การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดโดยใช้การเรียนรู้แบบเสริมกำลัง

MINIMAL CONSISTENT SUBSET SELECTION USING REINFORCEMENT LEARNING

เอกพล อนันตพรกิจ

EKAPHOL ANANTAPORNKIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2552

KMITL-2009-EN-M-070-140

การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดโดยใช้การเรียนรู้แบบเสริมกำลัง

MINIMAL CONSISTENT SUBSET SELECTION USING REINFORCEMENT LEARNING

เอกพล อนันตพรกิจ

EKAPHOL ANANTAPORNKIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2552

KMITL 2009-EN-M-070-140

MINIMAL CONSISTENT SUBSET SELECTION USING REINFORCEMENT LEARNING

EKAPHOL ANANTAPORNKIT

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

2009

KMITL 2009-EN-M-070-140

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดโดยใช้การเรียนรู้แบบเสริมกำลัง

Thesis Title Minimal Consistent Subset Selection Using Reinforcement Learning

นักศึกษา นายเอกพล อนันตพรกิจ

รหัสประจำตัว 47060851

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร.บุญธีร์ เครือตราชู

หมายเลขวิทยานิพนธ์ KMITL-2009-EN-M-070-140

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ดร.อรัญญา	วลัยรัชต์	
ผศ.ดร.ชุติเมษฎ์	ศรีนิลทา	
รศ.ดร.เอื้อน	ปิ่นเงิน	
รศ.กฤตวัน	ศิริบูรณ์	
รศ.ดร.บุญธีร์	เครือตราชู	

วัน / เดือน/ ปี ที่สอบ วันพุธที่ 30 กันยายน พ.ศ. 2552 เวลา 09.00-11.00 น.

สถานที่สอบ ณ อาคาร A ชั้น 3 ห้องประชุม 2

คณะวิศวกรรมศาสตร์ รับรองแล้ว

(รองศาสตราจารย์ ดร.กอบชัย เดชหาญ)

คณบดี คณะวิศวกรรมศาสตร์

วันที่ 30 กันยายน พ.ศ. 2552

หัวข้อวิทยานิพนธ์	การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดโดยใช้การเรียนรู้แบบเสริมกำลัง
นักศึกษา	นายเอกพล อนันตพรกิจ
รหัสนักศึกษา	47060851
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2552
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.บุญธีร์ เครือตราชู

บทคัดย่อ

การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด (ปัญหา MCSS) เป็นโจทย์ปัญหาเกี่ยวกับการเลือกโปรโตไทป์จำนวนน้อยที่สุดจากชุดข้อมูลฝึกหัด โดยที่ยังคงคุณสมบัติความสอดคล้อง วิทยานิพนธ์ฉบับนี้นำเสนอแนวทางการแก้ปัญหา MCSS โดยใช้การเรียนรู้แบบเสริมกำลัง ทาสก์การเรียนรู้สำหรับปัญหา MCSS ในงานวิจัยนี้แบ่งเป็นสองกลุ่มใหญ่ คือ ทาสก์การเลือกโปรโตไทป์เพิ่มเติม กับทาสก์การคัดโปรโตไทป์ออก นอกจากการกำหนดทาสก์การเรียนรู้พื้นฐานแล้ว ยังมีการพัฒนาเทคนิคการดัดแปลงทาสก์การเรียนรู้เพื่อเพิ่มประสิทธิผลของอัลกอริธึมอีกด้วย จากผลการทดลองแสดงให้เห็นว่า เอเจนต์ RL สามารถค้นหาคำตอบของปัญหา MCSS ได้ดีในทาสก์การเรียนรู้บางชนิด ซึ่งให้ผลที่น่าพอใจสำหรับชุดข้อมูลที่เลือกใช้ในการทดลอง เมื่อเปรียบเทียบกับผลที่ได้จากอัลกอริธึม MCS ของดาซาราธี ซึ่งเป็นวิธีที่เป็นที่นิยมใช้ในการแก้ปัญหา MCSS

Thesis Title	Minimal Consistent Subset Selection Using Reinforcement Learning
Student	Mr. Ekaphol Anantapornkit
Student ID.	47060851
Degree	Master of Engineering
Program	Computer Engineering
Year	2009
Thesis Advisor	Assoc. Prof. Dr. Boontee Kruatrachue

ABSTRACT

The minimal consistent subset selection (MCSS) problem is the problem of selecting a minimum number of prototypes from a training dataset while maintaining the consistency property. This thesis presents a reinforcement learning approach to the MCSS problem. In this study, the learning task for the MCSS problem is divided into two major groups, the prototype insertion task and the prototype removal task. To improve the results obtained, modifications to the original learning task is developed. The experiments show that for some type of learning task, the RL agent to perform better in solving the MCSS problem. The results obtained from those tasks for the selected datasets are promising compared to the popular MCS algorithm by Dasarathy.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จสมบูรณ์ได้ ด้วยคำแนะนำจาก รศ.ดร.บุญธีร์ เครือตราชู ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ข้าพเจ้ารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณกรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้กรุณาให้คำแนะนำในเรื่องต่าง ๆ ซึ่งทำให้วิทยานิพนธ์ฉบับนี้มีความสมบูรณ์มากยิ่งขึ้น

ขอขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณเพื่อน ๆ พี่ ๆ น้อง ๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง สำหรับกำลังใจและคำแนะนำต่าง ๆ ที่มีให้กับข้าพเจ้า

ขอขอบคุณบัณฑิตศึกษาและบัณฑิตวิทยาลัย คณะวิศวกรรมศาสตร์ที่ให้ความช่วยเหลือในเรื่องต่าง ๆ

ขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้า สำหรับกำลังใจ และการสนับสนุนในทุก ๆ เรื่อง ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

สุดท้ายนี้ หากวิทยานิพนธ์ฉบับนี้มีข้อผิดพลาดแต่ประการใด ข้าพเจ้าขอน้อมรับไว้แต่เพียงผู้เดียว

เอกพล อนันตพรกิจ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด (minimal consistent subset selection: MCSS) [5] เป็นโจทย์ปัญหาเกี่ยวกับการเลือกโปรโตไทป์จำนวนน้อยที่สุดจากชุดข้อมูลฝึกหัดโดยที่ยังคงคุณสมบัติความสอดคล้อง (consistency property) [3] ซึ่งมีที่มาจากการพัฒนาเทคนิคการเลือกโปรโตไทป์ (prototype selection) ชุดข้อมูลย่อยที่ได้มาจากการเลือกโปรโตไทป์ในปัญหานี้สามารถนำไปใช้ในการจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุด (nearest neighbor classification) [2] แทนที่การใช้ข้อมูลทั้งหมดในชุดข้อมูลฝึกหัดเพื่อเป็นโปรโตไทป์ในการจำแนกข้อมูล ทั้งนี้เพื่อลดเวลาที่ใช้ในการประมวลผลโดยที่ยังคงรักษาประสิทธิผลของการจำแนกข้อมูล เซตของโปรโตไทป์ที่เลือกใช้จะยังคงคุณสมบัติความสอดคล้อง ถ้าเซตนั้นสามารถนำไปใช้จำแนกข้อมูลทุกตัวในชุดข้อมูลฝึกหัดโดยใช้การจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุดได้อย่างถูกต้อง

แม้ว่าเซตของโปรโตไทป์ที่ได้จากการแก้ปัญหา MCSS อาจไม่ใช่ตัวแทนที่เหมาะสมสำหรับใช้ในการจำแนกข้อมูลอื่น ๆ นอกเหนือจากข้อมูลในชุดข้อมูลฝึกหัด เนื่องจากผลจากการจำแนกข้อมูลอาจมีความผิดพลาดมากขึ้นเมื่อใช้โปรโตไทป์น้อยลง โดยเฉพาะในกรณีที่ลักษณะของข้อมูลนั้นมีความเหลื่อมล้ำระหว่างข้อมูลต่างชนิดมาก อัลกอริธึมที่ใช้แก้ปัญหา MCSS ก็ยังคงเป็นที่นิยมใช้ เนื่องจากสามารถลดจำนวนโปรโตไทป์ที่ใช้ในการจำแนกข้อมูลลงจากชุดข้อมูลเริ่มต้นได้เป็นอย่างดี นอกเหนือจากการแก้ปัญหา MCSS แล้ว ยังมีเทคนิคอื่น ๆ ที่ใช้ในการเลือกโปรโตไทป์อีกมากมายที่ไม่เกี่ยวข้องกับการแก้ปัญหา MCSS และไม่ได้ใช้คุณสมบัติความสอดคล้อง แต่มีประโยชน์ในการปรับแต่งโปรโตไทป์เพื่อลดความเหลื่อมล้ำระหว่างข้อมูลต่างชนิดกันและการปรับปรุงผลการจำแนกข้อมูลอื่น ๆ นอกเหนือจากข้อมูลในชุดข้อมูลฝึกหัด [1]

ปัญหา MCSS เป็นปัญหาการจัดกลุ่มที่มีความซับซ้อนสูง [4] ทำให้ไม่สามารถค้นหาคำตอบที่ดีที่สุดได้ด้วยการสำรวจปริภูมิการค้นหาอย่างทั่วถึงในเวลาที่จำกัด เบเลอร์ วี ดาซาราธี (Belur V. Dasarathy) ได้นำเสนออัลกอริธึม MCS [5] ซึ่งเป็นวิธีการแก้ปัญหา MCSS โดยการใช้ความครอบคลุมข้อมูลในการตัดสินใจ และเลือกเฉพาะโปรโตไทป์ที่จำเป็นในการจำแนกข้อมูล อัลกอริธึม MCS ของดาซาราธีเป็นอัลกอริธึมหนึ่งที่ได้รับความนิยมมาก เนื่องจากให้ผลลัพธ์ที่ค่อนข้างดีและมีประสิทธิภาพสูง อย่างไรก็ตาม อัลกอริธึมนี้จัดอยู่ในประเภทอัลกอริธึมแบบกรีดี (greedy algorithm) ดังนั้น คำตอบที่ได้จากวิธีนี้จึงอาจไม่ใช่คำตอบที่ดีที่สุด นอกเหนือจากอัลกอริธึมที่นำเสนอโดยดาซาราธีแล้ว ยังมีการพัฒนาอัลกอริธึมอื่น ๆ ในการแก้ปัญหา MCSS อีกด้วย เช่น อัลกอริธึมเจเนติก [7] การค้นหาแบบทาบู (tabu search) [4] เป็นต้น ทั้งนี้ ประสิทธิผล

ของอัลกอริธึมแต่ละแบบขึ้นอยู่กับลักษณะเฉพาะของชุดข้อมูล ด้วยเหตุนี้ จึงมีการพัฒนาอัลกอริธึมสำหรับใช้ในการแก้ปัญหา MCSS อย่างต่อเนื่องด้วยวิธีการที่หลากหลาย อย่างไรก็ตาม การแก้ปัญหา MCSS ก็ยังไม่มีอัลกอริธึมที่ให้ผลที่สมบูรณ์แบบ จึงเป็นที่มาของแนวคิดในการประยุกต์ใช้การเรียนรู้แบบเสริมกำลัง (reinforcement learning: RL) [8] เพื่อแก้ปัญหา MCSS

การเรียนรู้แบบเสริมกำลัง เป็นหนึ่งในระเบียบวิธีการแก้ปัญหาที่สำคัญในศาสตร์การเรียนรู้ของเครื่องคำนวณ (machine learning) ซึ่งใช้การเรียนรู้แบบลองผิดลองถูก และการเรียนรู้การกระทำที่ให้ผลระยะยาว ผ่านการโต้ตอบระหว่างระบบผู้เรียนหรือเอเจนต์กับสภาพแวดล้อมเป็นพื้นฐาน การเรียนรู้ของเอเจนต์ในการเรียนรู้แบบเสริมกำลังจะมาจากผลของการตัดสินใจเลือกการกระทำตอบสนองต่อสภาพแวดล้อม เอเจนต์จะได้รับรางวัลในรูปแบบของค่าตัวเลขซึ่งแสดงถึงผลความสำเร็จของการกระทำ เป้าหมายของเอเจนต์คือการเรียนรู้ที่จะเลือกการกระทำที่ทำให้ผลรวมของรางวัลที่ได้รับมีค่าสูงที่สุดในระยะยาว การเรียนรู้แบบเสริมกำลังเป็นเทคนิคพื้นฐานที่สำคัญในการออกแบบและพัฒนาอัลกอริธึมที่ใช้ในการแก้ปัญหา MCSS ซึ่งนำเสนอในวิทยานิพนธ์ฉบับนี้

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้มุ่งหวังเพื่อศึกษาการประยุกต์ใช้การเรียนรู้แบบเสริมกำลัง สำหรับการค้นหาคำตอบที่ดีที่สุดภายใต้เงื่อนไขจำกัด รวมถึงเทคนิคต่าง ๆ ที่เป็นประโยชน์ในการเพิ่มประสิทธิภาพของอัลกอริธึม RL โดยใช้ปัญหา MCSS เป็นกรณีศึกษา และสร้างรากฐานความรู้เกี่ยวกับการค้นหาคำตอบที่ดีที่สุดภายใต้เงื่อนไขโดยใช้อัลกอริธึม RL

1.3 สมมติฐานของการศึกษา

เนื่องจากปัญหา MCSS เป็นปัญหาการจัดกลุ่มที่มีความซับซ้อนสูง ทำให้ไม่สามารถค้นหาคำตอบที่ดีที่สุดได้ด้วยการสำรวจปริภูมิการค้นหาอย่างทั่วถึงในเวลาที่จำกัด ดังนั้นอัลกอริธึมต่าง ๆ ที่พัฒนาขึ้นมาสำหรับแก้ปัญหานี้ จึงมีทั้งข้อดีและข้อด้อยที่แตกต่างกันไป และไม่มีอัลกอริธึมที่ให้ผลดีที่สุดโดยสมบูรณ์ การประยุกต์ใช้การเรียนรู้แบบเสริมกำลังเพื่อแก้ปัญหา MCSS จึงถือเป็นอีกทางเลือกหนึ่งที่น่าสนใจ เนื่องจากใช้หลักการตัดสินใจที่แตกต่างจากอัลกอริธึมอื่น ๆ ที่มีการพัฒนามาก่อนหน้านี้

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

แนวคิดหลักที่ใช้ในวิทยานิพนธ์นี้ ได้แก่ เทคนิคการคัดเลือกโปรโตไทป์ ซึ่งเป็นที่มาของโจทย์ปัญหา MCSS คุณสมบัติความสอดคล้องของเซตของโปรโตไทป์ที่เลือกมาใช้ในการจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุด ซึ่งเป็นเงื่อนไขสำหรับเซตคำตอบที่เหมาะสมสำหรับปัญหา MCSS การเรียนรู้แบบเสริมกำลัง ซึ่งเป็นพื้นฐานสำหรับนำไปประยุกต์ใช้ในการแก้ปัญหา MCSS ข้อดีของการเรียนรู้แบบเสริมกำลัง คือ การเรียนรู้การค้นหาคำตอบจากการลองผิดลองถูก ทำให้เกิดการปรับเปลี่ยนวิธีการตัดสินใจจากการสุ่มไปสู่การตัดสินใจที่มีระบบ

1.5 ขอบเขตการวิจัย

วิทยานิพนธ์ฉบับนี้นำเสนอผลการศึกษาและพัฒนาวิธีการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด โดยมีขอบเขตการวิจัยดังนี้

1. อัลกอริธึมที่พัฒนาขึ้นในงานวิจัยนี้มีพื้นฐานมาจากการเรียนรู้แบบเสริมกำลัง และอัลกอริธึมสำหรับปัญหา MCSS ที่มีการพัฒนามาก่อนหน้านี้
2. ในการเปรียบเทียบผลการทดลอง จะใช้ผลที่ได้จากอัลกอริธึม MCS ของดาซาราธีเป็นตัวอ้างอิงในทุก ๆ ชุดข้อมูล ยกเว้นในกรณีที่พบคำตอบที่ดีที่สุดจากวิธีอื่น ๆ ซึ่งมาจากแหล่งข้อมูลที่เชื่อถือได้
3. ในชุดข้อมูลทุก ๆ ชุดที่ใช้ในการทดลอง ข้อมูลแต่ละตัวในชุดข้อมูลจะมีค่าคุณลักษณะเฉพาะเป็นจำนวนเท่า ๆ กัน และแต่ละค่าเป็นจำนวนจริงเท่านั้น
4. ชุดข้อมูลทั้งหมดที่เลือกมาใช้ในการทดลองมาจากเว็บไซต์ของมหาวิทยาลัยแคลิฟอร์เนีย เออร์ไวน์ (University of California, Irvine) [9] ซึ่งเป็นแหล่งรวมชุดข้อมูลที่เหมาะสำหรับใช้ในการทดลองเกี่ยวกับการเรียนรู้ของเครื่องคำนวณ

1.6 ขั้นตอนการศึกษา

1. ศึกษาทฤษฎีและความรู้พื้นฐานที่เกี่ยวข้องกับการจำแนกข้อมูลโดยใช้กฎการจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุด
2. ศึกษาทฤษฎีและความรู้พื้นฐานที่เกี่ยวข้องกับการเลือกโปรโตไทป์
3. ศึกษาทฤษฎีและความรู้พื้นฐานที่เกี่ยวข้องกับการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด
4. ศึกษาวิธีการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดที่มีผู้นำเสนอมาก่อนหน้านี้
5. ศึกษาทฤษฎีและความรู้พื้นฐานที่เกี่ยวข้องกับการเรียนรู้แบบเสริมกำลัง

6. ออกแบบทาสก์การเรียนรู้ที่เหมาะสมสำหรับการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด ในการประยุกต์ใช้การเรียนรู้แบบเสริมกำลัง

7. พัฒนาอัลกอริธึม RL และเทคนิคที่ช่วยในการเพิ่มประสิทธิผลของอัลกอริธึม

8. ทดสอบอัลกอริธึมที่พัฒนาขึ้นและวิเคราะห์ผลที่ได้

9. สรุปผลการทดลองและจัดทำเอกสารประกอบวิทยานิพนธ์

1.7 โครงสร้างของวิทยานิพนธ์

วิทยานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 6 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความเป็นมาของงานวิจัย ความมุ่งหมายและวัตถุประสงค์ สมมติฐาน ทฤษฎีที่ใช้ ขอบเขตของการวิจัย และขั้นตอนการศึกษา

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่เกี่ยวข้องกับการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด และการเรียนรู้แบบเสริมกำลัง

บทที่ 3 กล่าวถึงหลักการประยุกต์ใช้การเรียนรู้แบบเสริมกำลังในการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด และแนวคิดพื้นฐานในการออกแบบอัลกอริธึมที่นำเสนอในงานวิจัยนี้

บทที่ 4 กล่าวถึงเทคนิคที่พัฒนาขึ้นเพื่อเพิ่มประสิทธิผลของอัลกอริธึม RL ในการแก้ปัญหา MCSS

บทที่ 5 กล่าวถึงการเปรียบเทียบผลที่ได้จากอัลกอริธึม RL โดยใช้ทาสก์การเรียนรู้แบบต่าง ๆ และการเปรียบเทียบผลที่ได้กับอัลกอริธึม MCS ของดาซาราธี

บทที่ 6 กล่าวถึงบทสรุปผลการวิจัยและข้อเสนอแนะ

บทที่ 2
ทฤษฎีพื้นฐานที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึง ทฤษฎีพื้นฐานที่เกี่ยวข้องกับการจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุด เทคนิคการเลือกโปรโตไทป์ ปัญหาการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด และทฤษฎีการเรียนรู้แบบเสริมกำลัง

2.1 การจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุด

การจำแนกข้อมูล เป็นกระบวนการที่ใช้ในการแบ่งกลุ่มข้อมูลตามปริมาณที่วัดได้จากตัวข้อมูล ซึ่งเรียกว่า ค่าคุณลักษณะเฉพาะ (feature) ซึ่งกำหนดขึ้นจากลักษณะการประยุกต์ใช้ข้อมูลโดยทั่วไป ข้อมูลแต่ละตัวสามารถกำหนดได้ในรูปของเวคเตอร์ของค่าคุณลักษณะเฉพาะ x ซึ่งอยู่ในปริภูมิคุณลักษณะเฉพาะ (feature space) เช่น ในกรณีที่ x อยู่ในปริภูมิเวคเตอร์จำนวนจริงขนาด d มิติ ($\Re^d$) ตัวแทนของข้อมูล x เมื่อ $x \in \Re^d$ ก็คือจุดบนปริภูมิขนาด d มิตินั่นเอง ในกระบวนการจำแนกข้อมูล ข้อมูลทั้งหมดจะแบ่งออกเป็นกลุ่มหรือชนิดต่าง ๆ ตามค่าคุณลักษณะเฉพาะและจัดอยู่ในเซตชนิดของข้อมูล $\Theta = \{c_1, c_2, \ldots, c_{nc}\}$ ซึ่งกำหนดชนิดของข้อมูล nc ชนิดที่เป็นไปได้ในข้อมูลทั้งหมด โดยทั่วไป ข้อมูลชนิดเดียวกันจะมีค่าคุณลักษณะเฉพาะในรูปแบบที่ใกล้เคียงกัน

วิธีการจำแนกข้อมูลวิธีหนึ่งที่เป็นที่นิยมมากคือ การจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุด (nearest neighbor classification) เนื่องจากให้ผลลัพธ์ที่ดี และสามารถพัฒนาขึ้นได้ง่าย สิ่งที่จำเป็นในการจำแนกข้อมูลด้วยวิธีนี้ ได้แก่

1) เซตของตัวแทนข้อมูล (reference set) หรือเซตตัวแทน ซึ่งมีการกำกับด้วยชนิดของข้อมูลแต่ละตัวหรือโปรโตไทป์ในเซต โดยทั่วไป เซตตัวแทน V สามารถกำหนดได้ดังนี้

$$V = \{v_1, v_2, \ldots, v_m\} = \{(x_1, \theta_1), (x_2, \theta_2), \ldots, (x_m, \theta_m)\}$$

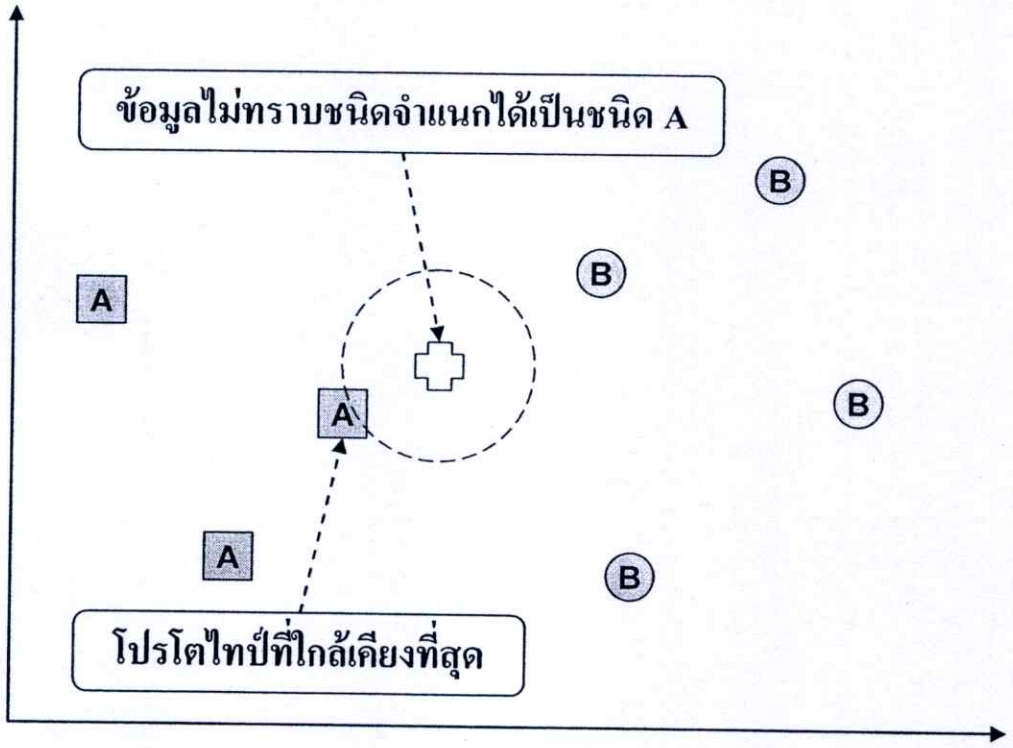
เมื่อ $x_i \in \Re^d$ เป็นจุดบนปริภูมิคุณลักษณะเฉพาะซึ่งแทนข้อมูลในเซตตัวแทน $v_i \in V$ และ $\theta_i \in \Theta$ เป็นชนิดของข้อมูล v_i จำนวนสมาชิกของเซต V หรือ $|V|$ มีค่าเท่ากับ m

2) การคำนวณหาโปรโตไทป์ k ตัวที่มีความใกล้เคียงกับข้อมูลที่ไม่ทราบชนิดมากที่สุด โดยใช้ฟังก์ชันคำนวณระยะทางในการวัดความใกล้เคียงกันระหว่างข้อมูลแต่ละคู่ ฟังก์ชันคำนวณระยะทาง $\delta : \Re^d \times \Re^d \rightarrow \Re$ สามารถกำหนดได้ตามความต้องการ เช่น ฟังก์ชันคำนวณระยะทางแบบยูคลิด ซึ่งเป็นฟังก์ชันคำนวณระยะทางแบบพื้นฐานและเหมาะสำหรับใช้กับข้อมูลที่แทนได้ด้วยเวคเตอร์จำนวนจริง

กฎที่สำคัญสำหรับการจำแนกข้อมูลด้วยวิธีนี้เรียกว่า กฎ k-NN (k-nearest neighbor rule) กล่าวคือ จำแนกชนิดของข้อมูลที่ไม่ทราบชนิด ตามชนิดของโปรโตไทป์ที่ปรากฏมากที่สุดในกลุ่มของโปรโตไทป์ k ตัวที่มีความใกล้เคียงกับข้อมูลนั้นมากที่สุด ในกรณีที่ k=1 (หรือเรียกว่ากฎ 1-NN) การกำหนดชนิดของข้อมูลที่ต้องการจำแนก จะเป็นไปตามชนิดข้อมูลของโปรโตไทป์ตัวที่มีความใกล้เคียงกับข้อมูลที่ต้องการจำแนกมากที่สุด กฎ 1-NN หรือ $NN(x, V, \delta)$ ซึ่งจำแนกข้อมูลโดยใช้เซตตัวแทน V และใช้ฟังก์ชันคำนวณระยะทาง δ สามารถเขียนในรูปของสมการคณิตศาสตร์ได้ดังนี้

$$NN(x, V, \delta) = \theta \mid (x', \theta) \in V \mid \delta(x', x) \le \delta(x_i, x), \quad \forall x_i \mid (x_i, \theta_i) \in V; 1 \le i \le m \qquad (2.1)$$

เมื่อ x เป็นข้อมูลที่ไม่ทราบชนิดซึ่งจำแนกเป็นได้ชนิด θ ตามกฎ 1-NN



รูปที่ 2.1 การจำแนกข้อมูลตามกฎ 1-NN

รูปที่ 2.1 แสดงตัวอย่างการจำแนกข้อมูลตามกฎ 1-NN โดยใช้เซตตัวแทนซึ่งประกอบด้วยโปรโตไทป์จำนวน 7 ตัว แบ่งเป็นโปรโตไทป์ชนิด A จำนวน 3 ตัว (แทนด้วยรูปสี่เหลี่ยม) และโปรโตไทป์ชนิด B จำนวน 4 ตัว (แทนด้วยรูปวงกลม) โปรโตไทป์ทั้งหมดอยู่ในรูปของจุดที่มีตำแหน่งแตกต่างกันบนปริภูมิคุณลักษณะเฉพาะ 2 มิติ และใช้ฟังก์ชันคำนวณระยะทางแบบยูคลิด ในรูปแสดงการจำแนกข้อมูลไม่ทราบชนิด (แทนด้วยรูปเครื่องหมายบวก) ซึ่งเป็นจุดบนปริภูมิคุณลักษณะเฉพาะเดียวกัน ข้อมูลนี้เมื่อวัดระยะทางโดยใช้การคำนวณระยะทางแบบยูคลิดจะพบว่าอยู่ใกล้กับโปรโตไทป์ตัวหนึ่งซึ่งเป็นชนิด A มากที่สุด (ดังแสดงด้วย

ขอบเขตวงกลมเส้นประ) ดังนั้น ในการจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุดโดยใช้กฎ 1-NN ข้อมูลนี้จะจำแนกได้เป็นชนิด A เช่นเดียวกับชนิดของโปรโตไทป์ที่อยู่ใกล้กับข้อมูลนั้นมากที่สุด จากรูปเห็นได้ชัดว่า ในการจำแนกข้อมูลที่ไม่ทราบชนิด 1 ตัว จะต้องวัดระยะทางระหว่างข้อมูลตัวนั้นกับโปรโตไทป์ทุกตัวในเซตตัวแทน ดังนั้น ถ้าเซตตัวแทนมีจำนวนโปรโตไทป์มาก การจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุดจะใช้เวลาในการคำนวณมากขึ้นตามลำดับ

2.2 เทคนิคการเลือกโปรโตไทป์

แม้ว่าการเลือกโปรโตไทป์ที่ใช้เป็นตัวแทนในเซตตัวแทนสำหรับการจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุดจะสามารถทำได้ง่าย แต่ในบางครั้งผลที่ได้จากการจำแนกข้อมูลก็อาจไม่ดีเท่าที่ควร ข้อเสียของการใช้การจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุดก็คือ โปรโตไทป์บางตัวในเซตของตัวแทนนั้นอาจให้ผลลัพธ์ที่ผิดพลาด ทำให้ผลการจำแนกข้อมูลไม่ดี นอกจากนี้ เซตของโปรโตไทป์ที่เป็นตัวแทนอาจมีขนาดใหญ่ ทำให้ต้องใช้เวลาในการจำแนกข้อมูลมาก

การแก้ปัญหาที่เกิดขึ้นในการจำแนกข้อมูลตามกฎ NN นั้นมีหลากหลายวิธีด้วยกัน หนึ่งในนั้นคือเทคนิคการเลือกโปรโตไทป์ [1] โดยมีวัตถุประสงค์เพื่อให้ได้เซตของโปรโตไทป์ซึ่งผ่านการคัดเลือก หรือลดจำนวนลงจากเซตของข้อมูลเริ่มต้น (original set) แต่ยังคงให้ผลที่ดีจากการจำแนกข้อมูลโดยใช้กฎ 1-NN โดยทั่วไป เทคนิคการเลือกโปรโตไทป์แบ่งเป็นเทคนิคการปรับแก้ (editing) และเทคนิคการลดรูป (condensing) เทคนิคการปรับแก้มีเป้าหมายหลักคือการปรับปรุงผลการจำแนกข้อมูลให้ดีขึ้น โดยการกำจัดข้อมูลที่ไม่เข้าพวก และการลดความเหลื่อมล้ำระหว่างข้อมูลที่ต่างชนิดกัน เทคนิคการลดรูปมีเป้าหมายหลักคือการลดจำนวนโปรโตไทป์ที่ใช้ในการจำแนกข้อมูลลงจากชุดข้อมูลเริ่มต้น

ในการเลือกโปรโตไทป์โดยใช้เทคนิคการลดรูป โปรโตไทป์ที่เลือกมาใช้ในการจำแนกข้อมูลอาจได้มาจากการสร้างขึ้นมาหรือดัดแปลงมาจากชุดข้อมูลเริ่มต้นก็ได้ ทำให้โปรโตไทป์ที่ได้ไม่จำเป็นต้องเป็นเซตย่อยของชุดข้อมูลเริ่มต้น เทคนิคการเลือกโปรโตไทป์แบบนี้เรียกว่าการแทนที่โปรโตไทป์ (prototype replacement) [6], [7] เพื่อเน้นความแตกต่างจากการลดรูปโดยใช้เทคนิคการคัดเลือกโปรโตไทป์ (prototype selection) [3] ซึ่งเป็นการคัดเลือกโปรโตไทป์มาจากข้อมูลในชุดข้อมูลเริ่มต้นเท่านั้น เซตของโปรโตไทป์ที่ได้มาจากเทคนิคการแทนที่โปรโตไทป์เรียกว่า เซตของโปรโตไทป์แบบอาร์ (R-prototypes) และเซตของโปรโตไทป์ที่ได้มาจากเทคนิคการคัดเลือกโปรโตไทป์ เรียกว่า เซตของโปรโตไทป์แบบเอส (S-prototypes)

หลักสำคัญของเทคนิคการคัดเลือกโปรโตไทป์ คือ การคงคุณสมบัติความสอดคล้อง (consistency property) [3] เซตของโปรโตไทป์ที่ได้จะยังคงคุณสมบัติความสอดคล้องกันกับชุดข้อมูลเริ่มต้น ซึ่งใช้เป็นชุดข้อมูลฝึกหัด เมื่อสามารถนำไปใช้จำแนกข้อมูลทุกตัวในชุดข้อมูลฝึกหัดได้อย่างถูกต้อง โดยใช้การจำแนกข้อมูลตามกฎ 1-NN การคัดเลือกโปรโตไทป์จึงนับว่าเป็น

กระบวนการฝึกหัดเพื่อเรียนรู้ชุดข้อมูลย่อยที่ยังคงคุณสมบัติความความสอดคล้องกันกับชุดข้อมูลที่ใช้ฝึกหัด

2.3 การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด

การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุด หรือปัญหา MCSS เป็นโจทย์ปัญหาซึ่งมีที่มาจากการพัฒนาเทคนิคการเลือกโปรโตไทป์สำหรับใช้ในการจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุด ปัญหา MCSS เป็นปัญหาเกี่ยวกับการหาคำตอบที่ดีที่สุดภายใต้เงื่อนไข (constraint optimization) โดยเงื่อนไขในปัญหานี้คือ การคงคุณสมบัติความสอดคล้องของเซตตัวแทนซึ่งเป็นคำตอบที่ได้จากการแก้ปัญหานี้กับชุดข้อมูลเริ่มต้น

สำหรับชุดข้อมูลเริ่มต้น O ซึ่งกำหนดได้ในรูป

$$O = \{o_1, o_2, \ldots, o_n\} = \{(x_1, \theta_1), (x_2, \theta_2), \ldots, (x_n, \theta_n)\}$$

เมื่อ $x_i \in \Re^d$ เป็นจุดบนปริภูมิคุณลักษณะเฉพาะซึ่งแทนข้อมูลในชุดข้อมูลเริ่มต้น $o_i \in O$ และ $\theta_i \in \Theta$ เป็นชนิดของข้อมูล o_i จำนวนสมาชิกของเซต O หรือ $|O|$ มีค่าเท่ากับ n

และเซตตัวแทน V ซึ่งกำหนดได้ในรูป

$$V = \{v_1, v_2, \ldots, v_m\} = \{(x_1, \theta_1), (x_2, \theta_2), \ldots, (x_m, \theta_m)\}$$

เมื่อ $x_i \in \Re^d$ เป็นจุดบนปริภูมิคุณลักษณะเฉพาะซึ่งแทนข้อมูลในเซตตัวแทน $v_i \in V$ และ $\theta_i \in \Theta$ เป็นชนิดของข้อมูล v_i จำนวนสมาชิกของเซต V หรือ $|V|$ มีค่าเท่ากับ m โดยกำหนดให้ $m \le n$

เซตตัวแทน V จะคงคุณสมบัติความสอดคล้องกับชุดข้อมูลเริ่มต้น O ก็ต่อเมื่อการจำแนกข้อมูลตามกฎ 1-NN โดยใช้เซตตัวแทน V และใช้ฟังก์ชันคำนวณระยะทาง δ ($NN(x, V, \delta)$) สามารถนำไปใช้จำแนกข้อมูลทุกตัวในชุดข้อมูลเริ่มต้น O ได้อย่างถูกต้อง คุณสมบัติความสอดคล้อง $CP(V, O)$ สามารถเขียนได้ในรูปของความสัมพันธ์ทางคณิตศาสตร์ดังนี้

$$CP(V, O) \leftrightarrow \theta_i = NN(x_i, V, \delta), \quad \forall o_i \mid o_i = (x_i, \theta_i) \in O; 1 \le i \le n \qquad (2.2)$$

เซตตัวแทน V เรียกว่าเป็นเซตย่อยสอดคล้อง (consistent subset) ของชุดข้อมูลเริ่มต้น O เมื่อ $V \subset O$ และ $CP(V, O)$

จากคุณสมบัติความสอดคล้องข้างต้น ปัญหา MCSS สามารถกำหนดได้ดังนี้

$$MCS(O) = \min_{V \in P(O)} |V| \quad (2.3)$$

ภายใต้เงื่อนไข $CP(V, O)$

เมื่อ $P(O)$ หมายถึงพาวเวอร์เซตของเซต O ซึ่งเป็นชุดข้อมูลเริ่มต้น

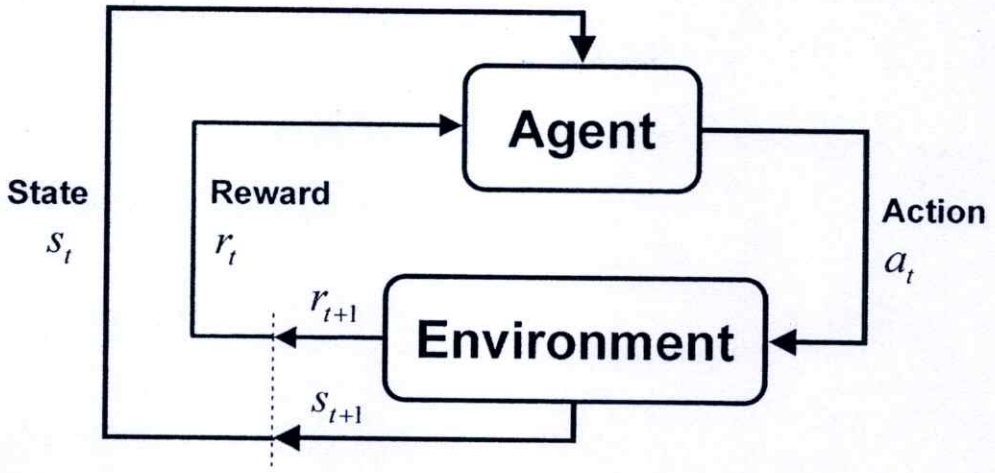
จากนิยามของปัญหา MCSS แสดงให้เห็นว่า ปัญหา MCSS เป็นปัญหาการจัดกลุ่มที่มีความซับซ้อนสูง ทำให้การสำรวจปริภูมิการค้นหาอย่างทั่วถึงไม่สามารถค้นหาคำตอบที่ดีที่สุดได้ในเวลาที่จำกัด อัลกอริธึมที่ใช้ในการแก้ปัญหา MCSS โดยทั่วไปจึงมีที่มาจากอัลกอริธึมที่ใช้ได้ดีในการหาคำตอบที่ดีที่สุด เช่น เจเนติกอัลกอริธึม และการค้นหาแบบทาบู เป็นต้น

2.4 การเรียนรู้แบบเสริมกำลัง

การเรียนรู้แบบเสริมกำลัง (reinforcement learning: RL) [8] เป็นแนวคิดเกี่ยวกับการเรียนรู้ของเครื่องคำนวณ การเรียนรู้จะได้มาจากการลองผิดลองถูก และการคาดหวังผลระยะยาว ซึ่งอยู่ในรูปแบบของรางวัลที่กำหนดเป็นค่าตัวเลขซึ่งแสดงผลความสำเร็จของการกระทำ

2.4.1 การโต้ตอบระหว่างเอเจนต์กับสภาพแวดล้อม

โจทย์ปัญหาเกี่ยวกับการเรียนรู้แบบเสริมกำลัง เปรียบเสมือนกับโครงสร้างพื้นฐานของปัญหาเกี่ยวกับการเรียนรู้ผ่านการโต้ตอบเพื่อให้บรรลุเป้าหมายที่ต้องการ ระบบที่ทำการตัดสินใจและเรียนรู้เรียกว่า **เอเจนต์** (agent) ซึ่งเป็นระบบที่โต้ตอบกับ**สภาพแวดล้อม** (environment) หรือระบบที่อยู่ภายนอกเอเจนต์ การโต้ตอบระหว่างกันของเอเจนต์กับสภาพแวดล้อมจะเป็นไปอย่างต่อเนื่อง จากการเลือกการกระทำของเอเจนต์ และการตอบสนองต่อการกระทำเหล่านั้นของสภาพแวดล้อม ด้วยการให้ข้อมูลเกี่ยวกับสภาวะปัจจุบัน นอกจากนี้ สภาพแวดล้อมยังเป็นสิ่งที่กำหนดรางวัลที่เอเจนต์จะได้รับ ซึ่งเป็นผลของการตัดสินใจของเอเจนต์ ตลอดอายุการเรียนรู้ของเอเจนต์ รางวัลที่มีค่าสูงที่สุดคือสิ่งที่เอเจนต์พยายามเรียนรู้และปรับตัวเพื่อให้ได้มา รายละเอียดทั้งหมดของสภาพแวดล้อมเป็นสิ่งที่กำหนด**ทาสก์** (task) ซึ่งเป็นหนึ่งในองค์ประกอบของโจทย์ปัญหาเกี่ยวกับการเรียนรู้แบบเสริมกำลัง



รูปที่ 2.2 การโต้ตอบระหว่างเอเจนต์และสภาพแวดล้อมในการเรียนรู้แบบเสริมกำลัง

รูปที่ 2.2 แสดงแผนภาพการโต้ตอบระหว่างเอเจนต์กับสภาพแวดล้อมตามลำดับขั้นเวลา $t = 0, 1, 2, 3, \ldots$ ที่แต่ละลำดับขั้นเวลา t เอเจนต์จะได้รับข้อมูลเกี่ยวกับ**สถานะ** (state) ของสภาพแวดล้อม ซึ่งแทนด้วยสัญลักษณ์ $s_t \in S$ เมื่อ S คือเซตของสถานะทั้งหมดที่เป็นไปได้ และเลือก**การกระทำ** (action) ตามสถานะนั้น ซึ่งแทนด้วยสัญลักษณ์ $a_t \in A(s_t)$ เมื่อ $A(s_t)$ คือเซตของการกระทำที่สามารถทำได้ในสถานะ s_t ในลำดับขั้นเวลาขั้นถัดไปซึ่งเป็นผลจากการกระทำของเอเจนต์ เอเจนต์จะได้รับ**รางวัล** (reward) ในรูปของค่าตัวเลข $r_{t+1} \in \Re$ และสถานะของสภาพแวดล้อมจะเปลี่ยนไปเป็น s_{t+1}

การตัดสินใจเลือกการกระทำของเอเจนต์ในแต่ละลำดับขั้นเวลา จะเป็นไปตามฟังก์ชันซึ่งกำหนดค่าความเป็นไปได้ในการเลือกการกระทำในสถานะต่าง ๆ ของสภาพแวดล้อม ฟังก์ชันนี้เรียกว่า**โพลิซี** (policy) ซึ่งแทนด้วยสัญลักษณ์ π_t โดย $\pi_t(s,a)$ หมายถึง ค่าความน่าจะเป็นที่ $a_t = a$ เมื่อ $s_t = s$ วิธีการต่าง ๆ ในการเรียนรู้แบบเสริมกำลังจะกำหนดการปรับเปลี่ยนโพลิซีของเอเจนต์ผ่านประสบการณ์ของเอเจนต์เอง เป้าหมายของเอเจนต์ คือ การทำให้ได้มาซึ่ง**ค่าคาดหมายของผลตอบแทน** (expected return) ที่สูงที่สุด ในที่นี้ ผลตอบแทน R_t หมายถึง ฟังก์ชันของลำดับของรางวัลที่ได้รับหลังจากลำดับขั้นเวลา t ซึ่งสามารถเขียนได้ในรูปแบบของลำดับเลขคณิต $r_{t+1}, r_{t+2}, r_{t+3}, \ldots$ ฟังก์ชัน R_t สามารถเขียนให้อยู่ในรูปทั่วไปได้ดังนี้

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{2.4}$$

เมื่อ γ เป็นค่า**อัตราการลดค่า** (discount rate) และ T เป็นค่าของลำดับขั้นเวลาขั้นสุดท้าย ในกรณีที่ทาสก์ของเอเจนต์สามารถแบ่งเป็น**เอพิโสด** (episode) ได้ γ จะมีค่าเป็น 1 และ T จะต้องมีค่าจำกัดซึ่งไม่เท่ากับ ∞ ในกรณีที่ทาสก์ของเอเจนต์เป็นแบบต่อเนื่อง γ จะมีค่าอยู่ในช่วง (0, 1) และ T จะมีค่าเข้าใกล้ ∞ เนื่องจากเป็นทาสก์ที่ไม่มีสถานะจบ และไม่มีการเริ่มต้นเอพิโสดใหม่

2.4.2 กระบวนการตัดสินใจแบบมาร์คอฟ

โดยทั่วไป ทาสก์ในการเรียนรู้แบบเสริมกำลังส่วนใหญ่สามารถอธิบายได้ในรูปแบบกระบวนการตัดสินใจแบบมาร์คอฟ (Markov decision process: MDP) [10] ซึ่งประกอบด้วย เซตของสถานะที่เป็นไปได้ เซตของการกระทำทั้งหมดที่เป็นไปได้ ความน่าจะเป็นของการเปลี่ยนสถานะ (transition probabilities) และค่าคาดหมายของรางวัลที่จะได้รับโดยทันที (expected immediate reward) ความน่าจะเป็นของการเปลี่ยนสถานะจากสถานะ s ไปสถานะ s' เมื่อเลือกการกระทำ a สามารถเขียนในรูปทั่วไปได้ดังนี้

$$P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{2.5}$$

และค่าคาดหมายของรางวัลที่จะได้รับโดยทันทีหลังจากเปลี่ยนสถานะจากสถานะ s ไปสถานะ s' จากการเลือกการกระทำ a สามารถเขียนในรูปทั่วไปได้ดังนี้

$$R_a(s, s') = E(r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s') \tag{2.6}$$

MDP มีประโยชน์ในการกำหนดรายละเอียดทั้งหมดเกี่ยวกับสภาพแวดล้อมในการเรียนรู้แบบเสริมกำลัง และมีส่วนสำคัญในการทำความเข้าใจอัลกอริธึมที่เกี่ยวกับการเรียนรู้แบบเสริมกำลัง เนื่องจาก MDP สามารถอธิบายการเปลี่ยนแปลงต่าง ๆ ที่เกิดขึ้นในการเรียนรู้แบบเสริมกำลังได้เป็นอย่างดี

2.4.3 ฟังก์ชันมูลค่า

ฟังก์ชันมูลค่า (value function) คือ ฟังก์ชันที่ใช้ในอัลกอริธึมเกี่ยวกับการเรียนรู้แบบเสริมกำลัง ในการประเมินมูลค่าหรือประโยชน์ของการอยู่ในสถานะที่กำหนด หรือประโยชน์ของการเลือกการกระทำใด.ๆ ในสถานะที่กำหนด ซึ่งเป็นฟังก์ชันของสถานะ หรือฟังก์ชันของคู่ของสถานะและการกระทำ ซึ่งมูลค่าในที่นี้กำหนดโดยค่าคาดหมายของผลตอบแทน และแปรเปลี่ยนไปตามโพลิซีที่ใช้

มูลค่าของสถานะ $s \in S$ ภายใต้โพลิซี π ซึ่งแทนด้วยสัญลักษณ์ $V^{\pi}(s)$ หมายถึง ค่าคาดหมายของผลตอบแทนซึ่งนับเริ่มต้นจากสถานะ s และดำเนินไปตามโพลิซี π ซึ่งเป็นตัวกำหนดความน่าจะเป็น $\pi(s, a)$ ในการเลือกการกระทำ $a \in A(s)$ เมื่ออยู่ในสถานะ s การคำนวณค่า $V^{\pi}(s)$ สามารถเขียนในรูปของสมการสำหรับ MDP ได้ดังนี้

$$V^{\pi}(s) = E_{\pi}(R_t \mid s_t = s) = E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s) \tag{2.7}$$

เมื่อ E_{π} หมายถึง ค่าคาดหมายของผลตอบแทนเมื่อกำหนดให้เอเจนต์ดำเนินตามโพลิซี π และ t หมายถึง ลำดับขั้นเวลาใด ๆ ในกรณีที่ทาสก์มีสถานะสิ้นสุด มูลค่าของสถานะสิ้นสุดจะมีค่าเป็น

ศูนย์เสมอ ฟังก์ชัน V^{π} นี้เรียกว่า **ฟังก์ชันมูลค่าของสถานะตามโพลิซี** π (state-value function for policy π)

ในทำนองเดียวกัน มูลค่าของการเลือกการกระทำ a ในสถานะ s ภายใต้โพลิซี π ซึ่งแทนด้วยสัญลักษณ์ $Q^{\pi}(s,a)$ หมายถึง ค่าคาดหมายของผลตอบแทนซึ่งนับเริ่มต้นจากสถานะ s ผ่านตัดสินใจเลือกการกระทำ a และดำเนินไปตามโพลิซี π การคำนวณค่า $Q^{\pi}(s,a)$ สามารถเขียนในรูปของสมการสำหรับ MDP ได้ดังนี้

$$Q^{\pi}(s,a) = E_{\pi}(R_t \mid s_t = s, a_t = a) = E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a) \qquad (2.8)$$

ฟังก์ชัน Q^{π} นี้เรียกว่า **ฟังก์ชันมูลค่าของการกระทำตามโพลิซี** π (action-value function for policy π)

โดยทั่วไป วิธีการตามแนวทางการเรียนรู้แบบเสริมกำลังจะมุ่งเน้นไปที่การแก้ปัญหาสองเรื่องที่มีความสัมพันธ์กัน คือ ปัญหาการทำนายค่า (prediction problem) ซึ่งเป็นการเรียนรู้ฟังก์ชันมูลค่าเมื่อดำเนินไปตามโพลิซีใด ๆ และปัญหาการควบคุม (control problem) ซึ่งเป็นการปรับเปลี่ยนหรือค้นหาโพลิซีเพื่อให้ได้มาซึ่งโพลิซีที่เหมาะสมที่สุด

สิ่งที่สำคัญสำหรับการได้มาซึ่งผลตอบแทนที่สูงที่สุดในการเรียนรู้แบบเสริมกำลัง คือ การปรับสมดุลระหว่างการสำรวจและการใช้ประโยชน์จากความรู้เดิม ซึ่งเป็นผลจากการตัดสินใจเลือกการกระทำของเอเจนต์ วิธีการเลือกการกระทำแบบ**เอพซิลอน-กรีดี** (ε-greedy) เป็นวิธีหนึ่งที่เป็นที่นิยมที่สุดในวิธีการตัดสินใจเลือกการกระทำในการเรียนรู้แบบเสริมกำลัง ซึ่งใช้การเลือกการกระทำที่มีมูลค่าสูงที่สุดเป็นส่วนใหญ่เพื่อใช้ประโยชน์จากความรู้ที่มีอยู่เดิม แต่ในบางครั้งจะใช้การสุ่มเลือกการกระทำเพื่อสำรวจการกระทำอื่น ๆ ไปด้วย โดยกำหนดให้ความน่าจะเป็นในการสุ่มเลือกการกระทำที่มีมูลค่าสูงที่สุดมีค่าเป็น $1-\varepsilon+\frac{\varepsilon}{|A(s)|}$ และความน่าจะเป็นในการสุ่มเลือกการกระทำอื่น ๆ มีค่าเป็น $\frac{\varepsilon}{|A(s)|}$ เมื่อ $|A(s)|$ หมายถึงจำนวนการกระทำที่เป็นไปได้ และ ε เป็นอัตราการสำรวจซึ่งมักกำหนดด้วยค่าน้อย ๆ แต่ไม่เท่ากับศูนย์

นอกจากวิธีการเลือกการกระทำแบบเอพซิลอน-กรีดีแล้ว วิธีการเลือกการกระทำที่สำคัญอีกวิธีหนึ่งคือ การเลือกการกระทำแบบ**ซอฟท์แมกซ์** (softmax) ข้อดีของวิธีซอฟท์แมกซ์คือ ความน่าจะเป็นในการสุ่มเลือกการกระทำที่มีค่าสูงจะมีค่ามาก ถึงแม้ว่าการกระทำนั้นจะไม่ใช่การกระทำที่มีมูลค่าสูงที่สุด ทำให้ความน่าจะเป็นในการเลือกการกระทำที่มีคุณค่าน้อยมีน้อยกว่าความน่าจะเป็นในการเลือกการกระทำที่มีค่าใกล้เคียงกับการกระทำที่มีค่าสูงที่สุด ซึ่งต่างจากวิธีการเลือกการกระทำแบบเอพซิลอน-กรีดีที่ใช้การสุ่มเลือกการกระทำอื่น ๆ ที่ไม่ใช่การกระทำที่มีค่าสูงที่สุดด้วยความน่าจะเป็นที่เท่า ๆ กัน โดยทั่วไป วิธีการแบบซอฟท์แมกซ์ที่นิยมที่สุดจะใช้การแจกแจงแบบกิบส์ (Gibbs distribution) หรือที่เรียกโดยทั่วไปว่าการแจกแจงโบลทซ์มันน์

(Boltzmann distribution) ซึ่งกำหนดความน่าจะเป็นในการเลือกการกระทำ a ณ เวลา t ดังสมการ

$$\Pr(a_t = a) = \frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}} \tag{2.9}$$

เมื่อ $Q_t(a)$ เป็นฟังก์ชันมูลค่าของการกระทำ และ τ หมายถึง พารามิเตอร์ที่เรียกว่า**อุณหภูมิ** (temperature) ซึ่งมีค่าเป็นบวก ค่าอุณหภูมิที่สูงจะส่งผลให้ความน่าจะเป็นในการเลือกการกระทำต่าง ๆ มีค่าเท่า ๆ กัน อุณหภูมิที่ต่ำลงจะส่งผลให้เกิดความแตกต่างของความน่าจะเป็นในการเลือกการกระทำที่มีมูลค่าแตกต่างกัน เมื่อค่าพารามิเตอร์ τ มีค่าเข้าใกล้ศูนย์ การเลือกการกระทำจะให้ผลเหมือนกับการเลือกการกระทำแบบกรีดี

2.4.4 การเรียนรู้ผลต่างระหว่างเวลา

การเรียนรู้ผลต่างระหว่างเวลา (temporal difference learning: TD learning) [11] เป็นอัลกอริธึมหลักสำหรับใช้ในการทำนายค่าของฟังก์ชันมูลค่า และเป็นหนึ่งในอัลกอริธึมที่สำคัญที่สุดในการเรียนรู้แบบเสริมกำลัง เนื่องจากอัลกอริธึมส่วนใหญ่ที่เกี่ยวกับการเรียนรู้แบบเสริมกำลังใช้การเรียนรู้ผลต่างระหว่างเวลาเป็นพื้นฐานในการเรียนรู้ฟังก์ชันมูลค่า การปรับค่าประมาณของฟังก์ชันมูลค่าด้วยการเรียนรู้แบบนี้ จะเป็นไปตามค่าความผิดพลาดของผลต่างระหว่างเวลา (temporal difference error: TD error) ซึ่งแทนด้วยสัญลักษณ์ δ_t และนิยามได้ดังนี้

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \tag{2.10}$$

เมื่อ r_{t+1} หมายถึง รางวัลที่ได้รับ ณ เวลา $t+1$, V_t หมายถึง ฟังก์ชันมูลค่า ณ เวลา t, s_t หมายถึง ข้อมูลสถานะของสภาพแวดล้อมที่เอเจนต์รับรู้ และ γ หมายถึง อัตราการลดค่า การปรับค่าของฟังก์ชันมูลค่าจึงเป็นไปดังนี้

$$V_{t+1}(s) = V_t(s) + \alpha\delta_t e_t(s) \tag{2.11}$$

สำหรับทุก ๆ สถานะ $s \in S$ และกำหนดให้ α เป็นค่าคงที่ของ**ระดับการปรับค่า** (step-size) ฟังก์ชัน $e_t(s)$ หมายถึง **รอยทางการปรับมูลค่า** (eligibility trace) ซึ่งอยู่ในรูปแบบตัวแปรความจำชั่วคราวของเอเจนต์ โดยแสดงถึงระดับการปรับเปลี่ยนค่าของฟังก์ชันมูลค่าสำหรับสถานะ s ณ เวลา t และนิยามได้ดังนี้

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{เมื่อ } s \neq s_t \\ \gamma\lambda e_{t-1}(s)+1 & \text{เมื่อ } s = s_t \end{cases} \quad (2.12)$$

เมื่อ γ เป็นค่าอัตราการลดค่า และ λ เป็น**พารามิเตอร์การเสื่อมสลายของรอยทาง** (trace-decay parameter) ซึ่งรอยทางการปรับมูลค่าแบบนี้เรียกว่า **รอยทางสะสม** (accumulating trace) จากสมการข้างต้น รอยทางการปรับมูลค่าที่สถานะล่าสุดที่เอเจนต์ไปเยือนจะมีค่าเพิ่มขึ้นด้วยค่า 1 ขณะที่รอยทางของสถานะอื่น ๆ จะค่อย ๆ จางหายไปเนื่องจากไม่มีการมาเยือนของเอเจนต์

นอกจากการใช้รอยทางสะสมในการปรับฟังก์ชันมูลค่าแล้ว ยังมีการใช้รอยทางการปรับมูลค่าแบบอื่น ๆ อีก ตัวอย่างเช่น การใช้**รอยทางแทนที่** (replacing trace) ซึ่งอาจทำให้ได้อัตราการเรียนรู้ที่ดีขึ้น ฟังก์ชัน $e_t(s)$ สำหรับรอยทางแทนที่สามารถนิยามได้ดังนี้

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{เมื่อ } s \neq s_t \\ 1 & \text{เมื่อ } s = s_t \end{cases} \quad (2.13)$$

อัลกอริธึม TD ที่ใช้รอยทางการปรับมูลค่า สามารถเรียนรวมได้เป็น อัลกอริธึม TD(λ) การใช้รอยทางการปรับมูลค่าทำให้อัลกอริธึม TD มีความยืดหยุ่นสูง เนื่องจากส่งผลโดยตรงกับอัตราการปรับมูลค่า ซึ่งควบคุมได้ด้วยการปรับค่าพารามิเตอร์ λ อัลกอริธึม TD ที่ใช้ค่า $\lambda = 0$ นั้นเรียกว่า TD(0) หรือ TD หนึ่งขั้นตอน (one-step TD) จะปรับค่าเฉพาะมูลค่าของสถานะล่าสุดในทันที หลังจากที่มีการเลือกการกระทำที่สถานะนั้น และได้รับรางวัลกลับมาในแต่ละขั้นตอนการตัดสินใจ ทำให้อัลกอริธึมสามารถเรียนรู้ถึงการเปลี่ยนแปลงและปรับการทำนายค่าให้ถูกต้องได้ทันทีในแบบออนไลน์ และใช้การปรับเปลี่ยนทีละน้อยโดยสมบูรณ์ โดยไม่จำเป็นต้องจำลำดับการเยือนสถานะต่าง ๆ ที่เกิดขึ้นในเอพิโสดเดียวกัน ในทางกลับกัน อัลกอริธึม TD ที่ใช้ค่า $\lambda = 1$ หรือ TD(1) จะใช้การปรับมูลค่าของสถานะทุก ๆ สถานะที่มีการไปเยือนจากเอเจนต์ตั้งแต่เริ่มต้นเอพิโสดจนกระทั่งถึงสถานะปัจจุบันในแต่ละขั้นตอน ซึ่งทำให้ใช้เวลาในการคำนวณ รวมถึงหน่วยความจำชั่วคราวสำหรับเก็บลำดับการเยือนสถานะต่าง ๆ มากขึ้นเมื่อเหตุการณ์ดำเนินไป แต่ก็ทำให้การทำนายค่าของฟังก์ชันมูลค่ามีความถูกต้องมากขึ้น การกำหนดค่าพารามิเตอร์ λ นอกเหนือไปจากค่า 0 และค่า 1 จึงเป็นการปรับความสมดุลระหว่างคุณภาพของการทำนายค่าและเวลาที่ใช้ในการปรับค่าของอัลกอริธึม การประยุกต์ใช้อัลกอริธึม TD จึงสามารถปรับเปลี่ยนให้สอดคล้องกับลักษณะของโจทย์ปัญหาได้เป็นอย่างดี

2.4.5 อัลกอริธึม Sarsa(λ)

อัลกอริธึม Sarsa(λ) [12] เป็นวิธีการเรียนรู้แบบออนโพลิซี (on-policy learning) ซึ่งใช้การเรียนรู้ฟังก์ชันมูลค่าของโพลิซีที่เลือกใช้ในการตัดสินใจ อัลกอริธึม Sarsa(λ) ใช้การเรียนรู้แบบ TD เป็นพื้นฐานในการทำนายค่าของฟังก์ชันมูลค่า และเรียนรู้การปรับเปลี่ยนโพลิซีโดยใช้ฟังก์ชันมูลค่าของการกระทำ ในที่นี้ ค่าความผิดพลาดของผลต่างระหว่างเวลาหรือ δ_t สามารถนิยามได้ดังนี้

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \tag{2.14}$$

สมการนี้มาจากการแทนที่ฟังก์ชัน $V_t(s)$ ในสมการ (2.10) ด้วยฟังก์ชัน $Q_t(s, a)$ ซึ่งหมายถึงการทำนายค่าของการกระทำ ณ สถานะใดสถานะหนึ่ง แทนที่การทำนายเฉพาะค่าของสถานะ การปรับค่าของฟังก์ชันมูลค่าจึงเป็นไปดังนี้

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha\delta_t e_t(s, a) \tag{2.15}$$

สำหรับทุก ๆ สถานะ $s \in S$ และทุก ๆ การกระทำ $a \in A(s)$ ซึ่งในที่นี้ นอกจากรอยทาง e_t ที่ใช้จะมีค่าต่างกันไปตามสถานะแล้ว ยังมีค่าต่างกันตามการกระทำต่าง ๆ ที่สถานะนั้น ๆ อีกด้วย ในสมการนี้จึงกำหนดรอยทางเป็นฟังก์ชัน $e_t(s, a)$ แทนการใช้ $e_t(s)$ ดังนั้น ค่าของฟังก์ชันจึงสามารถคำนวณได้ดังนี้

$$e_t(s, a) = \begin{cases} \gamma\lambda e_{t-1}(s, a) + 1 & \text{เมื่อ } s = s_t \text{ และ } a = a_t \\ \gamma\lambda e_{t-1}(s, a) & \text{สำหรับกรณีอื่น ๆ} \end{cases} \tag{2.16}$$

สำหรับทุก ๆ สถานะ s และทุก ๆ การกระทำ a

รูปที่ 2.3 แสดงซูโดโค้ดของอัลกอริธึม Sarsa(λ) ในกรณีที่ใช้ตารางค่าของฟังก์ชันในการปรับค่าของฟังก์ชันมูลค่า

กำหนดค่า $Q(s, a)$ ตามต้องการ และ $e(s, a) = 0$ สำหรับทุก ๆ ค่าของ s, a
ทำซ้ำ (ในแต่ละเอพิโสด):
  กำหนดค่าเริ่มต้นให้กับ s, a
  ทำซ้ำ (ในแต่ละขั้นตอนของเอพิโสด):
    โต้ตอบด้วยการกระทำ a และพิจารณาค่าของ r, s' ที่ได้รับ
    เลือกการกระทำ a' ตามสถานะ s' โดยใช้โพลิซีที่คำนวณได้จากฟังก์ชัน Q
      (เช่น เอพซิลอน-กรีดี)
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
    $e(s, a) \leftarrow e(s, a) + 1$
    สำหรับทุก ๆ ค่าของ s, a :
      $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
      $e(s, a) \leftarrow \gamma\lambda e(s, a)$
    $s \leftarrow s'; a \leftarrow a'$
  จนกระทั่งสถานะ s เป็นสถานะสิ้นสุด

รูปที่ 2.3 อัลกอริธึม Sarsa(λ) แบบใช้ตารางค่าของฟังก์ชัน

2.4.6 อัลกอริธึม Q(λ)

อัลกอริธึม Q(λ) [13] เป็นอัลกอริธึมการเรียนรู้แบบออฟโพลิซี (off-policy learning) ซึ่งใช้การเรียนรู้ฟังก์ชันมูลค่าสำหรับโพลิซีแบบหนึ่ง ขณะที่ตัดสินใจด้วยโพลิซีอีกแบบหนึ่งที่แตกต่างกัน อัลกอริธึม Q(λ) ใช้การเรียนรู้แบบ TD เป็นพื้นฐานในการทำนายค่าของฟังก์ชันมูลค่า และเรียนรู้การปรับเปลี่ยนโพลิซีโดยใช้ฟังก์ชันมูลค่าของการกระทำเช่นเดียวกับอัลกอริธึม Sarsa(λ) แต่ต่างกันที่การคำนวณค่าความผิดพลาดของผลต่างระหว่างเวลาหรือ δ_t ซึ่งสามารถนิยามได้ดังนี้

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \qquad (2.17)$$

สมการนี้มาจากการแทนที่ $Q_t(s_{t+1}, a_{t+1})$ ในสมการ (2.14) ด้วย $\max_{a'} Q_t(s_{t+1}, a')$ ซึ่งหมายถึงมูลค่าของการกระทำที่คาดคะเนว่าเป็นการกระทำที่ดีที่สุด ณ สถานะถัดไป การปรับค่าของฟังก์ชันมูลค่าในอัลกอริธึม Q(λ) เป็นไปตามสมการ (2.15) เช่นเดียวกับอัลกอริธึม Sarsa(λ) แต่ใช้การคำนวณฟังก์ชัน $e_t(s, a)$ ดังนี้

$$e_t(s,a) = I_{ss_t \cap aa_t} + \begin{cases} \gamma\lambda e_{t-1}(s,a) & \text{เมื่อ } Q_{t-1}(s_t,a_t) = \max_a Q_{t-1}(s_t,a) \\ 0 & \text{สำหรับกรณีอื่น ๆ} \end{cases} \quad (2.18)$$

สำหรับทุก ๆ สถานะ s และทุก ๆ การกระทำ a และกำหนดให้ $I_{ss_t \cap aa_t} = I_{ss_t} \cdot I_{aa_t}$ โดย I_{xy} หมายถึง ฟังก์ชันบ่งชี้เอกลักษณ์ (identity indicator function) ซึ่งมีค่าเท่ากับ 1 เมื่อ $x = y$ และมีค่าเท่ากับ 0 สำหรับกรณีอื่น ๆ

รูปที่ 2.4 แสดงซูโดโค้ดของอัลกอริธึม Q(λ) แบบวัตคิน (Watkin's Q(λ)) ในกรณีที่ใช้ตารางค่าของฟังก์ชันในการปรับค่าของฟังก์ชันมูลค่า

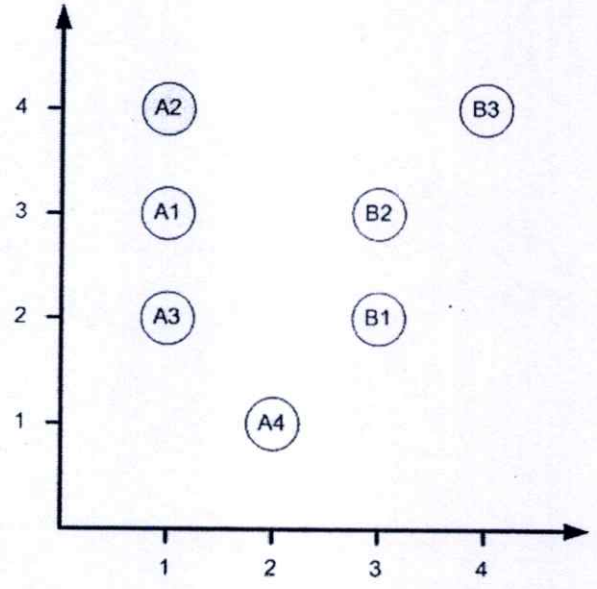
กำหนดค่า $Q(s,a)$ ตามต้องการ และ $e(s,a) = 0$ สำหรับทุก ๆ ค่าของ s, a
ทำซ้ำ (ในแต่ละเอพิโสด):
  กำหนดค่าเริ่มต้นให้กับ s, a
  ทำซ้ำ (ในแต่ละขั้นตอนของเอพิโสด):
    โต้ตอบด้วยการกระทำ a และพิจารณาค่าของ r, s' ที่ได้รับ
    เลือกการกระทำ a' ตามสถานะ s' โดยใช้โพลิซีที่คำนวณได้จากฟังก์ชัน Q
      (เช่น เอพซิลอน-กรีดี)
    $a^* \leftarrow \arg\max_b Q(s',b)$ (ถ้า a' มีค่าเท่ากับการกระทำที่มีค่ามากที่สุด แล้ว $a^* \leftarrow a'$)
    $\delta \leftarrow r + \gamma Q(s',a^*) - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    สำหรับทุก ๆ ค่าของ s, a:
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
      ถ้า $a' = a^*$ แล้ว $e(s,a) \leftarrow \gamma\lambda e(s,a)$ นอกจากนี้ $e(s,a) \leftarrow 0$
    $s \leftarrow s'; a \leftarrow a'$
  จนกระทั่งสถานะ s เป็นสถานะสิ้นสุด

รูปที่ 2.4 อัลกอริธึม Q(λ) แบบวัตคินในรูปแบบการใช้ตารางค่าของฟังก์ชัน

บทที่ 3
การเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดโดยใช้การเรียนรู้แบบเสริมกำลัง

ในบทนี้จะกล่าวถึง การประยุกต์ใช้การเรียนรู้แบบเสริมกำลังในการแก้ปัญหาการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดหรือปัญหา MCSS ซึ่งในที่นี้ จะพิจารณาการกำหนดทาสก์การเรียนรู้ปัญหา MCSS ใน 2 ลักษณะด้วยกัน คือ การค้นหาเซตย่อยสอดคล้องด้วยการเพิ่มจำนวนโปรโตไทป์ (prototype insertion approach) และการค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ (prototype removal approach)

เพื่อให้ง่ายต่อการทำความเข้าใจ การอธิบายทาสก์การเรียนรู้ปัญหา MCSS จะใช้ชุดข้อมูลตัวอย่างที่ 1 ซึ่งมีลักษณะดังแสดงในรูปที่ 3.1 เป็นการอ้างอิง นอกจากมีการระบุเป็นอย่างอื่น



รูปที่ 3.1 กราฟแสดงคุณลักษณะ 2 มิติของชุดข้อมูลตัวอย่างที่ 1

ชุดข้อมูลตัวอย่างในรูปที่ 3.1 ประกอบด้วยตัวอย่างข้อมูลทั้งหมด 7 ตัว ซึ่งจัดเป็นชนิด A 4 ตัว (A1-A4) และชนิด B 3 ตัว (B1-B3) คุณลักษณะของข้อมูลแต่ละตัวประกอบด้วยข้อมูลพิกัด 2 มิติ คือ x และ y และมีรายละเอียดดังแสดงในตารางที่ 3.1

ตารางที่ 3.1 คุณลักษณะของข้อมูลในชุดข้อมูลตัวอย่างที่ 1

ข้อมูล	ชนิด	(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 3.1 เมทริกซ์ระยะทาง (distance matrix) ระหว่างข้อมูลแต่ละคู่ของชุดข้อมูลตัวอย่างที่ 1 ซึ่งได้มาจากการคำนวณระยะทางแบบยูคลิด (Euclidean distance) มีค่าดังแสดงในตารางที่ 3.2

ตารางที่ 3.2 เมทริกซ์ระยะทางของข้อมูลในชุดข้อมูลตัวอย่างที่ 1

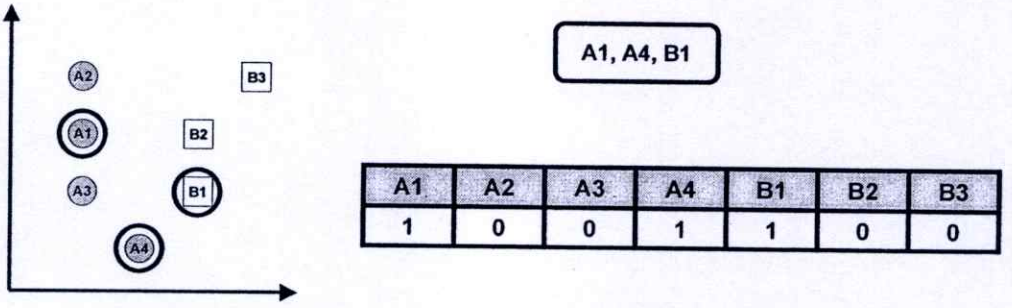
	A1	**A2**	**A3**	**A4**	**B1**	**B2**	**B3**
A1	**0.00**	**1.00**	**1.00**	**2.24**	**2.24**	**2.00**	**3.16**
A2	**1.00**	**0.00**	**2.00**	**3.16**	**2.83**	**2.24**	**3.00**
A3	**1.00**	**2.00**	**0.00**	**1.41**	**2.00**	**2.24**	**3.61**
A4	**2.24**	**3.16**	**1.41**	**0.00**	**1.41**	**2.24**	**3.61**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

3.1 การประยุกต์ใช้การเรียนรู้แบบเสริมกำลังในการแก้ปัญหา MCSS

รายละเอียดเกี่ยวกับองค์ประกอบต่าง ๆ ที่ต้องพิจารณาในการประยุกต์ใช้การเรียนรู้แบบเสริมกำลังในการแก้ปัญหา MCSS มีดังนี้

3.1.1 รูปแบบสถานะของสภาพแวดล้อม

สำหรับปัญหา MCSS สถานะปัจจุบันของสภาพแวดล้อมของการเรียนรู้แบบเสริมกำลังจะต้องมีข้อมูลเกี่ยวกับโปรโตไทป์ที่ได้เลือกมาใช้เป็นสมาชิกของเซตตัวแทนในขณะนั้น ดังนั้นเซตตัวแทนซึ่งอยู่ในรูปของบิตเวคเตอร์ (bit vector) หรือบิตเซต (bit set) จึงเป็นรูปแบบสถานะที่เหมาะสมที่สุด เนื่องจากมีความสอดคล้องกับโจทย์ปัญหาโดยตรง เช่น ในชุดข้อมูลตัวอย่างที่ 1 ถ้ามีการเลือกโปรโตไทป์ A1, A4 และ B1 เป็นสมาชิกของเซตตัวแทนแล้ว รูปแบบสถานะปัจจุบันจะเป็นไปดังแสดงในรูปที่ 3.2



รูปที่ 3.2 รูปแบบสถานะของสภาพแวดล้อมในปัญหา MCSS

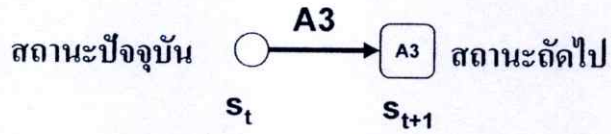
จากรูปที่ 3.2 บิตที่มีค่า 1 จะแสดงถึงการเลือกโปรโตไทป์นั้นเป็นสมาชิกของเซตตัวแทน และบิตที่มีค่า 0 จะแสดงว่าโปรโตไทป์นั้นไม่ได้เป็นสมาชิกของเซตตัวแทน ซึ่งในที่นี้ เซตตัวแทนที่ได้คือ เซต {A1, A4, B1}

จากรูปแบบสถานะดังกล่าว การกำหนดสถานะเริ่มต้นสามารถทำได้หลายรูปแบบด้วยกัน เช่น การเริ่มจากสถานะเซตว่าง การเริ่มจากสถานะเซตที่แทนชุดข้อมูลเริ่มต้น (เซตเริ่มต้น) หรือการเริ่มจากสถานะเซตย่อยใด ๆ ของเซตเริ่มต้น ก็สามารถทำได้

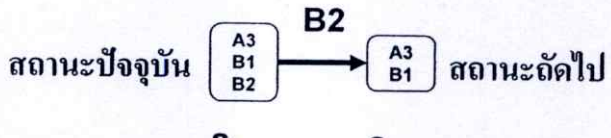
ในทำนองเดียวกัน การกำหนดสถานะสิ้นสุดก็สามารถทำได้หลายรูปแบบ เช่น การสิ้นสุดที่สถานะเซตว่าง การสิ้นสุดที่สถานะเซตเริ่มต้น หรือการสิ้นสุดที่สถานะเซตย่อยใด ๆ ของเซตเริ่มต้น นอกจากนี้ เราอาจกำหนดให้สถานะสิ้นสุดเป็นสถานะที่เป็นเซตย่อยที่มีคุณสมบัติความสอดคล้อง หรือเราอาจกำหนดให้ไม่มีสถานะสิ้นสุด ซึ่งหมายถึง การกำหนดทาสก์การเรียนรู้ในรูปแบบทาสก์แบบต่อเนื่อง ก็สามารถทำได้เช่นกัน

3.1.2 รูปแบบการกระทำของเอเจนต์

จากรูปแบบสถานะในหัวข้อที่แล้ว เราสามารถกำหนดรูปแบบการกระทำของเอเจนต์ได้ 2 ลักษณะด้วยกันคือ การเพิ่มโปรโตไทป์ในเซตตัวแทน และการคัดโปรโตไทป์ออกจากเซตตัวแทน การย้ายสถานะที่เกิดจากการกระทำทั้ง 2 ลักษณะของเอเจนต์เป็นไปดังแสดงในรูปที่ 3.3



(ก) การเพิ่มโปรโตไทป์ในเซตตัวแทน



(ข) การคัดโปรโตไทป์ออกจากเซตตัวแทน

รูปที่ 3.3 การย้ายสถานะจากการกระทำของเอเจนต์ในปัญหา MCSS

จากรูปที่ 3.3 การกระทำทั้ง 2 ลักษณะนี้ก็คือการเปลี่ยนค่าบิตที่ใช้แทนสถานะการเลือกโปรโตไทป์ต่าง ๆ ภายในเซตตัวแทน โดยการเปลี่ยนจากค่า 0 ไปเป็นค่า 1 เป็นการเพิ่มสมาชิกหรือโปรโตไทป์ในเซตตัวแทน และการเปลี่ยนจากค่า 1 ไปเป็นค่า 0 ก็เป็นการคัดโปรโตไทป์ออกจากเซตตัวแทนนั่นเอง

3.1.3 การกำหนดรางวัล

เนื่องจากเอเจนต์ RL เรียนรู้การกระทำที่ดีที่สุดจากรางวัลที่ได้รับ การกำหนดรางวัลให้เหมาะสมจึงเป็นส่วนสำคัญในการออกแบบอัลกอริธึม RL สำหรับการแก้ปัญหา MCSS โดยใช้ RL เกณฑ์การให้รางวัลที่เหมาะสมคือ การให้รางวัลเมื่อสถานะที่พบเป็นสถานะที่มีความสอดคล้อง และเป็นสถานะที่มีจำนวนโปรโตไทป์น้อย ยิ่งมีสถานะที่มีความสอดคล้องมีจำนวนโปรโตไทป์น้อย ก็จะต้องให้รางวัลที่มีค่าสูงขึ้นตามสัดส่วน แต่ในกรณีที่สถานะนั้นไม่มีความสอดคล้อง รางวัลที่ได้ก็ควรจะมีค่าที่น้อยกว่าสถานะที่มีความสอดคล้องหรือกำหนดให้รางวัลที่ได้มีค่าเป็นศูนย์ ดังเช่นตัวอย่างการให้รางวัลในสถานะต่าง ๆ ซึ่งแสดงในรูปที่ 3.4

รูปที่ 3.4 ตัวอย่างการกำหนดรางวัลในสถานะต่าง ๆ ในปัญหา MCSS

จากรูปที่ 3.4 ถ้าเซตตัวแทนมีความสอดคล้อง รางวัลที่ได้จะมีค่าตามขนาดที่เล็กลง เช่น เซต {A1, A4, B1} จะให้รางวัลเป็นค่าเท่ากับ 4 ซึ่งมาจากจำนวนโปรโตไทป์ในเซตเริ่มต้น (7 ตัว) ลบด้วยจำนวนโปรโตไทป์ในเซต (3 ตัว) ในทางกลับกัน ถ้าเซตตัวแทนไม่มีความสอดคล้อง รางวัลที่ได้ก็จะมีค่าเป็นศูนย์ สำหรับเซตเริ่มต้น รางวัลที่ได้ก็มีค่าเป็นศูนย์เช่นกัน (7 - 7 = 0)

3.2 การค้นหาเซตย่อยสอดคล้องด้วยการเพิ่มจำนวนโปรโตไทป์

จากการพิจารณารายละเอียดต่าง ๆ ในหัวข้อที่แล้ว วิธีการพื้นฐานวิธีหนึ่งซึ่งใช้ในการค้นหาเซตย่อยสอดคล้องโดยใช้การเรียนรู้แบบเสริมกำลังคือ การเริ่มค้นหาโดยเริ่มต้นจากเซตตัวแทนซึ่งเป็นเซตว่าง จากนั้น ให้เพิ่มสมาชิกหรือโปรโตไทป์จากชุดข้อมูลเริ่มต้นเข้ามาเป็นส่วนประกอบของเซตทีละตัว หลังจากการเพิ่มสมาชิกหนึ่งตัว ให้นำเซตตัวแทนที่ได้ไปใช้ในการจำแนกข้อมูลทั้งหมดเพื่อทดสอบคุณสมบัติความสอดคล้อง ถ้าเซตตัวแทนที่ได้สามารถใช้จำแนกข้อมูลทั้งหมดได้อย่างถูกต้อง เซตตัวแทนนั้นก็จะเป็นเซตคำตอบที่เหมาะสม เนื่องจากมีความสอดคล้อง ทำให้กระบวนการค้นหาสิ้นสุดสำหรับหนึ่งเอพิโสด

เนื่องจากคำตอบที่ได้จากการค้นหาด้วยวิธีนี้ ขึ้นอยู่กับลำดับการเลือกโปรโตไทป์เพื่อใช้เป็นสมาชิกของเซตตัวแทน การเลือกโปรโตไทป์ด้วยลำดับที่ต่างกันจึงมีผลให้คำตอบที่ได้ต่างกัน การค้นหาเซตย่อยสอดคล้องด้วยวิธีนี้ จึงต้องใช้การทดลองเลือกโปรโตไทป์โดยการสุ่มเลือกให้ได้เซตย่อยสอดคล้องจำนวนมาก เพื่อหาว่าเซตย่อยที่มีขนาดเล็กที่สุดที่พบมีขนาดเท่าใด และประกอบด้วยโปรโตไทป์ตัวใดบ้าง

เทคนิคการค้นหาข้างต้น สามารถใช้กำหนดทาสก์การเรียนรู้แบบเสริมกำลังได้ในรูปแบบของทาสก์ที่แบ่งได้เป็นเอพิโสด (episodic task) ได้ดังนี้

1) กำหนดให้เซตตัวแทนเป็นข้อมูลเกี่ยวกับสถานะปัจจุบันดังที่ได้อธิบายในหัวข้อที่แล้ว
2) สถานะเริ่มต้นของแต่ละเอพิโสดในทาสก์นี้คือ สถานะเซตว่าง และสถานะสิ้นสุดก็คือ สถานะเซตตัวแทนที่มีความสอดคล้อง ซึ่งเป็นเซตคำตอบที่ได้ในแต่ละเอพิโสด

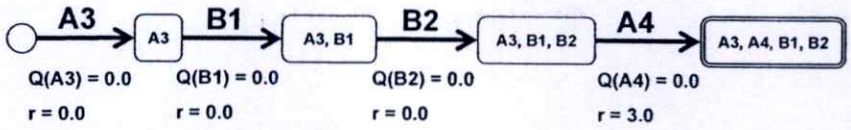
3) การกระทำที่เอเจนต์สามารถทำได้ คือ การเลือกโปรโตไทป์ที่ไม่ได้เป็นสมาชิกของสถานะปัจจุบันเพิ่มเข้ามาเป็นสมาชิกใหม่ ทำให้เกิดการเปลี่ยนสถานะไปสู่สถานะใหม่ที่มีสมาชิกมากขึ้น
4) รางวัลที่ได้รับจะมีค่าไม่เท่ากับศูนย์ ก็ต่อเมื่อ การเลือกโปรโตไทป์เพิ่มเข้ามาเป็นสมาชิกใหม่ทำให้เกิดการเปลี่ยนแปลงจากสถานะที่ไม่มีความสอดคล้องไปสู่สถานะที่มีความสอดคล้อง ซึ่งเป็นจุดสิ้นสุดของแต่ละเอพิโสด ทั้งนี้ ค่าของรางวัลจะขึ้นกับจำนวนสมาชิกของเซตตัวแทน ถ้าเซตตัวแทนที่ได้มีจำนวนสมาชิกมาก รางวัลจะมีค่าต่ำ ในทางกลับกัน ถ้าเซตมีสมาชิกน้อย รางวัลก็จะมีค่าสูงตามสัดส่วน

รางวัลที่ได้รับจากการเลือกการกระทำ a จากสถานะ s และเปลี่ยนไปสู่สถานะถัดไป หรือ s_n สามารถนิยามในรูปของฟังก์ชัน $R(s, a, s_n)$ ได้ดังนี้

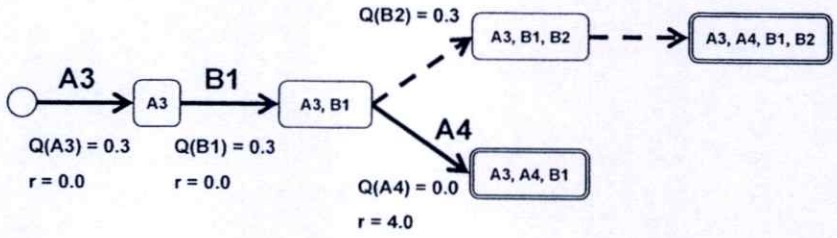
$$R(s, a, s_n) = \begin{cases} 0 & \text{เมื่อ } s_n \text{ ไม่มีความสอดคล้อง} \\ N - |s_n| & \text{เมื่อ } s_n \text{ มีความสอดคล้อง} \end{cases} \quad (3.1)$$

เมื่อ N หมายถึง จำนวนโปรโตไทป์ทั้งหมดในชุดข้อมูลเริ่มต้น และ $|s|$ หมายถึง จำนวนสมาชิกหรือโปรโตไทป์ของสถานะ s

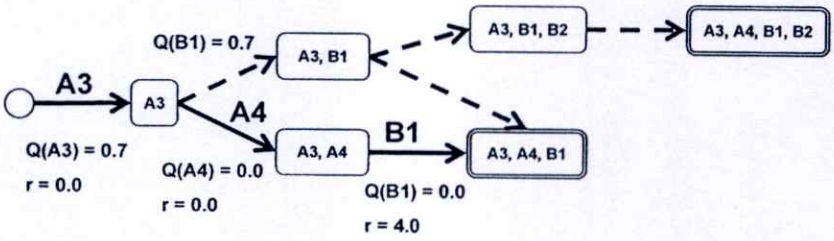
ตัวอย่างเช่น ในการค้นหาเซตย่อยสอดคล้องขนาดเล็กที่สุดของชุดข้อมูลตัวอย่างที่ 1 ลำดับการตัดสินใจของเอเจนต์ RL ในการโต้ตอบกับสภาพแวดล้อมการค้นหาด้วยเทคนิคการเพิ่มจำนวนโปรโตไทป์ อาจเป็นไปดังแสดงในรูปที่ 3.5



(ก) เอพิโสด e1

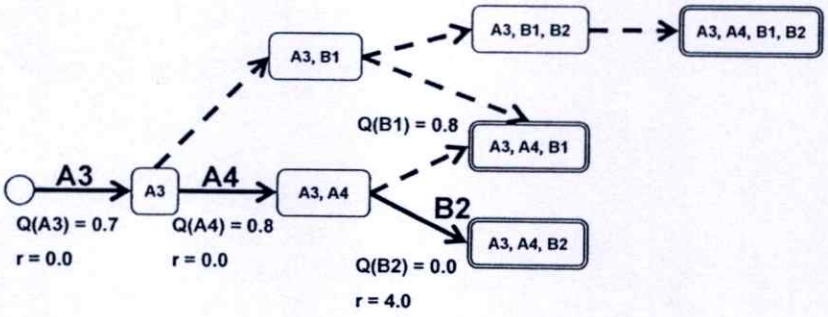


(ข) เอพิโสด e2

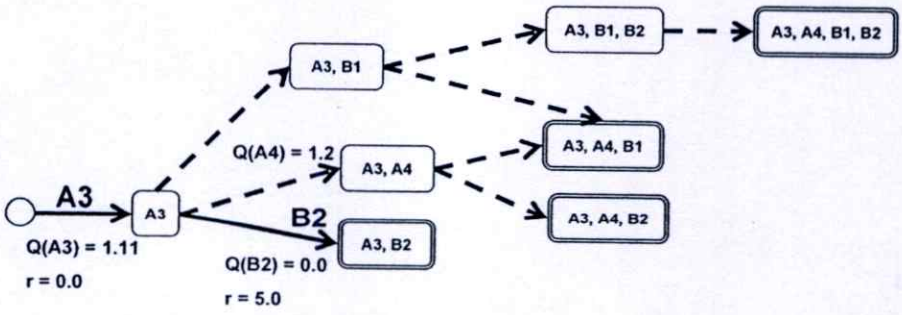


(ค) เอพิโสด e3

...



(ง) เอพิโสด e4



(จ) เอพิโสด e5

รูปที่ 3.5 ตัวอย่างลำดับการตัดสินใจในการเพิ่มจำนวนโปรโตไทป์ของเอเจนต์ RL

รูปที่ 3.5 แสดงลำดับการตัดสินใจเลือกโปรโตไทป์ของเอเจนต์ RL ภายในการทดลองหนึ่งครั้ง ซึ่งแบ่งเป็นหลายเอพิโสดด้วยกัน จากรูปที่ 3.5 (ก) ในเอพิโสด e1 ซึ่งเป็นเอพิโสดแรกของการทดลอง เอเจนต์ยังไม่มีข้อมูลสำหรับใช้ในการตัดสินใจ ทำให้ค่าของฟังก์ชันมูลค่าหรือฟังก์ชัน Q สำหรับทุก ๆ การกระทำในสถานะเริ่มต้น (ซึ่งแทนด้วยรูปวงกลม) และสถานะอื่น ๆ มีค่าเป็นศูนย์ ดังนั้น การตัดสินใจในทุก ๆ ขั้นตอนจึงมาจากการสุ่มเลือกการกระทำ ในเอพิโสดนี้เอเจนต์ได้เลือกโปรโตไทป์ A3, B1, B2 และ B4 ตามลำดับ เข้ามาเป็นสมาชิกในเซตตัวแทน และสิ้นสุดเอพิโสดที่สถานะ {A3, A4, B1, B2} (ซึ่งแทนด้วยรูปสี่เหลี่ยมที่มีเส้นรอบรูปเป็นเส้นคู่ขนาน) ซึ่งเป็นเซตคำตอบของเอพิโสด รางวัลที่เอเจนต์ได้รับในแต่ละขั้นตอนมีค่าเท่ากับศูนย์ยกเว้นที่ขั้นตอนสุดท้าย ซึ่งมีค่ารางวัลเท่ากับ +3 ($7-4$ โดย $N=7$ และ $|s_n|=4$)

ในแต่ละขั้นตอนการเลือกโปรโตไทป์ เอเจนต์จะปรับค่าของฟังก์ชัน Q ซึ่งในตัวอย่างนี้ใช้การปรับค่าตามวิธีการเรียนรู้แบบ Q(λ) โดยใช้ค่า $\lambda=1$ ซึ่งให้ผลเทียบเท่ากันกับการใช้วิธีการมอนติคาร์โล (Monte Carlo method) [8] และใช้ค่าคงที่ของระดับการปรับค่าหรือ $\alpha=0.1$ ดังนั้น หลังจากจบเอพิโสด e1 การกระทำที่มีค่า Q สูงที่สุดในสถานะเริ่มต้น จึงเป็นการเลือก A3 เป็นโปรโตไทป์ และมีค่าเท่ากับ +0.3 ดังนั้น จากรูปที่ 3.5 (ข) ในการเริ่มต้นเอพิโสด e2 เอเจนต์จึงตัดสินใจเลือกโปรโตไทป์ A3 เป็นตัวแรกเหมือนเดิม เนื่องจากโอกาสในการสุ่มเลือกโปรโตไทป์อื่นมีน้อยกว่าโอกาสในการเลือก A3 (กำหนดโดยค่าอัตราการสำรวจ)

ในเอพิโสด e2 เอเจนต์ได้ใช้ประโยชน์จากความรู้เดิมในการเลือกโปรโตไทป์ A3 และ B1 ตามลำดับ จนมาอยู่ในสถานะ {A3, B1} ถ้าเอเจนต์ใช้ประโยชน์จากความรู้เดิมที่สถานะนี้ โปรโตไทป์ที่เพิ่มเข้ามาเป็นสมาชิกจะต้องเป็นโปรโตไทป์ B2 (ตามที่แสดงด้วยเส้นประในภาพ) และย้ายสถานะไปที่สถานะ {A3, B1, B2} เช่นเดียวกันกับในเอพิโสด e1 แต่เนื่องจากในสถานะนี้เอเจนต์เลือกโปรโตไทป์แบบสุ่มเพื่อสำรวจตามหลักการเลือกการกระทำแบบเอพซิลอน-กรีดี ทำให้เอเจนต์สุ่มเลือกได้โปรโตไทป์ A4 และสิ้นสุดเอพิโสดที่สถานะ {A3, A4, B1} พร้อมกับได้รางวัล +4

หลังจากจบเอพิโสด e2 และเริ่มเอพิโสด e3 ดังแสดงในรูปที่ 3.5 (ค) จะสังเกตเห็นได้ว่าค่า Q ของโปรโตไทป์ A3 ที่สถานะเริ่มต้นยังมีค่าเพิ่มขึ้น เนื่องจากค่านี้ยังไม่ลู่เข้าสู่ค่าที่ดีที่สุด ซึ่งในที่นี้ เอเจนต์เลือกสำรวจที่สถานะ {A3} จนกระทั่งไปสิ้นสุดเอพิโสดที่สถานะ {A3, A4, B1} พร้อมกับได้รางวัล +4 ซึ่งเป็นผลลัพธ์เดียวกันกับเอพิโสด e2 แต่ใช้ลำดับการเลือกโปรโตไทป์ที่ต่างกัน

หลังจากที่เอเจนต์เรียนรู้ด้วยการลองผิดลองถูกสักระยะหนึ่งจนกระทั่งเริ่มเอพิโสด e4 ดังแสดงในรูปที่ 3.5 (ง) สิ่งที่เกิดขึ้นก็คือ ลำดับการเลือกโปรโตไทป์ที่ดีที่สุดตามประสบการณ์ของเอเจนต์ก็เริ่มเปลี่ยนจากการเริ่มจากการเลือก A3 และ B1 ตามลำดับ ไปเป็นเริ่มจากการเลือก A3 และ A4 ตามลำดับ แต่เซตคำตอบที่ดีที่สุดก็ยังไม่ดีกว่าเอพิโสดก่อน ๆ ซึ่งได้เซตคำตอบที่มีขนาด 3 ตัวเหมือนกัน การทดลองดำเนินต่อไปจนกระทั่งสิ้นสุดเอพิโสด e5 ดังแสดงในรูปที่ 3.5 (จ) ซึ่ง

ได้เซตคำตอบที่ดีที่สุดคือ {A3, B2} ซึ่งมีขนาด 2 ตัว เนื่องจากมีข้อมูลจากชนิดข้อมูล A และ B อย่างละ 1 ตัวเท่านั้น ข้อสังเกตคือ เอเจนต์จะไม่รู้และไม่ต้องสนใจว่าเซตคำตอบที่ดีที่สุดมีขนาดเท่าใด และการลองผิดลองถูกของเอเจนต์อาจทำให้พบเซตคำตอบอื่น ๆ ที่ดีที่สุดนอกเหนือไปจากเซต {A3, B2} ก็ได้

การสิ้นสุดการทดลองสามารถกำหนดอย่างไรก็ได้ กล่าวคือ จะให้เอเจนต์เรียนรู้ต่อไปอย่างไม่สิ้นสุด หรือจะให้จบการทดลองเมื่อสังเกตได้ว่าคำตอบที่ได้ไม่มีการเปลี่ยนแปลงอีก ก็สามารถทำได้ตามที่ผู้ออกแบบระบบต้องการ

3.3 การค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์

วิธีการพื้นฐานอีกวิธีหนึ่งซึ่งใช้ในการค้นหาเซตย่อยสอดคล้องโดยใช้การเรียนรู้แบบเสริมกำลังก็คือ การเริ่มค้นหาโดยเริ่มต้นจากเซตตัวแทนที่มีข้อมูลทุกตัวในชุดข้อมูลเริ่มต้นเป็นสมาชิก ซึ่งหมายถึง การเริ่มค้นหาโดยเริ่มต้นจากเซตเริ่มต้นซึ่งเป็นเซตที่มีความสอดคล้อง และจำเซตตัวแทนนี้เป็นเซตคำตอบ เนื่องจากชุดข้อมูลเริ่มต้นย่อมสามารถนำไปใช้จำแนกข้อมูลทุกตัวในชุดข้อมูลได้ถูกต้อง จากนั้น ให้ลดจำนวนโปรโตไทป์โดยการเลือกสมาชิกที่ไม่ต้องการออกทีละตัว หลังจากการเลือกสมาชิกออกหนึ่งตัว ให้นำเซตตัวแทนที่ได้ไปใช้ในการจำแนกข้อมูลทั้งหมดเพื่อทดสอบคุณสมบัติความสอดคล้อง ถ้าเซตตัวแทนที่ได้สามารถใช้จำแนกข้อมูลทั้งหมดได้อย่างถูกต้อง ให้จำเซตตัวแทนนั้นเป็นเซตคำตอบ แทนที่เซตคำตอบเดิมที่มีขนาดใหญ่กว่า การค้นหาจะสิ้นสุดเมื่อเซตตัวแทนเป็นเซตว่าง หรือขาดสมาชิกที่จำเป็นในการจำแนกข้อมูลบางชนิดในชุดข้อมูลเริ่มต้น ซึ่งเป็นกรณีที่สามารถสรุปได้ว่า เซตย่อยทุก ๆ เซตของเซตตัวแทนเซตนั้นไม่มีความสอดคล้องกับชุดข้อมูลเริ่มต้น ทำให้ได้คำตอบเป็นเซตย่อยสอดคล้องเซตสุดท้ายที่พบจากการค้นหาในหนึ่งเอพิโสด

เนื่องจากคำตอบที่ได้จากการค้นหาด้วยวิธีนี้ ขึ้นอยู่กับลำดับการเลือกสมาชิกออกจากเซตตัวแทน การเลือกด้วยลำดับที่ต่างกันจึงมีผลให้คำตอบที่ได้ต่างกัน การค้นหาเซตย่อยสอดคล้องด้วยวิธีนี้ จึงต้องใช้การทดลองเลือกโปรโตไทป์โดยการสุ่มเลือกสมาชิกออกเพื่อให้ได้เซตย่อยสอดคล้องจำนวนมาก เพื่อหาว่าเซตย่อยที่มีขนาดเล็กที่สุดที่พบมีขนาดเท่าใด และประกอบด้วยโปรโตไทป์ตัวใดบ้าง

เทคนิคการค้นหาข้างต้น สามารถใช้กำหนดทาสก์การเรียนรู้แบบเสริมกำลังได้ในรูปแบบของทาสก์ที่แบ่งได้เป็นเอพิโสด เช่นเดียวกันกับการค้นหาด้วยการเพิ่มจำนวนโปรโตไทป์ ได้ดังนี้

1) กำหนดให้เซตตัวแทนเป็นข้อมูลเกี่ยวกับสถานะปัจจุบัน
2) สถานะเริ่มต้นของแต่ละเอพิโสดในทาสก์นี้คือ สถานะที่มีข้อมูลทุกตัวในชุดข้อมูลเริ่มต้นเป็นสมาชิก และสถานะสิ้นสุดก็คือ สถานะที่สามารถตรวจสอบได้ว่าเซตย่อยทุก ๆ เซตของสถานะนั้นไม่มีความสอดคล้องกับชุดข้อมูลเริ่มต้น

3) การกระทำที่เอเจนต์สามารถทำได้ คือ การเลือกโปรโตไทป์ที่เป็นสมาชิกในสถานะปัจจุบันออก ทำให้เกิดการเปลี่ยนสถานะไปสู่สถานะใหม่ที่มีสมาชิกน้อยลง

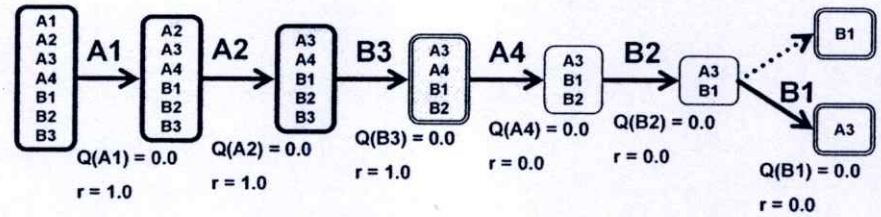
4) รางวัลที่ได้รับจะมีค่าไม่เท่ากับศูนย์ ก็ต่อเมื่อ สถานะใหม่ที่เกิดจากการเลือกสมาชิกออกมีความสอดคล้องกับชุดข้อมูลเริ่มต้น ทั้งนี้ รางวัลจะมีค่าตามจำนวนสมาชิกที่ต้องคัดออกจากเซตคำตอบเซตสุดท้ายที่พบในเอพิโสดเดียวกัน

รางวัลที่ได้รับจากการเลือกการกระทำ a จากสถานะ s และเปลี่ยนไปสู่สถานะถัดไปหรือ s_n สามารถนิยามในรูปของฟังก์ชัน $R(s, a, s_n)$ ได้ดังนี้

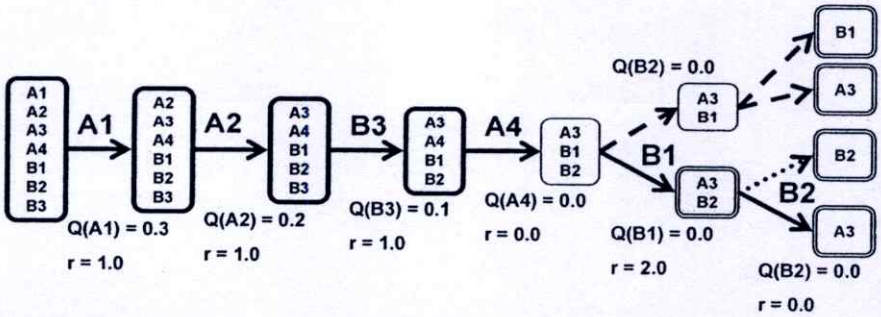
$$R(s,a,s_n) = \begin{cases} 0 & \text{เมื่อ } s_n \text{ ไม่มีความสอดคล้อง} \\ |s_{con}| - |s_n| & \text{เมื่อ } s_n \text{ มีความสอดคล้อง} \end{cases} \quad (3.2)$$

เมื่อ s_{con} หมายถึง สถานะที่มีความสอดคล้องลำดับสุดท้ายที่พบในเอพิโสด และ $|s|$ หมายถึงจำนวนโปรโตไทป์ที่เป็นสมาชิกของสถานะ s ซึ่งหลังจากสภาพแวดล้อมให้รางวัลกับเอเจนต์และเปลี่ยนสถานะแล้ว s_{con} จะเปลี่ยนเป็นค่าเดียวกันกับ s_n ก็ต่อเมื่อ s_n มีความสอดคล้อง ดังนั้น หลังจากจบเอพิโสด s_{con} จะเป็นคำตอบสำหรับเอพิโสดนั้น และผลตอบแทนรวมที่เอเจนต์ได้รับสำหรับเอพิโสดนั้นจะมีค่าเท่ากับ $N - |s_n|$ เมื่อ N เป็นจำนวนโปรโตไทป์ทั้งหมดในชุดข้อมูลเริ่มต้น

ตัวอย่างเช่น ในการค้นหาเซตย่อยสอดคล้องขนาดเล็กที่สุดของชุดข้อมูลตัวอย่างที่ 1 ลำดับการตัดสินใจของเอเจนต์ RL ในการโต้ตอบกับสภาพแวดล้อมการค้นหาด้วยเทคนิคการลดจำนวนโปรโตไทป์ อาจเป็นไปดังแสดงในรูปที่ 3.6



(ก) เอพิโสด e1



(ข) เอพิโสด e2

รูปที่ 3.6 ตัวอย่างลำดับการตัดสินใจในการลดจำนวนโปรโตไทป์ของเอเจนต์ RL

รูปที่ 3.6 แสดงลำดับการตัดสินใจลดจำนวนโปรโตไทป์ของเอเจนต์ RL ภายในการทดลองหนึ่งครั้ง ซึ่งในที่นี้ จะพิจารณาการเรียนรู้วิธีลดจำนวนโปรโตไทป์จากการลองผิดลองถูกของเอเจนต์ใน 2 เอพิโสดด้วยกัน จากรูปที่ 3.6 (ก) ในเอพิโสด e1 ซึ่งเป็นเอพิโสดแรก เช่นเดียวกับการเรียนรู้ในหัวข้อที่ 3.2 เอเจนต์ยังไม่มีข้อมูลสำหรับใช้ในการตัดสินใจ ทำให้ค่าของฟังก์ชัน Q สำหรับทุก ๆ การกระทำในสถานะเริ่มต้น (ซึ่งแทนด้วยรูปสี่เหลี่ยมมนที่ไม่มีการแรเงา) และสถานะอื่น ๆ มีค่าเป็นศูนย์ ดังนั้น การตัดสินใจทุกขั้นตอนจึงมาจากการสุ่มเลือกการกระทำ ในเอพิโสดนี้ เอเจนต์ได้คัดโปรโตไทป์ A1, A2, B3, A4, B2 และ B1 ตามลำดับ ออกจากการเป็นสมาชิกของเซตตัวแทน และสิ้นสุดเอพิโสดที่สถานะ {A3} (ซึ่งแทนด้วยรูปสี่เหลี่ยมที่มีเส้นรอบรูปเป็นเส้นคู่ขนาน และไม่มีการแรเงา) เนื่องจาก {A3} มีเฉพาะสมาชิกชนิด A เท่านั้น จึงไม่สามารถจำแนกข้อมูลชนิด B ได้ถูกต้อง จึงสรุปได้ว่า เซตย่อยทุก ๆ เซตของสถานะนี้ไม่มีความสอดคล้องกับชุดข้อมูลเริ่มต้น (มีเพียงเซตว่างเท่านั้นที่เป็นเซตย่อยของ {A3}) เซตคำตอบของเอพิโสดนี้คือ {A3, A4, B1, B2} (ซึ่งแทนด้วยรูปสี่เหลี่ยมแรเงาที่มีเส้นรอบรูปเป็นเส้นคู่ขนาน) ซึ่งมีสมาชิก 4 ตัว รางวัลที่เอเจนต์ได้รับในแต่ละขั้นตอนมีค่า +1 เมื่อสถานะถัดไปเป็นสถานะที่มีความสอดคล้อง และมีค่าศูนย์สำหรับกรณีอื่น ๆ ทำให้ผลตอบแทนรวมมีค่าเท่ากับ +3 ซึ่งคำนวณได้จาก จำนวนโปรโตไทป์ทั้งหมดของชุดข้อมูลเริ่มต้น ลบด้วยจำนวนโปรโตไทป์ในเซตคำตอบ ($N - |s_{con}|$)

ในแต่ละขั้นตอนการเลือกโปรโตไทป์ เอเจนต์จะปรับค่าของฟังก์ชัน Q ตามวิธีการเรียนรู้แบบ Q(λ) เช่นเดียวกันกับในหัวข้อ 3.2 โดยใช้ค่าพารามิเตอร์เดียวกัน ซึ่งหมายถึง การตัดสินใจของเอเจนต์ยังคงใช้วิธีการเดิม ตามสมการการปรับค่าฟังก์ชัน Q ในการเรียนรู้แบบ Q(λ) ดังนั้น หลังจากจบเอพิโสด e1 การกระทำที่มีค่า Q สูงที่สุดในสถานะเริ่มต้น จึงเป็นการเลือก A1 ออกจากการเป็นสมาชิกของเซตตัวแทน และมีค่าเท่ากับ +0.3

ในเอพิโสด e2 ดังแสดงในรูปที่ 3.6 (ข) เอเจนต์ได้ใช้ประโยชน์จากความรู้เดิมในการคัดโปรโตไทป์ A1, A2 และ B3 ออกจากการเป็นสมาชิกตามลำดับ จนมาถึงการคัด A4 ออกจากการเป็นสมาชิกของสถานะ {A3, A4, B1, B2} ซึ่งเป็นสถานะที่เอเจนต์ไม่มีความรู้สำหรับใช้ในการตัดสินใจ และย้ายไปสถานะ {A3, B1, B2} เช่นเดียวกันกับในเอพิโสด e1 ที่สถานะนี้ เอเจนต์ได้ลองสำรวจด้วยการคัด B1 ออกจากเซตตัวแทน ทำให้ค้นพบสถานะ {A3, B2} ซึ่งเป็นคำตอบที่ดีที่สุดในเอพิโสด ทำให้ได้รางวัล +2 สำหรับการเลือก B1 ออก (เซตสอดคล้องที่พบก่อนหน้า {A3, B2} คือ {A3, A4, B1, B2} ซึ่งมีจำนวนสมาชิกมากกว่า {A3, B2} อยู่ 2 ตัว) สุดท้าย เอพิโสดนี้จึงสิ้นสุดที่สถานะ {A3} และได้ผลตอบแทนรวมมีค่าเท่ากับ +5

บทที่ 4
การเพิ่มประสิทธิผลของอัลกอริธึม RL

จากการวิเคราะห์ผลการทดลองที่ได้มาจากการใช้อัลกอริธึม RL พื้นฐานทั้ง 2 ลักษณะซึ่งได้อธิบายในบทที่แล้วในการแก้ปัญหา MCSS พบว่าผลที่ได้ไม่ดีเท่าที่ควร เนื่องจากการค้นหาของอัลกอริธึมทั้ง 2 แบบติดอยู่ในบางพื้นที่ของปริภูมิการค้นหาซึ่งอยู่ห่างจากคำตอบที่ดีที่สุดค่อนข้างมาก โดยเฉพาะในกรณีที่ชุดข้อมูลเริ่มต้นมีขนาดใหญ่ จึงเป็นที่มาของการดัดแปลงอัลกอริธึม RL เพื่อเพิ่มประสิทธิผลการค้นหาให้ดีขึ้นกว่าเดิม

ในบทนี้จะกล่าวถึง ปัญหาที่พบเมื่อใช้อัลกอริธึม RL พื้นฐานทั้ง 2 ลักษณะในการแก้ปัญหา MCSS และแนวทางการแก้ปัญหาด้วยการเพิ่มประสิทธิผลของอัลกอริธึม RL ในการค้นหาคำตอบของปัญหา MCSS สำหรับชุดข้อมูลเริ่มต้นที่มีขนาดใหญ่ ซึ่งในที่นี้ จะพิจารณาถึงวิธีการเพิ่มประสิทธิผล 2 วิธี คือ การใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น (episode continuation and exploration start) และการย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง (returning to the last known consistent state) ทั้ง 2 วิธีนี้เป็นการดัดแปลงอัลกอริธึม RL พื้นฐานซึ่งสามารถปรับใช้ได้กับเทคนิคการค้นหาเซตย่อยด้วยการลดจำนวนโปรโตไทป์เท่านั้น เนื่องจากในแต่ละเอพิโสดของการค้นหาด้วยการเพิ่มจำนวนโปรโตไทป์จะมีสถานะเดียวที่เป็นสถานะที่มีความสอดคล้อง คือ สถานะสิ้นสุด ขณะที่รางวัลที่เอเจนต์ได้รับในสถานะอื่น ๆ ก็มีค่าเป็นศูนย์เสมอ ทำให้ต้องเริ่มต้นการค้นหาใหม่จากเซตว่างเสมอเมื่อต้องการหาคำตอบอื่น ๆ

สิ่งหนึ่งที่สังเกตได้จากการดัดแปลงอัลกอริธึมทั้ง 2 วิธีนี้คือ จะไม่มีการเปลี่ยนแปลงในส่วนของการเรียนรู้และปรับเปลี่ยนการตัดสินใจของเอเจนต์ เนื่องจากเอเจนต์ยังคงปรับค่าของฟังก์ชันมูลค่าตามสมการการเรียนรู้แบบเสริมกำลังโดยทั่วไป แต่การเปลี่ยนแปลงทั้งหมดจะอยู่ที่วิธีการกำหนดสถานะเริ่มต้น สถานะสิ้นสุด การให้รางวัล และวิธีการกำหนดฟังก์ชันการเปลี่ยนสถานะของทาสก์การค้นหา ซึ่งทำให้สรุปได้ว่า ลักษณะการกำหนดทาสก์การเรียนรู้มีผลอย่างมากในประสิทธิภาพการเรียนรู้ของเอเจนต์ในการเรียนรู้แบบเสริมกำลัง รายละเอียดการดัดแปลงอัลกอริธึมทั้ง 2 วิธีนี้จะเป็นไปดังที่ได้อธิบายไว้ในหัวข้อย่อยของบทนี้

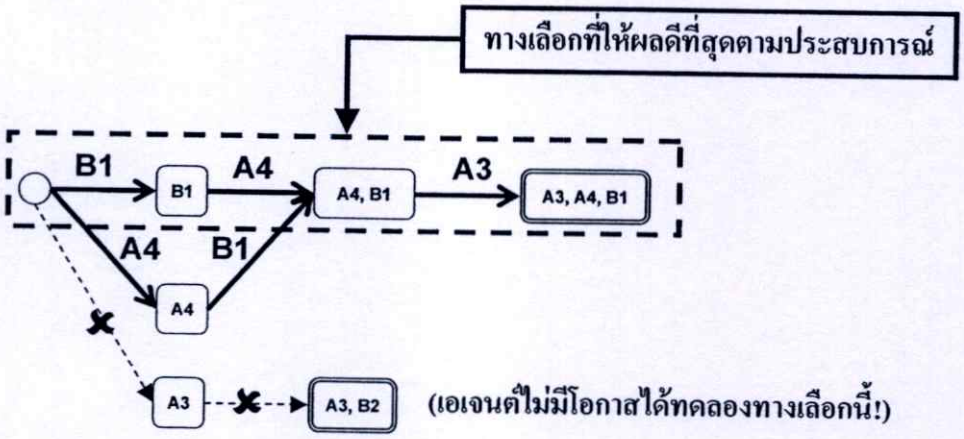
4.1 ข้อจำกัดของอัลกอริธึม RL พื้นฐาน

ปริภูมิการค้นหาของปัญหา MCSS นั้นมีขนาดใหญ่ขึ้นตามจำนวนข้อมูลในชุดข้อมูลเริ่มต้นในระดับเอกซ์โพเนนเชียล กล่าวคือ ถ้าชุดข้อมูลเริ่มต้นมีจำนวนข้อมูล N ตัว ปริภูมิการค้นหาของปัญหาจะมีขนาด 2^N คำตอบ ตัวอย่างเช่น ถ้าชุดข้อมูลเริ่มต้นมีจำนวนข้อมูล 150 ตัว ปริภูมิการค้นหาจะมีขนาด $2^{150} = 1.427 \times 10^{45}$ ในกรณีที่ชุดข้อมูลมีขนาดเล็ก ทั้งอัลกอริธึม RL

แบบที่ใช้การเพิ่มและการลดจำนวนโปรโตไทป์ สามารถลู่เข้าสู่คำตอบที่ดีที่สุดได้โดยการทดลองสำรวจทั่วทั้งปริภูมิการค้นหา แต่สำหรับชุดข้อมูลขนาดใหญ่ เอเจนต์อาจไม่สามารถเรียนรู้วิธีการเลือกที่เหมาะสมได้ในเวลาอันจำกัด และเป็นยังเป็นสาเหตุให้เกิดการลู่เข้าสู่คำตอบที่ไม่ใช่คำตอบที่ดีที่สุด

4.1.1 ข้อจำกัดของการค้นหาเซตย่อยสอดคล้องด้วยการเพิ่มจำนวนโปรโตไทป์

ในการค้นหาด้วยการเพิ่มจำนวนโปรโตไทป์ ถ้าเอเจนต์เลือกโปรโตไทป์ที่ไม่ใช่สมาชิกของคำตอบที่ดีที่สุดตั้งแต่ในสถานะต้น ๆ ซึ่งยังมีสมาชิกอยู่น้อย ก็จะส่งผลให้เซตคำตอบที่ได้มีขนาดใหญ่ และโอกาสในการพบเซตสอดคล้องก็จะมีน้อยลงไปมาก นอกจากนี้เอเจนต์ RL ยังมีแนวโน้มที่จะตัดสินใจแบบเดิมซ้ำอีกเป็นเวลานาน เนื่องจากเอเจนต์ RL ตัดสินใจจากทางเลือกที่ดีที่สุดตามประสบการณ์ที่ได้จากการลองผิดลองถูก ด้วยเหตุนี้ การสุ่มเลือกในเอพิโสดต้น ๆ จึงมีผลอย่างมากต่อการลู่เข้าสู่คำตอบของเอเจนต์ RL



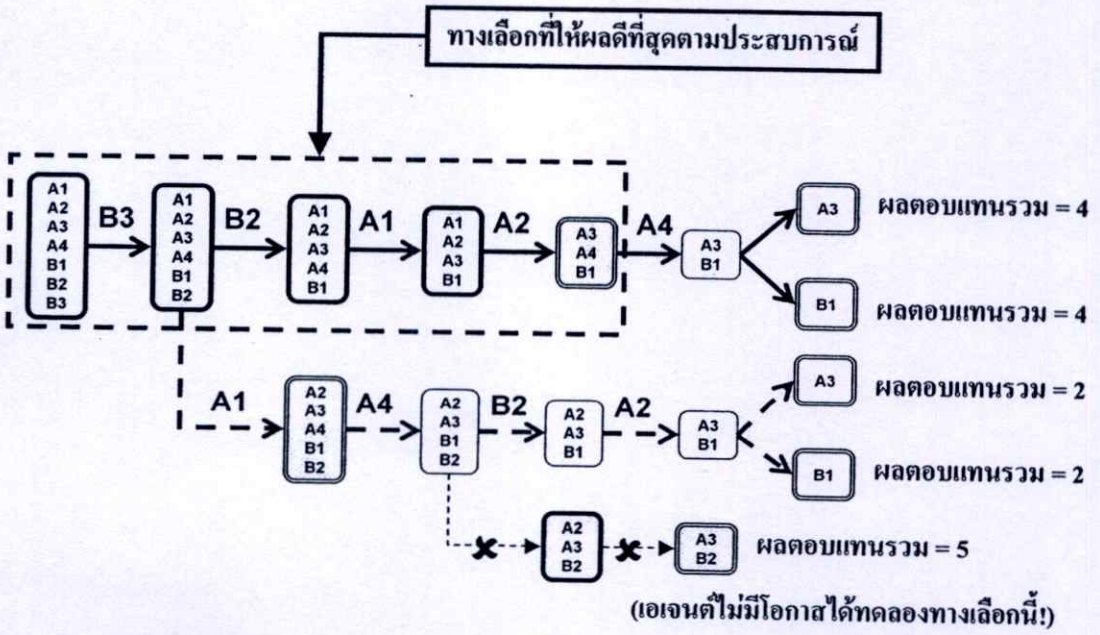
รูปที่ 4.1 แนวโน้มการตัดสินใจในการเพิ่มจำนวนโปรโตไทป์

จากการพิจารณารูปที่ 4.1 ซึ่งเป็นลำดับการเพิ่มจำนวนโปรโตไทป์ในชุดข้อมูลตัวอย่างที่ 1 ซึ่งมีรายละเอียดดังที่ได้อธิบายในบทที่แล้ว สิ่งที่สามารถสังเกตได้คือ ถ้ามีการเลือกโปรโตไทป์ B1 เป็นสมาชิกแล้ว เซตย่อยสอดคล้องที่ได้จะต้องมีจำนวนโปรโตไทป์อย่างน้อย 3 ตัว แต่เนื่องจากการเลือกโปรโตไทป์ B1, A4 และ A3 ตามลำดับ หรือการเลือกโปรโตไทป์ A4, B1 และ A3 ตามลำดับ เป็นทางเลือกที่เอเจนต์ RL เลือกเป็นอันดับแรก และได้ปรับค่าที่สถานะที่เกี่ยวข้อง ทำให้เอเจนต์ RL เลือกโปรโตไทป์ B1 หรือ A4 ที่สถานะเริ่มต้นในแทบทุกครั้งที่เริ่มเอพิโสด และได้เซตย่อยสอดคล้องขนาด 3 ตัวเป็นคำตอบที่ดีที่สุดที่พบ (เซต {A3, A4, B1}) แต่จากการสำรวจทั่วทั้งปริภูมิการค้นหาจะพบว่าเซต {A3, B2} เป็นเซตย่อยสอดคล้องที่มีขนาดเล็กที่สุด ในสถานการณ์นี้ เอเจนต์ RL จะพบคำตอบนี้ก็ต่อเมื่อการใช้โพลิซีเอพซิลอน-กรีดี ทำให้เกิดการสุ่มเลือกโปรโตไทป์ A3 และ B2 (หรือ B2 และ A3 ตามลำดับ) เป็น 2 ตัวแรก ซึ่งความน่าจะเป็นที่จะ

เกิดเหตุการณ์เช่นนี้มีน้อยกว่า การเลือก B1, A4 และ A3 อย่างมาก เนื่องจากเอเจนต์ RL จะตัดสินใจเลือกทางเลือกที่ดีที่สุดตามประสบการณ์บ่อยครั้งกว่าการสำรวจทางเลือกอื่น ๆ ปัญหานี้จะมีผลกับคำตอบมากขึ้นในกรณีที่ชุดข้อมูลมีขนาดใหญ่ขึ้นกว่าตัวอย่างนี้มาก ๆ เช่นในกรณีที่ชุดข้อมูลมีขนาดถึง 100 ตัวขึ้นไป เอเจนต์จะไม่สามารถสำรวจได้ทุก ๆ สถานะในปริภูมิการค้นหา และทำให้ไม่พบคำตอบที่ดีที่สุดที่เป็นไปได้ในโจทย์ปัญหา

4.1.2 ข้อจำกัดของการค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์

ในการค้นหาด้วยการลดจำนวนโปรโตไทป์ ถ้าเอเจนต์เลือกโปรโตไทป์ที่จำเป็นสำหรับคำตอบที่ดีที่สุดออกตั้งแต่ในสถานะต้น ๆ ก็จะส่งผลให้เซตคำตอบที่ได้มีขนาดใหญ่ และทำให้ไม่พบเซตสอดคล้องที่มีขนาดเล็กที่สุด นอกจากนี้เอเจนต์ RL ยังมีแนวโน้มที่จะตัดสินใจแบบเดิมซ้ำอีกเป็นเวลานาน เนื่องจากเอเจนต์ RL ตัดสินใจจากทางเลือกที่ดีที่สุดตามประสบการณ์ที่ได้จากการลองผิดลองถูก ด้วยเหตุนี้ การสุ่มเลือกในเอพิโสดต้น ๆ จึงมีผลอย่างมากต่อการลู่เข้าสู่คำตอบของเอเจนต์ RL เช่นเดียวกันกับการค้นหาด้วยการเพิ่มจำนวนโปรโตไทป์ อย่างไรก็ตาม การค้นหาด้วยการลดจำนวนโปรโตไทป์ก็ยังมีข้อดีอย่างหนึ่งที่ไม่มีในการเพิ่มจำนวนโปรโตไทป์ คือ วิธีการนี้เริ่มต้นจากชุดข้อมูลเริ่มต้นซึ่งเป็นเซตที่มีคุณสมบัติความสอดคล้องตั้งแต่แรก และการคัดโปรโตไทป์ออกทีละตัวก็มักจะทำให้พบเซตย่อยที่ยังคงมีความสอดคล้องอยู่ ซึ่งเซตเหล่านี้ก็อาจจะมีสมาชิกที่ไม่จำเป็นอยู่อีกเป็นจำนวนมาก ทำให้ยังมีทางเลือกในการตัดทอนต่อไปได้อีกจนกระทั่งได้เซตสอดคล้องที่มีขนาดเล็กลง ขณะที่ในแต่ละเอพิโสดของการค้นหาเซตย่อยสอดคล้องด้วยการเพิ่มจำนวนโปรโตไทป์ เอเจนต์ RL จะพบเซตสอดคล้องเพียง 1 เซตเท่านั้น และการเพิ่มโปรโตไทป์จากเซตนั้นก็ไม่สามารถทำให้พบคำตอบที่ดีขึ้นกว่าคำตอบเดิม



รูปที่ 4.2 แนวโน้มการตัดสินใจในการลดจำนวนโปรโตไทป์

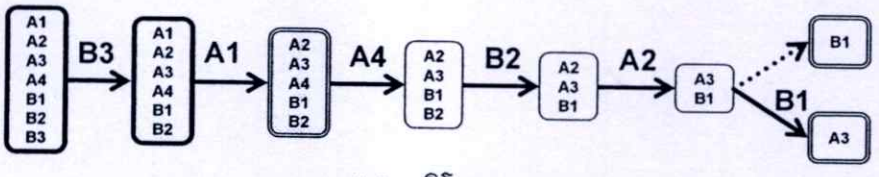
จากการพิจารณารูปที่ 4.2 ซึ่งเป็นลำดับการลดจำนวนโปรโตไทป์ในชุดข้อมูลตัวอย่างที่ 1 เช่นเดียวกันกับในหัวข้อย่อยที่แล้ว สิ่งที่สามารถสังเกตได้คือ ถ้าทางเลือกที่ดีที่สุดตามประสบการณ์ของเอเจนต์ RL มีการคัดโปรโตไทป์ B3 และ B2 ออกไปก่อนเป็นลำดับแรกแล้ว เซตย่อยสอดคล้องที่ได้จะต้องมีจำนวนโปรโตไทป์อย่างน้อย 3 ตัว เช่น เซต {A3, A4, B1} ในสถานการณ์นี้ เอเจนต์ RL จะพบเซต {A3, B2} ได้จากการสุ่มในการสำรวจเท่านั้น ซึ่งก็มีโอกาสน้อยกว่าการใช้ประโยชน์จากความรู้เดิมเป็นอันมาก และแม้ว่าเอเจนต์ RL จะเลือกสุ่มมาในทางที่จะมีโอกาสมาถึงสถานะ {A3, B2} ได้ แต่จากรูปจะเห็นได้ว่า ถ้ามีการตัดสินใจสุ่มไปทางอื่น ผลที่ได้ก็คือ ได้เซต {A2, A3, A4, B1, B2} เป็นเซตสอดคล้องขนาดเล็กที่สุดในเอพิโสดนั้น (และได้ผลตอบแทนรวมเท่ากับ 2) ซึ่งเป็นคำตอบที่แย่กว่าเซต {A3, A4, B1} ทำให้เอเจนต์ไม่เปลี่ยนพฤติกรรมการตัดสินใจ ผลตอบแทนรวมที่ดีที่สุดจึงมีค่าเพียง 4 แทนที่จะเป็น 5 ซึ่งเป็นค่าที่ดีที่สุดที่เป็นไปได้

4.2 อัลกอริธึม RL กับการใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น

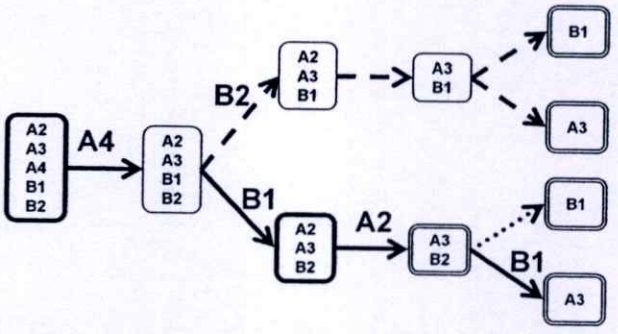
จากปัญหาที่ได้กล่าวถึงในหัวข้อที่แล้ว ทำให้สรุปได้ว่าการปรับค่า ε ในการเลือกการกระทำแบบเอพซิลอน-กรีดีอาจไม่เพียงพอสำหรับการปรับปรุงคุณภาพการเรียนรู้ของเอเจนต์ ค่า ε ที่สูงแม้จะทำให้เอเจนต์สำรวจด้วยการสุ่มมากขึ้น แต่ก็ทำให้การลู่เข้าสู่คำตอบที่ดีเป็นไปได้ช้ามาก และคำตอบที่พบภายในเวลาที่จำกัดก็อาจมีคุณภาพต่ำ เนื่องจากการสำรวจในแต่ละเอพิโสดอาจเกิดขึ้นเร็วเกินไป และการกระทำที่มาจากการสุ่มเลือกโดยส่วนใหญ่ก็อาจไม่ใช่ตัวเลือกที่ดี จึงทำให้ไม่พบคำตอบที่ดีกว่าเดิมจากการสำรวจเลย ในทางกลับกัน ค่า ε ที่ต่ำก็ทำให้ลำดับการตัดสินใจส่วนใหญ่ของเอเจนต์อยู่ในรูปแบบเดิมเป็นเวลานาน ซึ่งทำให้ไม่เกิดการค้นหาคำตอบใหม่ ๆ ที่ดีขึ้นจากการใช้ลำดับการตัดสินใจที่แตกต่างไปจากเดิม

เทคนิคหนึ่งที่ได้พัฒนาขึ้นมาเพื่อแก้ปัญหานี้คือ เทคนิคการดัดแปลงอัลกอริธึม RL โดยใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น (RL+ECES) ซึ่งแบ่งได้เป็น 2 ส่วน การดัดแปลงในส่วนแรกคือ การใช้ความต่อเนื่องของเอพิโสด ซึ่งหมายถึง การเริ่มต้นจากสถานะที่มีผลตอบแทนรวมดีที่สุด เพื่อสำรวจหาทางเลือกที่ทำให้ได้รางวัลมากขึ้นเมื่อเหตุการณ์ดำเนินต่อไปจากสถานะนั้น แทนที่การเริ่มต้นเอพิโสดของอัลกอริธึม RL พื้นฐาน ซึ่งใช้การเริ่มเอพิโสดจากสถานะเดียวกัน (สถานะที่มีสมาชิกทุกตัวของชุดข้อมูลเริ่มต้น) เสมอ การดัดแปลงในส่วนที่สองคือ การสำรวจสถานะเริ่มต้น ซึ่งหมายถึง การค้นหาทั้งจากการเลือกจุดเริ่มต้นเอพิโสดจากการใช้ความต่อเนื่องของเอพิโสด และการเลือกจุดเริ่มต้นเอพิโสดจากสถานะอื่น ๆ ดังนั้น อัลกอริธึม RL+ECES จึงแตกต่างจากอัลกอริธึม RL พื้นฐานในส่วนของการดัดแปลงวิธีการเลือกสถานะเริ่มต้นของเอพิโสดเพียงอย่างเดียวเท่านั้น

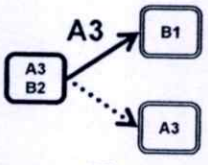
การเริ่มต้นจากสถานะที่มีผลตอบแทนรวมดีที่สุด ทำให้เกิดการค้นหาเฉพาะที่ ในพื้นที่ที่มีโอกาสเจอคำตอบที่ดีขึ้น แทนที่การค้นหาแบบกว้างจากการเริ่มต้นจากสถานะเดียวกัน ซึ่งอาจจะอยู่ห่างจากคำตอบที่ต้องการเป็นระยะทางที่มาก นอกจากนี้ การค้นหาเฉพาะที่ยังทำให้การลู่เข้าหาคำตอบที่ดีเป็นไปได้อย่างรวดเร็ว แม้ว่าจะใช้ค่า ε ที่สูง สถานะที่เหมาะสำหรับใช้เป็นจุดเริ่มต้นสำหรับเอพิโสดถัดไปในการค้นหาด้วยการลดจำนวนโปรโตไทป์ในปัญหา MCSS มากที่สุด คือ สถานะที่เป็นเซตสอดคล้องขนาดเล็กที่สุดที่พบในเอพิโสดล่าสุด การค้นหาโดยใช้ความต่อเนื่องของเอพิโสดมีลักษณะดังแสดงในตัวอย่างในรูปที่ 4.3



(ก) เอพิโสด e1



(ข) เอพิโสด e2



(ค) เอพิโสด e3

รูปที่ 4.3 การใช้ความต่อเนื่องของเอพิโสด

จากรูปที่ 4.3 แสดงให้เห็นการใช้ความต่อเนื่องของเอพิโสด e2 จากเอพิโสด e1 โดยในเอพิโสด e2 มีสถานะเริ่มต้นเป็นสถานะ {A2, A3, A4, B1, B2} ซึ่งเป็นเซตสอดคล้องขนาดเล็กที่สุดที่พบในเอพิโสด e1 และจากการค้นหาในเอพิโสด e2 เอเจนต์จึงพบเซตสอดคล้องที่มีขนาดเล็กลงจากเซตนี้ และได้เซต {A3, B2} เป็นเซตสอดคล้องขนาดเล็กที่สุดที่พบในเอพิโสดนี้ จากรูปที่ 4.3 (ค) แสดงให้เห็นว่า การค้นหาต่อจากสถานะ {A3, B2} นั้นจะไม่ทำให้พบคำตอบใหม่ ๆ อีก เนื่องจากมีทางเลือกที่ไปได้เพียง 2 ทางเท่านั้น และยังเป็นการย้ายไปสู่สถานะสิ้นสุดเอพิโสดอีกด้วย จึงจำเป็นต้องใช้การดัดแปลงในส่วนที่สองในการแก้ปัญหานี้

เนื่องจากในการค้นหาด้วยการลดจำนวนโปรโตไทป์ เอเจนต์ไม่สามารถย้ายจากสถานะที่มีจำนวนโปรโตไทป์น้อยไปยังสถานะที่มีจำนวนโปรโตไทป์มากได้ การเริ่มต้นเอพิโสดใหม่จากเซตสอดคล้องขนาดเล็กที่สุดที่พบในเอพิโสดล่าสุดในทุก ๆ เอพิโสดจึงทำให้การค้นหาติดอยู่ในพื้นที่ที่จำกัด นอกจากการใช้ความต่อเนื่องของเอพิโสดแล้ว จุดเริ่มต้นที่เหมาะสมของเอพิโสดใหม่อาจมาจากการเลือกเซตสอดคล้องขนาดเล็กที่สุดของเอพิโสดที่ผ่านมาในอดีต ด้วยค่าความน่าจะเป็น ε_{start} ค่าพารามิเตอร์ ε_{start} นี้เป็นค่าที่กำหนดการเลือกสถานะเริ่มต้นของเอพิโสดใหม่ว่าจะเลือกมาจากเซตคำตอบของเอพิโสดล่าสุด (การใช้ประโยชน์จากผลสืบเนื่องจากเอพิโสดล่าสุด) หรือจะมาจากเซตคำตอบของเอพิโสดในอดีต ซึ่งเก็บไว้ในบันทึก (pool) ของเซตคำตอบที่ดีที่สุด (การสำรวจโดยการเลือกจุดเริ่มต้น) การสำรวจสถานะเริ่มต้นแบบนี้ทำให้เอเจนต์สามารถค้นหาด้วยการสำรวจในพื้นที่ต่าง ๆ ในปริภูมิการค้นหาได้ทั่วถึงมากขึ้น อัลกอริธึม RL ที่ใช้การดัดแปลงทั้ง 2 ส่วนนี้ ทำให้การปรับสมดุลระหว่างการสำรวจและการใช้ประโยชน์จากความรู้เดิมเกิดขึ้นในอีกระดับนอกเหนือจากการใช้โพลิซีแบบเอพซิลอน-กรีดี

วิธีการเลือกสถานะเริ่มต้นของเอพิโสดใหม่ในอัลกอริธึม RL+ECES สามารถกำหนดได้ดังแสดงในรูปที่ 4.4

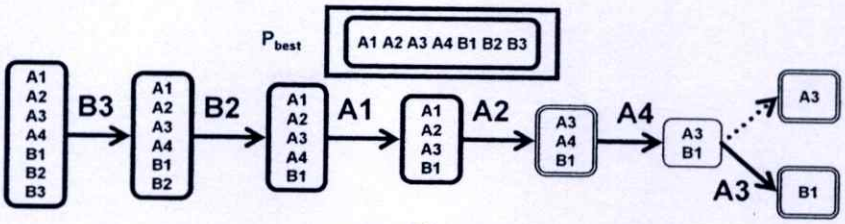
ถ้า เซตสอดคล้องขนาดเล็กที่สุดในเอพิโสดล่าสุดหรือ s_{ep}
ไม่เคยมีอยู่ในบันทึกของเซตคำตอบที่ดีที่สุดหรือ P_{best} แล้ว
    เพิ่ม s_{ep} เข้าไปใน P_{best}

ด้วยความน่าจะเป็น ε_{start} :
    สุ่มเลือกสถานะเริ่มต้น s_0 จากข้อมูลใน P_{best}
ในกรณีอื่น ๆ:
    กำหนดค่าของสถานะเริ่มต้น s_0 ด้วยค่าของ s_{ep}

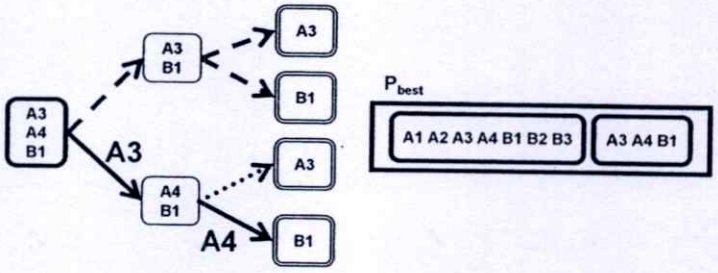
รูปที่ 4.4 วิธีการเลือกสถานะเริ่มต้นของเอพิโสดใหม่ในอัลกอริธึม RL+ECES

ในที่นี้ สถานะที่มีโปรโตไทป์ทุกตัวในชุดข้อมูลเริ่มต้น ก็เป็นสถานะที่บันทึกอยู่ใน P_{best} ด้วยเช่นกัน ทั้งนี้ เพื่อให้เอเจนต์สามารถเริ่มต้นเอพิโสดใหม่ได้จากสถานะเริ่มต้นที่มีโปรโตไทป์ทุกตัวในชุดข้อมูลเริ่มต้น เช่นเดียวกับในกรณีของอัลกอริธึม RL พื้นฐานที่ใช้การค้นหาด้วยการลดจำนวนโปรโตไทป์

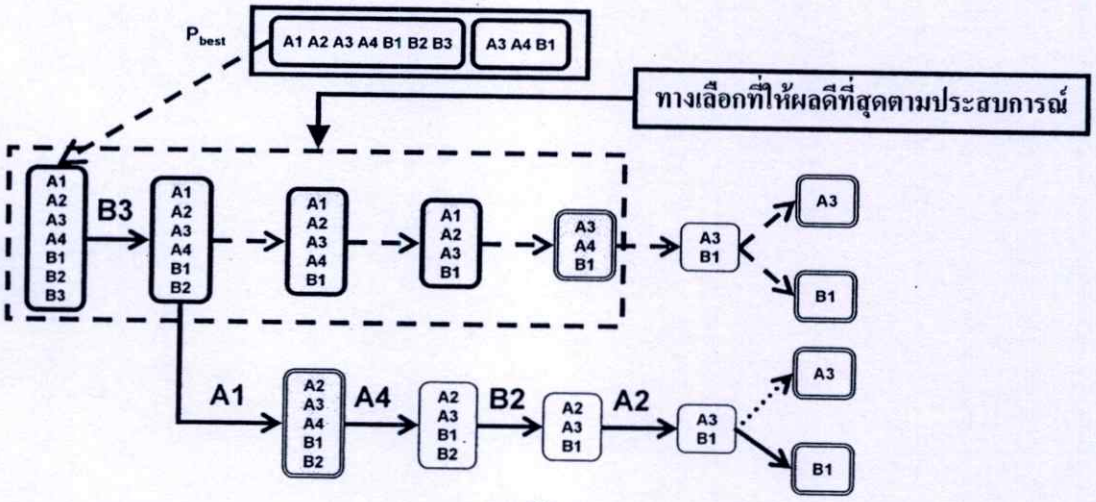
ตัวอย่างเช่น ในการค้นหาเซตย่อยสอดคล้องขนาดเล็กที่สุดของชุดข้อมูลตัวอย่างที่ 1 ลำดับการตัดสินใจในการลดจำนวนโปรโตไทป์ของเอเจนต์ RL ในอัลกอริธึม RL+ECES อาจเป็นไปดังแสดงในรูปที่ 4.5



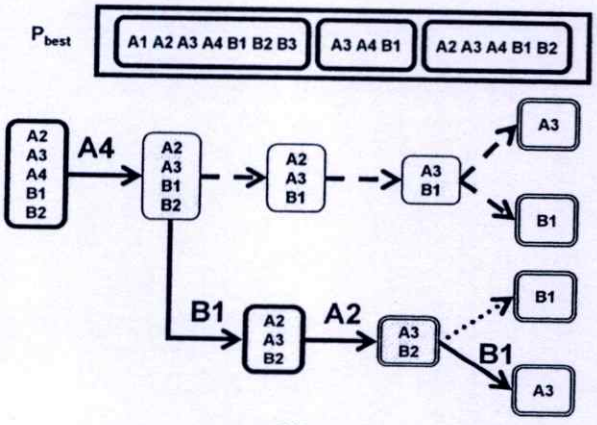
(ก) เอพิโสด e1



(ข) เอพิโสด e2



(ค) เอพิโสด e3



(ง) เอพิโสด e4

รูปที่ 4.5 ตัวอย่างลำดับการตัดสินใจในการลดจำนวนโปรโตไทป์ของเอเจนต์ RL ในอัลกอริธึม RL+ECES

รูปที่ 4.5 แสดงลำดับการตัดสินใจลดจำนวนโปรโตไทป์ของเอเจนต์ RL ในอัลกอริธึม RL+ECES ภายในการทดลองหนึ่งครั้ง ซึ่งในที่นี้ จะพิจารณาการเรียนรู้วิธีลดจำนวนโปรโตไทป์จากการลองผิดลองถูกของเอเจนต์ใน 4 เอพิโสดด้วยกัน ในเอพิโสด e1 ซึ่งเป็นเอพิโสดแรกดังแสดงในรูปที่ 4.5 (ก) ในตอนต้นเอเจนต์ใช้การตัดสินใจจากการสุ่มโดยเริ่มจากสถานะเริ่มต้นที่มีโปรโตไทป์ทุกตัวในชุดข้อมูลเริ่มต้นเป็นสมาชิก (ซึ่งแทนด้วยรูปสี่เหลี่ยมมนที่ไม่มีการแรเงา) และได้คัดโปรโตไทป์ B3, B2, A1, A2, A4 และ A3 ตามลำดับ ออกจากการเป็นสมาชิกของเซตตัวแทน และสิ้นสุดเอพิโสดที่สถานะ {B1} (ซึ่งแทนด้วยรูปสี่เหลี่ยมที่มีเส้นรอบรูปเป็นเส้นคู่ขนาน และไม่มีการแรเงา) เซตคำตอบของเอพิโสดนี้คือ {A3, A4, B1} (ซึ่งแทนด้วยรูปสี่เหลี่ยมแรเงาที่มีเส้นรอบรูปเป็นเส้นคู่ขนาน) ซึ่งมีสมาชิก 3 ตัว สิ่งที่เพิ่มเติมขึ้นมาจากอัลกอริธึม RL ดั้งเดิมในที่นี้คือ บันทึกของเซตคำตอบที่ดีที่สุดหรือ P_{best} เนื่องจากเอพิโสด e1 เป็นเอพิโสดเริ่มต้นของการทดลอง P_{best} จึงมีสมาชิก 1 ตัว คือเซต {A1, A2, A3, A4, B1, B2, B3} ซึ่งเป็นสถานะเริ่มต้นของอัลกอริธึม RL ดั้งเดิม และยังเป็นสถานะเริ่มต้นของเอพิโสดนี้อีกด้วย หลังจากจบเอพิโสด e1 จะมีการเพิ่มเซต {A3, A4, B1} เข้าไปใน P_{best} เนื่องจากเป็นเซตคำตอบที่ไม่เคยมีอยู่ในบันทึกมาก่อน เพื่อใช้เป็นสถานะเริ่มต้นของเอพิโสดในอนาคต

ในเอพิโสด e2 ดังแสดงในรูปที่ 4.5 (ข) มีการใช้ความต่อเนื่องของเอพิโสดและกำหนดให้สถานะเริ่มต้นเป็นเซต {A3, A4, B1} ซึ่งเป็นเซตคำตอบของเอพิโสดที่แล้ว (เอพิโสด e1) ทำให้เอเจนต์ไม่ต้องเสียเวลาในการคัดโปรโตไทป์ B3, B2, A1 และ A2 ออกเพื่อมาถึงสถานะนี้ แต่ก็ไม่พบคำตอบที่ดีกว่าเดิม และถ้าเอเจนต์เริ่มจากสถานะนี้ต่อไป ก็จะไม่สามารถพบคำตอบที่ดีกว่านี้ได้เลย เนื่องจากเอเจนต์ RL ติดอยู่ในขอบเขตของสถานะที่เป็นเซตย่อยของเซต {A3, A4, B1}

ในเอพิโสด e3 ถ้ามีการใช้ความต่อเนื่องของเอพิโสดอีกครั้ง เอเจนต์ RL จะต้องเริ่มต้นเอพิโสดจากสถานะ {A3, A4, B1} เช่นเดิม แต่เนื่องจากมีการสำรวจสถานะเริ่มต้นซึ่งควบคุมด้วยพารามิเตอร์ ε_{start} จึงส่งผลให้ในเอพิโสดนี้เริ่มต้นจากสถานะเซตเริ่มต้น ซึ่งเลือกมาจาก P_{best} ดังที่แสดงในรูปที่ 4.5 (ค) ในเอพิโสดนี้ เอเจนต์มีการสุ่มเลือก A1 ออกในขั้นตอนที่ 2 ทำให้เกิดการสำรวจสถานะนอกเหนือจากสถานะที่ให้ผลดีที่สุดตามประสบการณ์ และได้เซต {A2, A3, A4, B1, B2} เป็นคำตอบซึ่งยังเป็นคำตอบที่มีขนาดใหญ่กว่าเซต {A3, A4, B1} และเพิ่มเซตคำตอบใหม่เข้าไปใน P_{best}

ในที่สุด จากการเริ่มต้นจากสถานะ {A2, A3, A4, B1, B2} ก็ทำให้พบคำตอบที่เล็กลงในเอพิโสด e4 ดังแสดงในรูปที่ 4.5 (ง) และได้เซต {A3, B2} เป็นคำตอบ ซึ่งเป็นคำตอบที่ดีกว่าเซต {A3, A4, B1} จากตัวอย่างนี้ทำให้สามารถสรุปได้ว่า อัลกอริธึม RL+ECES มีการค้นหาในพื้นที่ที่มีแนวโน้มที่จะพบคำตอบใหม่ ๆ บ่อยครั้งกว่าการค้นหาในอัลกอริธึม RL พื้นฐาน

4.3 อัลกอริธึม RL กับการย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง

จากปัญหาที่ได้กล่าวถึงในหัวข้อที่ 4.1 ข้อสังเกตอีกอย่างหนึ่งที่สังเกตได้คือ ถ้ามีการคัดโปรโตไทป์ที่สามารถจำแนกข้อมูลได้ดีออกจากเซตตัวแทนแล้ว โอกาสที่เอเจนต์จะได้รางวัลเพิ่มจากการคัดโปรโตไทป์ออกจะลดลงไปอย่างมาก และอาจทำให้รางวัลที่ได้หลังจากนั้นมีค่าเป็นศูนย์ตลอดไปจนกระทั่งจบเอพิโสด เนื่องจากเอเจนต์ไม่พบเซตย่อยสอดคล้องที่มีขนาดเล็กลงอีกเลยในเอพิโสดนั้น ในกรณีเช่นนี้ ถ้าเอเจนต์สามารถย้อนกลับไปที่สถานะที่เป็นเซตย่อยสอดคล้องเซตสุดท้ายที่พบในเอพิโสด ซึ่งเป็นสถานะก่อนที่เอเจนต์จะได้รางวัลที่มีค่าศูนย์จนจบเอพิโสด และคัดโปรโตไทป์ตัวอื่นออกจากสถานะนั้น ก็เป็นไปได้ที่เอเจนต์จะพบคำตอบอื่นที่ดีขึ้นอีก เนื่องจากโปรโตไทป์บางตัวในสถานะนั้นอาจไม่จำเป็นสำหรับการจำแนกข้อมูลอย่างถูกต้อง อย่างไรก็ตาม ในอัลกอริธึม RL พื้นฐาน เอเจนต์ไม่สามารถย้อนกลับไปในสถานะก่อนหน้าได้ภายในเอพิโสดเดียวกัน อีกทั้งโอกาสที่เอเจนต์จะไปถึงเซตคำตอบของเอพิโสดในอดีตอีกครั้งหนึ่งจากการใช้โพลิซีพื้นฐาน (เช่น โพลิซีแบบเอพซิลอน-กรีดี) เมื่อเซตคำตอบนั้นไม่ได้ให้ผลตอบแทนรวมที่ดีที่สุดก็มีน้อยมาก ทำให้เป็นอุปสรรคของเอเจนต์ในการกระจายการค้นหาไปในพื้นที่อื่น ๆ ที่มีโอกาสเจอคำตอบที่ดีขึ้น

ตัวอย่างเช่น ถ้ากำหนดค่า $\varepsilon = 0.1$ ในโพลิซีแบบเอพซิลอน-กรีดี เอเจนต์ก็จะเลือกการกระทำตามวิธีที่ดีที่สุดตามประสบการณ์ประมาณ 9 ครั้ง ก่อนที่จะเริ่มสำรวจในลำดับขั้นที่ 10 ถ้าการสำรวจจากสถานะนั้นนำไปสู่สถานะที่ไม่สอดคล้อง ก็อาจทำให้เอเจนต์ไม่ได้รับรางวัลอีกเลยจนกระทั่งจบเอพิโสด และไม่ทำให้เกิดการปรับเปลี่ยนรูปแบบการใช้ประโยชน์จากความรู้เดิม ในที่สุด เอเจนต์จึงไปถึงสถานะต่าง ๆ ที่เป็นเซตคำตอบที่อยู่ไกลกว่าขั้นที่ 10 น้อยครั้งมาก แต่ใช้การตัดสินใจส่วนใหญ่เพื่อไปถึงสถานะที่เป็นคำตอบที่ดีที่สุดที่พบ ซึ่งทำให้ไม่พบคำตอบที่ดีขึ้น

แม้ว่าอัลกอริธึม RL+ECES จะให้ผลดีพอสมควรในการทดลองกับชุดข้อมูลที่มีขนาดใหญ่ แต่ก็เป็นผลที่ได้หลังจากที่ใช้หน่วยความจำของระบบเป็นจำนวนมาก และในบางครั้งก็ยังต้องใช้เวลาค่อนข้างสูงมากในการหาคำตอบที่ดีขึ้นจากคำตอบเดิมเมื่ออัลกอริธึมเริ่มมีการลู่เข้าใกล้กับคำตอบที่ดีที่สุด และสำหรับชุดข้อมูลที่มีขนาดใหญ่ขึ้น คำตอบที่ได้จากอัลกอริธึม RL+ECES ก็อาจไม่ดีพอ ด้วยเหตุนี้จึงต้องพัฒนาเทคนิคใหม่เพื่อให้อัลกอริธึม RL มีประสิทธิผลในการแก้ปัญหา MCSS มากยิ่งขึ้น

เทคนิคการดัดแปลงอัลกอริธึม RL ที่พัฒนาขึ้นมาอีกเทคนิคหนึ่งก็คือ การย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง (RL+RCS) แนวคิดหลักของการดัดแปลงนี้คือ การกำหนดให้เอเจนต์ย้อนกลับไปที่สถานะสอดคล้องที่พบเป็นสถานะล่าสุดในเอพิโสด เมื่อพบว่าสถานะปัจจุบันอยู่ไกลจากสถานะสอดคล้องนั้นมาก ๆ หรือเมื่อเหตุการณ์ดำเนินมาจนถึงสถานะสิ้นสุดของการค้นหาด้วยการลดจำนวนโปรโตไทป์ เพื่อให้เอเจนต์ได้มีโอกาสทดลองทางเลือกอื่น ๆ จากสถานะนั้นมากเท่าที่จำเป็น และพบเซตคำตอบที่มีขนาดเล็กลงได้ในที่สุด (ในกรณีที่

สามารถพบได้) เซตคำตอบที่มีขนาดเล็กลงนั้นก็จะกลายเป็นเซตคำตอบของเอพิโสด สิ่งที่จำเป็นอีกอย่างหนึ่งในการใช้เทคนิคนี้คือ การกำหนดจำนวนขั้นที่มากที่สุดสำหรับแต่ละเอพิโสด เพื่อเป็นเงื่อนไขในการจบเอพิโสด ดังนั้น อัลกอริธึม RL+RCS จึงแตกต่างจากอัลกอริธึม RL พื้นฐานในส่วนของการดัดแปลงวิธีการกำหนดเงื่อนไขการสิ้นสุดเอพิโสด การให้รางวัล และการเปลี่ยนสถานะของทาสก์การค้นหา

ฟังก์ชันการเปลี่ยนสถานะ $T(s, a) \rightarrow r, s_n$ ในอัลกอริธึม RL+RCS ซึ่งกำหนดการย้ายสถานะจาก s ไป s_n พร้อมกับได้รับรางวัล r เมื่อเอเจนต์เลือกการกระทำ a สามารถกำหนดได้ดังแสดงในรูปที่ 4.6

```
หลังจากที่เอเจนต์ได้เลือกการกระทำ a และสภาพแวดล้อมได้
คำนวณสถานะถัดไปหรือ s_n ตามปกติ แล้ว
ถ้า s_n เป็นเซตสอดคล้อง:
    คำนวณรางวัลและย้ายสถานะต่อไปตามปกติ
ในกรณีอื่น ๆ:
    ถ้า s_n เป็นเซตที่ไม่มีเซตย่อยที่มีความสอดคล้องเลยหรือ
    ด้วยความน่าจะเป็น x :
        s_n = สถานะสอดคล้องที่พบล่าสุดในเอพิโสด
        รางวัลที่ได้ = -1
    ในกรณีอื่น ๆ:
        ย้ายสถานะต่อไปตามปกติ
```

รูปที่ 4.6 ฟังก์ชันการเปลี่ยนสถานะในอัลกอริธึม RL+RCS

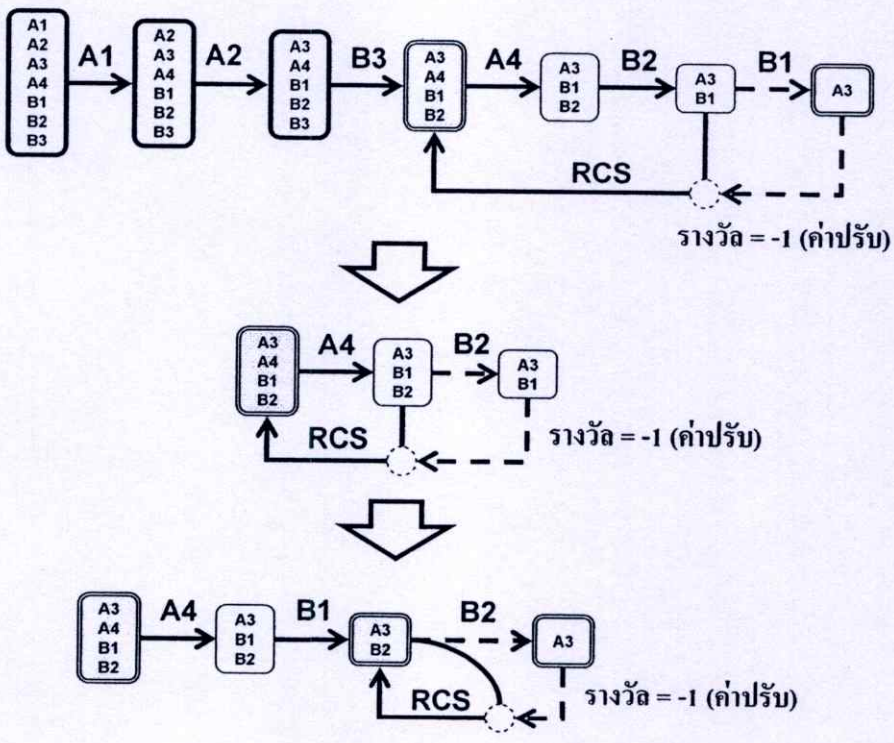
การเปลี่ยนสถานะแบบนี้ ทำให้เอเจนต์ไม่ต้องเดินทางผ่านสถานะที่ไม่สอดคล้องเป็นระยะทางที่ยาวจนเกินไป และสิ้นสุดเอพิโสดโดยไม่มีโอกาสได้ลองสำรวจทางเลือกอื่น ๆ การกำหนดค่าปรับ -1 ในค่ารางวัลสำหรับกรณีที่มีการย้อนกลับไปที่สถานะก่อนหน้ามีวัตถุประสงค์เพื่อเป็นสัญญาณบอกให้เอเจนต์หลีกเลี่ยงเส้นทางนี้และเลือกสำรวจในเส้นทางอื่นแทน

ค่าความน่าจะเป็น x สามารถกำหนดเป็นค่าใดก็ได้ในช่วง [0, 1] การกำหนดค่าศูนย์หมายถึง ไม่มีการย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้องจนกว่าเอเจนต์จะไปถึงสถานะที่ไม่สามารถคัดโปรโตไทป์ออกได้อีก (ซึ่งในกรณีปกติเป็นสถานะสิ้นสุดเอพิโสด) ในทางกลับกัน การกำหนดค่า 1 หมายถึง การกำหนดให้เอเจนต์ย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้องในทุกครั้งที่การคัดโปรโตไทป์ออกทำให้สถานะถัดไปไม่มีความสอดคล้อง ซึ่งในที่นี้ขอเสนอการใช้ค่า 0.2 ซึ่งเป็นค่าที่เหมาะสมเนื่องจากเป็นค่าที่ควบคุมไม่ให้เกิดการเดินทางผ่าน

สถานะที่ไม่สอดคล้องเป็นระยะทางที่ยาวเกินไป แต่ก็ยังอนุญาตให้เอเจนต์เดินทางผ่านสถานะที่ไม่ใช่เซตสอดคล้องจนไปถึงสถานะสอดคล้องในที่สุด (เช่นในกรณีที่แสดงในรูปที่ 3.6) ซึ่งในบางกรณี เอเจนต์อาจจะต้องเดินทางผ่านสถานะที่ไม่สอดคล้องมากกว่า 2 สถานะจึงจะไปถึงสถานะสอดคล้อง

อีกสิ่งหนึ่งที่เปลี่ยนไปจากอัลกอริธึมเดิมคือ เงื่อนไขการจบเอพิโสดของอัลกอริธึม ซึ่งเปลี่ยนจากการสิ้นสุดเอพิโสดเมื่อไปถึงสถานะสิ้นสุด (ในอัลกอริธึมนี้จะไม่มีสถานะสิ้นสุดตามปกติ) ไปเป็นการสิ้นสุดเอพิโสดเมื่อเอเจนต์เดินทางเป็นจำนวนขั้นที่มากเกินกว่าจำนวนขั้นที่มากที่สุดสำหรับแต่ละเอพิโสด ซึ่งโดยทั่วไปสามารถกำหนดให้เป็นค่าใดก็ได้ที่มีค่ามากกว่าขนาดของชุดข้อมูลเริ่มต้น ซึ่งในที่นี้ ขอเสนอการกำหนดจำนวนขั้นที่มากที่สุดให้เป็น 5 เท่าของขนาดของชุดข้อมูลเริ่มต้น เนื่องจากเป็นค่าที่ไม่มากและไม่น้อยจนเกินไป

ตัวอย่างเช่น ในการค้นหาเซตย่อยสอดคล้องขนาดเล็กที่สุดของชุดข้อมูลตัวอย่างที่ 1 ลำดับการตัดสินใจของเอเจนต์ RL ในการโต้ตอบกับสภาพแวดล้อมการค้นหาด้วยเทคนิคการลดจำนวนโปรโตไทป์ เมื่อมีการย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง อาจเป็นไปดังแสดงในรูปที่ 4.7



รูปที่ 4.7 ตัวอย่างลำดับการตัดสินใจในการลดจำนวนโปรโตไทป์ของเอเจนต์ RL ในอัลกอริธึม RL+RCS

รูปที่ 4.7 แสดงลำดับการตัดสินใจลดจำนวนโปรโตไทป์ของเอเจนต์ RL ในอัลกอริธึม RL+RCS ภายในหนึ่งเอพิโสด โดยเริ่มจากสถานะเริ่มต้นที่มีโปรโตไทป์ทุกตัวในชุดข้อมูลเริ่มต้น

เป็นสมาชิก (ซึ่งแทนด้วยรูปสี่เหลี่ยมมนที่ไม่มีการแรเงา) เช่นเดียวกันกับอัลกอริธึม RL แบบพื้นฐานซึ่งใช้การลดจำนวนโปรโตไทป์ ในตอนต้นเอเจนต์ใช้การตัดสินใจจากการสุ่มและได้คัดโปรโตไทป์ A1, A2, B3, A4, B2 และ B1 ตามลำดับ ออกจากการเป็นสมาชิกของเซตตัวแทน จนกระทั่งไปถึงจุดที่ปกติจะต้องย้ายจากสถานะ {A3, B1} ไปสู่สถานะ {A3} (ซึ่งตามปกติจะต้องเป็นสถานะสิ้นสุดของเอพิโสด) ณ จุดนี้ เซตย่อยสอดคล้องขนาดเล็กที่สุดที่พบในเอพิโสดคือเซต {A3, A4, B1, B2} ในอัลกอริธึมนี้เอเจนต์จึงย้ายจากสถานะ {A3, B1} ไปสู่สถานะ {A3, A4, B1, B2} แทนที่การไปยังสถานะ {A3} เพื่อให้โอกาสในการสำรวจทางเลือกอื่นจากสถานะ {A3, A4, B1, B2} ใหม่อีกครั้ง แทนที่การสิ้นสุดเอพิโสดตามปกติ

สำหรับทุก ๆ การเปลี่ยนสถานะไปยังสถานะที่ไม่มีความสอดคล้อง พารามิเตอร์ x จะกำหนดโอกาสที่จะเกิดการย้อนกลับไปที่สถานะสอดคล้องขนาดเล็กที่สุดที่พบในเอพิโสดนี้ ดังนั้น การย้ายสถานะไปยังสถานะ {A3, B1, B2} และ {A3, B1} ในตัวอย่างนี้ จึงมีโอกาสเปลี่ยนไปเป็นการย้อนกลับไปสู่สถานะ {A3, A4, B1, B2} แทนการย้ายสถานะตามปกติด้วยความน่าจะเป็น x

เมื่อเอเจนต์คัดโปรโตไทป์ออกจากสถานะ {A3, A4, B1, B2} ซ้ำอีกครั้ง เอเจนต์อาจตัดสินใจเลือกทางเดิมซ้ำอีกก็ได้ (เนื่องจากค่าปรับ -1 อาจจะยังไม่ส่งผลถึงค่าของฟังก์ชันมูลค่าในสถานะก่อนหน้าที่อยู่ไกลจากสถานะนั้น) ในที่นี้เอเจนต์ได้คัดโปรโตไทป์ A4 ออก ทำให้เกิดการย้ายสถานะไปยังสถานะ {A3, B1, B2} อีกครั้งหนึ่ง จากนั้น เอเจนต์ได้คัดโปรโตไทป์ B2 ออกจากเซต ซึ่งปกติจะทำให้เกิดการย้ายสถานะจากสถานะ {A3, B1, B2} ไปยังสถานะ {A3, B1} ซึ่งไม่ใช่สถานะสอดคล้อง แต่ความน่าจะเป็น x ได้กำหนดให้เอเจนต์ย้อนกลับไปที่สถานะ {A3, A4, B1, B2} อีกครั้งหนึ่ง (ดังแสดงในแผนภาพลำดับที่สองของรูปที่ 4.7) เอเจนต์จึงมีโอกาสทดลองคัดโปรโตไทป์ออกจากสถานะ {A3, A4, B1, B2} ได้อีกหลายครั้งจนกระทั่งพบเซตสอดคล้อง {A3, B2} ในที่สุด รางวัลที่มีค่า -1 ซึ่งเป็นค่าปรับได้ส่งผลให้เอเจนต์หลีกเลี่ยงการเลือกเส้นทางเดิม เอเจนต์จึงมีโอกาสในการสำรวจเส้นทางที่ไม่เคยเลือกบ่อยครั้งขึ้นกว่าในอัลกอริธึม RL พื้นฐาน และหลีกเลี่ยงการเดินทางมาถึงสถานะ {A3, B1} ซ้ำอีกในอนาคต

เมื่อเอเจนต์พบเซต {A3, B2} แล้ว ในเอพิโสดนี้จะไม่มีการย้อนกลับไปยังสถานะ {A3, A4, B1, B2} อีก และทำให้เอเจนต์คัดโปรโตไทป์ A3 หรือ B2 ออกจากเซต {A3, B2} ไปเรื่อย ๆ และพบเซต {B2} หรือ {A3} ซึ่งไม่ใช่เซตสอดคล้อง จึงย้ายสถานะกลับมาที่สถานะ {A3, B2} อีก ด้วยเหตุนี้ จึงต้องมีการกำหนดจำนวนขั้นที่มากที่สุดสำหรับแต่ละเอพิโสด เพื่อเป็นเงื่อนไขในการจบเอพิโสด ในจำนวนที่ไม่สูงเกินไป และไม่ต่ำจนเกินไป ในที่นี้ ถ้ากำหนดจำนวนขั้นที่มากที่สุดเป็น 5 เท่าของขนาดของชุดข้อมูลเริ่มต้นซึ่งมี 7 ตัว แต่ละเอพิโสดก็จะมีจำนวนขั้นมากที่สุดไม่เกิน 35 ขั้น

ในที่สุด เมื่อจบเอพิโสดนี้ เซตคำตอบที่ได้จึงเป็นเซต {A3, B2} ซึ่งเป็นเซตคำตอบที่ดีที่สุดสำหรับชุดข้อมูลตัวอย่างนี้ จากตัวอย่างนี้แสดงให้เห็นว่า ทาสก์การเรียนรู้แบบนี้มีส่วนสนับสนุนให้เอเจนต์สามารถค้นหาคำตอบที่ดีได้ตั้งแต่ในเอพิโสดต้น ๆ

บทที่ 5

การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึง การทดลองเพื่อเปรียบเทียบผลลัพธ์ของแก้ปัญหา MCSS โดยใช้การเรียนรู้แบบเสริมกำลังแบบต่าง ๆ ซึ่งได้อธิบายในบทที่ 3 และบทที่ 4

5.1 ชุดข้อมูลที่ใช้ในการทดลอง

ชุดข้อมูลที่ใช้ในวิทยานิพนธ์ฉบับนี้นำมาจากฐานข้อมูลมาตรฐานของ "UCI Machine Learning Repository" [9] ซึ่งเป็นชุดข้อมูลที่เก็บมาจากข้อมูลจริง ชุดข้อมูลทั้งหมดที่ใช้ในการทดลองมีรายละเอียดดังนี้

ตารางที่ 5.1 รายละเอียดของชุดข้อมูลที่ใช้ในการทดลอง

ชุดข้อมูล	จำนวนข้อมูล	คุณลักษณะเฉพาะ	ชนิด
IRIS	150	4	3
GLASS	214	9	7
ECOLI	336	7	8

5.1.1 ชุดข้อมูล IRIS

ชุดข้อมูล IRIS เป็นชุดข้อมูลของดอกไอริส ใช้ในการจำแนกชนิดของดอกไอริสโดยพิจารณาจากความกว้าง และความยาวของกลีบเลี้ยง และกลีบดอก มีจำนวนข้อมูลทั้งหมด 150 ตัว มีค่าคุณลักษณะเฉพาะ 4 ค่า และแบ่งเป็น 3 ชนิด คือ Setosa, Versicolor และ Virginica

5.1.2 ชุดข้อมูล GLASS

ชุดข้อมูล GLASS เป็นชุดข้อมูลของแก้ว ใช้ในการจำแนกชนิดของแก้ว โดยพิจารณาจากสารที่ใช้เป็นส่วนประกอบของวัสดุ มีที่มาจากการพิสูจน์หลักฐานทางด้านนิติวิทยาศาสตร์ ชุดข้อมูลนี้มีจำนวนข้อมูลทั้งหมด 214 ตัว มีค่าคุณลักษณะเฉพาะ 9 ค่า และแบ่งเป็น 7 ชนิด

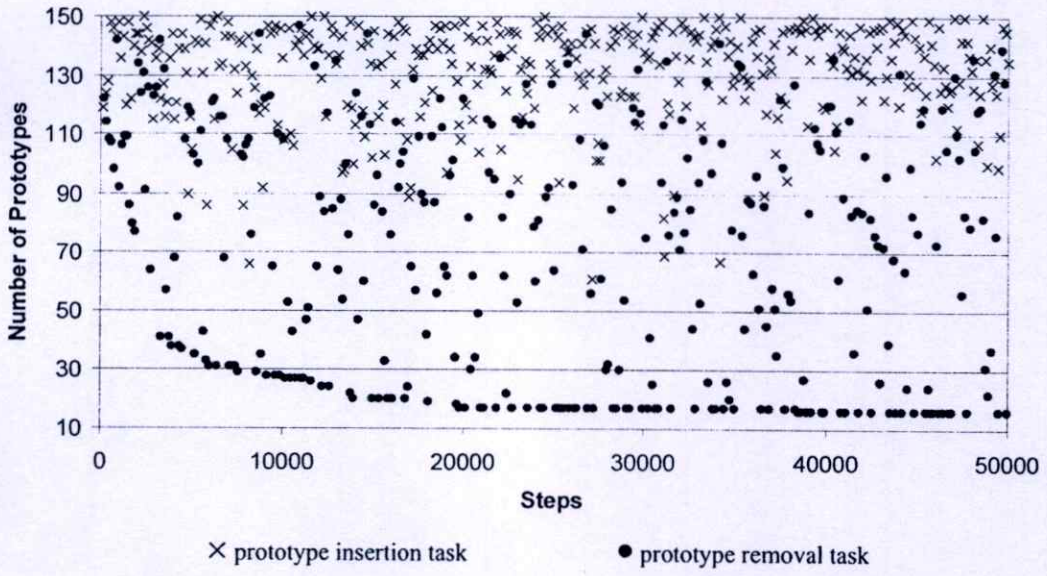
5.1.3 ชุดข้อมูล ECOLI

ชุดข้อมูล ECOLI เป็นชุดข้อมูลของแบคทีเรียอีโคไล ใช้ในการจำแนกชนิดของแบคทีเรียอีโคไล โดยพิจารณาจากรายละเอียดต่าง ๆ มีจำนวนข้อมูลทั้งหมด 336 ตัว มีค่าคุณลักษณะเฉพาะ 7 ค่า และแบ่งเป็น 8 ชนิด

5.2 การเปรียบเทียบผลลัพธ์ระหว่างทาสก์การเพิ่มและการลดจำนวนโปรโตไทป์

ในหัวข้อนี้ จะพิจารณาผลลัพธ์ที่ได้จากอัลกอริธึม RL ที่ใช้ทาสก์การค้นหาเซตย่อยสอดคล้องด้วยการเพิ่มจำนวนโปรโตไทป์ กับผลลัพธ์ที่ได้จากอัลกอริธึม RL ที่ใช้ทาสก์การค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์

สำหรับชุดข้อมูล IRIS จำนวนโปรโตไทป์ของเซตสอดคล้องที่มีขนาดเล็กที่สุดที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL ทั้ง 2 วิธี เป็นไปดังแสดงในกราฟในรูปที่ 5.1

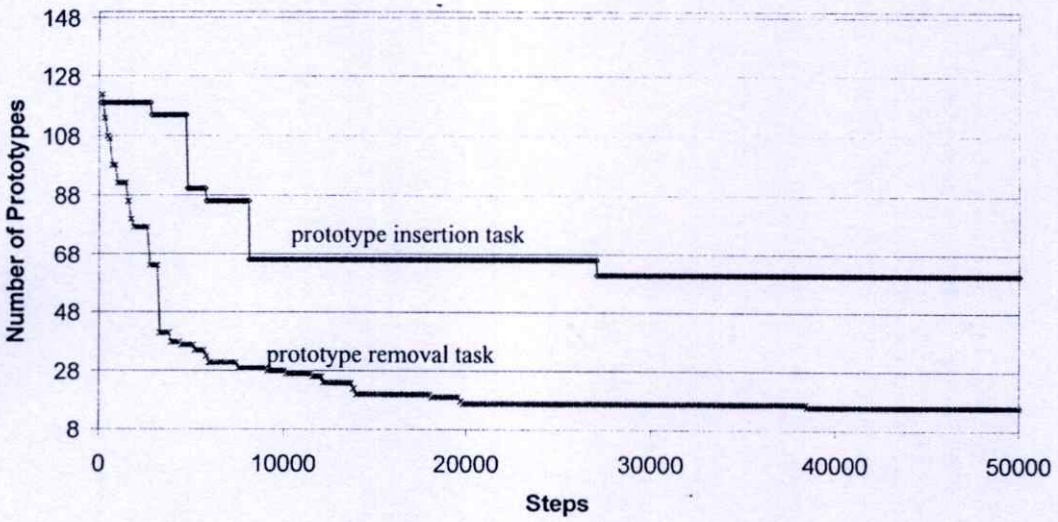


รูปที่ 5.1 กราฟเปรียบเทียบจำนวนโปรโตไทป์ในเซตสอดคล้องขนาดเล็กที่สุดของชุดข้อมูล IRIS ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL

กราฟในรูปที่ 5.1 แสดงจำนวนโปรโตไทป์ของเซตสอดคล้องขนาดเล็กที่สุดที่พบโดยอัลกอริธึม RL ในแต่ละเอพิโสด สำหรับกรณีที่ใช้ทาสก์การค้นหาด้วยการเพิ่มจำนวนโปรโตไทป์ (prototype insertion task ซึ่งในกราฟแทนด้วยจุดกากบาท) เปรียบเทียบกับกรณีที่ใช้ทาสก์การค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ (prototype removal task ซึ่งในกราฟแทนด้วยจุดกลมทึบ) โดยแต่ละจุดในกราฟแสดงถึงความสัมพันธ์ระหว่างจำนวนโปรโตไทป์ของเซตสอดคล้องขนาดเล็กที่สุดที่พบในแต่ละเอพิโสดเมื่อสิ้นสุดเอพิโสด กับจำนวนขั้นสะสมที่ใช้นับตั้งแต่เริ่มต้นอัลกอริธึม

ข้อมูลจากกราฟในรูปที่ 5.1 นั้นสะท้อนให้เห็นถึงลักษณะที่ต่างกันในการลู่เข้าสู่คำตอบของอัลกอริธึมทั้งสองแบบ อัลกอริธึมที่ใช้ทาสก์การคัดโปรโตไทป์ออกสามารถใช้ประโยชน์จากความรู้เดิมได้มากกว่าอัลกอริธึมที่ใช้ทาสก์การเพิ่มโปรโตไทป์ ทำให้เซตคำตอบที่ได้ส่วนใหญ่อยู่ในกลุ่มที่มีขนาดเล็กกว่ามาก และจำนวนโปรโตไทป์ของเซตคำตอบก็มีแนวโน้มลดลงมากกว่า

อย่างเห็นได้ชัด จำนวนโปรโตไทป์ในเซตสอดคล้องที่มีขนาดเล็กที่สุดที่พบโดยอัลกอริธึม RL ทั้ง 2 วิธี นับตั้งแต่เริ่มต้นการทดลอง เป็นไปดังแสดงในกราฟในรูปที่ 5.2



รูปที่ 5.2 กราฟเปรียบเทียบจำนวนโปรโตไทป์ในเซตสอดคล้องขนาดเล็กที่สุดของชุดข้อมูล IRIS ที่พบโดยอัลกอริธึม RL นับตั้งแต่เริ่มต้นการทดลอง

กราฟในรูปที่ 5.2 แสดงจำนวนโปรโตไทป์ของเซตสอดคล้องขนาดเล็กที่สุดที่พบโดยอัลกอริธึม RL นับตั้งแต่เริ่มต้นการทดลอง สำหรับกรณีที่ใช้ทาสก์การค้นหาด้วยการเพิ่มจำนวนโปรโตไทป์ เทียบกับกรณีที่ใช้ทาสก์การค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ โดยเส้นกราฟแสดงความสัมพันธ์ระหว่าง จำนวนโปรโตไทป์ในเซตสอดคล้องขนาดเล็กที่สุดที่พบ กับจำนวนขั้นสะสมที่ใช้นับตั้งแต่เริ่มต้นอัลกอริธึม

จากข้อมูลดังแสดงในรูปที่ 5.2 ในเอพิโสดแรก การคัดโปรโตไทป์ออกทำให้พบเซตคำตอบขนาด 122 ตัว ขณะที่เซตคำตอบที่พบจากการเลือกโปรโตไทป์เพิ่มจนได้เซตสอดคล้องนั้นมีขนาด 119 ตัว แต่ในเอพิโสดถัดมา คำตอบที่ได้จากการคัดโปรโตไทป์ออกมีขนาด 114 ตัว (ลดจำนวนลง 8 ตัว) ขณะที่การเลือกโปรโตไทป์เพิ่มไม่พบคำตอบที่ดีกว่า 119 ตัว หลังจากนั้นการคัดโปรโตไทป์ออกก็พบคำตอบที่ดีขึ้นอย่างต่อเนื่อง ขณะที่การเลือกโปรโตไทป์เพิ่มสามารถหาคำตอบที่ดีขึ้นได้เพียงเล็กน้อยเท่านั้น หลังจากอัลกอริธึมดำเนินไปประมาณ 50000 ขั้น การคัดโปรโตไทป์ออกทำให้พบเซตคำตอบขนาด 16 ตัว และใช้การทดลองทั้งหมด 380 เอพิโสด ขณะที่การเลือกโปรโตไทป์เพิ่มจนได้เซตสอดคล้องนั้นพบเซตคำตอบขนาด 61 ตัว และใช้การทดลองทั้งหมด 345 เอพิโสด

จำนวนขั้นที่ใช้ในทาสก์การเพิ่มโปรโตไทป์ จะเท่ากับจำนวนโปรโตไทป์ที่เป็นสมาชิกของเซตคำตอบ ขณะที่จำนวนขั้นที่ใช้ในทาสก์การคัดโปรโตไทป์ออก จะมีค่าประมาณใกล้เคียง

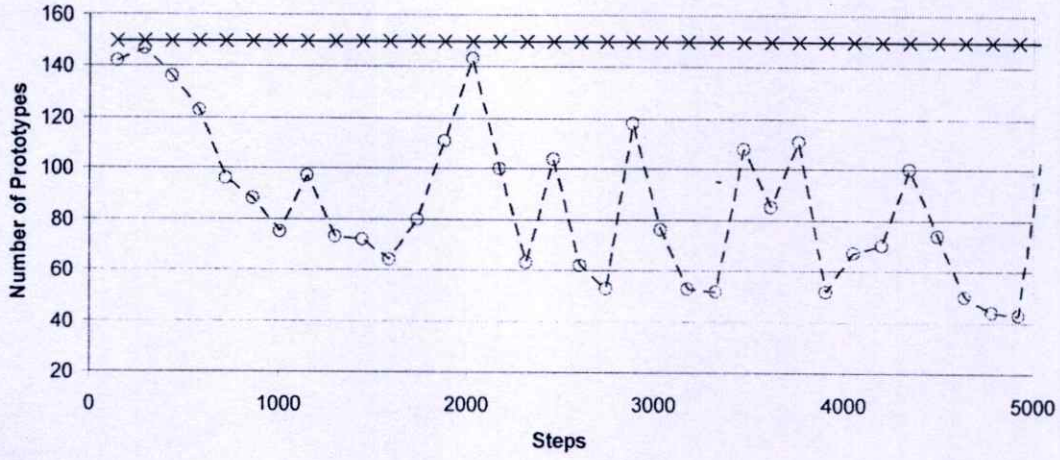
กับจำนวนข้อมูลทั้งหมดในชุดข้อมูลเริ่มต้น (150 ตัวสำหรับชุดข้อมูล IRIS) ซึ่งหมายความว่า การคัดโปรโตไทป์ออกต้องใช้จำนวนขั้นมากในการหาคำตอบที่ดี เมื่อเทียบกับการเพิ่มโปรโตไทป์จนได้เซตสอดคล้อง แต่เนื่องจากการคัดโปรโตไทป์ออกให้ผลลัพธ์ที่ดีกว่า ทำให้จำนวนขั้นที่ใช้ในแต่ละเอพิโสดโดยเฉลี่ยน้อยกว่าการค้นหาเซตคำตอบด้วยการเพิ่มโปรโตไทป์เล็กน้อย

ข้อมูลจากกราฟในรูปที่ 5.1 และ 5.2 ทำให้สามารถสรุปได้ว่า ทาสก์การค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ มีประสิทธิภาพในการหาผลลัพธ์มากกว่าทาสก์การค้นหาด้วยการเพิ่มจำนวนโปรโตไทป์ ในการเปรียบเทียบผลลัพธ์ที่ได้ในชุดข้อมูลที่มีขนาดไม่ใหญ่นัก โดยสังเกตได้จากความแตกต่างของผลลัพธ์ที่ได้ตั้งแต่ในเอพิโสดต้น ๆ เนื่องจากผลลัพธ์ที่ได้จากอัลกอริธึมทั้งสองแบบให้ผลที่ต่างกันอย่างชัดเจนสำหรับในชุดข้อมูล IRIS ซึ่งมีขนาดเล็กที่สุดในชุดข้อมูลทั้งหมดที่ใช้ในการทดลอง ในที่นี้จึงไม่แสดงการเปรียบเทียบผลลัพธ์ที่ได้ในชุดข้อมูลอื่น ๆ อย่างไรก็ตาม เซตตัวแทนที่ได้จากการค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ก็ยังไม่มีขนาดค่อนข้างใหญ่เมื่อเทียบกับผลที่ได้จากอัลกอริธึมอื่น ๆ เช่น อัลกอริธึม MCS ของคาซาราธี ซึ่งให้ผลลัพธ์เป็นเซตตัวแทนขนาด 15 ตัวในชุดข้อมูล IRIS [5] จึงเป็นที่มาของการพัฒนาวิธีการดัดแปลงอัลกอริธึม RL เพื่อเพิ่มประสิทธิผล 2 วิธี คือ การใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น (RL+ECES) และการย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง (RL+RCS) ซึ่งในหัวข้อต่อจากนี้จะอภิปรายถึงผลการทดลองที่ได้จากอัลกอริธึม RL ที่ใช้การดัดแปลงทั้ง 2 วิธี

5.3 ผลของเทคนิคการใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น

ในหัวข้อนี้ จะพิจารณาผลลัพธ์ที่ได้จากอัลกอริธึม RL ที่ใช้ทาสก์การค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ตามปกติ เทียบกับอัลกอริธึม RL ที่มีการใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น

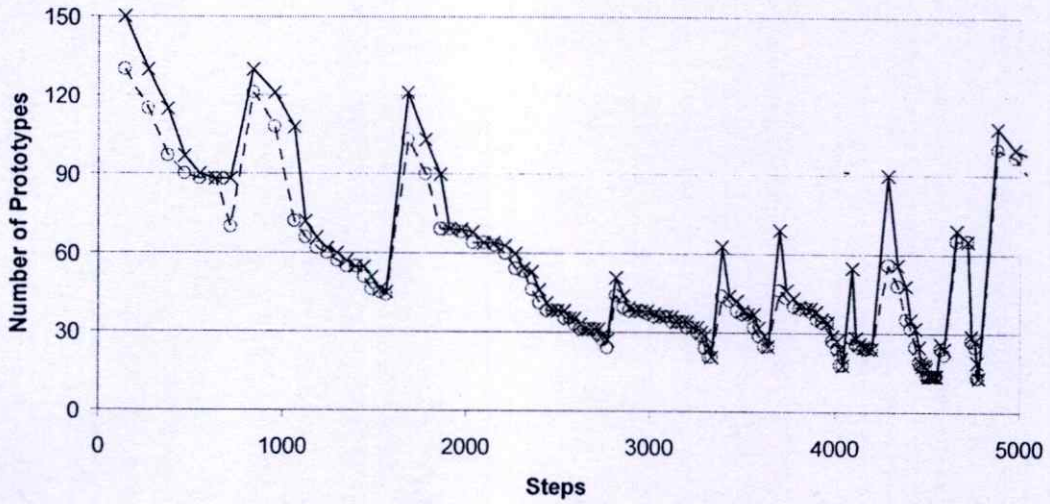
สำหรับชุดข้อมูล IRIS จำนวนโปรโตไทป์ของเซตสอดคล้องที่มีขนาดเล็กที่สุดที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL ที่ใช้การลดจำนวนโปรโตไทป์ตามปกติ เมื่อเปรียบเทียบกับจำนวนโปรโตไทป์ในสถานะเริ่มต้นในแต่ละเอพิโสด เป็นไปดังแสดงในกราฟในรูปที่ 5.3



รูปที่ 5.3 กราฟแสดงจำนวนโปรโตไทป์ในเซตสอดคล้องขนาดเล็กที่สุดของชุดข้อมูล IRIS ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL เทียบกับจำนวนโปรโตไทป์ในสถานะเริ่มต้น

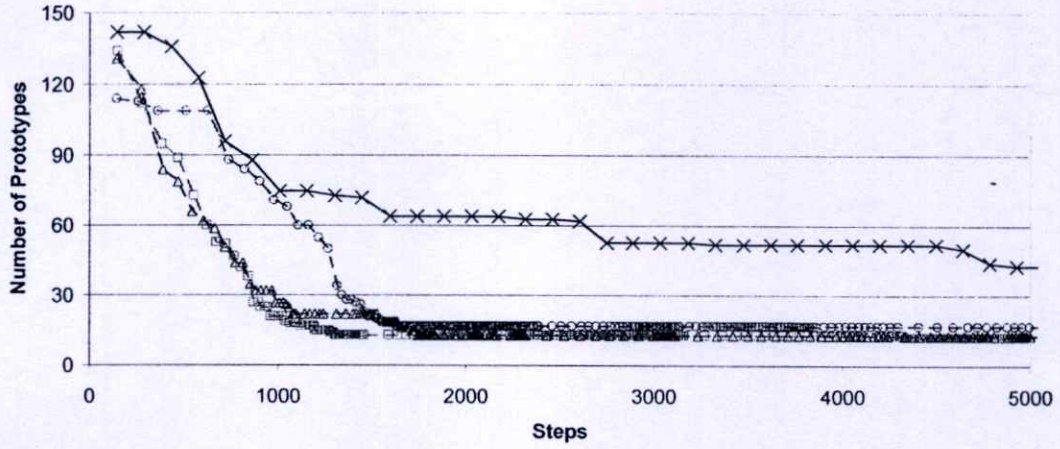
ข้อมูลจากกราฟในรูปที่ 5.3 สะท้อนให้เห็นถึงลักษณะการลู่เข้าสู่คำตอบของอัลกอริธึม RL ที่ใช้ทาสก์การคัดโปรโตไทป์ออกตามปกติ ในอัลกอริธึมนี้ เอเจนต์จะเริ่มเอพิโสดจากจุดเริ่มต้นเดียวกันเสมอ ซึ่งสถานะเริ่มต้นสำหรับชุดข้อมูล IRIS คือสถานะที่มีโปรโตไทป์ 150 ตัว จุดวงกลมทึบแต่ละจุดในกราฟเป็นตัวแทนข้อมูลที่บันทึกได้ และแสดงถึงความสัมพันธ์ระหว่างจำนวนโปรโตไทป์ของเซตคำตอบขนาดเล็กที่สุดที่พบในแต่ละเอพิโสดเมื่อสิ้นสุดเอพิโสด กับจำนวนขั้นสะสมที่ใช้นับตั้งแต่เริ่มต้นอัลกอริธึม จำนวนโปรโตไทป์ที่ได้มีแนวโน้มลดลงเมื่อเวลาผ่านไป แต่ก็มีเอพิโสดที่ได้คำตอบที่ไม่ดีสลับกันไป

จำนวนโปรโตไทป์ของเซตสอดคล้องที่มีขนาดเล็กที่สุดสำหรับชุดข้อมูล IRIS ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL ที่ใช้การลดจำนวนโปรโตไทป์และเทคนิคการใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น เมื่อเปรียบเทียบกับจำนวนโปรโตไทป์ที่เป็นสมาชิกในสถานะเริ่มต้นในแต่ละเอพิโสด เป็นไปดังแสดงในกราฟในรูปที่ 5.4



รูปที่ 5.4 กราฟแสดงจำนวนโปรโตไทป์ในเซตสอดคล้องขนาดเล็กที่สุดของชุดข้อมูล IRIS ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL ที่ใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้นเทียบกับจำนวนโปรโตไทป์ในสถานะเริ่มต้น

ข้อมูลจากกราฟในรูปที่ 5.4 สะท้อนให้เห็นถึงลักษณะการลู่เข้าสู่คำตอบของอัลกอริธึม RL ที่ใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น ซึ่งมีการกำหนดสถานะเริ่มต้นของแต่ละเอพิโสดต่างไปจากการคัดโปรโตไทป์ออกตามปกติ โดยในที่นี้ แต่ละเอพิโสดจะเริ่มต้นจากเซตคำตอบที่ดีที่สุดในเอพิโสดที่แล้ว และเพื่อไม่ให้การค้นหาของอัลกอริธึมติดอยู่ในพื้นที่ที่จำกัด สถานะเริ่มต้นสำหรับบางเอพิโสดจึงมาจากเซตคำตอบของเอพิโสดในอดีต ซึ่งเก็บไว้ในบันทึกของเซตคำตอบที่ดีที่สุด จุดกากบาทในกราฟแสดงจำนวนโปรโตไทป์ในสถานะเริ่มต้นในแต่ละเอพิโสด จุดวงกลมในกราฟแสดงจำนวนโปรโตไทป์ในเซตคำตอบในแต่ละเอพิโสดซึ่งสัมพันธ์กันกับจุดกากบาทที่จำนวนขั้นเดียวกัน ในกรณีที่มีความต่อเนื่องระหว่างเอพิโสด ข้อมูลที่จุดวงกลมในเอพิโสดที่แล้วจะกลายเป็นข้อมูลที่จุดกากบาทในเอพิโสดถัดไป ในกรณีที่มีการสำรวจสถานะเริ่มต้น จำนวนโปรโตไทป์ในสถานะเริ่มต้นในเอพิโสดถัดไปจะเปลี่ยนไปจากเซตคำตอบในเอพิโสดก่อนหน้าอย่างมาก ซึ่งสามารถสังเกตได้จากความไม่ต่อเนื่องในเส้นกราฟจำนวนโปรโตไทป์ในสถานะเริ่มต้นและในเซตคำตอบในแต่ละเอพิโสดมีแนวโน้มลดลง และมีการลู่เข้าหากันในช่วงที่มีความต่อเนื่องระหว่างเอพิโสด จำนวนขั้นที่ใช้ในแต่ละเอพิโสดของอัลกอริธึมนี้จะไม่แน่นอนและขึ้นอยู่กับจำนวนโปรโตไทป์ในสถานะเริ่มต้น ซึ่งโดยทั่วไปจำนวนขั้นที่ใช้ในเอพิโสดส่วนใหญ่จะน้อยกว่าจำนวนขั้นที่ใช้ในอัลกอริธึม RL ที่ใช้ทาสก์การคัดโปรโตไทป์ออกตามปกติ จำนวนโปรโตไทป์ในเซตสอดคล้องที่มีขนาดเล็กที่สุดที่พบโดยอัลกอริธึม RL ทั้ง 2 วิธี นับตั้งแต่เริ่มต้นการทดลอง เป็นไปดังแสดงในกราฟในรูปที่ 5.5



รูปที่ 5.5 กราฟเปรียบเทียบจำนวนโปรโตไทป์ในเซตสอดคล้องขนาดเล็กที่สุดของชุดข้อมูล IRIS ที่พบโดยอัลกอริธึม RL ที่ใช้การคัดโปรโตไทป์ออก ($\varepsilon = 0.01$) กับอัลกอริธึม RL ที่มีการดัดแปลง ($\varepsilon_{start} = 0.1$ และ $\varepsilon = 0.01, 0.05$ และ 0.1) นับตั้งแต่เริ่มต้นการทดลอง

ข้อมูลจากกราฟในรูปที่ 5.5 แสดงถึงอัตราการลู่เข้าสู่คำตอบของอัลกอริธึม RL ทั้ง 2 วิธี ซึ่งอัลกอริธึม RL ที่มีการดัดแปลงด้วยการใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น โดยใช้พารามิเตอร์ $\varepsilon_{start} = 0.1$ และ $\varepsilon = 0.01, 0.05$ และ 0.1 สามารถลู่เข้าสู่คำตอบที่ดีได้เร็วกว่าอัลกอริธึม RL ที่ไม่มีการดัดแปลงอย่างมาก กราฟเส้นที่กำกับด้วยจุดกากบาทแสดงผลลัพธ์ที่ได้จากการใช้อัลกอริธึม RL ที่ใช้การคัดโปรโตไทป์ออกและกำหนดค่า $\varepsilon = 0.01$

ค่า ε ในอัลกอริธึม RL ที่มีการดัดแปลงมีผลต่อการลู่เข้าสู่คำตอบเล็กน้อย โดยการกำหนดค่า ε ต่ำ ๆ จะทำให้การลู่เข้าสู่คำตอบช้าลง เนื่องจากอัลกอริธึมใช้เวลาในการสำรวจมากขึ้น ตัวอย่างเช่น การลู่เข้าสู่คำตอบเมื่อใช้ $\varepsilon = 0.1$ ในรูปที่ 5.5 จะช้ากว่าในกรณีที่ใช้ $\varepsilon = 0.01$ และ 0.05 ซึ่งให้ผลการลู่เข้าสู่คำตอบใกล้เคียงกัน แต่ในที่สุดทั้ง 3 กรณีก็มีการลู่เข้าสู่คำตอบที่ใกล้เคียงกัน

จำนวนโปรโตไทป์ในเซตสอดคล้องที่มีขนาดเล็กที่สุดที่ได้จากอัลกอริธึม RL ทั้ง 2 วิธี ซึ่งเป็นผลจากการทดลองกับชุดข้อมูล IRIS เมื่อนำไปเปรียบเทียบกับผลการเลือกโปรโตไทป์ที่ได้จากอัลกอริธึม MCS ของดาซาราธี เป็นไปดังแสดงในตารางที่ 5.2 เนื่องจากอัลกอริธึม RL เป็นอัลกอริธึมที่มีการใช้การสุ่มเป็นพื้นฐาน ทำให้ผลที่ได้จากการทดลองแต่ละครั้งอาจไม่เหมือนกัน ผลลัพธ์ที่นำมาเปรียบเทียบจึงมาจากการทดลอง 10 ครั้ง

ตารางที่ 5.2 การเปรียบเทียบผลลัพธ์ที่ได้จากจากการทดลอง 10 ครั้งของแต่ละอัลกอริธึม สำหรับชุดข้อมูล IRIS

จำนวนโปรโตไทป์	MCS	RL	RL+ECES		
		$\varepsilon = 0.01$	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$
10	-	-	2	-	4
11	-	1	5	7	3
12	-	-	2	3	2
13	-	2	1	-	1
14	-	2	-	-	-
15	10	2	-	-	-
16	-	2	-	-	-
17	-	1	-	-	-

เซตคำตอบที่ดีที่สุดที่พบได้ในชุดข้อมูล IRIS นั้นมีขนาด 10 ตัว [4], [14] จากตารางที่ 5.2 วิธี MCS ของดาซาราธีจะให้ผลลัพธ์ขนาด 15 เสมอ เนื่องจากเป็นอัลกอริธึมที่ไม่มีการสุ่ม ทำให้ผลลัพธ์เท่ากันเสมอในชุดข้อมูลเดียวกัน ในที่นี้ อัลกอริธึม RL ที่ใช้การคัดโปรโตไทป์ออกให้ผลลัพธ์ที่ดีกว่า MCS ในหลาย ๆ ครั้ง แต่ก็ไม่พบเซตคำตอบขนาด 10 ตัว ซึ่งเป็นคำตอบที่ดีที่สุดเลย จากอัลกอริธึม RL ที่มีการดัดแปลงด้วยการใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น (RL+ECES) ให้ผลลัพธ์ที่ดีกว่า MCS และอัลกอริธึม RL พื้นฐานในทุกครั้ง และพบเซตคำตอบที่ดีที่สุดในบางครั้งสำหรับในกรณีที่กำหนดค่า $\varepsilon = 0.01$ และ 0.1

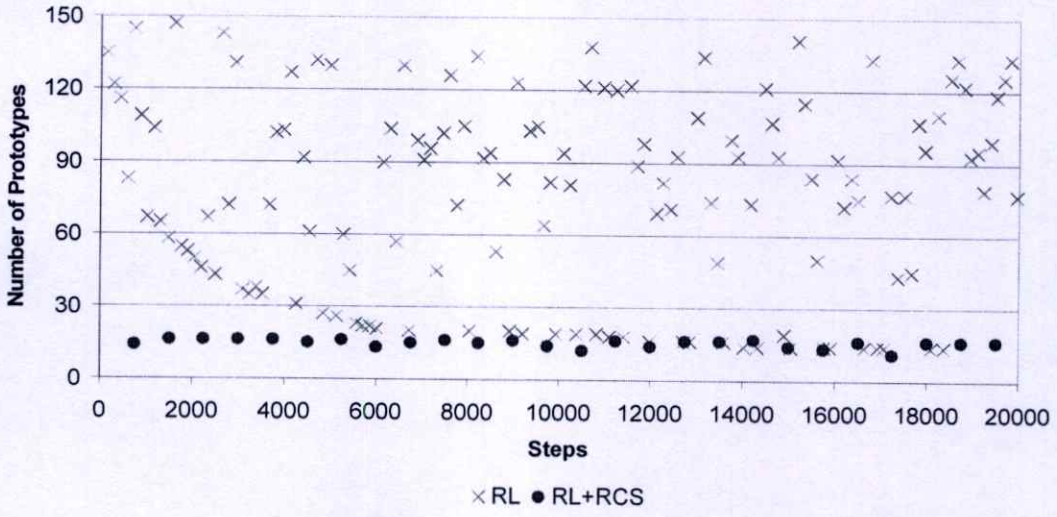
อัลกอริธึม RL+ECES ในกรณีที่กำหนดค่า $\varepsilon = 0.1$ พบเซตคำตอบที่ดีที่สุดบ่อยที่สุดเมื่อเทียบกับกรณีอื่น ๆ ซึ่งเป็นผลจากการใช้เวลาในการสำรวจที่มากขึ้น ซึ่งแตกต่างจากอัลกอริธึม RL พื้นฐาน ในกรณีของอัลกอริธึม RL ที่ใช้การคัดโปรโตไทป์ออก ถ้ากำหนดให้ค่า ε มีค่าสูงขึ้นจะทำให้คำตอบที่พบมีคุณภาพต่ำลงอย่างเห็นได้ชัด ในที่นี้จึงไม่แสดงผลจากอัลกอริธึม RL พื้นฐานในกรณีที่กำหนดค่า ε เป็นค่าอื่น ๆ

แม้ว่าอัลกอริธึม RL+ECES จะให้ผลที่ค่อนข้างดีสำหรับชุดข้อมูล IRIS แต่ผลที่ได้ก็ใช้เวลาและหน่วยความจำในการค้นหาที่สูงมาก ทำให้การทดลองกับชุดข้อมูลขนาดใหญ่ขึ้นเป็นไปได้ยาก ในที่นี้จึงไม่แสดงการเปรียบเทียบผลที่ได้สำหรับชุดข้อมูลอื่น ๆ ที่มีขนาดใหญ่กว่าชุดข้อมูล IRIS และเป็นที่มาของการดัดแปลงอีกวิธีหนึ่งคือ การย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง ซึ่งในหัวข้อถัดไปจะอภิปรายถึงผลการทดลองที่ได้จากอัลกอริธึม RL ที่ใช้การดัดแปลงนี้

5.4 ผลของเทคนิคการย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง

ในหัวข้อนี้ จะพิจารณาผลลัพธ์ที่ได้จากอัลกอริธึม RL ที่ใช้ทาสก์การค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ตามปกติ เทียบกับอัลกอริธึม RL ที่มีการย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง

สำหรับชุดข้อมูล IRIS จำนวนโปรโตไทป์ของเซตสอดคล้องที่มีขนาดเล็กที่สุดที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL ที่ใช้การลดจำนวนโปรโตไทป์ตามปกติ เมื่อเทียบกับผลลัพธ์ที่ได้จากอัลกอริธึม RL ที่มีการย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง (RL+RCS) ในแต่ละเอพิโสด เป็นไปดังแสดงในกราฟในรูปที่ 5.6

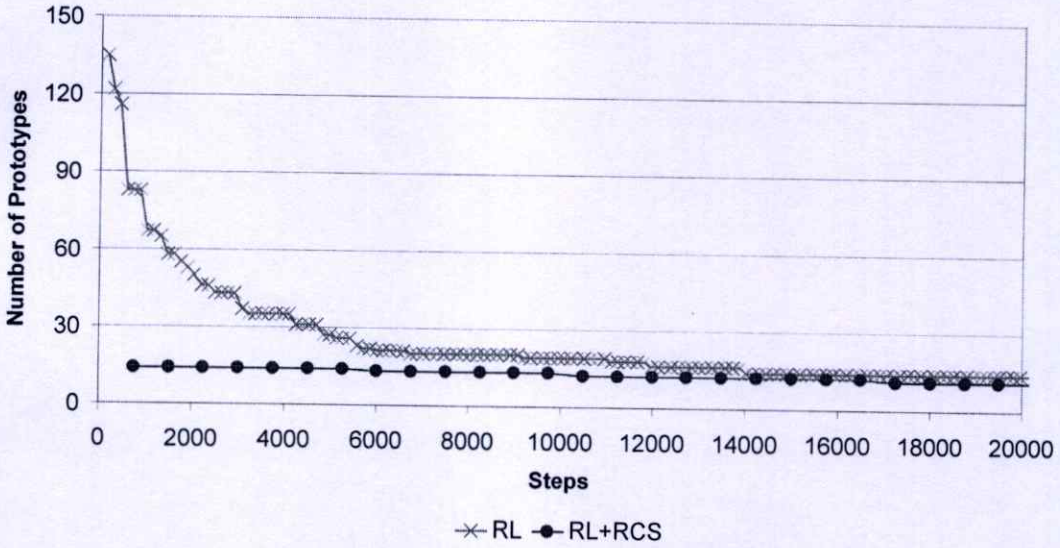


รูปที่ 5.6 กราฟเปรียบเทียบจำนวนโปรโตไทป์ในเซตสอดคล้องขนาดเล็กที่สุดของชุดข้อมูล IRIS ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL ที่ใช้การคัดโปรโตไทป์ออก กับอัลกอริธึม RL ที่มีการดัดแปลง

ข้อมูลจากกราฟในรูปที่ 5.6 สะท้อนให้เห็นถึงลักษณะการลู่เข้าสู่คำตอบของอัลกอริธึม RL ทั้ง 2 วิธี จุดกากบาทในกราฟแสดงผลลัพธ์ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL และจุดวงกลมทึบในกราฟแสดงผลลัพธ์ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL+RCS โดยแต่ละจุดในกราฟแสดงถึงความสัมพันธ์ระหว่าง จำนวนโปรโตไทป์ของเซตคำตอบขนาดเล็กที่สุดที่พบในแต่ละเอพิโสดเมื่อสิ้นสุดเอพิโสด กับจำนวนขั้นสะสมที่ใช้นับตั้งแต่เริ่มต้นอัลกอริธึม

จากข้อมูลดังแสดงในรูปที่ 5.6 ในแต่ละเอพิโสด อัลกอริธึม RL จะใช้จำนวนขั้นประมาณ 150 ขั้น (เท่ากับจำนวนข้อมูลในชุดข้อมูล IRIS) ขณะที่อัลกอริธึม RL+RCS จะใช้จำนวนขั้นประมาณ 750 ขั้น (5 เท่าของจำนวนข้อมูลในชุดข้อมูล IRIS) ขนาดของเซตคำตอบที่พบโดยอัลกอริธึม RL+RCS ในแต่ละเอพิโสดจะอยู่ในช่วง 10 ถึง 18 ตัว ขณะที่อัลกอริธึม RL มีการค้นหาที่ติดอยู่ในสถานะที่ไม่สอดคล้องเป็นจำนวนมากในบางเอพิโสด ทำให้ขนาดของเซต

คำตอบที่พบในแต่ละเอพิโสดอยู่ในช่วง 14 ถึง 147 ตัว นอกจากนี้ อัลกอริธึม RL+RCS ยังสามารถพบเซตคำตอบขนาด 14 ตัวได้ตั้งแต่ในเอพิโสดแรก ซึ่งเป็นผลจากการเรียนรู้จากบางสถานะที่มีทดลองการคัดโปรโตไทป์ออกด้วยวิธีการหลาย ๆ แบบในเอพิโสดเดียวกัน และในที่สุด อัลกอริธึม RL+RCS พบคำตอบที่ดีที่สุดสำหรับชุดข้อมูล IRIS คือ 10 ตัว ในขณะที่คำตอบที่ดีที่สุดที่อัลกอริธึม RL พบมีขนาด 14 ตัว จำนวนโปรโตไทป์ในเซตสอดคล้องที่มีขนาดเล็กที่สุดที่พบโดยอัลกอริธึม RL ทั้ง 2 วิธี นับตั้งแต่เริ่มต้นการทดลอง เป็นไปดังแสดงในกราฟในรูปที่ 5.7



รูปที่ 5.7 กราฟเปรียบเทียบจำนวนโปรโตไทป์ในเซตสอดคล้องขนาดเล็กที่สุดของชุดข้อมูล IRIS ที่พบโดยอัลกอริธึม RL ที่ใช้การคัดโปรโตไทป์ออก กับอัลกอริธึม RL ที่มีการดัดแปลง นับตั้งแต่เริ่มต้นการทดลอง

ข้อมูลจากกราฟในรูปที่ 5.7 แสดงถึงอัตราการลู่เข้าสู่คำตอบของอัลกอริธึม RL ทั้ง 2 วิธี จุดกากบาทในกราฟแสดงผลลัพธ์ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL และจุดวงกลมทึบในกราฟแสดงผลลัพธ์ที่พบในแต่ละเอพิโสดโดยอัลกอริธึม RL+RCS เช่นเดียวกันกับกราฟในรูปที่ 5.6 โดยแต่ละจุดในกราฟแสดงถึงความสัมพันธ์ระหว่าง จำนวนโปรโตไทป์ของเซตคำตอบขนาดเล็กที่สุดที่พบ กับจำนวนขั้นสะสมที่ใช้นับตั้งแต่เริ่มต้นอัลกอริธึม นับตั้งแต่เริ่มต้นการทดลอง

ข้อมูลจากกราฟในรูปที่ 5.7 แสดงให้เห็นอย่างชัดเจนว่า อัลกอริธึม RL+RCS สามารถลู่เข้าสู่คำตอบที่ดีได้เร็วกว่าอัลกอริธึม RL อย่างมาก นอกจากขนาดของผลลัพธ์ที่พบจากอัลกอริธึม RL+RCS จะมีขนาดเล็กกว่าอัลกอริธึม RL แล้ว ลักษณะของคำตอบที่พบในแต่ละเอพิโสดยังมีความหลากหลายมากกว่าอีกด้วย คำตอบที่พบจากอัลกอริธึม RL ค่อย ๆ มีขนาดเล็กลงในลักษณะที่เป็นเซตย่อยของคำตอบที่พบก่อนหน้า ขณะที่คำตอบที่พบจากอัลกอริธึม RL+RCS จะมีลักษณะที่กระจายไปที่คำตอบที่หลากหลายมากกว่า แต่มีขนาดของคำตอบที่ใกล้เคียงกัน

ในการเปรียบเทียบผลการทดลองที่ได้จากอัลกอริธึม RL+RCS กับอัลกอริธึม MCS ของดาซาราธี นอกจากการใช้โพลิซีแบบเอพซิลอน-กรีดีแล้ว ในที่นี้ จะใช้โพลิซีแบบซอฟท์แมกซ์ที่กำหนดขึ้นมาพิเศษสำหรับปัญหานี้ เพื่อให้มีโอกาสในการเลือกตัวเลือกที่มีค่าของฟังก์ชันมูลค่าเป็นอันดับรองลงมาจากค่าที่ดีที่สุดเพิ่มมากขึ้น โพลิซีแบบซอฟท์แมกซ์ที่ใช้ในการทดลองในหัวข้อนี้สามารถกำหนดได้ดังแสดงในรูปที่ 5.8

ด้วยความน่าจะเป็น ε :

เลือกการกระทำ a แบบสุ่ม

ในกรณีอื่น ๆ:

เลือกการกระทำ a ตามค่าของฟังก์ชันมูลค่า

ด้วยความน่าจะเป็น

$$\frac{Q(s,a)^{1.4}}{\sum_b Q(s,b)^{1.4}}$$

สำหรับทุก ๆ การกระทำ b ที่มีค่า $Q(s,b) > 0$

รูปที่ 5.8 โพลิซีแบบซอฟท์แมกซ์ที่ใช้ในการทดสอบอัลกอริธึม RL

โพลิซีนี้แตกต่างจากการใช้การแจกแจงแบบกิบส์ เนื่องจากมีการใช้พารามิเตอร์ ε ในการควบคุมการเลือกการกระทำเช่นเดียวกับโพลิซีแบบเอพซิลอน-กรีดี แต่ก็มีการเลือกการกระทำที่มีค่าของฟังก์ชันมูลค่าหรือ Q เป็นอันดับรองลงมาจากค่าที่ดีที่สุดตามสัดส่วนเช่นเดียวกับในการใช้โพลิซีแบบซอฟท์แมกซ์

การเปรียบเทียบผลลัพธ์ที่ได้จากอัลกอริธึม RL ในหัวข้อนี้กับผลที่ได้จากอัลกอริธึม MCS ของดาซาราธีสำหรับชุดข้อมูลต่าง ๆ จะเป็นลักษณะเช่นเดียวกันกับในหัวข้อที่ 5.3 คือใช้ผลที่มีจากการทดลอง 10 ครั้งสำหรับแต่ละอัลกอริธึม จากการทดลองในทาสก์การเรียนรู้แบบต่าง ๆ พบว่าค่า $\varepsilon = 0.01$ เป็นอัตราการสำรวจที่ให้ผลค่อนข้างดีในทุก ๆ กรณี ขณะที่ค่า ε ที่สูงกว่านี้มักทำให้เกิดการสำรวจที่มากเกินไป และค่า ε ที่ต่ำกว่านี้มักให้ผลใกล้เคียงกับอัลกอรธึมกรีดีจนทำให้เกิดการลู่เข้าสู่คำตอบที่ไม่ดี ดังนั้น ในหัวข้อนี้ จึงพิจารณาเฉพาะการทดลองที่ใช้ค่า $\varepsilon = 0.01$ เท่านั้น ผลการเปรียบเทียบสำหรับชุดข้อมูล IRIS เป็นไปดังแสดงในตารางที่ 5.3

ตารางที่ 5.3 การเปรียบเทียบผลลัพธ์ที่ได้จากจากการทดลอง 10 ครั้งของแต่ละอัลกอริธึม สำหรับชุดข้อมูล IRIS

จำนวนโปรโตไทป์	MCS	RL		RL+RCS	
		ε -กรีดี	ซอฟท์แมกซ์	ε -กรีดี	ซอฟท์แมกซ์
10	-	-	-	7	7
11	-	1	2	3	3
12	-	-	1	-	-
13	-	2	2	-	-
14	-	2	2	-	-
15	10	2	2	-	-
16	-	2	1	-	-
17	-	1	-	-	-

สำหรับชุดข้อมูล IRIS คำตอบที่ดีที่สุดมีขนาด 10 ตัว ซึ่งเป็นเซตคำตอบที่สามารถพบได้ด้วยอัลกอริธึมอื่น ๆ และผลที่ได้จากวิธี MCS มีขนาด 15 ตัวเสมอ ในที่นี้ อัลกอริธึม RL ที่ใช้การคัดโปรโตไทป์ออกให้ผลลัพธ์ที่ดีกว่า MCS ในหลาย ๆ ครั้ง แต่ก็ไม่พบเซตคำตอบขนาด 10 ตัว ซึ่งเป็นคำตอบที่ดีที่สุดเลย (ทั้งในกรณีที่ใช้โพลิซีเอฟซิลอน-กรีดี และซอฟท์แมกซ์) ขณะที่อัลกอริธึม RL+RCS นั้นให้ผลลัพธ์ที่ดีกว่าอัลกอริธึม RL ในทุก ๆ ครั้ง และพบเซตคำตอบขนาด 10 ตัว ถึง 7 ครั้งในการทดลอง 10 ครั้ง (ทั้งในกรณีที่ใช้โพลิซีเอฟซิลอน-กรีดี และซอฟท์แมกซ์)

ผลการเปรียบเทียบสำหรับชุดข้อมูล GLASS เป็นไปดังแสดงในตารางที่ 5.4

ตารางที่ 5.4 การเปรียบเทียบผลลัพธ์ที่ได้จากจากการทดลอง 10 ครั้งของแต่ละอัลกอริธึม สำหรับชุดข้อมูล GLASS

จำนวนโปรโตไทป์	MCS	RL+RCS	
		ε -กรีดี	ซอฟท์แมกซ์
80	-	-	-
81	-	-	1
82	-	1	3
83	-	3	3
84	-	6	2
85	10	-	1

สำหรับชุดข้อมูล GLASS ซึ่งไม่มีข้อมูลเกี่ยวกับเซตคำตอบที่ดีที่สุด ผลที่ได้จากวิธี MCS มีขนาด 85 ตัวเสมอ ในที่นี้ อัลกอริธึม RL+RCS นั้นให้ผลลัพธ์ที่ดีกว่าอัลกอริธึม MCS เล็กน้อย โดยโพลิซีแบบซอฟท์แมกซ์ให้ผลลัพธ์ที่มีความแตกต่างมากกว่าโพลิซีแบบเอพซิลอน-กรีดี และได้คำตอบที่ดีที่สุดที่มีขนาดเล็กกว่า สำหรับชุดข้อมูลนี้จะไม่ขอแสดงผลที่ได้จากอัลกอริธึม RL ที่ไม่มีการดัดแปลง

ผลการเปรียบเทียบสำหรับชุดข้อมูล ECOLI เป็นไปดังแสดงในตารางที่ 5.5

ตารางที่ 5.5 การเปรียบเทียบผลลัพธ์ที่ได้จากจากการทดลอง 10 ครั้งของแต่ละอัลกอริธึม สำหรับชุดข้อมูล ECOLI

จำนวน โปรโตไทป์	MCS	RL+RCS	
		ε -กรีดี	ซอฟท์แมกซ์
95	-	2	1
96	-	-	1
97	-	1	3
98	-	2	2
99	-	2	-
100	10	2	-
101	-	1	3

สำหรับชุดข้อมูล ECOLI ซึ่งไม่มีข้อมูลเกี่ยวกับเซตคำตอบที่ดีที่สุดเช่นกัน ผลที่ได้จากวิธี MCS มีขนาด 100 ตัวเสมอ ในที่นี้ อัลกอริธึม RL+RCS นั้นให้ผลลัพธ์ที่ดีกว่าอัลกอริธึม MCS เป็นส่วนใหญ่ แต่ก็มีบางครั้งที่ให้ผลลัพธ์ที่มีขนาดใหญ่กว่าอัลกอริธึม MCS (101 ตัว) ในที่นี้ การใช้โพลิซีแบบซอฟท์แมกซ์ให้ผลลัพธ์ที่มีความแตกต่างมากกว่าการใช้โพลิซีแบบเอพซิลอน-กรีดี และคำตอบที่ได้ส่วนใหญ่ก็มีขนาดเล็กกว่าคำตอบที่ได้มาจากการใช้โพลิซีแบบเอพซิลอน-กรีดี แต่ในกรณีที่ได้คำตอบขนาดใหญ่ คำตอบก็มักจะมีจำนวนโปรโตไทป์ค่อนข้างมากกว่าวิธีอื่น สำหรับชุดข้อมูลนี้จะไม่ขอแสดงผลที่ได้จากอัลกอริธึม RL ที่ไม่มีการดัดแปลงเช่นกัน

ผลการทดลองกับชุดข้อมูลทั้งสามชุดแสดงให้เห็นว่า อัลกอริธึม RL+RCS สามารถค้นหาคำตอบสำหรับชุดข้อมูลเหล่านี้ได้เป็นอย่างดี และสามารถหาคำตอบที่ดีกว่าอัลกอริธึม MCS ได้ นอกจากนี้ อัลกอริธึม RL+RCS ยังสามารถหาคำตอบที่ดีได้ตั้งแต่ในเอพิโสดต้น ๆ ดังนั้น อัลกอริธึม RL+RCS เมื่อเทียบกับอัลกอริธึม RL พื้นฐานและอัลกอริธึม RL+ECES จึงนับว่ามีประสิทธิผลที่ดีกว่ามาก

บทที่ 6
สรุปผลการทดลองและข้อเสนอแนะ

6.1 สรุปผลการทดลอง

ปัญหาการเลือกเซตย่อยสอดคล้องขนาดเล็กที่สุดหรือปัญหา MCSS เป็นปัญหาที่มีที่มาจากการพัฒนาเทคนิคการเลือกโปรโตไทป์ เพื่อลดจำนวนโปรโตไทป์ที่ใช้ในเซตตัวแทนในการจำแนกข้อมูลตามข้อมูลที่ใกล้เคียงที่สุด และทำให้การจำแนกข้อมูลมีประสิทธิภาพสูงขึ้น ในงานวิจัยที่เกี่ยวข้องกับการเลือกโปรโตไทป์พบว่า ปัญหา MCSS เป็นปัญหาที่มีความซับซ้อนสูง ทำให้ไม่สามารถค้นหาคำตอบที่ดีที่สุดได้ด้วยการสำรวจปริภูมิการค้นหาอย่างทั่วถึงในเวลาที่จำกัด จึงได้มีการพัฒนาอัลกอริธึมที่มีประสิทธิภาพในการแก้ปัญหานี้ อย่างไรก็ตาม การแก้ปัญหา MCSS ก็ยังไม่มีอัลกอริธึมที่ให้ผลที่สมบูรณ์แบบ จึงเป็นที่มาของแนวคิดในการประยุกต์ใช้การเรียนรู้แบบเสริมกำลังเพื่อแก้ปัญหา MCSS ซึ่งได้นำเสนอในวิทยานิพนธ์ฉบับนี้

ในการออกแบบอัลกอริธึมการเรียนรู้แบบเสริมกำลังหรืออัลกอริธึม RL เบื้องต้น ซึ่งมีพื้นฐานจากการใช้อัลกอริธึมการเรียนรู้แบบ Q ซึ่งเป็นอัลกอริธึมการเรียนรู้แบบเสริมกำลังที่เป็นที่นิยมที่สุดวิธีหนึ่ง ลักษณะการเรียนรู้ของอัลกอริธึม RL กำหนดขึ้นจากทาสก์การเรียนรู้ปัญหา ซึ่งแบ่งเป็นสองแนวทางในการเรียนรู้วิธีการเลือกโปรโตไทป์ คือ การค้นหาเซตย่อยสอดคล้องด้วยการเพิ่มจำนวนโปรโตไทป์ และการค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์หรือคัดโปรโตไทป์ออกจากการเป็นสมาชิก จากผลการทดลองเปรียบเทียบขนาดของเซตตัวแทนที่หาได้จากทาสก์การเรียนรู้ทั้งสองแบบในหัวข้อที่ 5.2 พบว่า ทาสก์การค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ มีประสิทธิภาพในการหาผลลัพธ์มากกว่าทาสก์การค้นหาด้วยการเพิ่มจำนวนโปรโตไทป์ อย่างไรก็ตาม เซตตัวแทนที่ได้จากการค้นหาเซตย่อยสอดคล้องด้วยการลดจำนวนโปรโตไทป์ก็ยังมีขนาดค่อนข้างใหญ่เมื่อเทียบกับผลที่ได้จากอัลกอริธึมอื่น ๆ จึงเป็นที่มาของการพัฒนาวิธีการดัดแปลงอัลกอริธึม RL เพื่อเพิ่มประสิทธิผล 2 วิธีคือ อัลกอริธึม RL+ECES ซึ่งใช้การใช้ความต่อเนื่องของเอพิโสดและการสำรวจสถานะเริ่มต้น และอัลกอริธึม RL+RCS ซึ่งใช้การย้อนกลับไปที่สถานะสุดท้ายที่มีความสอดคล้อง

จากผลการทดลองในหัวข้อที่ 5.3 พบว่า อัลกอริธึม RL+ECES ให้ผลที่ดีกว่าอัลกอริธึม RL พื้นฐาน และสามารถค้นหาคำตอบที่ดีกว่าอัลกอริธึม MCS ได้ในที่สุด อย่างไรก็ตาม เวลาและหน่วยความจำที่อัลกอริธึมใช้ในการค้นหาก็จัดอยู่ในปริมาณที่สูงมาก ทำให้การทดลองกับชุดข้อมูลขนาดใหญ่ขึ้นเป็นไปได้ยาก และแม้ว่าจะมีการทดลองปรับค่า ε หรืออัตราการสำรวจให้เป็นค่าต่าง ๆ แล้ว แนวโน้มของการลู่เข้าสู่คำตอบของอัลกอริธึมก็ยังไม่มีการเปลี่ยนแปลงที่เด่นชัด

จากผลการทดลองในหัวข้อที่ 5.4 พบว่า อัลกอริธึม RL+RCS ให้ผลที่ดีขึ้นจากอัลกอริธึม RL แบบอื่น ๆ ที่พัฒนาขึ้นมาก่อนหน้านี้เป็นอย่างมาก และสามารถหาเซตตัวแทนที่มีขนาดค่อนข้างเล็กได้ตั้งแต่ในเอพิโสดต้น ๆ แม้ว่าในแต่ละเอพิโสดจะใช้เวลาในการทำงานมากขึ้นกว่าอัลกอริธึม RL วิธีอื่น ๆ และในชุดข้อมูลทั้งหมดที่ใช้ในการทดลอง อัลกอริธึม RL+RCS ก็สามารถค้นหาคำตอบที่ดีกว่าหรือเท่ากันกับคำตอบที่ได้จากอัลกอริธึม MCS

จากผลการทดลองทั้งหมดทำให้สรุปได้ว่า เอเจนต์ RL สามารถค้นหาคำตอบของปัญหา MCSS ได้ดีขึ้นเมื่อกำหนดทาสก์การเรียนรู้ที่เหมาะสม โดยอัลกอริธึม RL ที่ให้ผลที่ดีที่สุดในการแก้ปัญหา MCSS ในงานวิจัยนี้ก็คือ อัลกอริธึม RL+RCS ซึ่งสามารถค้นหาคำตอบที่ดีกว่าหรือเท่ากันกับคำตอบที่ได้จากอัลกอริธึม MCS โดยเฉพาะในชุดข้อมูล IRIS ซึ่งอัลกอริธึมสามารถหาเซตคำตอบที่ดีที่สุดซึ่งมีขนาด 10 ตัวได้ โดยคำตอบนี้เป็นคำตอบที่พบจากอัลกอริธึมอื่น ๆ ที่มีการพัฒนามาก่อนหน้านี้ ขณะที่อัลกอริธึม MCS หาได้เพียงเซตคำตอบขนาด 15 ตัว

แม้ว่าอัลกอริธึม RL ทั้งหมดที่ได้นำเสนอในวิทยานิพนธ์ฉบับนี้อาจจะไม่ได้ให้ผลที่ดีที่สุดในทุก ๆ กรณี แต่ผลการทดลองในที่ได้นำเสนอไปนั้นก็แสดงให้เห็นว่าการเรียนรู้แบบเสริมกำลังนั้นก็สามารถนำไปใช้ในการแก้ปัญหาที่เกี่ยวกับการหาคำตอบที่ดีที่สุดได้เช่นเดียวกับอัลกอริธึมอื่น ๆ ที่มีใช้ทั่วไปในปัญหาลักษณะนี้ โดยทั่วไป ในปัญหาที่การโต้ตอบของผู้เรียนโดยการกระทำแบบเดียวกันในแต่ละครั้งภายใต้สถานการณ์เดียวกันอาจให้ผลลัพธ์ที่แตกต่างกัน เช่น ในระบบผู้เล่นเกมกระดานแบบต่าง ๆ การเรียนรู้แบบเสริมกำลังสามารถใช้ประโยชน์จากข้อมูลเหล่านี้ได้โดยตรง เนื่องจากการเรียนรู้แบบเสริมกำลังใช้หลักทางสถิติในการประเมินคุณค่าของการกระทำแต่ละแบบของผู้เรียนในสถานการณ์ที่ต่างกันบนพื้นฐานของความน่าจะเป็น การเรียนรู้แบบเสริมกำลังจึงเป็นทางเลือกที่เหมาะสมที่สุดในโจทย์ปัญหาลักษณะนี้ อย่างไรก็ตาม สำหรับปัญหาที่เกี่ยวกับการหาคำตอบที่ดีที่สุด การโต้ตอบด้วยการกระทำเดียวกันในสถานการณ์เดียวกันจะให้ผลแบบเดิมทุกครั้ง ทำให้อัลกอริธึมอื่น ๆ ที่ใช้ในการแก้ปัญหาลักษณะนี้โดยตรง เช่น อัลกอริธึมเจเนติก มีความเหมาะสมในการใช้งานมากกว่าการใช้อัลกอริธึมที่มีพื้นฐานมาจากการเรียนรู้แบบเสริมกำลัง

6.2 ข้อเสนอแนะ

แม้ว่าในที่สุดอัลกอริธึม RL ที่พัฒนาขึ้นในงานวิจัยนี้เมื่อผนวกกับแนวคิดในการดัดแปลงทาสก์การเรียนรู้การแก้ปัญหาแล้ว สามารถค้นหาคำตอบที่ดีได้ในการแก้ปัญหา MCSS สำหรับชุดข้อมูลที่นำมาทดลอง การใช้อัลกอริธึม RL ในการแก้ปัญหา MCSS ก็ยังมีอุปสรรคสำคัญ กล่าวคือ อัลกอริธึม RL ที่พัฒนาขึ้นและใช้ในการทดลองในงานวิจัยนี้ยังคงใช้โครงสร้างของฟังก์ชันมูลค่าของการกระทำในรูปของตารางค่า เนื่องจากเป็นโครงสร้างแบบพื้นฐานที่ใช้ในการเรียนรู้แบบเสริมกำลัง ข้อดีของการใช้ตารางค่าคือ การปรับค่าของฟังก์ชันมูลค่าจะไม่มีความ

กลาดเคลื่อน ทำให้สามารถวิเคราะห์พฤติกรรมต่าง ๆ ของอัลกอริธึมได้ง่าย ด้วยเหตุนี้ อัลกอริธึม RL จึงต้องใช้หน่วยความจำเป็นจำนวนมากขึ้นเป็นสัดส่วนที่สัมพันธ์โดยตรงกับขนาดของปริภูมิการค้นหาคำตอบของปัญหา ซึ่งมีความสัมพันธ์กับจำนวนข้อมูลในชุดข้อมูลเริ่มต้นในรูปแบบเอกซ์โพเนนเชียล ดังนั้น ถ้าชุดข้อมูลมีขนาดใหญ่ขึ้น หน่วยความจำที่ต้องการอาจมีขนาดใหญ่มากเกินกว่าที่ระบบคอมพิวเตอร์จะสามารถรองรับได้ ในขณะที่การเรียนรู้ที่เกิดขึ้นของอัลกอริธึมยังมีน้อยมาก ทำให้การทดลองอัลกอริธึมในแต่ละครั้งไม่สามารถทำได้ในระยะเวลานานมาก เนื่องจากหน่วยความจำของระบบจะเต็มก่อนที่อัลกอริธึมจะพบคำตอบที่ดี

โดยทั่วไป การเรียนรู้แบบเสริมกำลังที่มีปริภูมิสถานะขนาดใหญ่มักใช้เทคนิคการประมาณค่าของฟังก์ชันมูลค่า ซึ่งทำให้ลดปริมาณการใช้หน่วยความจำในการเก็บมูลค่าของการกระทำ ณ สถานะต่าง ๆ อย่างไรก็ตาม เทคนิคการประมาณค่าต่าง ๆ ก็มียังคงมีอุปสรรคในเรื่องของความคลาดเคลื่อนของการประมาณค่า และอาจทำให้พฤติกรรมของอัลกอริธึมเปลี่ยนไปในทางที่เข้าใจได้ยาก นอกจากนี้ การประมาณค่ากับสถานะที่ไม่ต่อเนื่องก็ทำได้ยาก ข้อมูลจากการทดลองเพิ่มเติมล่าสุดนอกเหนือจากที่นำเสนอในงานวิจัยนี้พบว่า เทคนิคการประมาณค่าทำให้คำตอบที่ได้จากอัลกอริธึม RL แตกต่างไปจากกรณีที่ใช้ตารางค่า และทำให้คำตอบที่ดีที่สุดที่หาได้ในการทดลองแต่ละครั้ง รวมถึงลักษณะการลู่เข้าสู่คำตอบของอัลกอริธึมเปลี่ยนไป อย่างไรก็ตาม เนื่องจากการวิเคราะห์ผลจากการประยุกต์ใช้เทคนิคการประมาณค่าในการเรียนรู้แบบเสริมกำลังนั้นก็เป็นสิ่งที่ทำได้ยาก และเทคนิคที่ใช้ในการประมาณค่าก็มีหลากหลาย แนวทางในการพัฒนาอัลกอริธึมด้วยเทคนิคเหล่านี้จึงยังต้องมีการค้นคว้าต่อไป ซึ่งผู้เขียนมีความคาดหวังว่าจะมีการพัฒนาอัลกอริธึม RL ด้วยเทคนิคการประมาณค่าฟังก์ชันในการแก้ปัญหา MCSS ต่อไปในอนาคต

นอกจากแนวทางการใช้เทคนิคการประมาณค่าแบบต่าง ๆ กับอัลกอริธึม RL ในการแก้ปัญหา MCSS แล้ว ยังมีแนวทางการพัฒนาอื่น ๆ ที่น่าสนใจอีก เช่น การใช้อัลกอริธึมอื่น ๆ ที่พัฒนาขึ้นมาสำหรับใช้ในการแก้ปัญหา MCSS (ตัวอย่างเช่น อัลกอริธึม MCS ของดาซาราธี อัลกอริธึมเจเนติก เป็นต้น) ร่วมกับอัลกอริธึม RL หรือการออกแบบทาสก์การเรียนรู้วิธีแก้ปัญหา MCSS สำหรับอัลกอริธึม RL ในรูปแบบอื่น ๆ นอกเหนือจากที่นำเสนอในงานวิจัยนี้ เป็นต้น ผู้เขียนหวังเป็นอย่างยิ่งว่างานวิจัยนี้จะเป็นจุดเริ่มต้นของการคิดค้นอัลกอริธึมที่มีประสิทธิภาพมากขึ้นในการหาคำตอบของปัญหา MCSS ในอนาคต

เอกสารอ้างอิง

[1] P. A. Devijver and J. Kittler, **"Pattern Recognition. A Statistical Approach."** Prentice-Hall, Englewood Cliffs, NJ, 1982.

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

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

[4] V. Cerverón and F. J. Ferri, **"Another Move Toward the Minimum Consistent Subset: A Tabu Search Approach to the Condensed Nearest Neighbor Rule,"** IEEE Transactions on Systems, Man and Cybernetics, Vol. 31, No. 3, 2001, pp. 408-413.

[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] J.C. Bezdek, T.R. Reichherzer, G.S. Lim and Y. Attikiouzel, **"Multiple-Prototype Classifier Design,"** IEEE Transactions on Systems, Man and Cybernetics, Vol. 28, 1998, pp. 67-79.

[7] 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.

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

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

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

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

[12] 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.

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

[14] K. Kangkan, B. Kruatrachue, **"Minimal Consistent Subset Selection as Interger Nonlinear Programming Problem,"** International Symposium on Communications and Information Technologies 2006 (ISCIT 2006), 2006, pp. 54-58.

ภาคผนวก

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

1. B. Kruatrachue and E. Anantapornkit, **"Episode Continuation and Exploration Start for Reinforcement Learning with Large Number of States,"** Asia Simulation Conference (JSST 2009), Shiga, Japan, October 7-9, 2009.
2. E. Anantapornkit and B. Kruatrachue, **"Reinforcement Learning Algorithm for the Minimal Consistent Subset Identification,"** International Conference on Machine Learning and Data Analysis (ICMLDA'09), pp. 848-852, San Francisco, USA, October 20-22, 2009.

ASIA SIMULATION CONFERENCE 2009

J S S T 2 0 0 9

October 7 - 9, 2009

Ritsumeikan University, Shiga, Japan

Japan Society for Simulation Technology (JSST)

Episode Continuation and Exploration Start for Reinforcement Learning with Large Number of States

Boontee Kruatrachue [1] and Ekaphol Anantapornkit [1]

[1] Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand

Abstract—This paper presents simple techniques to improve the learning rate of the RL algorithm in a task with large number of states that the algorithm cannot fully explore by going through all possible states. The problem with the standard RL algorithm is that varying the ε parameter in the ε-greedy policy is not sufficient to improve its learning performance. The techniques allow the algorithm to focus on exploring the local path to eventually obtain more rewards and occasionally switching to another path to avoid being trapped in the local region of the search space. In order to investigate the effectiveness of the techniques, the minimal consistent subset identification (MCSI) problem is used as a test problem. The paper concludes by comparing the size of the identified subset obtained from the standard RL algorithm and the proposed algorithm along with those of the standard MCSI method.

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

1 Introduction

Reinforcement Learning (RL) [4] has been used widely for Markov decision process (MDP). This paper proposes a simple method to improve its learning rate in a task with such a large number of states that the RL algorithm cannot fully explore by going through all possible states. This is close to a well known problem of exploration and exploitation trade-off in the RL algorithm. This trade-off is usually handled by varying the ε parameter in the ε-greedy policy to encourage the agent to keep exploring the search space. This is a problem because the RL algorithm keeps exploring and does not follow the greedy path long enough to be able to find the better reward (the optimum path). During an exploration, the number of states grows very fast and is finally out of control before the RL algorithm could find the optimum solution. On the other hand, if the ε parameter is reduced and thus the algorithm explores less, the RL algorithm tends to get stuck around the local optimum path and again the number of states is out of control after experiencing a large number of episodes. Hence, regardless of the ε parameter, the RL algorithm has to stop before a good solution could be found.

The paper proposes a way to encourage both the global search (exploration) and the local search (exploitation). This is in contrast to the ε-greedy policy that trades one for the other. In order to investigate the proposed method, it is tested on the minimal consistent subset identification (MCSI) problem.

Minimal consistent subset identification (MCSI) is the problem of selecting a minimum number of prototypes from training data set while maintaining the consistency property [2]. The set of prototypes selected can be used in the nearest neighbor classification [1] instead of using the original set. The selected prototype set is defined to be consistent if it is able to correctly classify the original data set by using the nearest neighbor classification. The MCSI problem is a hard combinatorial problem [5] that the optimum solution cannot be found by fully exploring the search space in limited time. The traditional MCSI method [3] by Dasarathy is one of the prototype selection methods that try to obtain the minimal consistent subset. The drawback of the method is that it employs the greedy strategy and is usually being trapped in a local optimum.

This paper is organized into 4 sections. Section 2 explains how to apply the reinforcement learning to the MCSI problem. Section 3 presents episode continuation and exploration start for the RL algorithm as a mean to improve the results obtained when solving a task with large number of states. The experimental results are shown in section 4. Section 5 is the conclusion.

2 Reinforcement Learning for the MCSI Problem

Reinforcement learning (RL) is a framework consisting of the agent going through states and the environment. In each state, the agent learns which action to take by obtaining the reward from an environment. After taking an action, the agent receives a reward and moves on to the other state. The goal is to repeat taking action and learn the best action to take in order to obtain the largest total rewards (returns).

It is natural to apply the RL framework to the MCSI problem as an episodic task by starting an episode from the start state with all of the training data being selected as a prototype, then let the RL agent repeat deselecting a prototype from the current state and move on to the next state which has one less prototype than the current one until reaching the terminal state which is the state that cannot be made consistent by deselecting more prototype (the state that represents an empty prototype set or a prototype set that doesn't include a member from some class) and thus the episode ends. A positive reward will be given to the agent if its action leads to the next state that is consistent (i.e. the state that represents a consistent prototype set) otherwise a reward of zero will be given. During an episode, the RL agent learns by averaging the expected returns of each action at each state and updating the state-action value function using a technique called Q-learning [4]. The agent can continue experiencing more episodes to learn more about a better way to deselect a prototype (thus gaining more returns).

The standard Q-learning algorithm is shown as follows:

For each episode:
  s = the start state with all prototypes
  While the episode does not end:
    1) the agent selects an action **a** to deselect a prototype from the state **s** using policy derived from **Q**
    2) the environment provides a reward for an action **a** at the state **s** and generates the next state $\mathbf{s_n}$
    3) the agent updates **Q(s,a)** using Watkin's **Q(λ)** [4]
    4) the agent goes to the next state $\mathbf{s_n}$ **(s=s_n)**

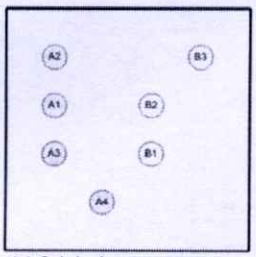
5) If the state **s** is the terminal state
episode end = true;

The reward is set to favor a path that leads to the consistent state with the minimum number of prototypes. The environment gives a reward of zero for an action that leads to an inconsistent state. All prototypes are selected in the start state. Any action that leads to the consistent next state will have a positive reward. The reward can be more than +1 if the previous action has a reward of zero.

The reward given for taking an action a from the state **s** and advancing to the next state $\mathbf{s_n}$ is defined as follows:

$$\mathbf{Reward(s, a, s_n)} = \mathbf{0}, \quad \text{if the state } \mathbf{s_n} \text{ is not consistent}$$
$$= \mathbf{P(s_{con}) - P(s_n)}, \quad \text{otherwise}$$

Where $\mathbf{s_{con}}$ is the latest consistent state found in the current episode, and **P(s)** is the number of prototypes in the state **s**.



(a) Original prototype set

	A1	A2	A3	A4	B1	B2	B3
A1	0.0	1.0	1.0	2.2	2.2	2	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	2.2
B2	2	2.2	2.2	2.2	1	0.0	1.4
B3	3.1	3.0	3.6	3.6	2.2	1.4	0.0

(b) Distance table

Fig. 1. Example prototype set.

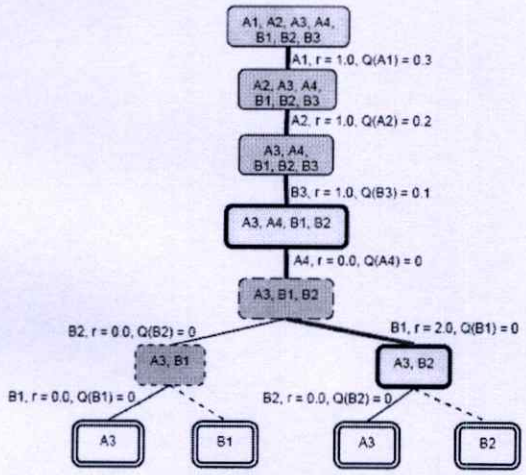


Fig. 2. The sequences of moves made by the RL agent.

Fig. 2 illustrates the sequences of moves that the agent made in 2 episodes. An example data set used in this figure is as shown in Fig. 1 which consists of 7 data from two classes (4 from class A and 3 from class B).

Initially, the value function **Q(a)** of each action is set to zero. In the first episode, the agent deselects A1, A2, B3, A4, B2, and B1 from the original set and the episode ends at the terminal state, represented by the double line rectangle, which has only A3 as a prototype in the set (thus cannot be made consistent by deselecting more prototypes). The best solution found in this episode is of size 4, {A3, A4, B1, B2} as represented by the thick line rectangle, thus the return of +3 is obtained and **Q(a)** is updated as shown in the figure.

In the next episode, the agent learns to exploit by deselecting A1, A2, and, B3 according to its **Q(a)** at its corresponding state, and incidentally deselects A4 from the state {A3, A4, B1, B2} in which the best action so far is not known, then tries to explore by deselecting B1 (as represented by the thick line) which ends up in a consistent state, {A3, B2}, and is given a reward of +2. In the end, the terminal state is the state {A3} and the best state in this episode is of size 2, the state {A3, B2}, which is the optimum solution, and the return of +5 is obtained. The inconsistent state is represented by the dashed line rectangle and the dashed lines from {A3, B1} to {B1} and from {A3, B2} to {B2} represent an unexplored path.

3 RL with Episode Continuation and Exploration Start

The number of states in the state space of the MCSI problem grows exponentially with respect to the size of the data set (2^N where N is the number of data in the data set). For a small data set, the RL algorithm can eventually converge to the optimum solution by simply exploring the entire state space. For a large data set, the RL agent may learn slowly and the optimum solution is rarely found in limited time. Controlling the ε parameter in the ε-greedy policy is not sufficient to improve the agent's learning rate. When the ε is large, the agent will explore more but converge very slow and the solution found in limited time may not be good. This is because the exploration is often made too early. On the other hand, the agent may follow the greedy path for a very long time even if the local minimum solution cannot be improved when the ε is reduced and the agent's opportunity to explore another path is very small.

To overcome this shortcoming, this paper proposes two modifications to the standard RL algorithm. The first modification is the episode continuation. In contrast to the standard RL that always starts a new episode at the same start state (with all prototypes), the agent is allowed to start an episode from the latest episode's state where the returns are maximum to find out whether more rewards can be gained from that point. This encourages the agent to search for the better solution locally by exploring from the potential state instead of starting over from the same starting point. Another advantage is that a convergence to a good solution is possible while the ε parameter is set to a large value. For the MCSI problem, the best state to continue from is the consistent state with the minimum number of prototypes found in the latest episode.

Because the agent is not allowed to move from any state to the state with larger number of prototypes, it is not a good idea to always start a new episode from the latest episode's best solution. Instead of always applying the episode continuation, the new episode starts by selecting the start state from the past episodes' best solutions with probability ε_{start}. The ε_{start} parameter controls whether the new episode will continue searching from the latest episode's best solution (exploitation by continuation) or start from the best state in the past episodes which is kept in the pool (exploration by selecting a starting point). This second modification is called the exploration start as it encourages the agent to explore many regions of the search space. With these two modifications to the RL algorithm, the balance between exploration and exploitation is made in two levels instead of relying only on the ε-greedy policy.

The selection of the start state of the new episode is defined as follows:

If the latest episode's best solution $\mathbf{S_{ep}}$ does not exist in the pool of the past episodes' best solutions $\mathbf{P_{best}}$:
Include $\mathbf{S_{ep}}$ in $\mathbf{P_{best}}$

With probability ε_{start}:
Select the start state $\mathbf{S_0}$ randomly from $\mathbf{P_{best}}$
Otherwise:
The start state $\mathbf{S_0}$ is assigned as $\mathbf{S_{ep}}$

The state with all prototypes being selected is also included in $\mathbf{P_{best}}$. This is done to make it possible for the agent to start over from the start state with all prototypes as done in the standard RL.

4 Experimental Results

All of the algorithms are tested with the IRIS data set of 150 data from three classes available at UCI site [6].

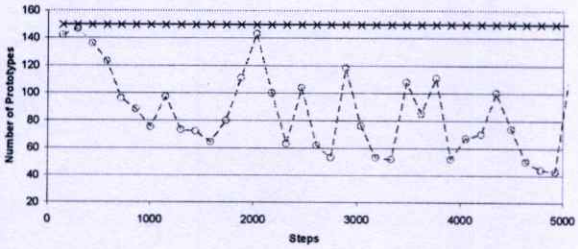


Fig. 3. The number of prototypes of the start state and the best solution found by the standard RL algorithm in each episode.

Fig. 3 shows how the standard RL algorithm converges. The RL agent always starts each episode at the same start state with 150 prototypes (an upper line with the cross marker in the graph). Each point is associated with the cumulative steps from the beginning of the execution to the end of each episode and represents the data of each episode. The agent then starts taking an action to deselect a prototype from the current state until the end of the episode. The lower line with circle marker shows the best consistent state with the smallest number of prototype found in each episode. The best solution in each episode keeps getting better but not monotonically depending on the exploration rate parameter (i.e. the ε). Each episode takes about 150 steps (actions) since all episodes start from the state with the original set.

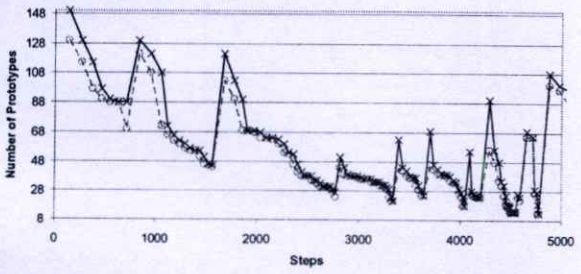


Fig. 4. The number of prototypes of the start state and the best solution found by the proposed RL algorithm in each episode.

Fig. 4 shows how the proposed RL algorithm converges. Compare to the standard RL algorithm, the main difference is the continuation from one episode to the next one as the start state of the next episode is the best state of the previous episode. In order to avoid getting stuck in the local minimum, some episode starts from the best state from the past which is kept in the pool. The cross marker in the graph shows the number of prototypes in the start state in each episode. The circle marker in the graph shows the best solution found in each episode corresponding to the cross marker at the same step. The circle marker becomes the next cross marker when the continuation happens. When the exploration start happens, the size of the start state of a new episode changes abruptly and is different from the size of the best solution of the latest episode. The size of the best solution in each episode decreases and eventually converges to the size of its associated start state when the episode continuation is applied. The number of steps in each episode varies depending on the start state. Most of the episodes have fewer steps compared to the standard RL episode.

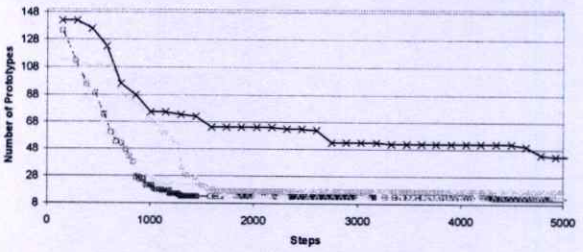


Fig. 5. Comparison of the number of prototypes of the best solution found by the standard RL (ε=0.01) and the proposed RL algorithm (ε=0.01, 0.05, and 0.1, ε_{start}=0.1).

Fig. 5 shows that the proposed RL algorithm with the ε parameter being set to 0.01, 0.05, and 0.1 and the ε_{start} being set to 0.1 converges much faster than the standard RL algorithm. The best solution of the standard RL algorithm with the ε=0.01 is represented by the straight line with the cross marker. The graph shows the progression of the best solution found by the algorithm since the start of the execution.

The size of the prototype set obtained from the proposed RL algorithm is compared to the standard RL algorithm and the standard MCSI [3] algorithm. Because RL-based algorithm has random elements, the algorithms are performed 10 times. The results are as shown in table 1.

Table 1. Comparison of the minimum consistent prototype set found by each algorithm.

Number of Prototypes	MCSI	RL	Proposed RL Algorithm		
		ε=0.01	ε=0.01	ε=0.05	ε=0.1
10	-	-	2	-	4
11	-	1	5	7	3
12	-	-	2	3	2
13	-	2	1	-	1
14	-	2	-	-	-
15	10	2	-	-	-
16	-	2	-	-	-
17	-	1	-	-	-

The optimum solution for the IRIS data set was found with size 10 [5], [7], the standard MCSI method's [3] result is 15. The standard RL does not perform well. It finds the result with the size smaller than 15 in some cases but performs poorer than the standard MCSI in the other cases. In addition, it could not find the optimum size (10 prototypes). The proposed RL algorithm outperforms the standard RL and even finds the optimum size in some cases. It always performs better than the standard MCSI.

For a larger value of the ε parameter, the proposed algorithm may obtain better solution more frequently as it spends more time on exploration. The standard RL algorithm will perform poorer when the ε parameter is lower as it learns slower when frequent exploration is made and the size of the state space is very large. The results obtained from setting a larger value of the ε parameter in the standard RL algorithm are omitted here.

5 Conclusion

This paper proposes a way to encourage exploration and exploitation in Reinforcement Learning in another level in addition to the action selection policy such as the ε-greedy policy. The learning rate of the standard RL can be too slow to find a good solution in a limited time when facing a task with large number of states. With episode continuation and exploration start, the RL algorithm can focus on exploring the local path to eventually obtain more rewards and occasionally switching to another path to avoid being trapped in the local minimum. As a result, the proposed algorithm learns faster and is able to obtain better solution than the standard RL. Compared to the standard MCSI method, which is a greedy local search that usually obtains a suboptimum solution, the proposed method can obtain better solution or optimum solution as it focuses on exploring many potential regions in the search space.

Episode continuation and exploration start can be useful in the planning phase of the learning task with large number of states. The authors expect that the technique presented here can be useful and further applied to other tasks in machine learning as well.

References

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

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

[3] 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.

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

[5] V. Cerveron and F. J. Ferri, "Another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule", IEEE Transactions on Systems, Man and Cybernetics, Vol. 31, No. 3, 2001, pp. 408-413.

[6] 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.

[7] K. Kangkan, B. Kruatrachue, "Minimal Consistent Subset Selection as Interger Nonlinear Programming Problem", ISCIT 2006, October 2006.

Lecture Notes in Engineering and Computer Science

WCECS 2009

World Congress on Engineering and Computer Science 2009

Volume II

S. I. Ao
Craig Douglas
W. S. Grundfest
Jon Burgstone (Eds.)

San Francisco, USA
20-22 October, 2009

IA ENG
International Association of Engineers

ISBN: 978-988-18210-2-7

Reinforcement Learning Algorithm for the Minimal Consistent Subset Identification

Ekaphol Anantapornkit and Boontee Kruatrachue

Abstract—**This paper describes the reinforcement learning (RL) algorithm for the minimal consistent subset identification (MCSI) problem. MCSI is widely used in pattern recognition to select prototypes from a training set to be used in nearest neighbor classification. The RL agent solves the MCSI problem by deselecting a prototype one by one from the original data set to search for the best subset. Because the algorithm rarely descends to the smaller solution via its exploration strategy, a simple modification to the algorithm is proposed. The modification encourages the agent to try as many actions as possible at the current best solution to improve the results obtained. The paper concludes by comparing the performance of the proposed algorithm in handling the MCSI problem with the RL algorithm and the standard MCSI method.**

Index Terms—**minimal consistent subset, nearest neighbor rule, prototype selection, reinforcement learning.**

I. Introduction

Minimal consistent subset identification (MCSI) is the problem of selecting a minimum number of prototypes from training data set while maintaining the consistency property [2]. The set of prototypes selected can be used in the nearest neighbor classification [1] instead of using the original set. The selected prototype set is defined to be consistent if it is able to correctly classify the original data set by using the nearest neighbor classification.

The MCSI problem is a hard combinatorial problem [5] that the optimum solution cannot be found by fully exploring the search space in limited time. The traditional MCSI method [3] by Dasarathy is one of the prototype selection methods that try to obtain the minimal consistent subset. The drawback of the method is that it employs the greedy strategy and is usually being trapped in a local minimum. This paper investigates the alternative solution by applying the reinforcement learning (RL) [4] to the MCSI problem. The standard RL method is also being trapped in a local minimum. In order to escape from the local minimum, a simple modification to the RL algorithm is proposed. The key modification is the new definition of the state transition function. This enables the agent to try as many actions as possible from the potential state and prevents the agent from exploring the unproductive region.

Manuscript received August 13, 2009.

Ekaphol Anantapornkit is with the Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand (e-mail: ekreal@yahoo.com).

Boontee Kruatrachue is with the Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand (e-mail: booontee@yahoo.com).

This paper is organized into 6 sections. The nearest neighbor classification is reviewed in section II. Section III reviews the minimal consistent set identification method by Dasarathy [3]. Section IV explains how to apply the reinforcement learning to the MCSI problem. Section V presents a modification to the RL algorithm to improve the results obtained. The experimental results are shown in section VI. Section VII is the conclusion.

II. Nearest Neighbor Rule

A prototype set $((x_1, y_1), c_1), \ldots, ((x_n, y_n), c_n)$ is given, where the (x_i, y_i) is the attribute value of the data point i^{th} and c_i is the category of the data i^{th}.

The unknown data ((a, b), z) can be classified as c_{nn}, which is the category of the prototype that is the nearest neighbor of (a, b). If $\mathbf{d((x, y), (a, b))}$ is the distance function that measure the difference (x, y) and (a, b). The nearest neighbor of unknown at (a, b) is defined as follows:

$$(x_{nn}, y_{nn}) = \min d((x_i, y_i), (a, b)) \; ; \; i = 1, 2, \ldots, n \qquad (1)$$

III. Minimal Consistent Set Identification Method

Minimal consistent set identification (MCSI) method is one of the condensing-selection prototype selection methods, which is based on the concept of covering defined by "NUN", the Nearest Unlike Neighbor [3]. Any data point A is covered by any data point B which has the same class as A, as long as B is nearer to A than A's NUN. Hence, A can be correctly classified, using nearest neighbor rule, to be the same class as B if B is selected as a prototype, as B is closer to A than A's NUN.

Once a distance matrix among all the data in the training set is sorted and all NUNs of each data point are located, a list of data points covered by any point B, B cover set, can be constructed. The cover list of B includes any data point A that has B located closer to A than A's NUN. The prototypes can be selected greedily incrementally by choosing a data point which has the largest cover list. Once that prototype is selected all the data point in that data cover list is covered (guarantee to be correctly classified) and that cover list is subtracted from all the cover lists. Then the data with the largest cover list is selected again as another prototype. The selection and subtraction process continues until all the cover lists are empty, which indicate the occurring of consistence property, where all the training data can be correctly classified by the selected prototypes.

The prototype set selected from the above algorithm is consistent but is still not minimal for two reasons. The first

one is the greedy order of the prototype selection which has no guarantee to minimal number of prototype. The second one is the inaccuracy of cover list calculation using NUN as boundary, since if the NUN is not selected as a prototype the cover lists can be expand. But before all prototypes are selected, it is safe to assume cover list boundary at NUN. To alleviate this problem, the prototypes selected are used again as boundary (instead of NUN) and the whole process is repeated until there is no change in the prototype set.

The MCSI Algorithm proceeds in the following steps. (as shown in Fig. 1)

1) Create a sorted distance table (c) among all data in the original prototype set (a).

2) Label all the NUN (Nearest Unlike Neighbor) by assumes that all the NUN are selected prototype (c).

3) Create a cover list for each prototype (d).

4) Select the prototype with maximum cover, and remove all the data in the covered list from all the cover lists (e).

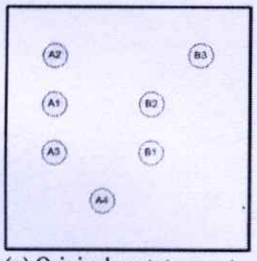
5) Repeat step 4 until all the cover lists is empty and obtains all new selected prototypes (f)

6) Stop if the new selected prototypes from step 5 are the same as the previous one.

7) Re-Label the sorted distance table (g) with the new selected prototypes from step 5.

8) Go to step 3.

The "cover" concept is shown in Fig. 1 (c), (d), where A1 covers A1, A2, A3 since A1 is closer to A1, A2 and A3 than B2, B2 and B1 (the nearest unlike neighbor of A1,A2 and A3). Hence, if A1 is selected as a prototype, A1, A2 and A3 can be correctly recognized.



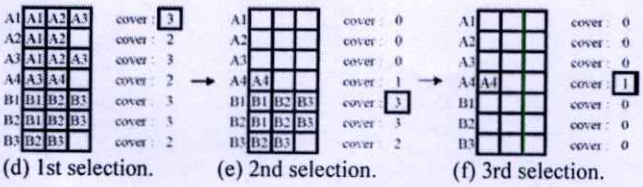
(a) Original prototype set.

	A1	A2	A3	A4	B1	B2	B3
A1	0.0	1.0	1.0	2.2	2.2	2	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	2.2
B2	2	2.2	2.2	2.2	1	0.0	1.4
B3	3.1	3.0	3.6	3.6	2.2	1.4	0.0

(b) Distance table.

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

(c) 1st sorted and labeled distance table.



(d) 1st selection. (e) 2nd selection. (f) 3rd selection.

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

(g) Re-labeled sorted distance table.

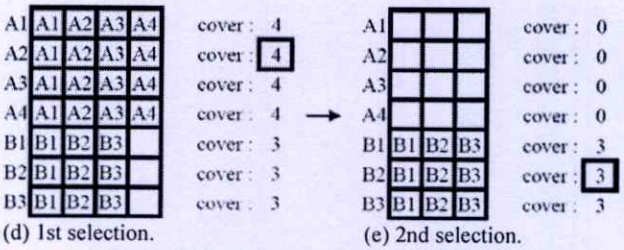
Fig. 1 Illustration of the MCSI algorithm

From Fig. 1, the first round of the MCSI obtains A1, A4, and B1 as a prototype set. After the second round, the MCSI obtains the same prototype set. So the MCSI obtains 3 prototypes (A1, A4 and B1) from 7 data.

Initially, if only A2 and B3 are selected as a prototype set and used as a boundary instead of NUN, as shown in Fig. 2 (a), the MCSI will obtain A2 and B3 as a prototype set. Hence, the optimality depends on how to initially label the boundary.

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

(a) Sorted and labeled distance table.



(d) 1st selection. (e) 2nd selection.

Fig. 2 Illustration of another selection approach

IV. Reinforcement Learning Algorithm for the MCSI Problem

Reinforcement learning (RL) [4] is a framework consisting of the agent going through states and the environment. In each state, the agent learns which action to take by obtaining the reward from an environment. After taking an action, the agent receives a reward and moves on to the other state. The goal is to repeat taking action and learn the best action to take in order to obtain the largest total rewards (returns).

It is natural to apply the RL framework to the MCSI problem as an episodic task by starting an episode from the start state with all of the training data being selected as a prototype, then let the RL agent repeat deselecting a prototype from the current state and move on to the next state which has one less prototype than the current one until reaching the terminal state which is the state that cannot be made consistent by deselecting more prototype (the state that represents an empty prototype set or a prototype set that doesn't include a member from some class) and thus the episode ends. A positive reward will be given to the agent if

its action leads to the next state that is consistent (i.e. the state that represents a consistent prototype set) otherwise a reward of zero will be given. During an episode, the RL agent learns by averaging the expected returns of each action at each state and updating the state-action value function using a technique called Q-learning [4]. The agent can continue experiencing more episodes to learn more about a better way to deselect a prototype (thus gaining more returns).

The standard Q-learning algorithm is shown as follows:
For each episode:
s = the start state with all prototypes
While the episode does not end:
1) the agent selects an action **a** to deselect a prototype from the state **s** using policy derived from **Q**
2) the environment provides a reward for an action **a** at the state **s** and generates the next state $\mathbf{s_n}$
3) the agent updates **Q(s,a)** using Watkin's **Q(λ)** [4]
4) the agent goes to the next state $\mathbf{s_n}$ **($s=s_n$)**
5) If the state **s** is the terminal state
episode end = true;

The reward is set to favor a path that leads to the consistent state with the minimum number of prototypes. The environment gives a reward of zero for an action that leads to an inconsistent state. There are N prototypes in the start state. Any action that leads to the consistent next state will have a positive reward. The reward can be more than +1 if the previous action has a reward of zero.

The reward given for taking an action **a** from the state **s** and advancing to the next state $\mathbf{s_n}$ is defined as follows:

$$\mathbf{Reward(s, a, s_n)} = \mathbf{0}, \quad \text{if the state } \mathbf{s_n} \text{ is not consistent}$$
$$= \mathbf{P(s_{con}) - P(s_n)}, \quad \text{otherwise}$$

Where $\mathbf{s_{con}}$ is the latest consistent state found in the current episode, and **P(s)** is the number of prototypes in the state **s**.

There are many ways to balance between exploration and exploitation. The popular method is the ε-greedy policy which is to choose an action that has maximum action value with probability $\mathbf{1 - \varepsilon + (\varepsilon / |A|)}$ and choose all other actions with probability $\mathbf{\varepsilon / |A|}$ where **A** is a set of all possible actions at the current state. The other method is the softmax policy.

The softmax policy used in this paper is defined as follows:
With probability ε:
Select an action **a** randomly
With probability **1 - ε**:
Select an action **a** based on its action value
with probability
$\mathbf{(1 - \varepsilon)\, Q(s, a)1.4 / \sum Q(s, b)1.4}$
For all action **b** that has **Q(s, b) > 0**.

This policy is adjusted to suit the MCSI problem which has many actions to choose at each state.

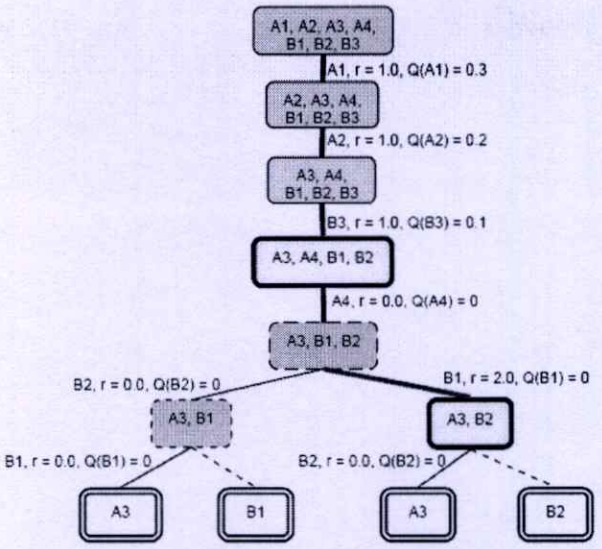


Fig. 3 The sequences of moves made by the RL agent

Fig. 3 illustrates the sequences of moves that the agent made in 2 episodes. An example data set used in this figure is the same data set as shown in Fig. 1 which consists of 7 data from two classes (4 from class A and 3 from class B).

Initially, the value function **Q(a)** of each action is set to zero. In the first episode, the agent deselects A1, A2, B3, A4, B2, and B1 from the original set and the episode ends at the terminal state, represented by the double line rectangle, which has only A3 as a prototype in the set (thus cannot be made consistent by deselecting more prototypes). The best solution found in this episode is of size 4, {A3, A4, B1, B2} as represented by the thick line rectangle, thus the return of +3 is obtained and **Q(a)** is updated as shown in the figure.

In the next episode, the agent learns to exploit by deselecting A1, A2, and, B3 according to its **Q(a)** at its corresponding state, and incidentally deselects A4 from the state {A3, A4, B1, B2} in which the best action so far is not known, then tries to explore by deselecting B1 (as represented by the thick line) which ends up in a consistent state, {A3, B2}, and is given a reward of +2. In the end, the terminal state is the state {A3} and the best state in this episode is of size 2, the state {A3, B2}, which is the optimum solution, and the return of +5 is obtained. The inconsistent state is represented by the dashed line rectangle and the dashed lines from {A3, B1} to {B1} and from {A3, B2} to {B2} represent an unexplored path.

V. Improving the Effectiveness of the RL Algorithm

The example in the previous section indicates that it is unlikely that the agent will obtain more rewards once it deselects a wrong prototype and advance to the inconsistent state. From that point on, the agent may just experience the sequence of zero rewards till the end of the episode. If the agent can return to the previous state before the wrong prototype is deselected, it may be able to find the better solution by trying another action from that state. However, the agent is not able to return to the previous state in the same episode and it rarely returns to the best state of the previous episodes via the normal action selection policy such as the

ε-greedy policy if the agent fails to obtain the largest returns from that state. This hampers the agent's effort to locate the better solution from many potential states.

For example, if the ε=0.1 is used in the ε-greedy policy, the agent will follow the path that leads to the smallest consistent state experienced so far for about 9 steps before the exploration step happens in the 10th step. If the exploration step at that state leads to an inconsistent state, the episode may end with no more rewards and fail to influence the change to the agent's exploitation strategy. Consequently, the agent rarely returns to many consistent states located deeper than 10 steps unless they are part of the current best path.

To overcome this shortcoming, a simple modification to the algorithm is presented. The idea is to force the agent to return to the last known consistent state if the current path from that state is overly long or the agent moves to the dead end. This encourages the agent to try as many actions as possible from that consistent state. The agent will eventually reach the smaller consistent state if there is one. The smaller consistent state then becomes the last known consistent state. The maximum steps allowed in the episode must be defined otherwise the episode will never end. The detail of the modification is as described below.

The new state transition function is defined as follows:
After the agent selects an action **a**
and the environment generates the next state $\mathbf{s}_n$ as usual
If $\mathbf{s}_n$ is consistent:
  The environment provides a reward
  and proceeds as usual
Else:
  If $\mathbf{s}_n$ cannot leads to the smaller consistent state
  or with probability **x**
    $\mathbf{s}_n$ = the last known consistent state in the episode
    Reward = -1
  Else:
    Proceeds as usual

This transition function prevents the agent from moving through a long sequence of inconsistent states and ending the episode prematurely without a chance to explore a better path. The reward -1 is given as a penalty to signal the agent that it is better to explore another path instead.

The probability **x** can be any value between [0, 1]. The zero value means no backward transition is made unless the agent cannot deselect more prototypes. The value of 1 means the agent always moves back to the last known consistent state within the episode if the move does not lead to the next consistent state with one less prototype. The value of 0.2 is selected here because it prevents a move through a very long sequence of inconsistent states, but still allows an agent to move through a sequence of states that has some intermediate inconsistent states (this case is as shown in Fig. 3) which may be longer than 2 steps.

The termination condition of the algorithm is also changed from reaching the terminal state (there are no terminal states here) to reaching the maximum steps allowed in each episode. The maximum steps can be any number larger than the size of the original data set, but 5 times of the size of the original data set is usually good.

It is clear that the agent's behavior is not changed since the update rule for the state-action value function is still the same. The major change is in the state transition function which is defined by the environment.

The performance comparison of the standard RL and the modified RL for the MCSI problem is as shown in Fig. 4. Both algorithms use the ε-greedy policy. The data set used here is the IRIS data set of 150 data from 3 classes.

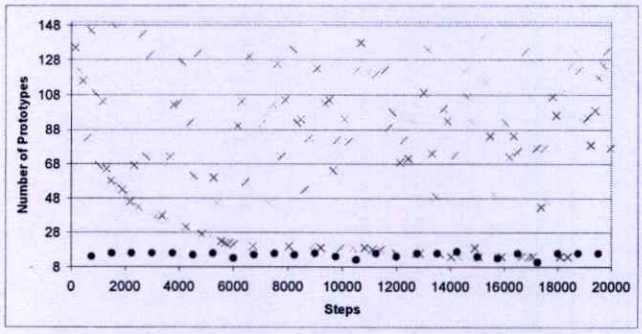


Fig. 4 Performance comparison of the standard RL and the modified RL for the MCSI problem

Each point in the graph shows the best consistent state with the smallest number of prototypes found in each episode. The cross points show the results of the standard RL algorithm while the round points show the results of the modified RL algorithm. Each point in the graph is the size of the best consistent subset found in each episode associated with the cumulative steps used by the algorithm. The best state in the later episode keeps getting better but not monotonically depending on the exploration rate parameter (i.e. the ε). Each episode takes about 150 steps in the standard RL and about 750 steps in the modified RL. The size of the best solutions found by the modified RL algorithm in each episode vary from 10 to 18 prototypes while the standard RL gets stuck in the long sequence of inconsistent states in some episodes and the size of the best solutions vary from 14 to 147 prototypes. The modified RL algorithm finds the solution of size 14 at the first episode as it also learns within an episode. The optimum set of 10 prototypes [5], [7] is eventually found by the modified RL algorithm while the best that the standard RL algorithm can find is of size 14 in this case.

VI. Experimental Results

The size of the prototype set obtained from the standard RL algorithm is compared to the proposed modified RL algorithm and the standard MCSI [3] algorithm on the test data sets available at UCI site [6]. The test data sets are as shown in Table I.

TABLE I
DETAIL OF DATA SETS

	NUMBER OF DATA	DIMENSION	CLASS
IRIS	150	4	3
GLASS	214	9	7

Because RL-based algorithm has random elements, the algorithms are performed 10 times for each data set. The best solution found in each run for the IRIS data set is as shown in

table II.

TABLE II
THE RESULTS OF THE ALGORITHMS FOR THE IRIS DATA SET

NUMBER OF PROTOTYPES	10	11	12	13	14	15	16	17
MCSI	-	-	-	-	-	10	-	-
Standard RL (ε-greedy)	-	1	-	2	2	2	2	1
Standard RL (softmax)	-	2	1	2	2	2	1	-
Modified RL (ε-greedy)	7	3	-	-	-	-	-	-
Modified RL (softmax)	7	3	-	-	-	-	-	-

In the IRIS data set case, the optimum set was found with size 10 [5], [7], the standard MCSI method's [3] result is 15. The standard RL does not perform well (for both ε-greedy and softmax). It finds the result with the size smaller than 15 in some cases but performs poorer than the standard MCSI in the other cases. In addition, it could not find the optimum size (10 prototypes). The modified RL outperforms the standard RL and even finds the optimum size in most cases. It always performs better than the standard MCSI.

TABLE III
THE RESULTS OF THE ALGORITHMS FOR THE GLASS DATA SET

NUMBER OF PROTOTYPES	80	81	82	83	84	85
MCSI	-	-	-	-	-	10
Modified RL (ε-greedy)	-	-	1	3	6	-
Modified RL (softmax)	-	1	3	3	2	1

As shown in table III, the modified RL algorithm performs slightly better than the standard MCSI method in this data set. The softmax algorithm has more variance on the results obtained, but the best solution obtained by this method is better than the ε-greedy method in this case. The results from the standard RL are omitted here.

VII. CONCLUSION

This paper presents a reinforcement learning approach to the solution of the MCSI problem. The traditional RL method has some drawbacks and is easily being trapped in the local optimum. However, with a simple modification, the RL agent is able to get better results or obtain the optimum solution. Compared to the standard MCSI method, which is a greedy local search that usually obtains a suboptimum solution, the proposed method performs better as it learns to make a correct move through many trials in the potential region in the search space.

REFERENCES

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

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

[3] 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.

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

[5] V. Cerveron and F. J. Ferri, "Another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule", IEEE Transactions on Systems, Man and Cybernetics, Vol. 31, No. 3, 2001, pp. 408-413.

[6] 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.

[7] K. Kangkan, B. Kruatrachue, "Minimal Consistent Subset Selection as Interger Nonlinear Programming Problem", ISCIT 2006, October 2006.

ประวัติผู้เขียน

นายเอกพล อนันตพรกิจ เกิดเมื่อวันที่ 25 มีนาคม พ.ศ. 2523 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2544 จากนั้นได้เข้าศึกษาต่อในระดับปริญญาโท หลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2547 และได้เข้าทำงานในตำแหน่งอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปี พ.ศ. 2546 จนถึงปี พ.ศ. 2551