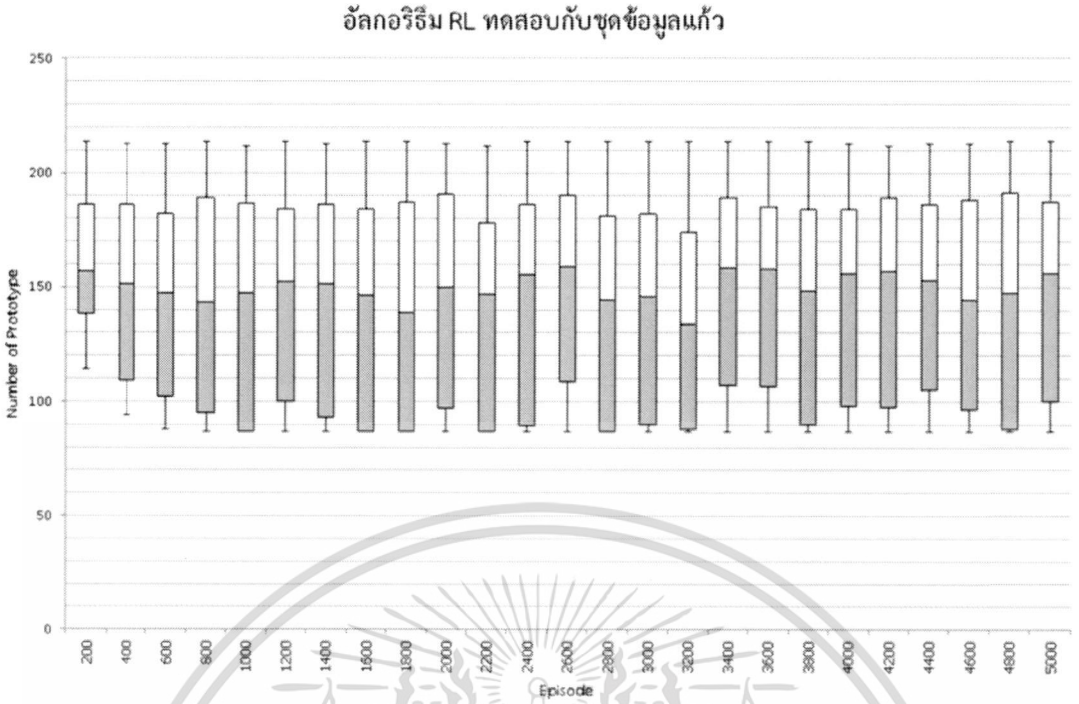


รูปที่ ค.4 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลไอริสด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ เอพิโสด

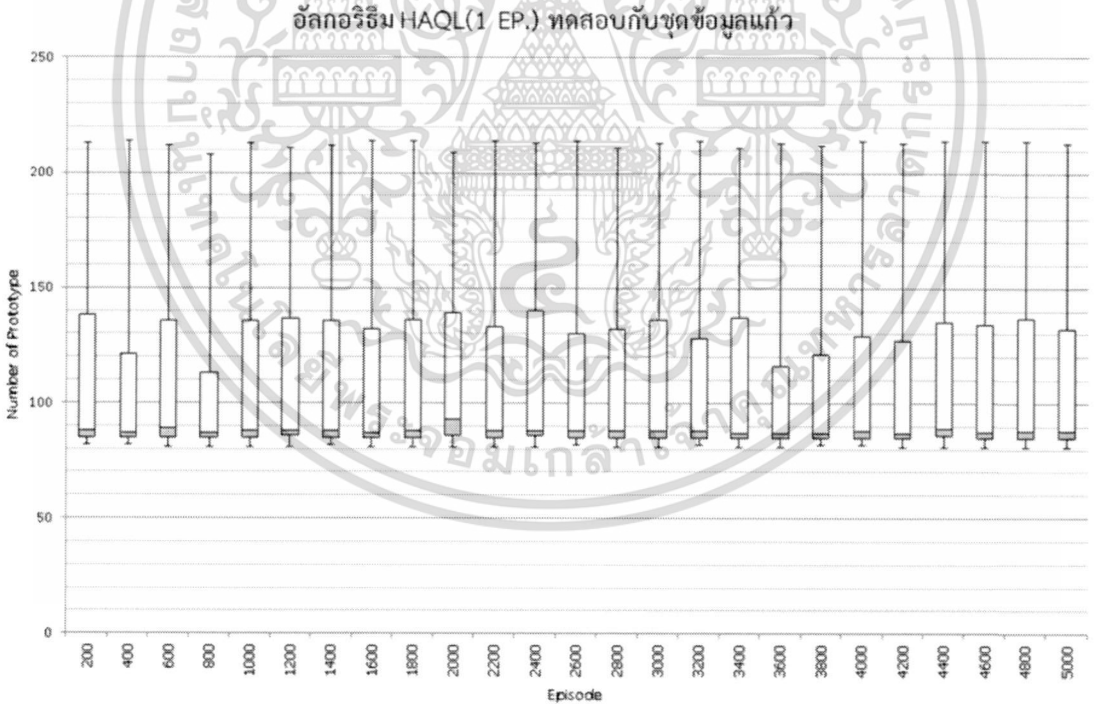


รูปที่ ค.5 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลไอริสด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ 10 เอพิโสด

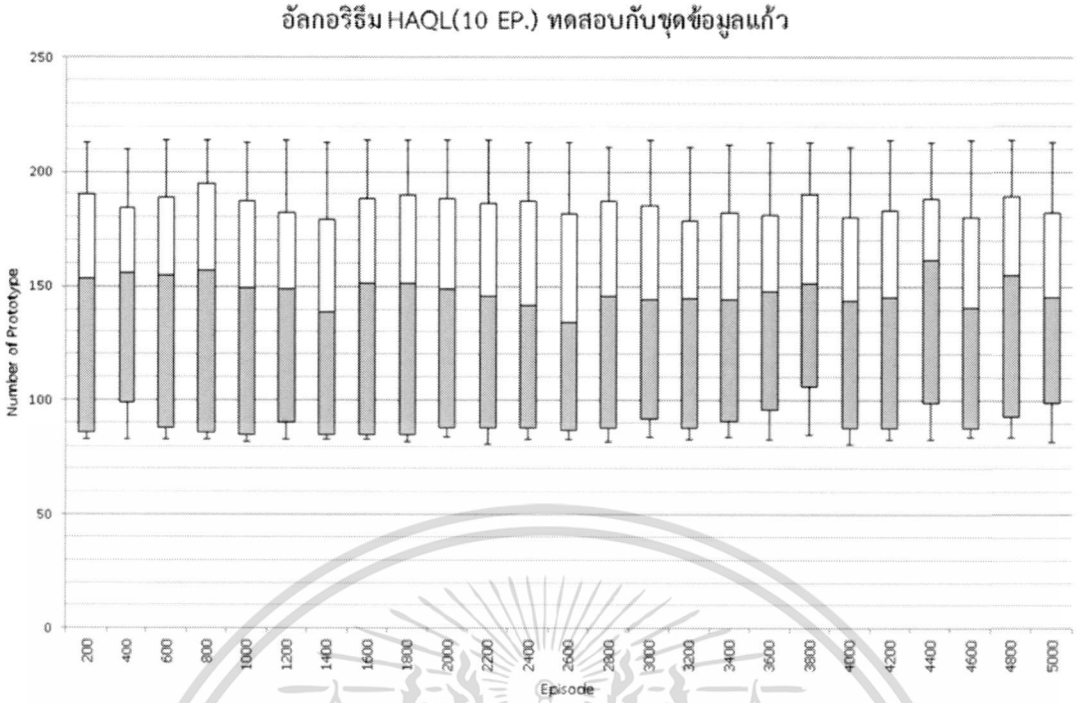
รูปที่ ค.6 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลไอริสด้วยอัลกอริธึม RL-RCS



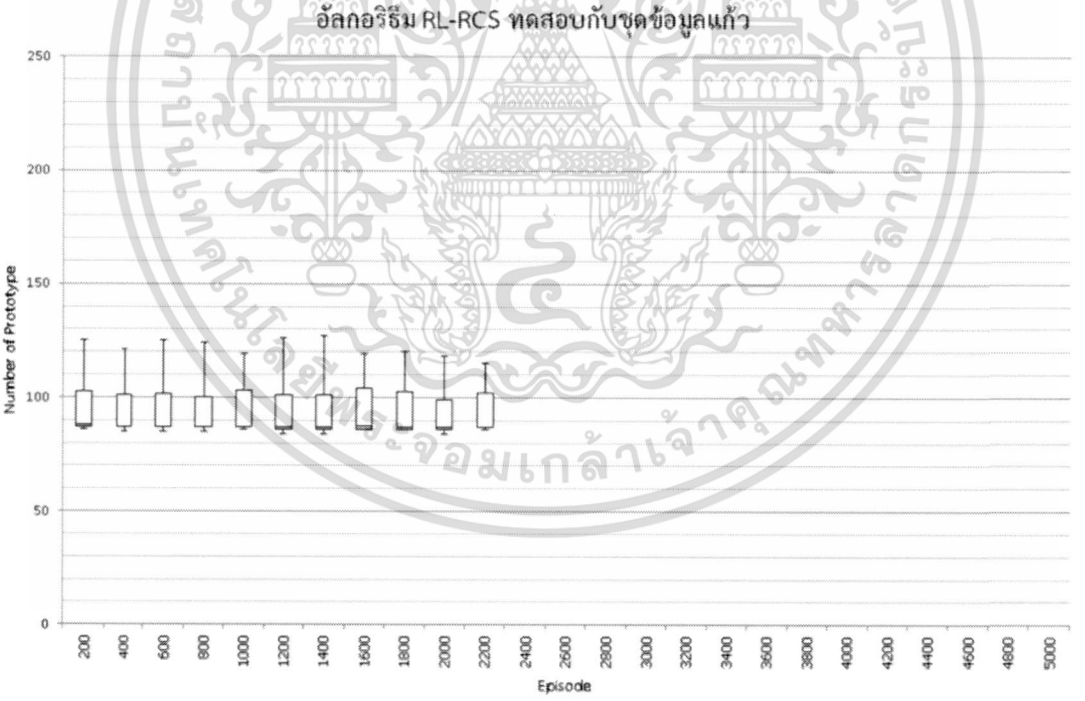
รูปที่ ค.7 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลแก้วด้วยอัลกอริธึมRL



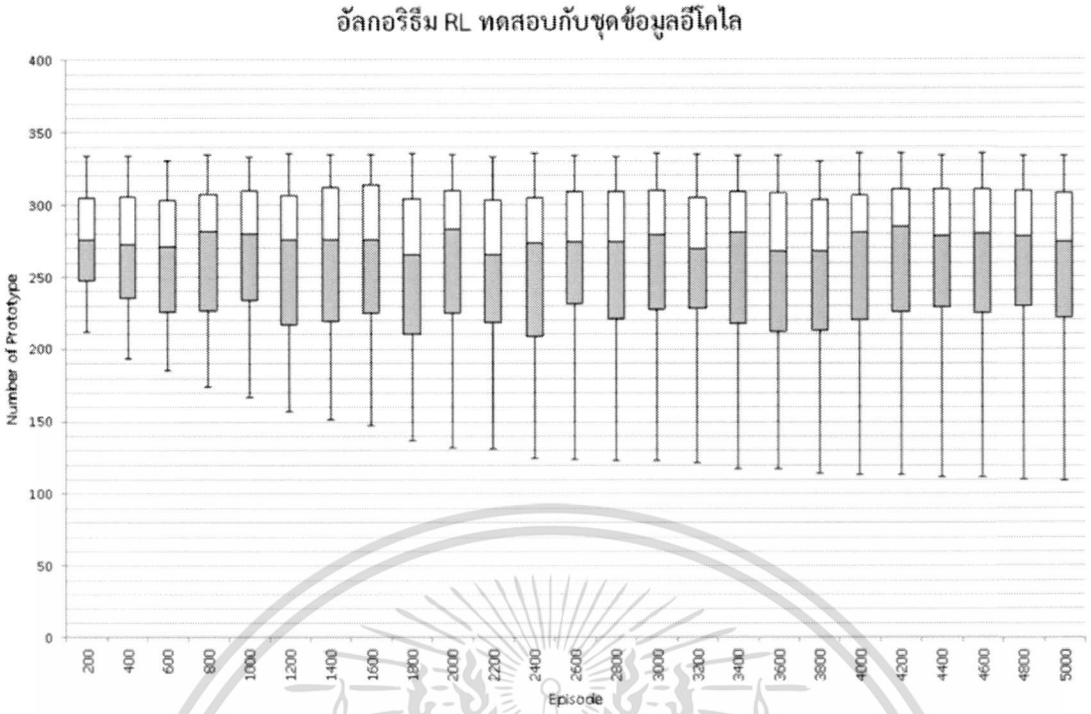
รูปที่ ค.8 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลแก้วด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ เอพิโสด



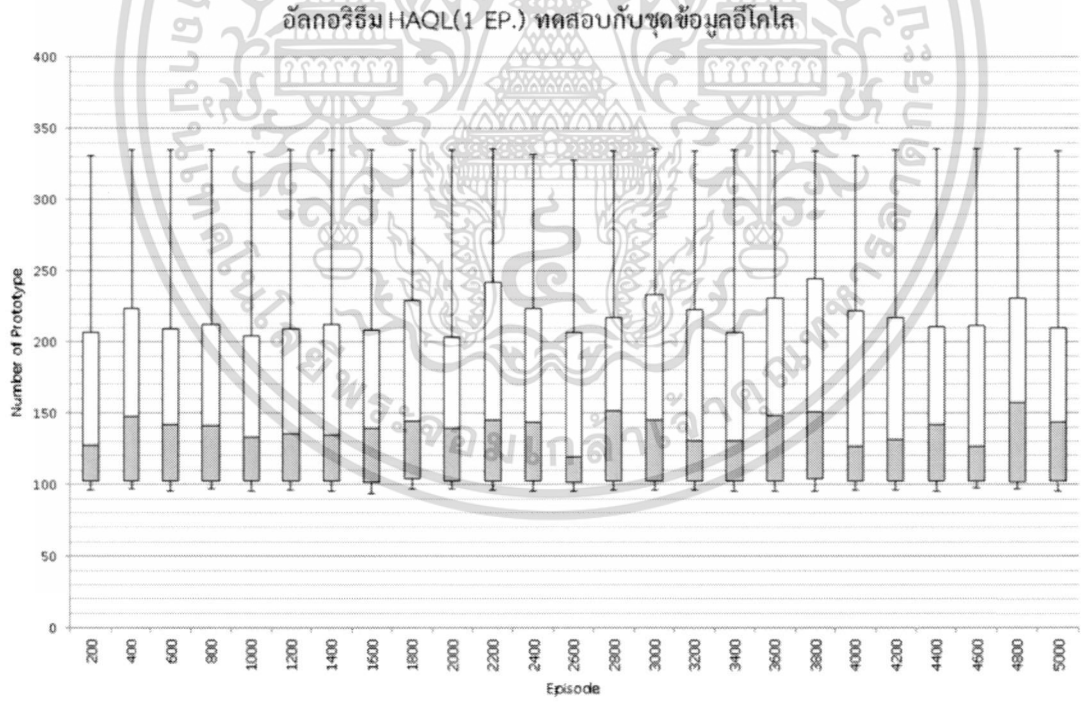
รูปที่ ค.9 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลแก้วด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ 10 เอพิโสด



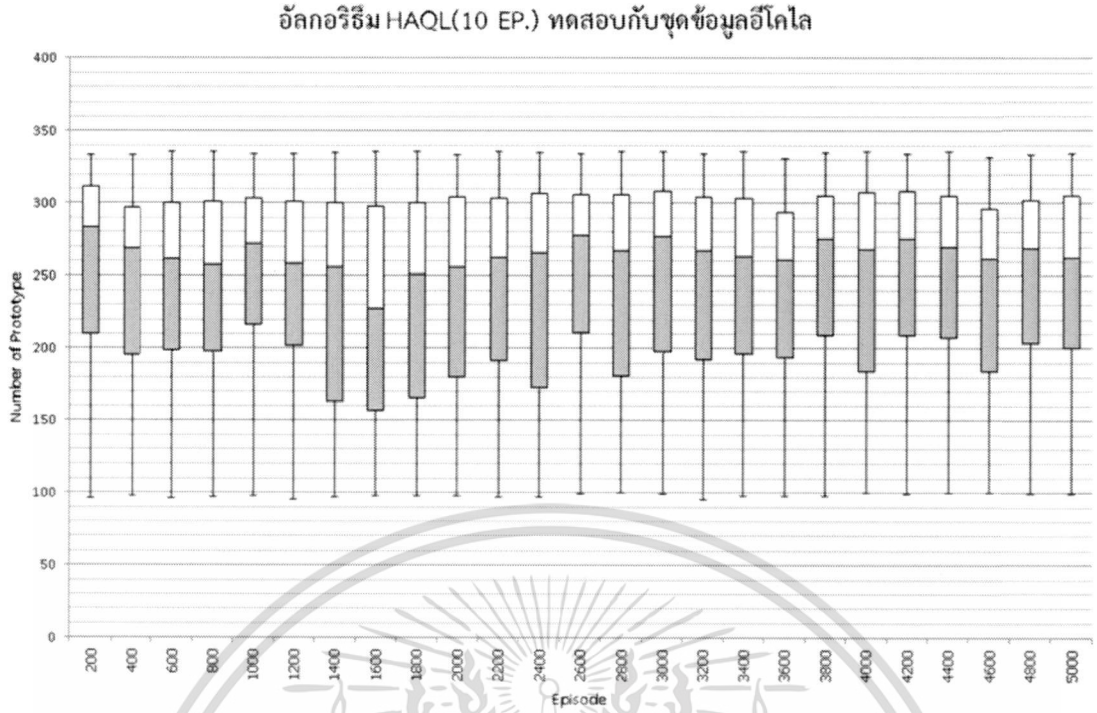
รูปที่ ค.10 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลแก้วด้วยอัลกอริธึม RL-RCS



รูปที่ ค.11 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลอีโคไลด้วยอัลกอริธึม RL

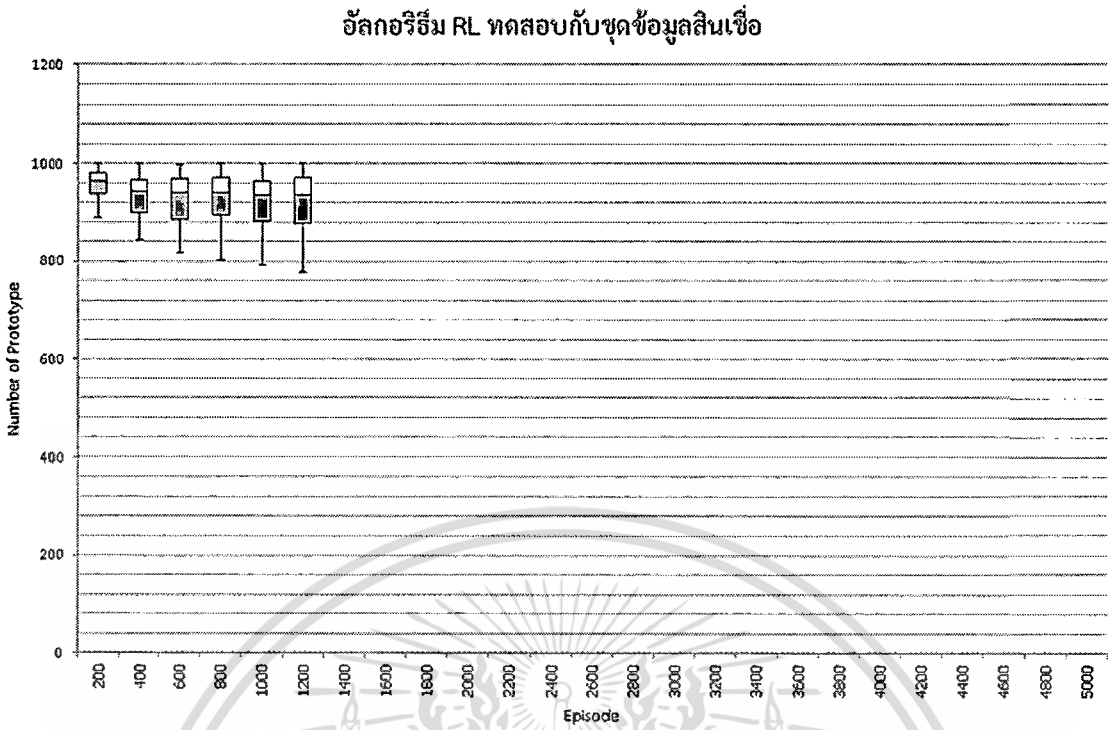


รูปที่ ค.12 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลอีโคไลด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ เอพิโสด

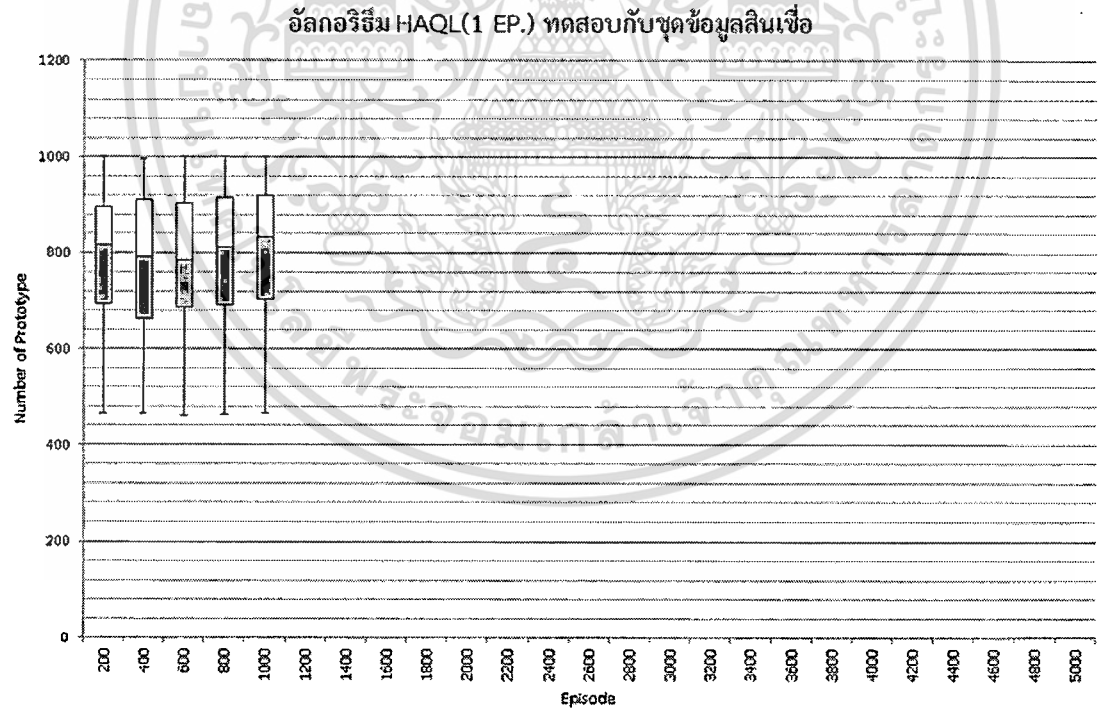


รูปที่ ค.13 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลอีโคไลด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ 10 เอพิโสด

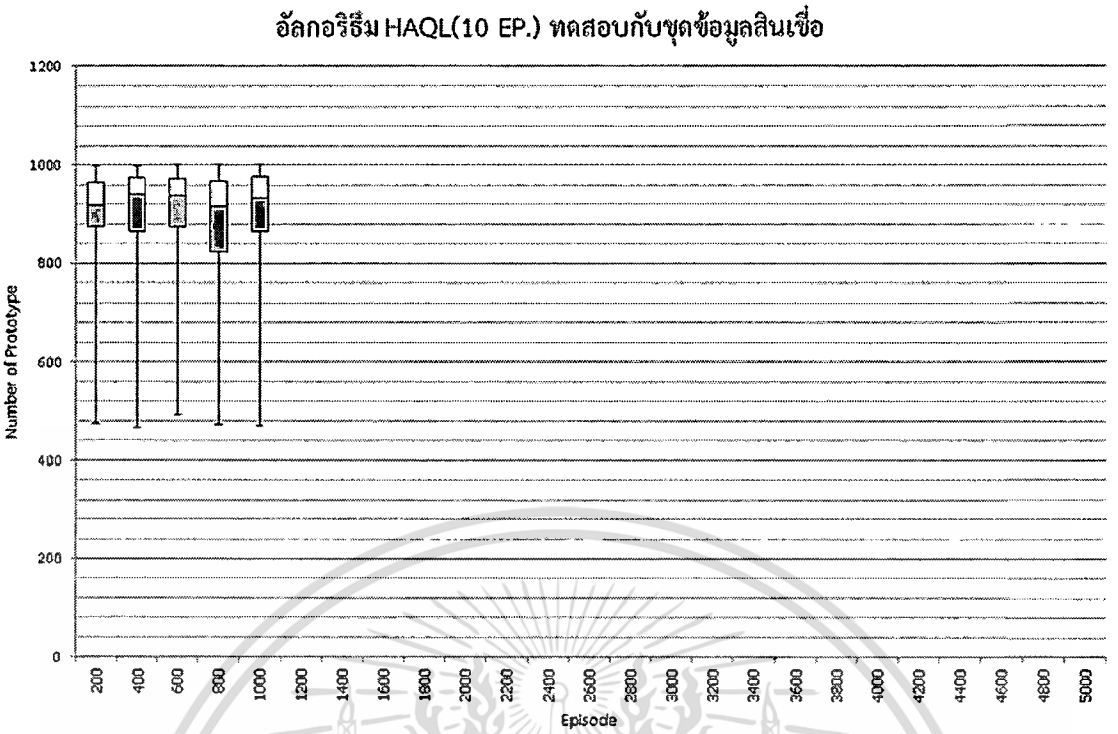
รูปที่ ค.14 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลอีโคไลด้วยอัลกอริธึม RL-RCS



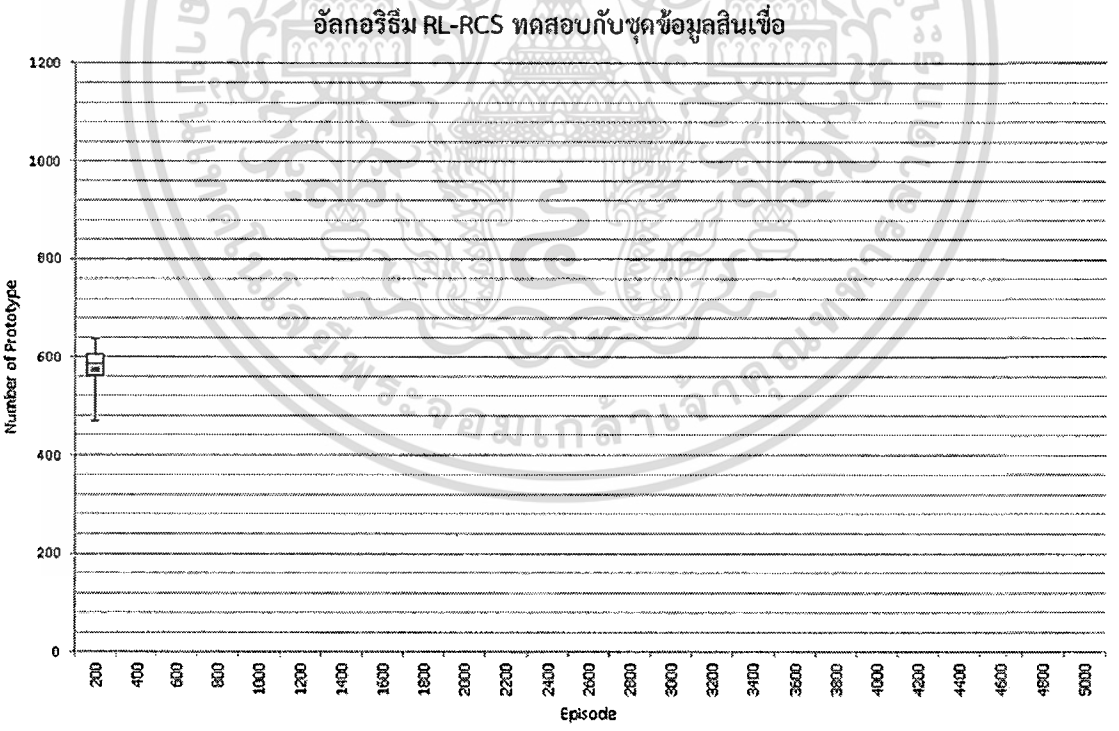
รูปที่ ค.15 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสินเชื่อด้วยอัลกอริธึม RL



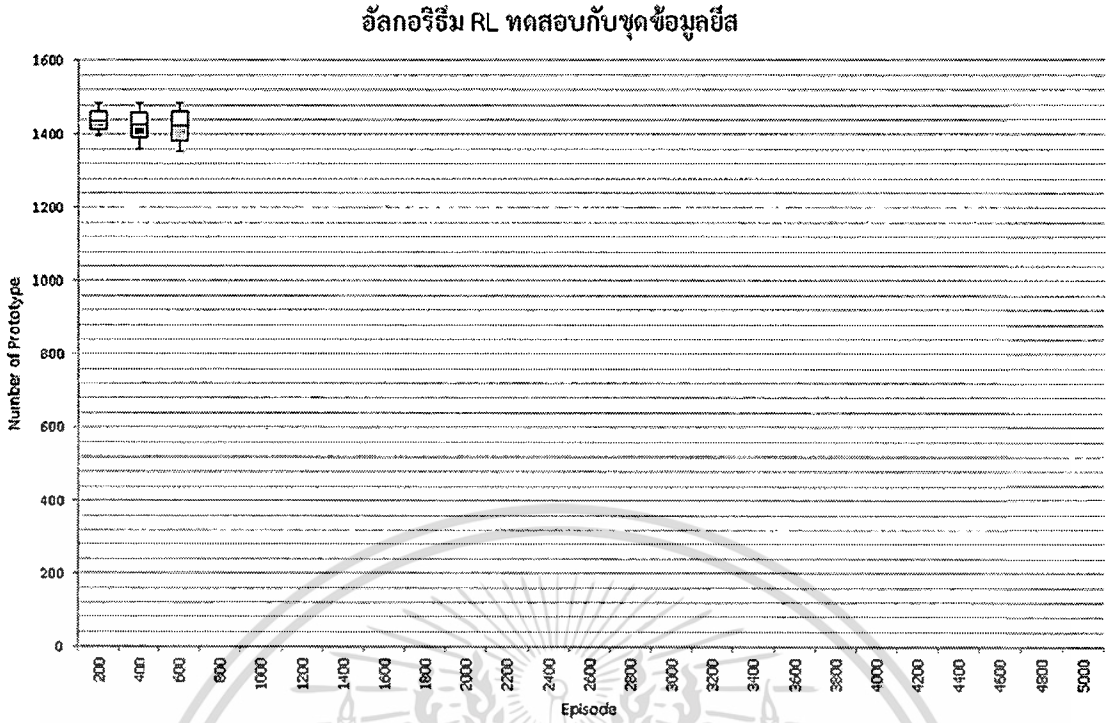
รูปที่ ค.16 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสินเชื่อด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ เอพิโสด



รูปที่ ค.17 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสินเชื่อด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ 10 เอพิโสด

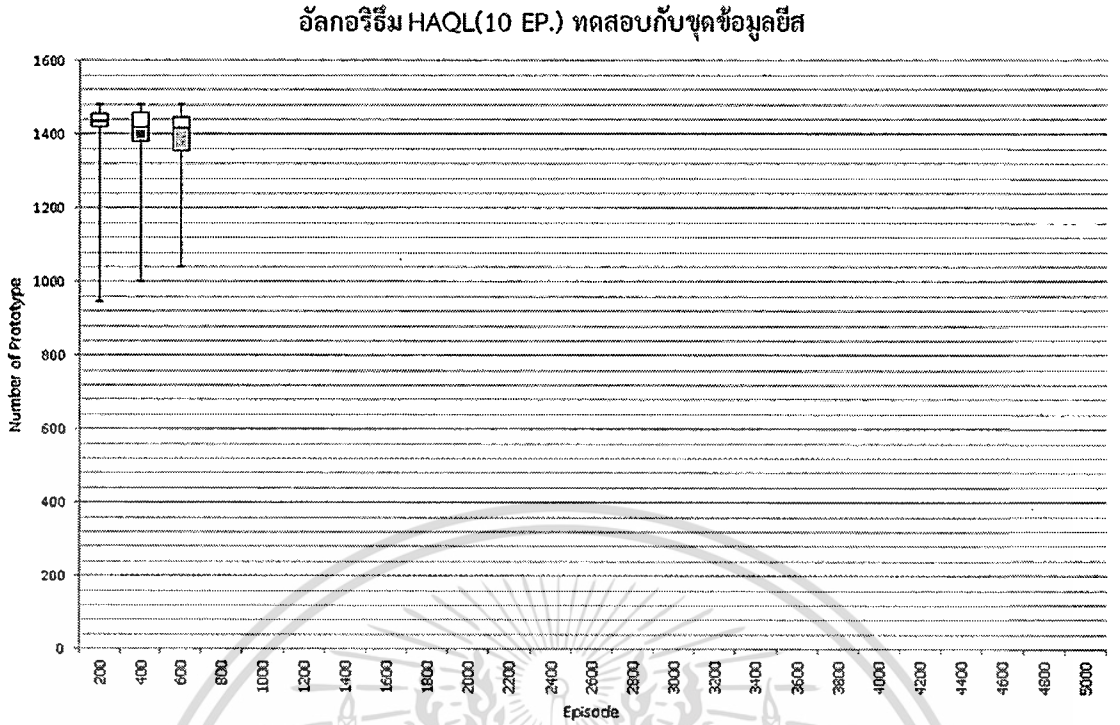


รูปที่ ค.18 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสินเชื่อด้วยอัลกอริธึม RL-RCS

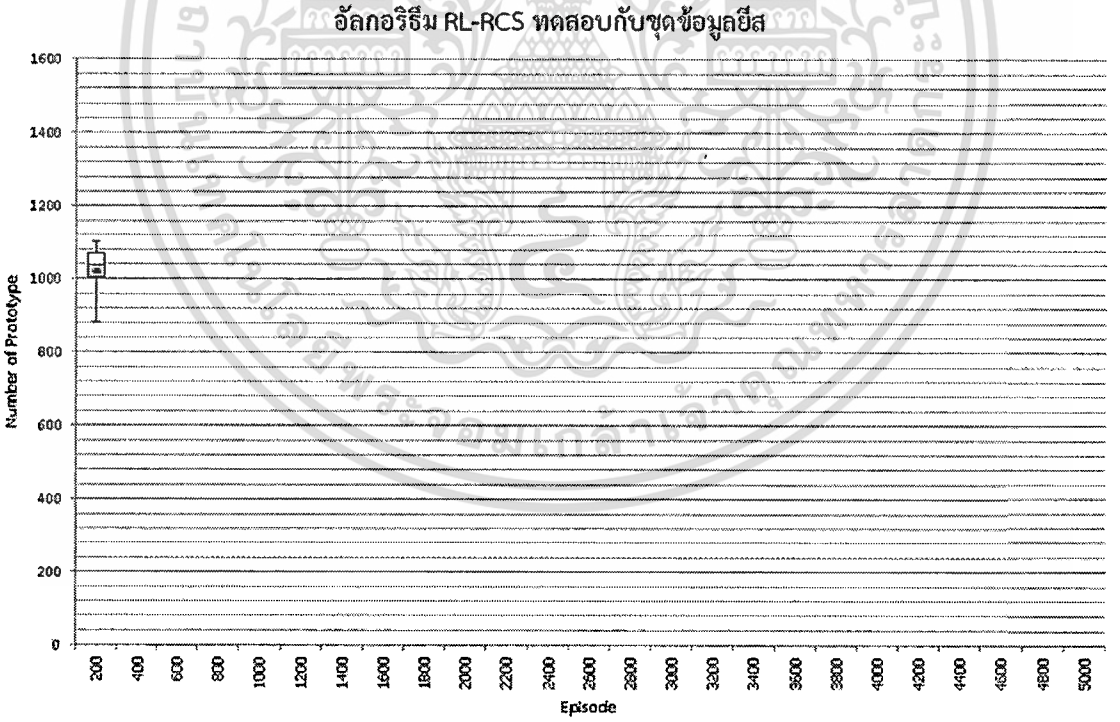


รูปที่ ค.19 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลยีสด้วยอัลกอริธึม RL

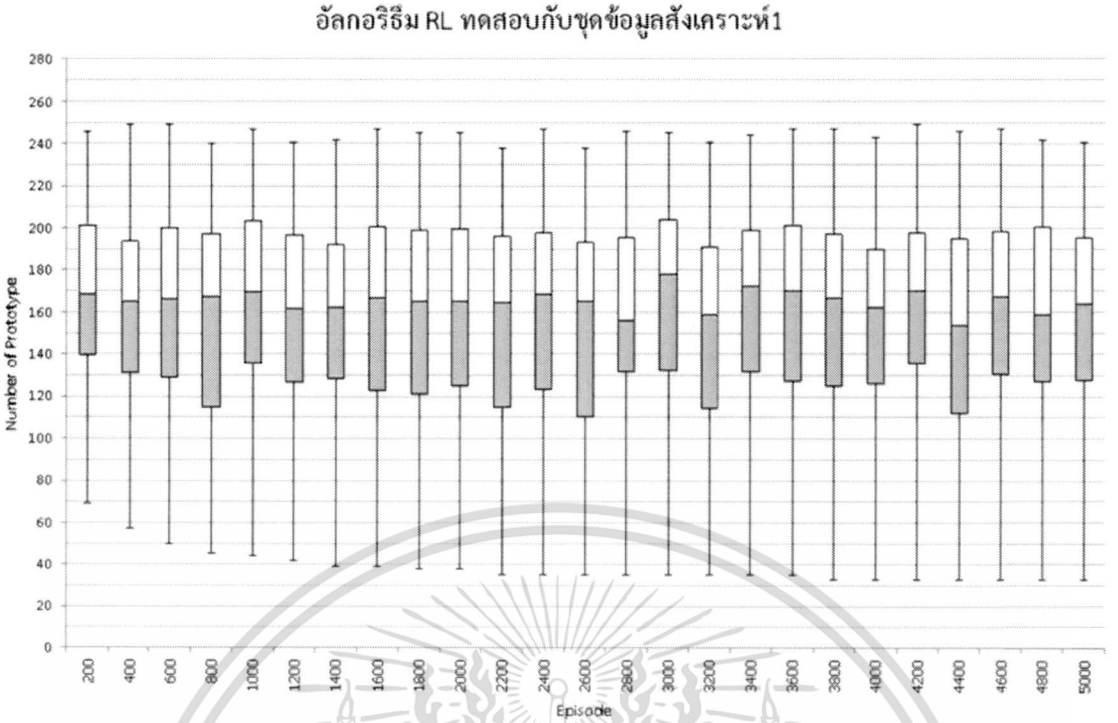
รูปที่ ค.20 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลยีสด้วยอัลกอริธึม HAQLที่ใช้คำแนะนำทุกๆ เอพิโสด



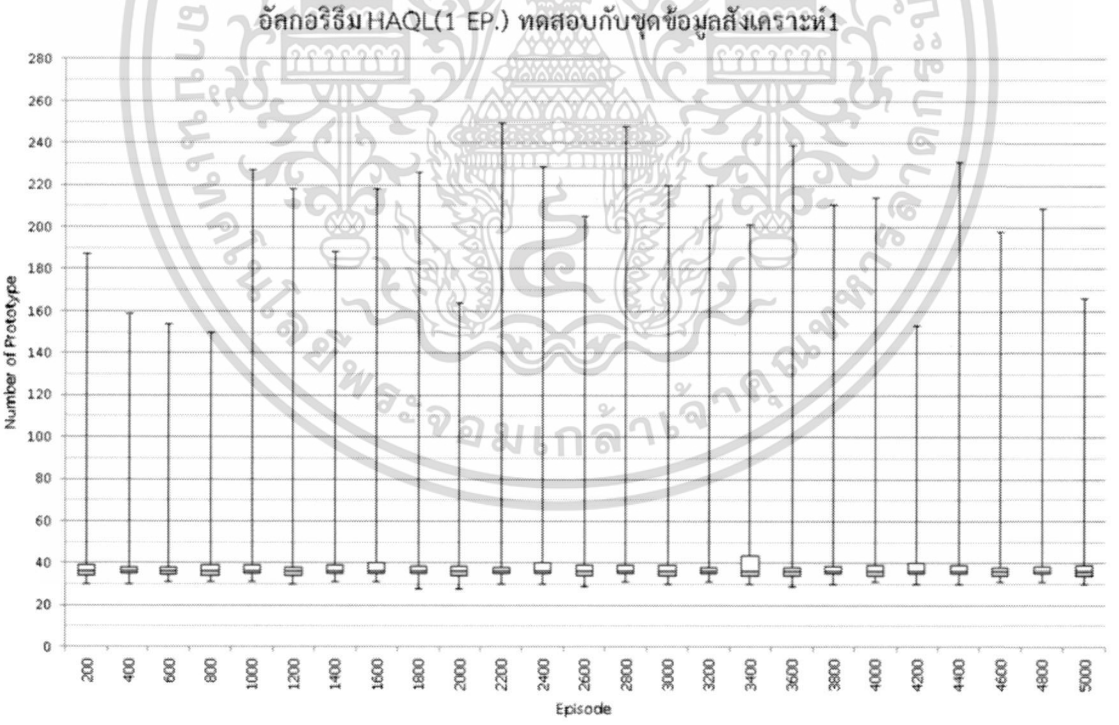
รูปที่ ค.21 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลยีสด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ 10 เอพิโสด



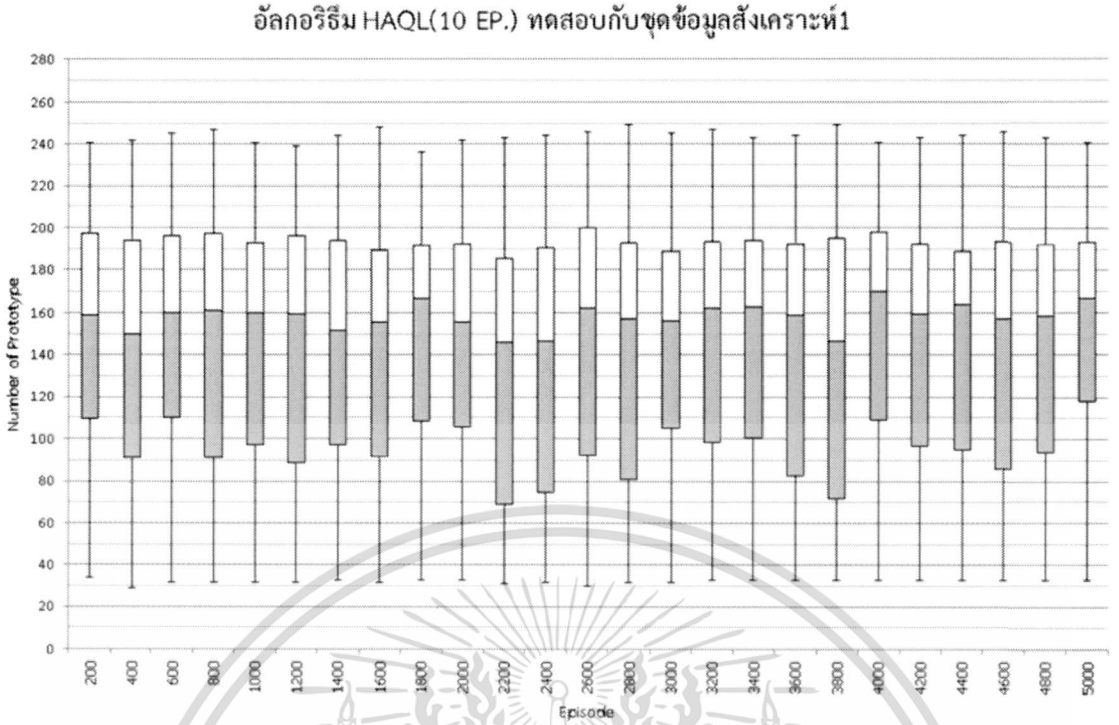
รูปที่ ค.22 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลยีสด้วยอัลกอริธึม RL-RCS



รูปที่ ค.23 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสังเคราะห์1 ด้วยอัลกอริธึม RL

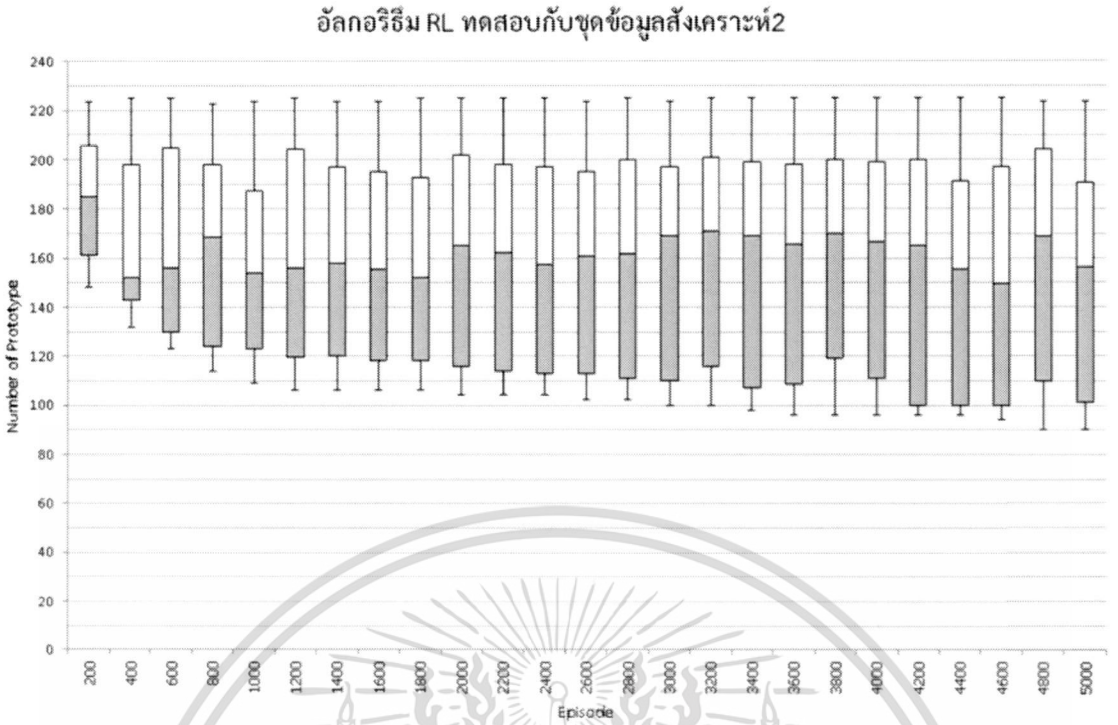


รูปที่ ค.24 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสังเคราะห์1 ด้วยอัลกอริธึม HAQLที่ใช้คำแนะนำทุกๆ เอพิโสด

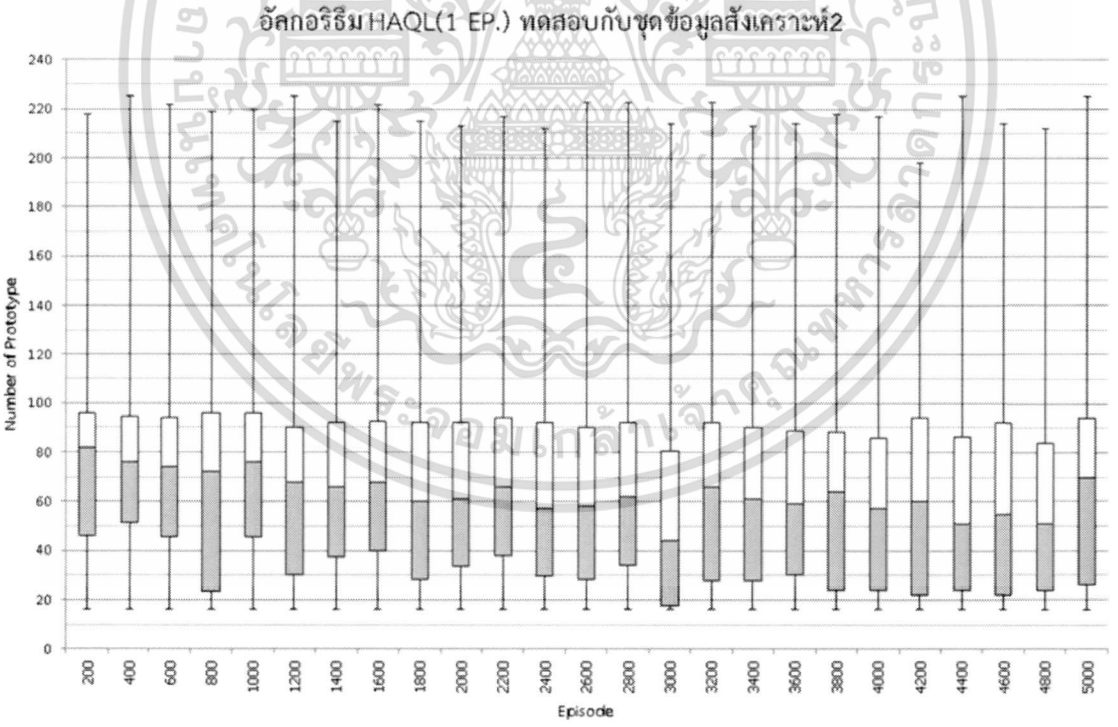


รูปที่ ค.25 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสังเคราะห์1 ด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ 10 เอพิโสด

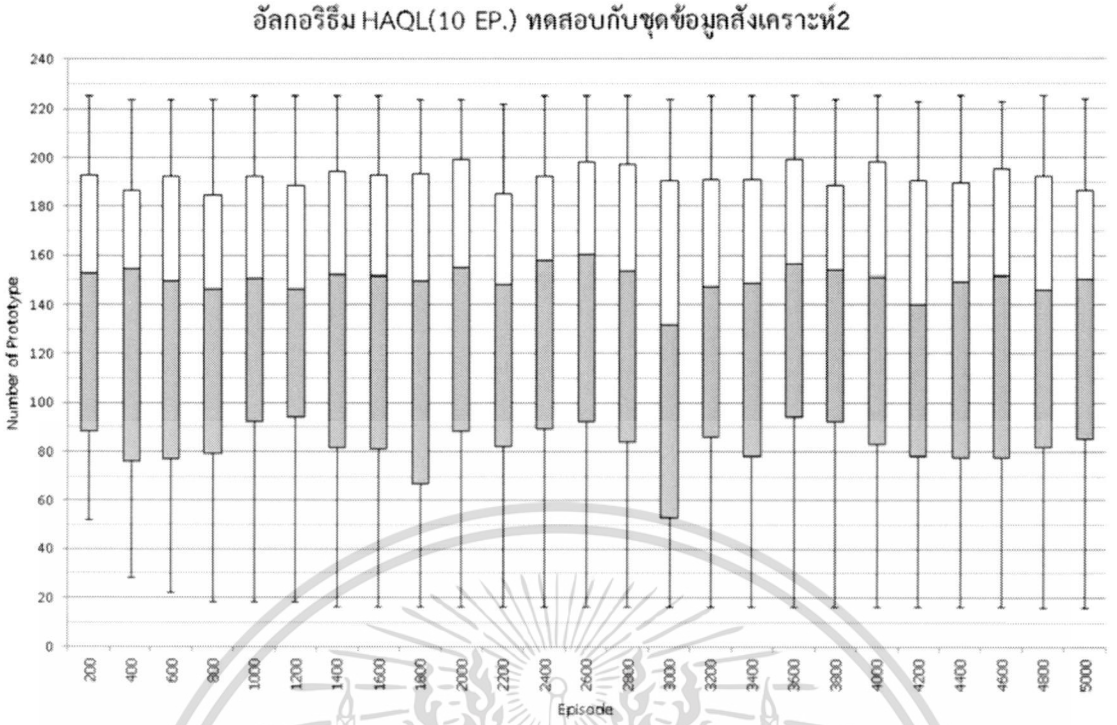
รูปที่ ค.26 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสังเคราะห์1 ด้วยอัลกอริธึม RL-RCS



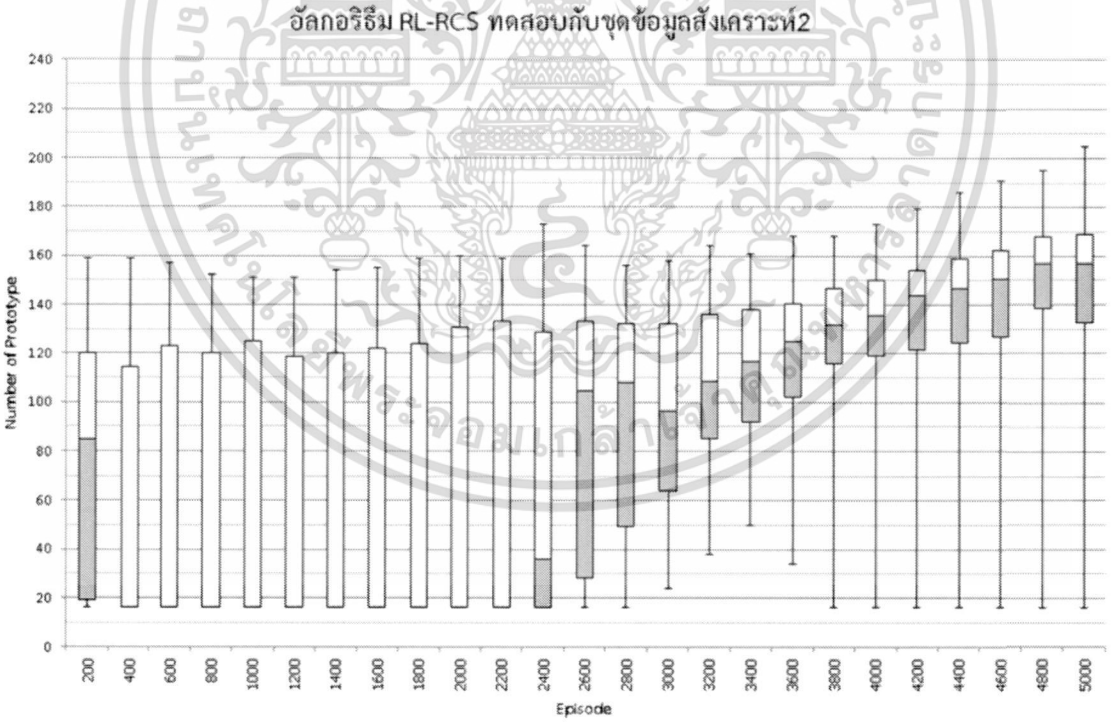
รูปที่ ค.27 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสังเคราะห์2 ด้วยอัลกอริธึม RL



รูปที่ ค.28 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสังเคราะห์2 ด้วยอัลกอริธึม HAQLที่ใช้คำแนะนำทุกๆ เอพิโสด

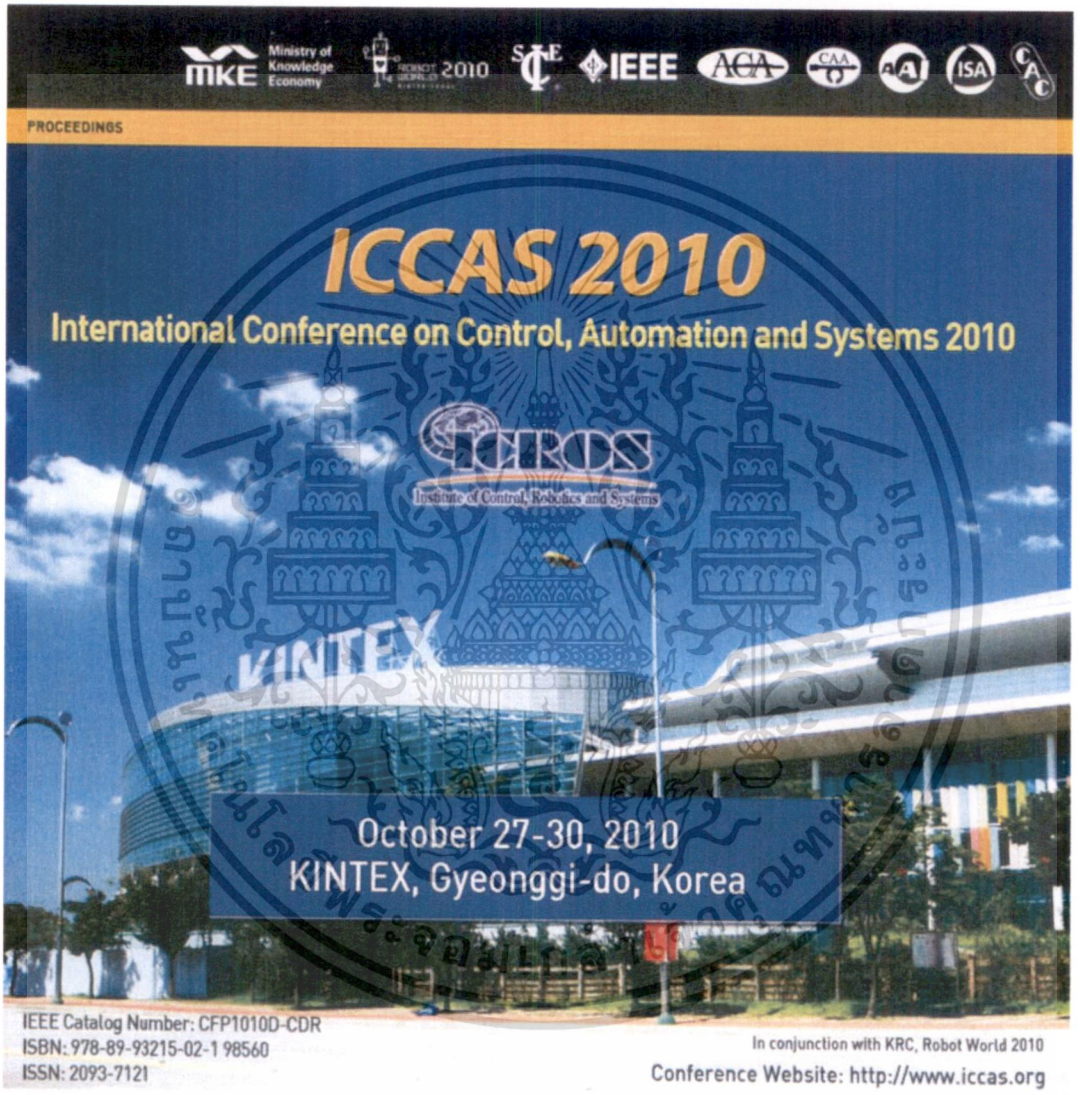


รูปที่ ค.29 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสังเคราะห์2 ด้วยอัลกอริธึม HAQL ที่ใช้คำแนะนำทุกๆ 10 เอพิโสด



รูปที่ ค.30 แผนภูมิกล่องแสดงเซตคำตอบที่พบในหนึ่งรอบการทดลองของชุดข้อมูลสังเคราะห์2 ด้วยอัลกอริธึม RL-RCS

ภาคผนวก ง.
งานวิจัยที่ได้รับการตีพิมพ์



International Conference on Control, Automation and Systems 2010
Oct. 27-30, 2010 in KINTEX, Gyeonggi-do, Korea

Prototype Selection using Reinforcement Learning and Minimal Consistent Subset Identification guide

Boontee Kruatrachue[1] and Teeratorn Choowong[2]

[1] Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand
(Tel : +66-86-533-0212; E-mail: booontee@yahoo.com)
[2] Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand
(Tel : +66-81-887-8683; E-mail: teeraongawa@hotmail.com)

Abstract: This paper try to apply Reinforcement Learning (RL) to a task with large number of states. This usually is a difficult task since RL has less chance to visit all state or has enough number of visit to learn average reward accurately. Moreover, RL may not be able to learn or obtain any optimal solution as RL learn by averaging rewards from each action performing in each state. In order to alleviate this RL learning problem, any solution to a task such as, non-optimal algorithm or heuristics can collaborate with RL by using their knowledge to prune the non-optimal action in each state. This reduces search space of RL and helps it learn faster. A Minimal consistent subset problem is used as an example to demonstrate how RL can learn faster with the help of other heuristics.

Keywords: reinforcement learning, prototype selection, minimal consistent subset, nearest neighbor classification.

1. INTRODUCTION

RL [1] is a learning framework for the Markov decision process (MDP). RL agent learns by interact with the environment. First, the environment provides the start state and a set of actions available in the state. The agent then selects an action from the set. Each action takes the agent to a different state and provides back different rewards. Once the agent select an action, the environment tell the next state and rewards. The agent keep select an action and go through each state obtaining more rewards. The goal of the RL agent is to find the path that has the most summation of rewards (return). By maintaining average return (Q value) for each action in each state, then the agent can learn to choose action with higher return. This learning process forces the agent to fully explore each state and go through the entire path in order to correctly averaging return. The agent always has to trade off and choose an action with less return to discover a new better path. In this paper, we focus on problems with a large number of states that RL cannot fully explore. RL usually sticks to the local optima with this kind of problem. In order to help RL learn faster or find a better path, any heuristic for the problems can be used to guide RL in choosing or not choosing action due to its probability of getting better return.

The Minimal Consistent Subset Identification (MCSI) [2] is use as an example problem with large number of state. MCSI is widely used in the pattern recognition to select minimum set of prototypes from training data. The prototype set can be used in nearest neighbor (NN) classification technique [4]. The smaller the set the lesser the reduce recognition time. The set is called consistent set if it can correctly classify all the training data. The consistent set is considered as having the same recognition boundary as their training data.

The rest of this paper is organized as follows. First, we describe heuristic to find Minimal Consistent Subset in Section 2. Reinforcement Learning for MCSI problem is introduced in Section 3. Section 4 then presents the result of our experiments. Finally the conclusion of this paper is in Section 5.

2. MINIMAL CONSISTENT SUBSET IDENTIFICATION

MCSI method is one of the selected Prototype techniques [5]. It was introduced by Dasarathy [2], based on the covering concept and Nearest Unlike Neighbor (NUN). A given Training data set is shown in Fig 1 (a). Distance table was calculated with a distance among all data using Euclidean distance shown in Fig 1 (b).

(a). Training Set.

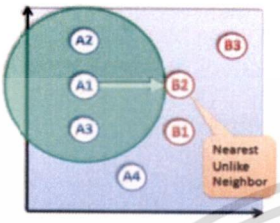
	A1	A2	A3	A4	B1	B2	B3
A1	0.0	1.0	1.0	2.2	2.2	2.0	3.1
A2	1.0	0.0	2.0	3.1	2.8	2.2	3.0
A3	1.0	2.0	0.0	1.4	2.0	2.2	3.6
A4	2.2	3.1	1.4	0.0	1.4	2.2	3.6
B1	2.2	2.8	2.0	1.4	0.0	1.0	2.2
B2	2.0	2.2	2.2	2.2	1.0	0.0	1.4
B3	3.1	3.0	3.6	3.6	2.2	1.4	0.0

(b). Distance table.

Fig 1. Training data set and its distance table.

978-89-93215-02-1 98560/10/$15 ©ICROS

As shown in Fig 2 (a), B2 is the nearest unlike neighbor of A1. Any data of type A closer to A1 than B2 (A2, A3) can be used as a prototype that can correctly classify A1. In other words, both A2 and A3 cover A1 since it closer to A1 than A1's NUN B2. Sorting distance table from minimum to maximum and NUN data points is shown in Fig 2 (b). All NUN are bolded and colored in Fig 2.b.



(a). Nearest Unlike Neighbor Boundary

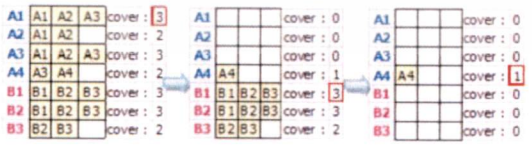
A1	A1	A2	A3	**B2**	B1	A4	B3
A2	A2	A1	A3	**B2**	B1	B3	A4
A3	A3	A1	A4	**B1**	A2	B2	B3
A4	A4	**B1**	A3	B2	A1	A2	B3
B1	B1	B2	**A4**	A3	A1	B3	A2
B2	B2	B1	B3	**A1**	A4	A3	A2
B3	B3	B2	B1	**A2**	A1	A4	A3

(b). Sorted distance table with NUN labeled

Fig 2. Nearest Unlike Neighbor (NUN).

The cover list for each data points (Fig 3 (a)) can be found using Fig 2 (b). Form the first row of Fig 2 (b), A1 is the closet to A1 then A2 and A3 followed by B2. Hence, A2 and A3 cover A1. From the second row, A1 and A3 cover A2. A1 and A4 also cover A3 in the third row. So the cover list of A1 is A1, A2 and A3 as shown in the first row of the cover table in Fig 3 (a).

The MCSI method starts by finding the cover list for each data point (Fig 3 (a)). Select the data point with maximum cover list as a prototype (A1). Remove data point in the selected prototype cover list (A1 list) from all the cover lists in the table (Fig 3 (b)). Repeat selecting prototype with max cover list and removing data in the list from the table until all the training data are covered (B1 and A4). At this point, the remaining cover lists are all empty and the selected prototype set (A1, B1 and A4) is consistent with the training data.



(a). cover table (b). remove A1 cover list (c). remove B1 cover list

Fig 3. Cover table use in MCSI

Also, now all the prototypes are defined and can be used instead of NUN to defined cover range. Hence, the previous process is repeated again but this time the construction of cover list use the selected prototype (A1, B1 and A4) instead of NUNs. Also, the prototype must be selected from the previous selected set.

This repetition continues until no further reduction of the set is possible. This method is very fast and produces much smaller prototype set from the training data. The main drawback is the use of all NUN to defined cover list in the first step which result in suboptimal prototype set.

3. REINFORCEMENT LEARNING FOR MCSI PROBLEM

It is natural to apply the RL framework to the MCSI problem as an episodic task as follow. RL Agent begins an episode in a start state with no selected prototype (Fig 4). The agent repeats taking action by adding a prototype and making a transition to a new state with one more prototype. The agent keep adding a prototype to a state until all the training data can be fully recognized using the prototype in the state. That state is call a consistent state and it is the end of that episode. In the worst case, the episode ends when all prototypes are chosen. At the end of each episode, RL learns by averaging the returns using Monte Carlo Method [1] and update the Q value in all visiting state in the episode. Agent start new episodes until the Q values converge.

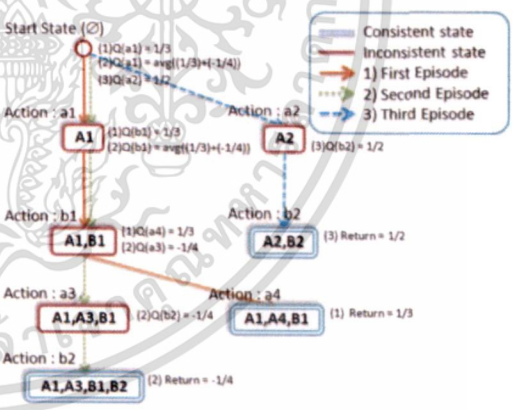


Fig 4. The state, actions sequence and their corresponding Q values.

The prototype covering concept [2] is used to prune the action of choosing prototype with similar covering with the previously selected prototypes. Each prototype is covering other training data if that prototype can be used to recognize them. Hence any adding prototype that does not cover more training data is useless and should not be selected. A cover table is maintained after each prototype selection to a state as in

978-89-93215-02-1 98560/10/$15 ©ICROS

Fig 3.

The RL-MCS algorithm is shown as follows:

Construct cover lists
For each episode:
Copy cover lists to cover table
Start state with no selected prototype.
While at least one of the cover list in the cover table is not null:
1)select a prototype with non-empty cover list randomly
2)remove data in the selected prototype cover list from all the cover lists in the table
update Q(s, a) with the return using –
Monte Carlo Method.

The return is calculated at the end of each episode. It is inverse proportional to the number of prototype at the final state of each episode. A positive return is given to the agent when the number of prototypes is less than the minimum number of prototypes founded in the former episodes. Conversely, the agent will get a negative return.

$$\text{Return}(\text{state, action, next state}) = \begin{cases} +(1/\text{No. of prototypes in the last state in the episode}) & \text{, No. of Prototypes} \le \text{Minimum No. of Prototypes found} \\ -(1/\text{No. of prototypes in the last state in the episode}) & \text{, No. of Prototypes} > \text{Minimum No. of Prototypes found} \end{cases}$$

4. EXPERIMENTAL RESULTS

This experiment compares the proposed RL-MCS and RL. RL is the same as RL-MCS describe in pseudo code in previous section. The only different is RL select prototype randomly without the use of cover lists knowledge, while RL-MCS will not select prototype with null cover list. The ε-greedy (parameter) is fixed at 0.1 (10% Explorations) for both RL and RL-MCS algorithm. The maximum number of episode is fixed to 20,000 episodes. The data used in this test is BezdekIris data with 150 instance, 4 features and 3 classes. It is available at UCI site [3].

Fig 5 and Fig 6 show the minimum number of prototype found from the start up to each episode. Since the number of step in taking action varies among each episode, episode number is shown in step in Fig 5. Fig 5 shows how the RL-MCS (red spot) converges in comparing with the RL (cyan spot). The RL-MCS algorithm can found an optimal prototype with 10 prototypes, but the RL algorithm minimal prototype is 23. In Fig 6, Show the minimum set of prototypes founded in each episode. Table 1 summarizes the experiment results. It shows that the RL-MCS can found the smaller prototype set in shorter number of episode and time.

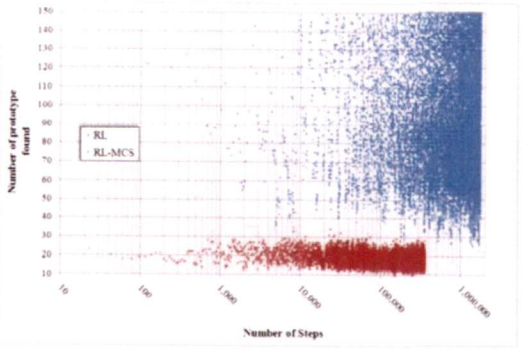


Fig 5. Number of best prototypes found in each episode in steps.

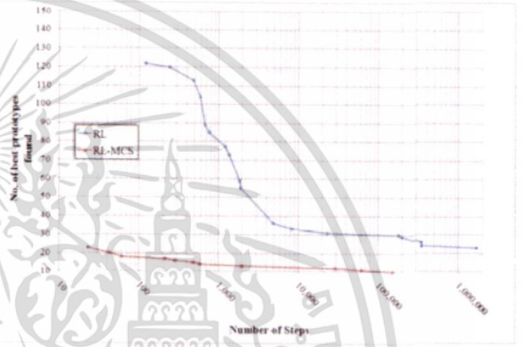


Fig 6. Number of best prototypes found in each episode in steps.

Table 1. Minimums set form experiment (at 20,000 episodes).

Algorithm	No. of Prototypes	Episode Found	Number of Steps
RL	23	19,079	1,693,399
RL-MCS	10	8,239	151,925

5. CONCLUSION

This paper proposed the use of other heuristic to guide RL agent in exploring by favor or non-favor some actions due to some probabilities suggested by the heuristic. In the MCS problem, RL with MCS heuristic can find the optimal solution while normal RL get stuck in local optimal. Moreover, RL-MCS also reduce prototype much faster than normal RL. The selection of heuristic is very important. It should not prune the optimal action; otherwise the optimal can't be reached.

REFERENCES

[1] R. S. Sutton and A. G. Barto, "*Reinforcement learning: An introduction.*" MIT Press, Cambridge, MA, 1998.

978-89-93215-02-1 98560/10/$15 ©ICROS

[2] B. V. Dasarathy, *"Minimal Consistent Set (MCS) Identification for Optimal Nearest Neighbor Decision Systems Design"*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 3, 1994, pp. 511-517.

[3] Department of Information and Computer Science, University of California, Irvine. *"UCI Machine Learning Repository"* [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html.1998.

[4] T. M. Cover and P. E. Hart, *"Nearest Neighbor Pattern Classification"*, IEEE Transactions on Information Theory, Vol. IT-13, No. 1, January 1967, p. 21-27.

[5] L.I. Kuncheva and J.C. Bezdek, *"Nearest Prototype Classification: Clustering, Genetic Algorithms, or Random Search?"*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 28, No. 1, 1998, pp.160-164

978-89-93215-02-1 98560/10/$15 ©ICROS

ภาคผนวก จ.
ผลการทดลองเพิ่มเติม

ภาคผนวก จ. เป็นการทดลองเพิ่มเติมของการแก้ปัญหาการเลือกโปรโตไทป์ โดยออกแบบให้เอเจนต์ทำการค้นหาเซตย่อยสอดคล้องเล็กที่สุดด้วยเทคนิคของอัลกอริธึม LCH-RCS แต่จะไม่ทำการบันทึกฟังก์ชันมูลค่า Q เพื่อหลีกเลี่ยงปัญหาทรัพยากรหน่วยความจำไม่เพียงพอ จึงถือเป็นการทดลองที่มีลักษณะของการค้นหาแบบสุ่ม (Random Search) บนปริภูมิที่แนะนำโดยฟังก์ชันฮิวริสติก เพื่อทดสอบว่าฟังก์ชันฮิวริสติกที่เลือกใช้ครอบคลุมปริภูมิคำตอบที่ลึกกว่าผลของคำตอบที่แสดงไว้ในบทที่ 5 หรือไม่ (มีคำแนะนำที่เข้าใกล้ปริภูมิคำตอบที่เหมาะสมยิ่งขึ้น) โดยตารางที่ จ.1 แสดงเซตย่อยสอดคล้องเล็กที่สุดที่พบในชุดข้อมูลต่างๆ

ตารางที่ จ.1 ผลการค้นหาเซตย่อยสอดคล้องเล็กที่สุดด้วยอัลกอริธึม LCH-RCS แบบไม่บันทึกฟังก์ชันมูลค่า Q

ชุดข้อมูล	ฟังก์ชันฮิวริสติกที่คำนวณจากอัตราการค้นคืน	ฟังก์ชันฮิวริสติกที่คำนวณจากการแก้ปัญหา MCSI
ไอริส (150)	10	10
แก้ว (214)	78	80
อีโคไล (336)	88	97
สินเชื่อ (1,000)	424	509
ยีส (1,484)	857	886
สังเคราะห์1 (250)	25	27
สังเคราะห์2 (225)	13	12

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายธีรธร ชูวงศ์
วัน เดือน ปีเกิด	16 มกราคม 2528
ประวัติการศึกษา	ปีการศึกษา 2540 -2545 ระดับมัธยมศึกษา โรงเรียน โพธิสัมพันธ์พิทยาคาร ปีการศึกษา 2546-2549 ระดับอุดมศึกษา คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมไฟฟ้า มหาวิทยาลัย บูรพา
ทุนการศึกษาที่ได้รับ	ทุนสนับสนุนการทำวิทยานิพนธ์จาก สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง