ปัญญาประดิษฐ์สำหรับการพัฒนาเกมส์

Artificial Intelligence for Game Development

จัดทำโดย

ผศ. ดร. ศันสนีย์ เอื้อพันธ์วิริยะกุล

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเชียงใหม่

พ.ศ. 2549

คำนำ

เนื้อหาในหนังสือฉบับนี้เป็นเพียงเนื้อหาที่เกี่ยวกับการนำปัญญาประดิษฐ์ไปใช้ในการพัฒนาเกมส์ และปัญญาประดิษฐ์ที่อธิบายในหนังสือเป็นปัญญาประดิษฐ์อย่างง่าย ดังนั้นหนังสือเล่มนี้จึงเหมาะกับนักศึกษาในระดับปริญญาตรี หรือผู้ที่ไม่มีความรู้ทางด้านนี้มาก่อน

สำหรับคำศัพท์ทางเทคนิคที่เป็นภาษาอังกฤษที่ใช้ในเอกสารประกอบการสอนนี้ ผู้เขียนพยายามใช้คำภาษาไทยที่เหมาะสมที่สุดเท่าที่พึงกระทำได้ และให้สื่อความหมายที่ถูกต้องให้มากที่สุด โดยอ้างอิงถึงศัพท์เทคนิคของวิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์ และศัพท์บัญญัติของราชบัณฑิตย สถาน ทั้งนี้ทั้งนั้น ผู้เขียนได้เขียนคำศัพท์ภาษาอังกฤษประกอบไว้ด้วย เพื่อความเข้าใจที่ถูกต้อง นอกจากนี้ เพื่อความชัดเจนและเพื่อคุณภาพที่ดีของเอกสาร ผู้เขียนได้จัดทำรูปในเนื้อหาใหม่เองทั้งหมด

ผู้เขียนหวังเป็นอย่างยิ่งว่า หนังสือเล่มนี้จะเอื้อประโยชน์ให้ผู้อ่านได้รับความรู้ทางด้านการนำปัญญาประดิษฐ์ไปใช้ในการพัฒนาเกมส์ ผู้เขียนยังหวังว่าหนังสือเล่มนี้จะเป็นประโยชน์สำหรับบุคคลทั่วไปที่มีความสนใจทางด้านนี้อีกด้วย หากผู้อ่านมีข้อคิดเห็นหรือข้อแนะนำประการใด อันจะเป็นผลให้หนังสือเล่มนี้มีความสมบูรณ์มากยิ่งขึ้น โปรดส่งอีเมล์มายังผู้เขียนที่ sansanee@eng.cmu.ac.th

ศันสนีย์ เอื้อพันธ์วิริยะกุล

สารบัญ

เกริ่นนำ
General Wisdom
1

ในบทนี้จะกล่าวถึงปรัชญาโดยทั่วไปของการนำปัญญาประดิษฐ์ (Artificial Intelligence (AI)) มาช่วยในการสร้างเกมให้มีความฉลาด

1.1 บทนำ (Introduction)

ก่อนที่จะกล่าวถึงปัญญาประดิษฐ์ในเกมส์ (Game AI) ขอกล่าวอธิบายว่าอะไรคือปัญญาประดิษฐ์ในเกมส์ ปัญญาประดิษฐ์ในเกมส์คือโปรแกรมในเกมส์ที่สร้างศัตรูที่ควบคุมด้วยคอมพิวเตอร์ (computer-controlled) ที่ดูเหมือนว่าสามารถตัดสินใจได้อย่างฉลาดเมื่อเกมส์มีตัวเลือกหลายตัวเลือกสำหรับสภาวะใดๆ ซึ่งทำให้เกิดพฤติกรรมที่เกี่ยวเนื่อง มีประสิทธิภาพและมีประโยชน์

ปัญญาประดิษฐ์ในเกมส์ในสมัยก่อนเป็นปัญญาประดิษฐ์ที่ใช้การแทนสัญลักษณ์ (symbolic representations) เช่น board หรือ card เกมส์ ตัวอย่างเกมส์ที่นำปัญญาประดิษฐ์ไปใช้และประสบผลสำเร็จคือ checkers program Chinook [Schaeffer97, Miikkulainen06] ซึ่งภายหลังได้เป็น world champion ในปีค.ศ. 1994 และ chess program Deep Blue [Campbell02, Miikkulainen06] ซึ่งชนะ world champion ในปีค.ศ. 1997 จากการประสบความสำเร็จของทั้งสองเกมส์นี้ทำให้ความสนใจในการนำปัญญาประดิษฐ์มาใช้ในการพัฒนาเกมส์เป็นไปอย่างกว้างขวาง แต่ปัญญาประดิษฐ์แบบเก่า (Classical Artificial Intelligence) นี้ไม่ค่อยเหมาะสมกับวีดีโอเกมส์ เนื่องจากในวีดีโอเกมส์ มีตัวแทน (agent) เข้ามาเกี่ยวข้อง ซึ่งตัวแทนเหล่านี้ต้องโต้ตอบ (interact) กันในสภาวะจำลอง (simulated physical environment) โดยผ่าน เซนเซอร์ (sensor) และตัวคุม (effector) ซึ่งการทำงานลักษณะนี้เป็นการทำงานเชิงตัวเลข ไม่ใช่สัญลักษณ์

แต่ในทางกลับกัน การใช้ความฉลาดเชิงคำนวณ (Computational Intelligence (CI)) เช่น neural networks, fuzzy sets และ evolutionary computing เป็นสิ่งที่เหมาะสมกับการนำไปใช้ในวีดีโอเกมส์มากกว่า เนื่องจากเทคนิคเหล่านี้สามารถทำงานได้ในสภาวะแวดล้อมที่ใอยู่ในวีดีโอเกมส์ซึ่งเป็นสภาวะที่เปลี่ยนแปลงรวดเร็ว มีสัญญาณรบกวน (noise) และเป็นการคำนวณเชิงสถิติ และเป็นตัวเลข ดังนั้นเพื่อความเข้าใจที่ตรงกันคำว่าปัญญาประดิษฐ์ที่จะกล่าวถึงในหนังสือเล่มนี้ จะหมายถึงเทคนิคหรืออัลกอริทึมในความฉลาดเชิงคำนวณ (Computational Intelligence) เสมอ

ในการนำปัญญาประดิษฐ์มาใช้ในเกมส์นั้นเราไม่ได้ต้องการในเป็นสิ่งที่ฉลาดในทุกเรื่อง แต่สิ่งที่นักพัฒนาเกมส์ต้องการคือ บริบทที่ขึ้นกับความเชี่ยวชาญ (context-dependent expertise) หรือความฉลาดเฉพาะเรื่องนั่นเอง เช่น pathfinding, steering, flocking, unit deployment, tactical analysis, strategic planning, resource allocation, weapon handling, target selection, group coordination, simulated perception, situation analysis และ spatial reasoning เป็นต้น ดังนั้นเกมส์ที่จะประสบผลสำเร็จคือเกมส์ที่นักพัฒนาสามารถบ่งบอกได้ว่าสำหรับเกมส์ประเภทนี้ ปัญหาย่อย (subproblem) คืออะไร และหาเทคนิคมาใช้ในการแก้ปัญหานั้นๆ ในเกมส์นั้นๆ และในการสร้างบริบท

ที่ขึ้นกับความเชี่ยวชาญ นักพัฒนาจะต้องเป็นผู้เชี่ยวชาญในการแก้ปัญหานั้นก่อน ถึงจะพัฒนาเทคนิคหรือพัฒนาวิธีการเพื่อนำมาแก้ปัญหานั้นได้ดี

1.2 ความคาดหวังของผู้เล่นที่เป็นมนุษย์ (Hallmarks of a human player)

ถ้าผู้เล่นคิดว่ากำลังเล่นกับตัวแทนที่ฉลาด แล้วตัวแทนนี้จะฉลาดจริง ซึ่งเวลาที่ผู้เล่นที่เป็นมนุษย์เล่นเกมส์กับผู้เล่นที่เป็นคอมพิวเตอร์ หรือตัวแทน จะมีความคาดหวังหลายอย่างรวมทั้ง

1. ทำนายได้และทำนายไม่ได้ (predictability and unpredictability) คนโดยปกติจะทำอะไรที่ไม่สามารถทำนายได้ เช่นใน a real-time strategy (RTS) เกมส์ ผู้เล่นอาจจะโจมตีกองกำลังที่แข่งแรงกว่าเพื่อจะดึงความสนใจไปทางอื่น แต่ในทางกลับกันมีผู้เล่นหลายๆคนที่ทำอะไรที่สามารถทำนายได้ เช่นใน RTS เกมส์ผู้เล่นอาจจะเดินทางไปในทางที่ชอบเสมอ ดังนั้นคนสามารถเป็นได้ทั้งผู้เล่นที่สามารถทำนายได้ และทำนายไม่ได้ ซึ่งอาจจะเกิดในเกมส์เดียวกัน หรืออาจจะเป็นผู้เล่นที่ไม่สามารถทำนายได้ถ้าเกมส์เดิมหลายๆครั้ง แต่ภายในครั้งเดียวกันเป็นผู้เล่นที่ทำนายได้ ดังนั้นการพัฒนาปัญญาประดิษฐ์จะต้องทำให้มีสุ่มที่มากพอเพื่อให้เกมส์นั้นน่าสนใจ และในขณะเดียวกันต้องเป็นผู้เล่นที่ทำนายได้เพื่อให้ผู้เล่นที่เป็นคนสามารถบอกได้ว่าจะต้องทำอย่างรถึงจะชนะเกมส์นี้
2. สนับสนุน (support) ในบางครั้งผู้เล่นที่เป็นคนจะเลือกผู้เล่นที่เป็นคอมพิวเตอร์ให้ทำหน้าที่เป็นผู้ช่วยเหลือ เช่นโจมตีผู้เล่นที่เป็นคนคนอื่น หรือผู้เล่นที่เป็นคอมพิวเตอร์ เครื่องอื่น ซึ่งการทำหน้าที่นี้จะมีเรื่องของการติดต่อกับผู้เล่นที่เป็นคนเข้ามาเกี่ยวข้อง
3. ความผิดคาด (surprise) เกมส์ที่เป็นที่พูดถึงโดยส่วนใหญ่จะเป็นเกมส์ที่มีความผิดคาดเกิดขึ้น แต่ทั้งนี้ทั้งนั้นจะต้องระวังไม่ให้มีความผิดความที่ไม่ฉลาดเกิดขึ้น เช่นตัวแทนวิ่งชน หรือติดอยู่ในมุมและไม่สามารถหาทางออกได้ เป็นต้น

1.3 คำแนะนำ 12 ข้อ

คำแนะนำเที่จะกล่าวถึงต่อไปนี้เป็นคำแนะนที่มาจากนักพัฒนาเกมส์ที่ใช้ปัญญาประดิษฐ์ ซึ่งมีดังต่อไปนี้

1. ทำการบ้าน (do your homework) ให้คิดไว้เสมอว่าไม่มี ระบบปัญญาประดิษฐ์ ระบบไหนที่สามารถแก้ปัญหาได้ทั้งหมด การหาคำตอบที่ถูกต้องขึ้นอยู่กับองค์ประกอบหลายองค์ประกอบเช่น
    ก. ปัญญาประดิษฐ์ที่ต้องการเน้นในเกมส์คืออะไร เช่นการหาเส้นทาง (pathfinding) เป็นจุดที่สำคัญ หรือการเรียน และการแสดงท่าทางสำคัญกว่า เมื่อได้เป้าหมายที่แน่นอนจึงทำการค้นหาว่าเคยมีการนำเทคนิคใดมาใช้ในการแก้ปัญหาบ้างและเทคนิดไหนที่จะเหมาะสม
    ข. เวลาและกำลังคนมีมากน้อยเท่าไร
    ค. การแบ่งกำลังคนเป็นอย่างไร เช่นในทีมมีโปรแกรมเมอร์ และนักออกแบบกี่คนและแต่ละคนมีความสามารถแตกต่างกันอย่างไร

2. ทำให้ง่าย (keep it simple) ระบบปัญญาประดิษฐ์ควรเป็นระบบที่นักออกแบบและโปรแกรมเมอร์สามารถสร้างสิ่งที่ทำงานซับซ้อนได้ด้วยองค์ประกอบที่ง่าย เช่นเข้าใจง่าย ง่ายต่อการดีบัค debug นำกลับมาใช้ใหม่ และปรับปรุง (maintain)
3. ทดลองในกระดาษ (Try it out on paper first) ลองเขียนอัลกอริทึมคร่าวๆในกระดาษก่อน และนำไปให้นักออกแบบที่จะต้องใช้ระบบนี้อ่านก่อนเพื่อเช็คว่าสิ่งที่นักออกแบบต้องการใช้ มีครบแล้วหรือไม่
4. หาทิศทางไว้ก่อน (precompute navigation) ในการทิศทางในสภาวะแวดล้อมที่เป็น 3 มิติจะใช้เวลาในการหาทิศทางนาน วิธีที่ง่ายคือหาทิศทางให้ตัวแทนไว้ล่วงหน้าก่อนโดยไม่ให้ผู้เล่นทราบ และสิ่งที่จะทำให้เกิดการเสียเวลาคือการทดสอบการชน (collision checking) ซึ่งตัวแทนสามารถใช้ข้อมูลที่สร้างไว้ล่วงหน้า โดยที่นักออกแบบเป็นคนสร้างไว้ให้ ตัวอย่างเช่น
    ก. นักออกแบบอาจทาสีพื้นที่ ที่ไม่ต้องการให้ตัวแทนผ่าน
    ข. นักออกแบบอาจสร้างส่วนของสันตรงเพื่อเป็นเส้นทางให้ตัวแทนเดินทางจากที่หนึ่งไปยังอีกที่หนึ่ง
    ค. นักออกแบบอาจสร้างพื้นที่ ที่ตัวแทนสามารถเดินทางได้โยไม่ต้องทดสอบการชน
5. ทำให้โลกฉลาด แทนที่ปัญญาประดิษฐ์ (put the smarts in the world, not in the AI) การเขียนระบบปัญญาประดิษฐ์ง่ายๆ โดยที่ทำให้ตัวแทนเลือกจุดหมายและเดินทางไปถึงจุดหมาย และวัตถุหรือตำแหน่งนั้นๆออกคำสั่งให้กับตัวแทนในการทำอะไรบางอย่างทำได้ง่ายกว่าการเขียนโปรแกรมให้ตัวแทนรู้ทุกอย่าง ตัวอย่างเช่น ตัวแทนหิว ตัวแทนจะมองหาวัตถุที่ประกาศว่าเป็นอาหาร และตัวแทนเลือกเดินทางไปหาอาหารที่ใกล้ที่สุด เมื่อตัวแทนมาถึงวัตถุนั้นจะบอกให้ตัวแทนทำอะไรเช่นหยิบลูกแอปเปิลจากต้นไม้ หรือเปิดตู้เย็น หรือใช้เครื่องขายอาหาร ดังนั้นตัวแทนจะถูกมองเหมือนว่าตัวแทนนั้นฉลาด ทั้งที่ในความเป็นจริงตัวแทนนั้นเพียงแค่ทำตามคำสั่งเท่านั้น และข้อดีอีกอย่างของการทำแบบนี้คือสามารถเพิ่มสิ่งใหม่ๆเข้าไปในเกมส์ได้โดยไม่ต้องเปลี่ยนแปลง ตัวโปรแกรมของปัญญาประดิษฐ์เลย
6. ทำให้ทุกการกระทำมีการหมดเวลา และการย้อนกลับ (give every action a timeout and fallback) ควรจะมีการตรวจสอบถึงเงื่อนไขที่เป็นความสำเร็จว่า ตัวแทนสามารถทำอะไรบางอย่างสำเร็จภายในเวลาที่กำหนดหรือไม่ ถ้าไม่ ให้ยกเลิกการกระทำนั้นและลองทำอย่างอื่น และในบางครั้งตัวแทนอาจย้อนกลับไปทำการเคลื่อนไหวบางอย่างที่น่าสนใจ เช่นแสดงให้เห็นว่ากำลังสับสนหรือตกใจ หรืออาจจะทำให้ตัวแทนพิจารณาสถานการณ์และเริ่มแผนใหม่ถ้ายังมี processing power เหลือเพียงพอ
7. ใช้ลำดับชั้นของสถานะ (use a hierarchy of states) ไฟไนต์สเตทแมชชีน (finite state machine) เป็นเครื่องมือที่ใช้ในการควบคุมพฤติกรรมของตัวแทน ถ้าสถานะ (state) ถูกออกแบบให้ง่าย มีหลายเป้าหมาย (general-purpose) และนำกลับมาใช้ใหม่ได้ สถานะเหล่านี้สามารถถูกนำกลับมาใช้ใหม่ได้ในหลายสถานการณ์ และการนำสถานะเหล่านี้ใส่

ในลำดับชั้น (hierarchy) โดยให้สถานะในชั้นต่ำเป็นสถานะที่นำกลับมาใช้ใหม่ และสถานะในชั้นสูงเป็นสถานะที่จัดการเกี่ยวกับ การตัดสินใจ การวางแผน

8. ไม่ให้ตัวแทนยุ่งเกี่ยวกับเหตุการณ์สำคัญในการเล่าเรื่อง (do not let agents interfere with the crucial storytelling events) ตัวแทนควรจะรู้เกี่ยวกับเหตุการณ์ที่สำคัญกัยการเล่าเรื่อง เช่นเมื่อผู้เล่นกำลังคุย ฟัง หรือแก้ปัญหา ตัวแทนไม่ควรเข้าไปยุ่ง และควรจะถอยออกมา ถ้าผู้เล่นต้องการสู้กับศัตรูในขณะที่เรื่องกำลังถูกเล่า ผู้เล่นอาจจะไม่ได้รับข้อมูลบางอย่าง
9. ทำให้ตัวแทนทราบเกี่ยวกับสถานะส่วนกลางของโลก (keep agents aware of the global state of the world) โลกของเกมส์และตัวละครที่น่าเชื่อถือทำให้เกมส์เป็นเกมส์ที่น่าเล่น ดังนั้นตัวแทนควรจะต้องจำได้ว่าเกิดอะไรขึ้นต่อโลกของเกมส์ ตัวละคร หรือตัวเองตลอดทั้งเกมส์ และควรจะเปลี่ยนแกลงพฤติกรรม และบทพูด ตามเหตุการณ์นั้นๆ เพื่อทำให้ผู้เล่นเชื่อว่านี่คือ ของจริง การใช้ตัวบ่งชี้ส่วนกลาง (global flag) และข้อมูลส่วนกลาง (global data) อาจจะจะช่วยในการจดจำการกระทำของผู้เล่น
10. สร้างความหลากหลายให้กับข้อมูล ไม่ใช่ให้กับโปรแกรม (create variety through the data, not through the code) การที่พฤติกรรมของศัตรูในเกมส์มีความหลากหลายทำให้เกมส์มีความน่าสนใจ แต่ถ้าสร้างความหลากหลายในการเขียนโปแรกรมปัญญาประดิษฐ์อาจทำเกิดความยุ่งยาก แต่ถ้ามีโปรแกรมที่มีพฤติกรรมพื้นฐานหลายอย่างที่นักออกแบบสามารถนำไปใช้ในการสร้างพฤติกรรมต่างๆได้โดยผ่านทางข้อมูลจะทำได้ดีกว่า ถ้าปัญญาประดิษฐ์ใช้วิธีการ data-driven (รับข้อมูลเข้ามาก่อนที่จะประมวลผล) และระบบถูกสร้างโดยคำนึงถึงตัวแปรต่างๆที่อาจจะเกิดขึ้น จะช่วยให้การสร้างความหลากหลายเป็นไปได้ง่ายขึ้น แต่อาจสร้างความสับสนให้กับนักออกแบบได้
11. ทำให้นักออกแบบเข้าถึงข้อมูลได้ง่าย (make the data easily accessible to designers) นักออกแบบควรที่จะสามารถเปลี่ยนแปลงค่าต่างๆขณะที่เกมส์ทำงานได้ เพื่อทำให้เกมส์ทำงานได้ดีขึ้น สมการและสถิติทั้งหลายควรจะถูกเก็บในไฟล์ แทนที่จะเก็บในตัวโปรแกรม และเราสามารถใช้อินเตอร์เฟสในการควบคุมการโต้ตอบ (interaction) ต่างๆได้ เช่นอินเตอร์เฟสของสมการทำให้นักออกแบบใส่ค่าตัวแปรที่ต้องการได้
12. ใส่สมการสถิติในปัญญาประดิษฐ์ (factor stat formulas into AI) มาการทางสถิติควรถูกใส่เข้าไปในตัวแทนในทุกพฤติกรรม เช่น ตัวแทนเคลื่อนที่เร็วแค่ไหน การหาทางเดินฉลาดขนาดไหน ตัวแทนเลือกที่จะสู้แบบจู่โจมหรือป้องกันกับอะไร และตัวแทนใช้โลกของเกมส์อย่างไร

คำถามท้ายบทที่ 1

1. จงออกแบบลำดับชั้นของสถานะแบบง่ายๆโดยบอกว่าสถานะแบบไหนควรจะอยู่ในชั้นต่ำและสถานะแบบไหนควรจะอยู่ในชั้นสูง สำหรับ puzzle เกมส์ที่ผู้เล่นต้องเดินในช่องที่ถูกทำเครื่องหมายไว้ แต่ถ้าผู้เล่นเดินไปในช่องที่ผิด สภาวะแวดล้อมจะเปลี่ยนเป็นอันตรายขึ้น เช่นศัตรูจะอารมณ์เสียมากขึ้นมีการยิงเกิดขึ้น แต่ถ้าผู้เล่นเดินไปในช่องที่ถูกสภาวะแวดล้อมจะดีขึ้น และศัตรูอาจจะไปอยู่ที่อื่นที่ไกลออกไป

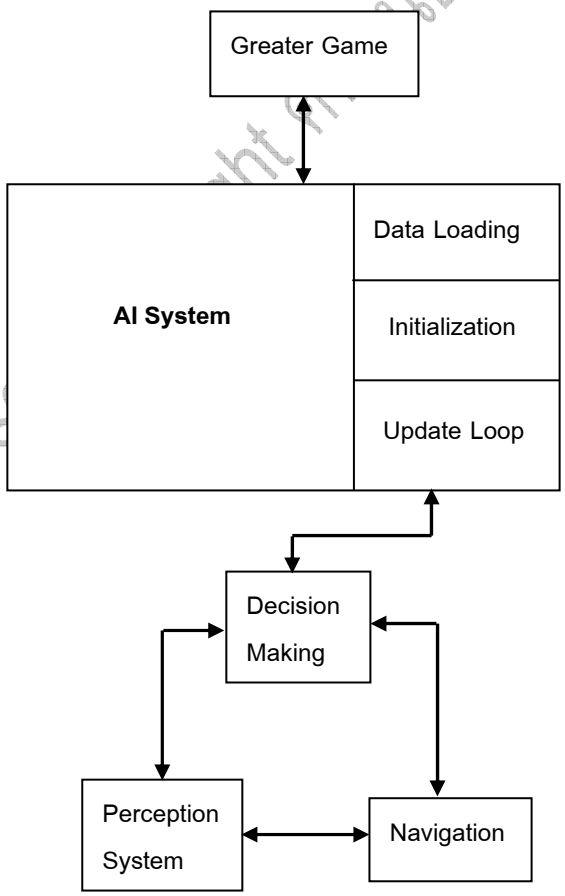
เทคนิคและระบบพิเศษที่มีประโยชน์
Useful Techniques and Specialized Systems

2

ในบทนี้จะกล่าวถึงเทคนิค หรือระบบต่างๆที่จะเป็นประโยชน์ในการใช้ปัญญาประดิษฐ์ในการสร้างเกมส์ ซึ่งระบบเหล่านี้มีมากมายแต่ที่จะกล่าวถึงคือ องค์ประกอบพื้นฐานที่จำเป็น ไฟไนต์สเตทแมชชีน (finite state machine) และการหาเหตุผลโดยประมาณ (approximate reasoning) เพื่อใช้ในการควบคุม

2.1 องค์ประกอบพื้นฐานและการออกแบบของเครื่องประมวลปัญญาประดิษฐ์ (The basic components and design of AI engine)

ก่อนที่จะกล่าวถึง ไฟไนต์สเตทแมชชีน (finite state machine) ในหัวข้อนี้จะกล่าวถึงองค์ประกอบพื้นฐานของเครื่องประมวลปัญญาประดิษฐ์ (AI engine) โดยที่เครื่องประมวลผลนี้ใช้ระบบการตัดสินใจ (decision making) และการอนุมาน (inference) รวมทั้งการเดินทาง (navigation) ไม่ทางใดก็ทางหนึ่ง โดยองค์ปรกอบพื้นฐานของเครื่องประมวลปัญญาประดิษฐ์แสดงในรูปที่ 2.1



รูปที่ 2.1 องค์ประกอบพื้นฐานของเครื่องประมวลปัญญาประดิษฐ์

2.1.1 การตัดสินใจ (Decision making) และการอนุมาน (Inference)

ระบบการตัดสินใจเป็นโครงสร้างพื้นฐานที่จะบ่งบอกถึงชนิดของเครื่องประมวลผลปัญญาประดิษฐ์ที่ต้องการสร้าง และการอนุมานเป็นการสรุปแบบมีลอจิกและมีเหตุผลโดยพิจารณาจากความเป็นจริงหรือสาเหตุที่เป็นจริง ดังนั้นในเกมส์ ศัตรูที่ถูกควบคุมโดยปัญญาประดิษฐ์จะรับข้อมูลเกี่ยวกับโลก และพยายามตัดสินใจอย่างฉลาดและมีเหตุผลในการตอบสนอง

ความแตกต่างระหว่างระบบการตัดสินใจจะบ่งบอกว่าผู้สร้างเกมส์จะเลือกใช้ชนิดระบบใดหรือจะใช้หลายระบบรวมกัน โดยความแตกต่างของระบบแยกออกได้เป็น

1. ชนิดของคำตอบ (Type of solutions) เช่นเป็นเกมส์ชนิดยุทธศาสตร์ (strategic) หรือกลวิธี (tactical) ถ้าเป็นยุทธศาสตร์เกมส์ ต้องการคำตอบที่เป็นระยะยาว มีจุดมุ่งหมายสูง ซึ่งมักจะเกี่ยวข้องกับการมีการกระทำหลายอย่างที่จะต้องทำ ส่วนคำตอบของกลวิธีเกมส์จะเป็นระยะสั้น จุดมุ่งหมายต่ำ และโดยปกติจะเกี่ยวข้องกับการกระทำทางกายภาพ (physical act) หรือความสามารถ เช่นใน เกมส์ที่เป็นแบบ Quake ในส่วนของการ HuntPlayer จะเป็นการกระทำที่มีจุดมุ่งหมายสูง ส่วนการ CircleStrafe เป็นการกระทำที่เป็นเพียงการเคลื่อนไหวในขณะต่อสู้กับผู้เล่น
2. ปฏิกิริยาของตัวแทน (Agent reactivity) ความต้องการให้ตัวแทนในเกมส์มีปฏิกิริยามากน้อยเพียงใด
3. ความเป็นจริงของระบบ (System Realism) แต่ละตัวแทนในเกมส์จะต้องทำในสิ่งที่ควรจะทำอย่างฉลาด และควรจะมีความเป็นมนุษย์อยู่ด้วย นั่นคือตัวแทนจำเป็นที่จะต้องแสดงถึงความอ่อนแอของมนุษย์
4. ประเภท (Genre) เกมส์ต่างประเภทกันต้องการปัญญาประดิษฐ์ที่ไม่เหมือนกัน
5. เนื้อหา (Content) ในเนื้อหาของเกมส์บางครั้งจะมีส่วนที่เป็นการเล่นที่เป็นเคสพิเศษ เช่น Black & White เกมส์ต้องการปัญญาประดิษฐ์พิเศษ สำหรับการสอนสัตว์ให้มีพฤติกรรมโดยการแสดงให้ดูหรือทำในสิ่งที่ถูกให้ดู
6. แพลตฟอร์ม (Platform) ถึงแม้ว่าการพิจารณาในเรื่องนี้ไม่คอยสำคัญนักแต่ในบางครั้งก็ยังต้องพิจารณาว่าเกมส์ที่สร้างจะถูกนำไปเล่นที่ใด
7. ข้อจำกัดของการพัฒนา (Development limitations) ซึ่งข้อจำกัดเหล่านี้รวมถึงจำนวนเงิน จำนวนคน และระยะเวลา ซึ่งในการพัฒนาปัญญาประดิษฐ์รวมทั้งการทดสอบปัญญาประดิษฐ์ ว่ายากเกินไป หรือไม่ มีข้อผิดพลาดใดบ้าง และจะทำให้เครื่องคอมพิวเตอร์หยุดทำงานหรือไม่ ใช้เวลานาน
8. ข้อจำกัดทางด้านความบันเทิง (Entertainment limitations) การสร้างความสมดุลระหว่างความยากของเกมส์และความสมจริง เป็นสิ่งที่เป็นขั้นตอนสุดท้ายของความสำเร็จของเกมส์ นั่นคือ คนควรจะทำในสิ่งที่ตัวแทนทำได้ และไม่ควรจะทำในสิ่งที่ตัวแทนทำไม่ได้ และระดับความยากของเกมส์เป็นอีกสิ่งหนึ่งที่ปัญญาประดิษฐ์เข้ามาช่วยได้ ทั้งนี้รวมทั้งการทำให้ความรู้สึกหรือความตั้งใจของตัวแทนส่งต่อไปที่ผู้เล่น

2.1.2 การจัดการอินพุต และการรับรู้ (Input handler and perception)

การรับรู้ของปัญญาประดิษฐ์คือสิ่งของในสภาวะแวดล้อมที่เราต้องการให้ตัวแทนในเกมส์สนองตอบ ซึ่งทำได้โดยการใช้ระบบการรับรู้กลางซึ่วแบ่งเป็น

1. ชนิดของการรับรู้ (Perception type) ชนิดของอินพุตมีหลายชนิดรวมทั้ง Boolean integer floating point และอื่นๆ ซึ่งอาจรวมถึงการรับรู้สถิต (static perception) เช่น การรับรู้ที่ต้องการสำหรับลอจิกในเกมส์บาสเกตบอลอาจเป็น ความสามารถในการจัดการลูกบาสเกตบอลต้องมากกว่า 75 เป็นต้น
2. การปรับสม่ำเสมอ (Update regularity) การรับรู้ต่างชนิดกันอาจต้องการการปรับตามความจำเป็น เพราะโดยปกติการรับรู้จะไม่ปรับบ่อย หรือการคำนวณใหม่เป็นการสิ้นเปลือง เช่นการตรวจเส้นสายตา (line of sight) ในเกมส์บาสเกตบอล เป็นการตรวจที่ค่อนข้างเปลืองดังนั้นควรตรวจเป็นครั้งคราว
3. เวลาในการตอบสนอง (Reaction time) การใช้เวลาในการตอบสอนงของตัวแทนจะทำในตัวเล่นในเกมส์มีลักษณะคล้ายมนุษย์มากขึ้น
4. เส้นแบ่ง (Threshold) คือค่าที่น้อยที่สุดและมากที่สุดที่ปัญญาประดิษฐ์จะตอบสนอง และค่านี้อาจมีการเปลี่ยนแปลงตามเหตุการณ์ในเกมส์
5. ความสมดุลของโหลด (Load balancing) การทำความสมดุลชองโหลดเพื่อไม่ให้ระบบการรับรู้ใช้ resource มากเกินไป
6. ราคาการคำนวณและภาวะก่อน (Computation cost and precondition) เพื่อเป็นการช่วยในการคำนวณควรจะทำการคำนวณในสิ่งที่เป็นภาวะก่อนไว้ล่วงหน้า ซึ่งการปรับการรับรู้ทำได้ 2 วิธีคือ
    - Polling ซึ่งเป็นการตรวจค่าเฉพาะค่าหนึ่ง หรือทำการคำนวณในวงของเกมส์ (game loop) เช่นคอยตรวจว่าผู้เล่นบาสเกตบอลว่างสำหรับรับลูกหรือไม่ในทุกช่วงเวลา
    - Events เป็นสิ่งที่ตรงข้ามกับ Polling ซึ่งอินพุตจะบอกระบบการรับรู้ว่ามีการเปลี่ยนแปลง และระบบการรับรู้จะจดจำการเปลี่ยนนั้นไว้

2.1.3 การเดินทาง (Navigation)

การเดินทางจากจุด A ไปยังจุด B เกี่ยวข้องสภาวะแวดล้อมที่ใหญ่และมีความซับซ้อนซึ่งมีหลากหลายลักษณะภูมิประเทศ สิ่งกีดขวาง วัตถุเคลื่อนที่ และอื่นๆ ดังนั้นการเดินทางจึงมีงานหลักอยู่ 2 งานคือ การหาเส้นทาง (pathfinding) และ การหลบสิ่งกีดขวาง (obstacle avoidance) วิธีการที่ทำให้ตัวแทนเดินทางจากจุด A ไปยังจุด B มีหลายวิธีการด้วยกัน เช่น Cell decomposition, Navigation Graph เป็นต้น

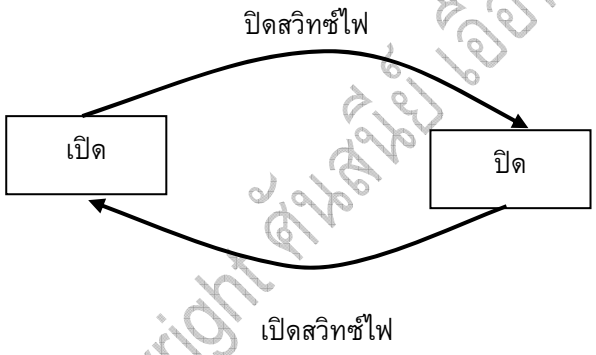
2.2 ไฟไนต์สเตทแมชชีน (Finite State Machine)

ไฟไนต์สเตทแมชชีน (Finite State Machine) หรือ FSM เป็นเครื่องมือที่ใช้สำหรับการโปรแกรมปัญญาประดิษฐ์เพื่อให้ตัวแทนมีลักษณะที่ฉลาดมาเป็นเวลานาน เนื่องจาก ง่ายต่อการ

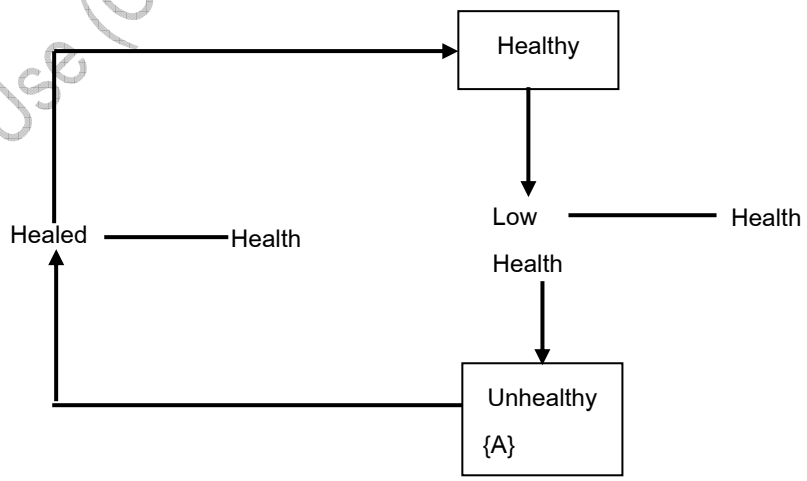
โปรแกรม ง่ายในการ debug มีค่าใช้จ่ายในการคำนวณต่ำ มีลักษณะตามความรู้สึกของมนุษย์ และง่ายต่อการเปลี่ยนแปลง

ไฟไนต์สเตทแมชชีนเป็นอุปกรณ์ทางคณิตศาสตร์ในการแก้ปัญหา ซึ่งนิยามของไฟไนต์สเตทแมชชีนสำหรับนักพัฒนาโปรแกรมปัญญาประดิษฐ์คือ ไฟไนต์สเตทแมชชีนเป็นอุปกรณ์ หรือแบบจำลองของอุปกรณ์ที่มีจำนวนสถานะ (state) ที่จำกัด ซึ่งสามารถทำงานในช่วงเวลาใดก็ได้และสามารถทำให้อินพุตเปลี่ยนจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง หรือสร้างเอาต์พุต หรือมีการกระทำเกิดขึ้น แต่ละสถานะในไฟไนต์สเตทแมชชีนทำงานในเวลาหนึ่งๆได้เพียงแค่ 1 สถานะเท่านั้น ตัวอย่างของไฟไนต์สเตทแมชชีนแสดงในตัวอย่างที่ 1

ตัวอย่างที่ 2.1 การเปิด/ปิดสวิทซ์ไฟเป็นไฟไนต์สเตทแมชชีนอย่างง่ายชนิดหนึ่ง ซึ่งเป็นไฟไนต์สเตทแมชชีนที่มีสถานะ 2 สถานะ นั่นคือ เปิดและปิด การเปลี่ยนสถานะทำได้ด้วยการปิด/เปิดสวิทซ์ไฟ ไฟไนต์สเตทแมชชีนนี้ไม่มีเอาต์พุต หรือการกระทำใดที่เกี่ยวข้องกับสถานะปิด แต่ในสถานะเปิดเกิดแสงสว่างในห้อง ไฟไนต์สเตทแมชชีนนี้แสดงในรูปที่ 2.2 สถานะจะถูกเขียนในสี่เหลี่ยม transition จะถูกเขียนในวงกลม



รูปที่ 2.2 ไฟไนต์สเตทแมชชีนของสวิทซ์ไฟ ■

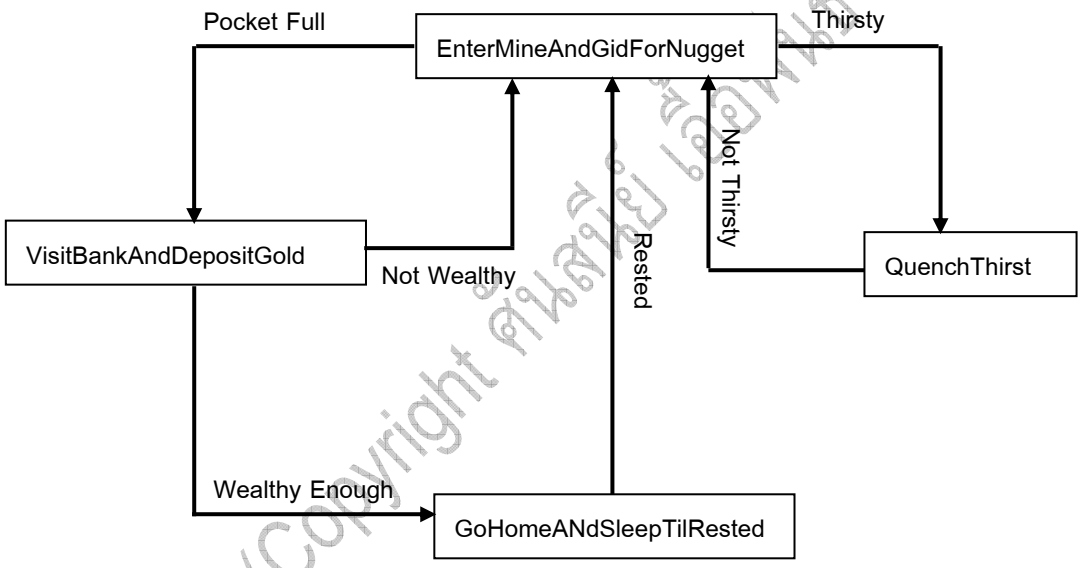


รูปที่ 2.3 สเตทแมชชีนของสิ่งมีชีวิตจำลอง

ตัวอย่างที่ 2.2 สมมุติให้ไฟไนต์สเตทแมชชีนสำหรับสิ่งมีชีวิตจำลองแสดงดังรูปที่ 2.3 ไฟไนต์สเตทแมชชีนนี้เป็นไฟไนต์สเตทแมชชีนที่เกี่ยวข้องกับสุขภาพของสิ่งมีชีวิตโดยที่สถานะโดยปริยายคือมีสุขภาพดี ซึ่งถ้าสิ่งมีชีวิตอยู่ใน state นี้จะไมมี action อะไร ระหว่างที่โปรแกรมทำงาน ถ้าตัวแปร

Health มีค่าต่ำกว่าค่าค่าหนึ่ง เช่น การปรับทำให้สิ่งมีชีวิตเสียหาย ทำให้ Low Health เป็นจริง จะมีการเปลี่ยน state จาก Healthy เป็น Unhealthy state และในไฟไนต์สเตทแมชชีนนี้การที่จะทำให้สิ่งมีชีวิตมีสุขภาพดีขึ้ ด้วยการที่สิ่งมีชีวิตกินอาหาร ซึ่ง action ของการกินอาหาร เขียนเป็น {A} ใน Unhealthy state นั่นเอง ■

ตัวอย่างที่ 2.3 ต้องการสร้างตัวแทนที่ใช้ไฟไนต์สเตทแมชชีนโดยที่ตัวแทนที่ชื่อ Bob อาศัยอยู่ในเมืองที่เป็น old west-style gold mining ซึ่งเกมส์นี้เป็น text-based console application การเปลี่ยนแปลง state หรือเอาต์พุตที่ออกมาจาก state action จะถูกส่งมาที่ console window ในรูปของ text ในเกมส์นี้มีสถานที่ 4 สถานที่คือ เหมืองทอง (gold mine) ธนาคาร (bank) ที่ Bob นำทองมาที่เขาหาเจอมาฝาก บาร์ (saloon) ที่ที่ Bob มาดับกระหาย และบ้านที่ Bob พักผ่อน ซึ่งที่ที่ Bob ไป และ Bob ทำอะไรเมื่อเขาไปถึงที่นั้นถูกกำหนดจาก state ปัจจุบัน และเขาจะเปลี่ยน state เนื่องจาก หิวน้ำ เหนื่อย และจำนวนทองที่เขาหาได้



รูปที่ 2.4 ไฟไนต์สเตท"ดอะแกรมของนักขุดเหมือง Bob

จะเห็นว่าในการที่ Bob เปลี่ยนสถานที่คือเขาเปลี่ยน state นั่นเอง โดยที่เหตุการณ์ต่างๆที่เกิด เกิดจาก action ในแต่ละ state นั่นเอง ดังนั้นในกรณีนี้มี 4 state นั่นคือ

1. EnterMineAndDigForNugget ถ้านักขุดเหมืองไม่ได้อยู่ในเหมืองทอง เขาจะเปลี่ยน state แต่ถ้าเขาอยู่ในเหมืองทอง เขาจะขุดทอง และเมื่อมีทองเต็มกระเป๋า เขาจะเปลี่ยนไป VisitBankAndDepositGold state และในขณะที่ขุดถ้าเขาหิวน้ำ เขาจถหยุดและไปยัง QuenchThrist state
2. VisitBakeAndDepositGold ใน state นี้นักขุดเหมืองจะเดินไปที่ธนาคารและฝากทองที่เขาถือมา ถ้าเขาคิดว่าเขารวยพอแล้วเขาจะไปยัง GoHomeAndSleepTilRestes state ถ้าไม่อย่างนั้นเขาจะไปยัง EnterMineAndDigForNugget state
3. GoHomeAndSleepTilRested ใน state นี้นักขุดเหมืองจะกลับไปบ้านและนอนหลับจนกว่าระดับความเหนื่อยลดต่ำลงกว่าค่าที่ยอมรับได้ และเขาจะกลับไปยัง EnterMineAndDigForNugget

4. QuenchThrist ถ้าในครั้งใดที่นักขุดเหมืองรู้สึกหิวน้ำ เขาจะมายัง state นี้และไปบาร์เพื่อสั่ง whiskey เมื่อความกระหายน้ำหายไป เขาจะกลับไปยัง EnterMineAndDigForNugget

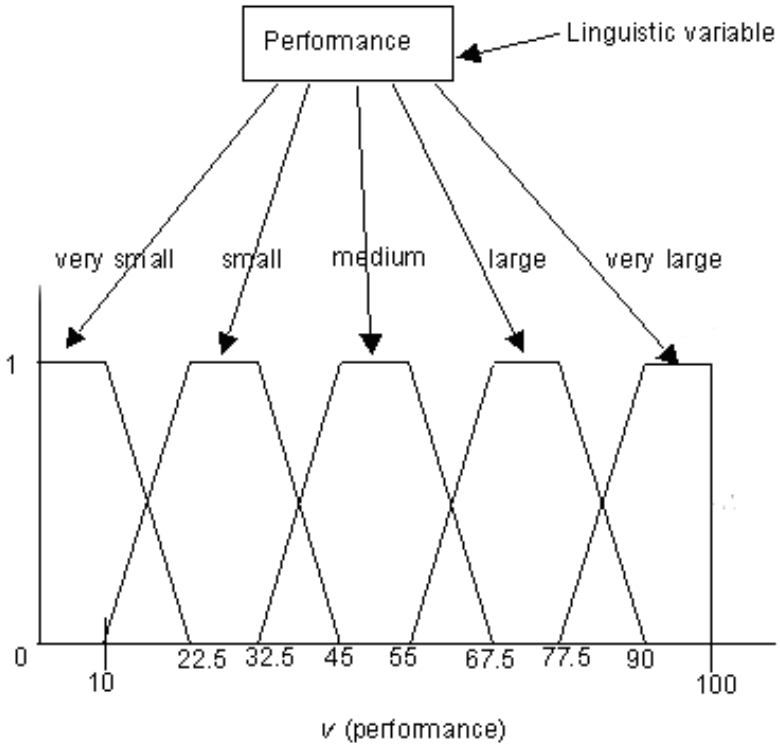
รูปที่ 2.4 แสดงไฟไนต์สเตทแมชชีนของปัญหานี้ และในการเขียนโปรแกรมนอกจากเขียน state class แล้วควรจะมี class ที่เป็น base class (BaseGameEntity) เพื่อเป็น class ที่เก็บเบอร์ Id ของสมาชิกซึ่งต้องมีแค่ค่าเดียวต่อ 1 สมาชิก และมี Update ฟังก์ชันที่ถูกเรียกโดย subclass ทุก subclass ในกรณีที่มีการเปลี่ยน state นอกจากนี้ยังต้องมี Miner class ซึ่งเป็น class ที่เป็นคลาสสืบต่อ (derived class) จาก BaseGameEntity ซึ่งเป็น class ที่เป็บข้อมูลของสมาชิกที่เป็นลักษณะต่างๆ ที่ Miner มีเช่นระดับสุขภาพ ระดับความเหนื่อย ตำแหน่ง และอื่นๆ และเพื่อเป็นความสะดวกในการแก้ไขของแต่ละ state ควรจะมี Enter และ Exit action ในแต่ละ state ด้วย ซึ่งaction นี้จะถูกเรียกเพียง 1 ครั้งในการเริ่ม state และออกจาก state ■

2.3 การหาเหตุผลโดยประมาณ (approximate reasoning) เพื่อใช้ในการควบคุม

การหาเหตุผลโดยประมาณ (approximate reasoning) ทำให้ตัวแทนในเกมส์ปัญญาประดิษฐ์สามารถทำในสิ่งที่คล้ายมนุษย์ทำได้ ซึ่งโดยปกติจะเป็นสิ่งที่ทำอะไรได้มากกว่า if-then-else ปกติ นักพัฒนาเกมส์สามารถใช้ โปรแกรม Free Fuzzy Logic Library (FFLL) ซึ่งเป็นแหล่งข้อมูลเปิด (open-source) ของฟัซซีลอจิกคลาสไลเบอร์รี (fuzzy logic class library) ได้ ในหัวข้อต่อไปนี้จะกล่าวถึงฟัซซีลอจิกอย่างย่อ

2.3.1 การหาเหตุผลโดยประมาณ (approximate reasoning)

การหาเหตุผลโดยประมาณ (approximate reasoning) ที่จะกล่าวถึงนี้เป็นวิธีการของ Mamdani (Mamdani model) โดยที่ ระบบมีมากกว่า 1 อินพุต (X_1, X_2,... X_n) และแต่ละอินพุตถูกนิยามตัวแปรภาษา (linguistic variable) (x_1, x_2,... x_n) และพจน์ภาษา (linguistic term) $\boldsymbol{T}(x_i)$ ของตัวแปรภาษา x_i ในเซตสากล X_i สำหรับ $1 \le i \le n$ ไว้แล้ว ในขณะเดียวกันเอาพุต Y ก็ถูกนิยามตัวแปรภาษา (linguistic variable) yและพจน์ภาษา (linguistic term) $\boldsymbol{T}(y)$ ของตัวแปรภาษา y ในเซตสากล Y ไว้แล้วเช่นกัน ตัวอย่างของตัวแปรภาษาแสดงในรูปที่ 2.5 ซึ่งมีตัวแปรภาษาชื่อหรือ x คือ 'performance' และ $\boldsymbol{T}(x)$ หรือพจน์ภาษาคือ 'very small', 'small', 'medium', 'large' และ 'very large' และแต่ละพจน์ภาษาถูกส่งทอดไปที่ฟัซซีเซตทั้ง 5 ในรูปที่ 2.5 โดยที่มีเซตสากลเป็น $[0,100]$ ซึ่งพจน์ภาษาเหล่านี้คือฟัซซีเซต (Fuzzy Set) ที่มีฟังก์ชันความเป็นสมาชิก (membership function) ซึ่งเป็นฟังก์ในการบอกถึงระดับความเป็นสมาชิกของแต่ละสมาชิกในฟัซซีเซตนั้น



รูปที่ 2.5 ตัวอย่างของตัวแปรภาษา

กฎสำหรับการหาเหตุผลโดยประมาณมีลักษณะดังนี้

If ξ_1 is $\boldsymbol{A}^{(1)}$, และ ξ_2 is $\boldsymbol{A}^{(2)}$ และ ... และ ξ_n is $\boldsymbol{A}^{(n)}$ then η is $\boldsymbol{B}$

โดยที่ $\boldsymbol{A}^{(1)}$, $\boldsymbol{A}^{(2)}$ และ ... และ $\boldsymbol{A}^{(n)}$ เป็นพจน์ภาษาใน $\boldsymbol{T}(x_i)$ สำหรับ $1 \le i \le n$ และ $\boldsymbol{B}$ พจน์ภาษาใน $\boldsymbol{T}(y)$ นั่นเองและถ้ามีกฎมากกว่า 1 กฎและให้

$\boldsymbol{T}(x_1)$ ประกอบด้วย $\boldsymbol{A}_1^{(1)}$, $\boldsymbol{A}_2^{(1)}$,... ,$\boldsymbol{A}_{\mathrm{N1}}^{(1)}$

$\boldsymbol{T}(x_2)$ ประกอบด้วย $\boldsymbol{A}_1^{(2)}$, $\boldsymbol{A}_2^{(2)}$,... ,$\boldsymbol{A}_{\mathrm{N2}}^{(2)}$

$\vdots$

$\boldsymbol{T}(x_n)$ ประกอบด้วย $\boldsymbol{A}_1^{(n)}$, $\boldsymbol{A}_2^{(n)}$,... ,$\boldsymbol{A}_{\mathrm{N}n}^{(n)}$ (2.1)

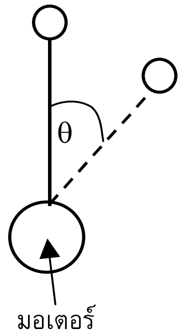
และ $\boldsymbol{T}(y)$ ประกอบด้วย $\boldsymbol{B}_1$, $\boldsymbol{B}_2$,... ,$\boldsymbol{B}_{\mathrm{N0}}$ (2.2)

จะเขียนกฎได้ว่า

กฎ j: If ξ_1 is $\boldsymbol{A}_{i1,j}^{(1)}$, และ ξ_2 is $\boldsymbol{A}_{i2,j}^{(2)}$ และ ... และ ξ_n is $\boldsymbol{A}_{in,j}^{(n)}$ then η is $\boldsymbol{B}_{i,j}$ (2.3)

โดยที่ $i1 \in \{1,2,...,\mathrm{N1}\}$, $i2 \in \{1,2,...,\mathrm{N2}\}$,..., $in \in \{1,2,...,\mathrm{N}n\}$ และ $i \in \{1,2,...,\mathrm{N0}\}$ นั่นเอง ตัวอย่าง 2.4 แสดงการหาเหตุผลโดยประมาณ

ตัวอย่างที่ 2.4 ระบบควบคุมฟัซซีที่ใช้ในการควบคุมเสถียรภาพของลูกตุ้มกลับหัว (inverted pendulum) แสดงในรูปที่ 2.6 ซึ่งสิ่งที่ระบบนี้ต้องทำคือพยายามทำให้ลูกตุ้มอยู่ตรงกลางนั่นคือทำให้ลูกตุ้มตีกลับในทิศทางตรงข้ามโดยการให้กระแสไฟฟ้ากับมอเตอร์ และถ้าการหมุนกลับเกินไปในทิศตรงข้ามก็ต้องให้กระแสไฟฟ้าในทิศตรงข้าม อีกเพื่อให้ลูกตุ้มตีกลับอีกครั้ง



รูปที่ 2.6 ลูกตุ้มกลับหัว

ลูกตุ้มเคลื่อนที่ไปในทิศทางที่ทำมุมกับเส้นกึ่งกลางเป็นมุม θ ด้วยความเร็ว $\Delta\theta$ ซึ่งทั้งมุมและความเร็วน่าจะเป็นอินพุตให้กับระบบควบคุมฟัซซี และระบบฟัซซีจะให้เอาพุตคือกระแสไฟฟ้า v และให้อินพุตทั้งสอง และเอาพุตมีค่าพจน์ภาษาเป็น negative large (NL) negative medium (NM) negative small (NS) zero (ZE) positive small (PS) positive medium (PM) และ positive large (PL) โดยที่ negative หมายถึงลูกตุ้มเคลื่อนที่ทวนเข็มนาฬิกา หรือให้กระแสไฟฟ้าในทิศไปทางซ้าย ในขณะที่เคลื่อนที่ตามเข็มนาฬิกาหรือให้กระแสไฟฟ้าในทิศไปทางขวาถ้าเป็น positive ดังนั้นกฎที่ j จะมีลักษณะเป็น

กฎ j: If θ is $\boldsymbol{A}_j$, และ $\Delta\theta$ is $\boldsymbol{B}_j$ then v is $\boldsymbol{C}_j$

เช่น

If θ is NL และ $\Delta\theta$ is NL, then v is PL

หรือ If θ is ZE และ $\Delta\theta$ is ZE, then v is ZE

ซึ่งโดยปกติเราสามารถเขียนกฎเหล่านี้ให้อยู่ในลักษณะทูเพิลได้เช่น $(\theta,\ \Delta\theta;\ v)$ ซึ่งในตัวอย่างของกฎทั้งสองเขียนได้เป็น (NL,NL;PL) สำหรับกฎแรก และ (ZE,ZE;ZE) และเราสามารถเขียนกฎโดยใช้เมตริกซ์ได้เช่น สมมุติให้มี 13 กฎ จะได้เมตริกซ์เป็น

$\Delta\theta$ \ θ	NL	NM	NS	ZE	PS	PM	PL
NL				PL			
NM				PM			
NS				PS			
ZE	PL	PM	PS	ZE	NS	NM	NL
PS				NS			
PM				NM			
PL				NL			

โดยที่ค่าที่อยู่ในเมตริกซ์คือค่าเอาพุตนั่นเอง เช่นกฎ (NL,ZE;PL) คือกฎที่อยู่ที่คอลัมภ์ที่ 1 แถวที่ 4 นั่นเอง จะเห็นได้ว่าในระบบนี้มีกฎที่เป็นไปได้ทั้งหมด 343 กฎคือแต่ละกฎให้ค่าพจน์ภาษาของอินพุต 2 อินพุตและ 1 เอาพุตโดยที่แต่ละอินพุตมี 7 พจน์ภาษาและเอาพุตมี 7 พจน์ภาษาเช่นกัน ซึ่งจะได้ 7 × 7 × 7 แต่ในการใช้งานจริงบางกฎอาจจะไม่ถูกใช้เลยดังนั้นกฎที่ใช้งานจริงจึงมีน้อยกว่า 343 กฎ ■

จากตัวอย่าง 2.4 สามารถบอกได้ว่ามีกฎที่เป็นไปได้ทั้งหมดเท่ากับ N1×N2×...×Nn×N0 แต่ในการใช้งานจริงมีเพียงบางกฎเท่านั้นที่นำไปใช้ได้จริง

ในการหาเหตุผล จะเริ่มจากการหาระดับความเข้ากันได้ของแต่ละอินพุต (x_i โดยที่ $i \in \{1,2,...,n\}$) กับพจน์ภาษาในกฎนั้น และเนื่องจากลักษณะของข้อตั้ง (premise) ของกฎต้องการให้ทุกอินพุตเป็นไปตามพจน์ภาษา ดังนั้นค่าความเป็นสมาชิกของแต่ละอินพุตในแต่ละพจน์ภาษาจะถูกรวมกันในลักษณะของตัวเชื่อม conjunction นั่นคือ ที่กฎ j

$$\alpha_j = \min\{ \boldsymbol{A}_{i1,j}^{(1)}(x_1), \boldsymbol{A}_{i2,j}^{(2)}(x_2),..., \boldsymbol{A}_{in,j}^{(n)}(x_n)\} \quad (2.4)$$

และเอาพุตของกฎ j เป็นฟัซซีเซตที่เกิดจากการตัด (cut off) พจน์ภาษา $\boldsymbol{B}_{i,j}$ ด้วย α_j หรืออาจจะพูดได้ว่า

$$\boldsymbol{OUT}_{x_1,x_2,...,x_n}^{(j)}(y) = \min\left[\boldsymbol{A}_{i1,j}^{(1)}(x_1), \boldsymbol{A}_{i2,j}^{(2)}(x_2),..., \boldsymbol{A}_{in,j}^{(n)}(x_n), \boldsymbol{B}_{i,j}(y)\right] \quad (2.5)$$

และเมื่อได้เอาพุตของแต่ละกฎแล้ว ฟัซซีเอาพุตจากทุกกฎจะถูกรวมกันโดยการหายูเนียน (ซึ่งเป็นการหายูเนียนมาตรฐานซึ่งจะได้ฟัซซีเอาพุตรวม (***OUT***) สมมุติให้มีกฎทั้งหมด k กฎ

$$\boldsymbol{OUT}_{x_1,x_2,...,x_n}(y) = \max_{j\in\{1,2,...,k\}} \min\left[\boldsymbol{A}_{i1,j}^{(1)}(x_1), \boldsymbol{A}_{i2,j}^{(2)}(x_2),..., \boldsymbol{A}_{in,j}^{(n)}(x_n), \boldsymbol{B}_{i,j}(y)\right] \quad (2.6)$$

โดยที่ฟัซซียูเนียนมาตรฐานของฟัซซีเซต $\boldsymbol{A}$ และ $\boldsymbol{B}$ สามารถเขียนฟังก์ชันสมาชิกที่แสดงถึงค่าความเป็นสมาชิกของทุก $x \in X$ ในฟัซซียูเนียนได้เป็น

$$(\boldsymbol{A} \cup \boldsymbol{B})(x) = \max[\boldsymbol{A}(x), \boldsymbol{B}(x)] \quad (2.7)$$

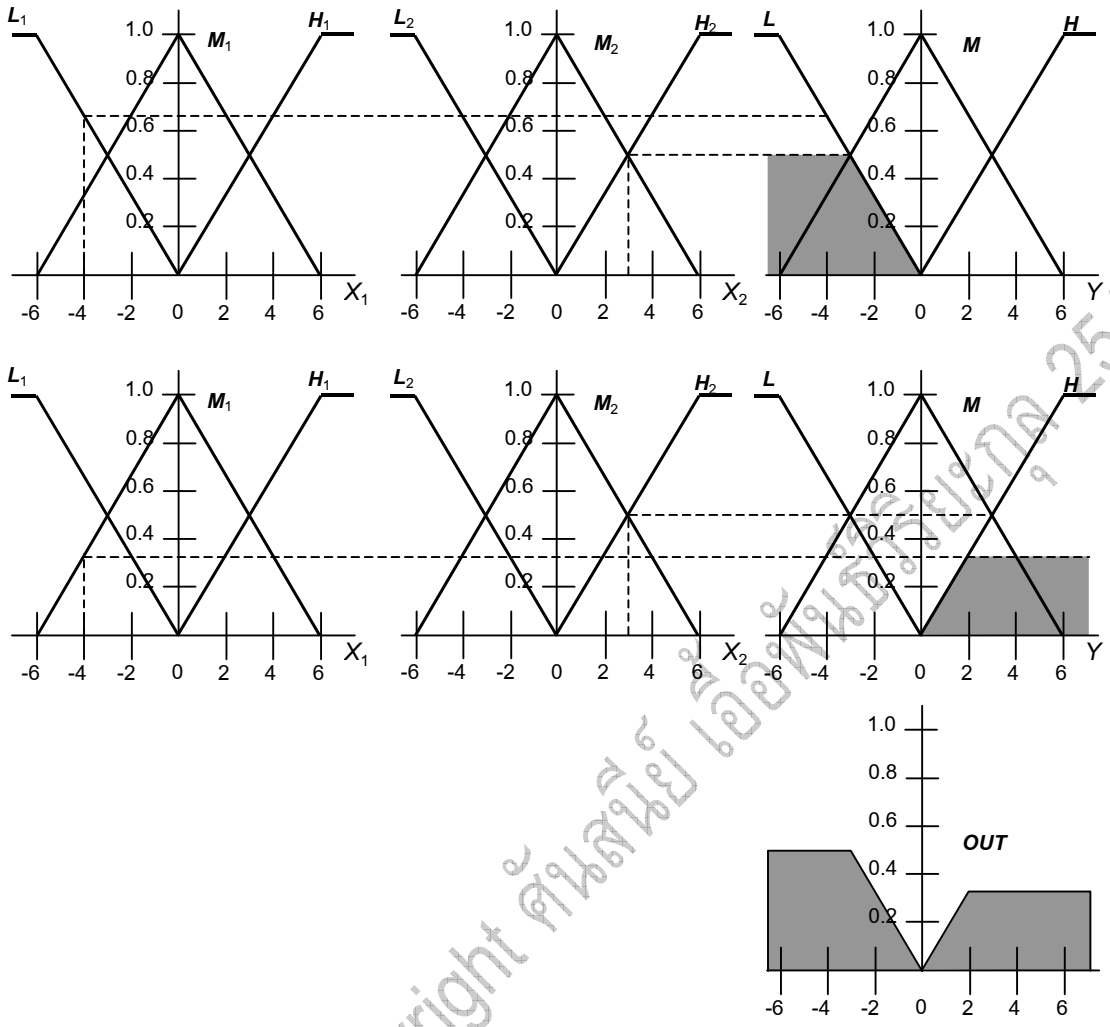
ดังนั้นเอาพุตที่ออกไปที่ controlled system จะเป็นค่าที่ได้จากการแปลงฟัซซีเอาพุตรวมเป็นตัวเลขโดย defuzzification interface ซึ่งการทำการแปลงฟัซซีเซตให้เป็นตัวเลข (defuzzification) สามารถทำได้หลายแต่วิธีที่นิยมมากที่สุดคือการหา Center of Area (Centroid) โดยให้เลือก *de_y* ที่เป็นค่า centroid ของฟัซซีเซตนั้นคือ ถ้าฟัซซีเซต $\boldsymbol{B}'$ ที่มีฟังก์ชันสมาชิกเป็นฟังก์ชันดิสครีต (discrete function) ค่า *de_y* คือ

$$de\_y = \frac{\sum_k \boldsymbol{B}'(y_k) y_k}{\sum_k \boldsymbol{B}'(y_k)} \quad (2.8)$$

ถ้าเป็นฟัซซีเซตมีฟังก์ชันสมาชิกเป็นฟังก์ชันต่อเนื่อง (continuous function) ค่า *de_y* คือ

$$de\_y = \frac{\int \boldsymbol{B}'(y) y dy}{\int \boldsymbol{B}'(y) dy} \quad (2.9)$$

ตัวอย่างที่ 2.5 แสดงการทำงานของวิธีการ Mamdani



รูปที่ 2.7 ระบบควบคุมฟัซซีโดยใช้วิธี Mamdani

ตัวอย่างที่ 2.5 สมมุติให้ระบบมีกฎ 2 กฎ โดยที่แต่ละกฎจะมีอินพุต 2 อินพุต และแต่ละอินพุตในแต่ละกฎมีพจน์ภาษาดังรูปที่ 2.7 โดยที่มีกฎดังนี้คือ

กฎ 1: If x_1 is $\boldsymbol{L}_1$ และ x_2 is $\boldsymbol{H}_2$, then y is $\boldsymbol{L}$

กฎ 2: If x_1 is $\boldsymbol{M}_1$ และ x_2 is $\boldsymbol{M}_2$, then y is $\boldsymbol{H}$

และสมมุติให้อินพุตที่เข้ามาที่ fuzzification interface มีค่าเป็น –4 และ 3 จะเห็นได้ว่า

$$\alpha_1 = \min\big(\boldsymbol{L}_1(-4), \boldsymbol{H}_2(3)\big) = \min(0.67, 0.5) = 0.5$$

และ $$\alpha_2 = \min\big(\boldsymbol{M}_1(-4), \boldsymbol{M}_2(3)\big) = \min(0.33, 0.5) = 0.5$$

ฟัซซีเอาพุตของกฎที่ 1 และ 2 และฟัซซีเอาพุตรวม (***OUT***) แสดงในรูปที่ 2.7 เช่นกัน สมมุติว่าในส่วนของ (defuzzification interface) ใช้วิธีการแปลงโดยใช้ centroid จะได้ค่า output เท่ากับ 1.2 ■

นอกเหนือจากระบบที่กล่าวถึงในบทนี้ยังคงมีระบบอื่นๆที่จะอำนวยความสะดวกในการเขียนโปรแกรมเกมส์ที่ใช้ปัญญาประดิษฐ์ ตัวอย่างเช่นระบบข่าวสาร (message-based system) หรือแม้กระทั่งระบบที่ช่วยในการเขียนเกมส์ 3 มิติ เป็นต้น

คำถามท้ายบทที่ 2

1. ในเกมส์ที่เป็นแบบ Quake จงอธิบายว่า HuntPlayer และ CircleStrafe ส่วนใดเป็นกลยุทธเกมส์และส่วนใดเป็นกลวิธีเกมส์

2. จงออกแบบ finite state machine สำหรับ ลิฟท์ (elevator) ซึ่งลิฟท์ตัวนี้จะเคลื่อนที่ระหว่าง 3 ชั้นโดยที่มี Input/output ดังข้างล่าง

Inputs:

- button1, button2, button3: จะ asserted เมื่อปุ่มถูกกดจากข้างในหรือข้างนอกลิฟท์และจะอยู่อย่างนั้นจนกว่าสัญญาณ acknowledge จะ asserted
- floor1, floor2, floor3: จะ asserted เมื่อ ลิฟท์ อยู่ที่ชั้นนั้นๆ
- start: start timer จับเวลาของการเปิดประตู

Outputs:

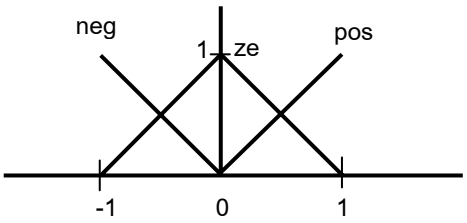
- open: เมื่อ asserted ประตูจะเปิดและเมื่อ deasserted ประตูจะปิด (ประตูจะเปิดแค่ 5 วินาที โดยใช้ timer จับเวลา)
- up, down: ลิฟท์เคลื่อนที่ขึ้นลง เมื่อลิฟท์อยู่กับที่ทั้งสองสัญญาณต้อง deasserted แต่ทั้งสองสัญญาณไม่ควรจะ asserted พร้อมกัน
- ack1, ack2, ack3: acknowledge ว่าลิฟท์ได้ไปถึงชั้นที่ต้องการไปแล้ว ทำให้ button1 หรือ button 2 deasserted
- expire: บอกว่าครบ 5 วินาทีแล้ว

3. จงเขียนโปรแกรมของเกมส์นักขุดเหมืองจากตัวอย่างที่ 2.3 โดยตัวอย่างของเอาต์พุตแสดงดังรูปที่ 2.8

```
Miner Bob: Pickin' up a nugget
Miner Bob: Pickin' up a nugget
Miner Bob: Ah'm leavin' the gold mine with mah pockets full o' sweet gold
Miner Bob: Goin' to the bank. Yes siree
Miner Bob: Depositin' gold. Total saving now: 3
Miner Bob: Leavin' the bank
                ⋮
```

รูปที่ 2.8 ตัวอย่างของเกมส์นักขุดทอง

4. ให้ระบบควบคุมฟัซซีใช้ Mamdani โดยมี input 2 ตัวแปรคือ $\zeta_1 \in X_1$ = [-1,1] และ $\zeta_2 \in X_2$ = [-1,1] และ output เป็น $\eta \in Y$ = [-1,1] โดยที่ Membership function ของตัวแปรทั้งสามเป็นดังรูปที่ 2.9



รูปที่ 2.9

โดยที่มีกฎดังนี้

ζ_2 \ ζ_1	neg	ze	pos
neg	neg		ze
ze		ze	
pos	ze		pos

จงหาว่า output ที่ออกจากระบบมีค่าเป็นเท่าไรถ้าอินพุตเป็น (0.25, 0.5)

การหาเส้นทางและการเคลื่อนไหว
Pathfinding and Movement

3

ในบทนี้จะกล่าวถึงการเคลื่อนไหวที่เกี่ยวข้องกับการที่ตัวแทนเคลื่อนที่จากที่หนึ่งไปยังอีกที่หนึ่งโดยอาศัยการหาเส้นทางในลักษณะต่างๆ

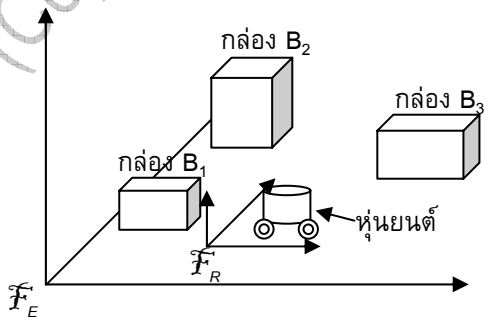
ก่อนที่จะกล่าวถึงการเส้นทางในการเคลื่อนที่ควรจะทำความรู้จักกับกราฟก่อน กราฟ G ถูกนิยามโดยเซตของ node หรือ จุด N ที่เชื่อมต่อกันด้วยเซตของเส้นเชื่อม E ซึ่งถูกเขียนเป็น

$$G = \{N, E\} \tag{3.1}$$

ถ้าแต่ละ node ในกราฟมี label ที่มีค่า 0 ถึง $N - 1$ เราสามารถอ้างถึงเส้นเชื่อมที่เชื่อม node ด้วยการอ้างถึง node นั้นๆเช่น 3 — 5 หรือ 19 — 7 เส้นเชื่อมในกราฟโดยส่วนมากจะมีน้ำหนัก (มีข้อมูลของค่าของการเคลื่อนที่จาก 1 node ไปยังอีก node หนึ่ง

นักเขียนโปรแกรมโดยส่วนใหญ่จะรู้จัก tree data structure ซึ่งที่จริงแล้ว tree เป็นเซตย่อย (subset) ของกราฟ ซึ่งคือเซตที่ประกอบด้วยกราฟที่ไม่มี cycle (acyclic graph) นั่นเอง

ในการหาเส้นทางโดยปกติแบบจำลองที่อธิบายวัตถุที่เคลื่อนไหวได้และวัตถุที่อยู่โดยรอบเป็นแบบจำลองแบบเรขาคณิต และโดยปกติการอธิบายแบบนี้จะใช้แนวความคิดของ space ของทุก configuration มาอธิบาย โดยที่พื้นที่ทั้งหมดถูกเรียกเป็น configuration space ซึ่งเป็นเซตของ configuration ทุก configuration ของวัตถุที่เคลื่อนไหวได้ โดยที่ configuration ของวัตถุที่เคลื่อนไหวได้คือ คุณลักษณะของตำแหน่งและทิศทางของคาร์ทีเซียนเฟรมอ้างอิง (Cartesian frame of reference) ($\mathcal{F}_R$) ในคาร์ทีเซียนเฟรมอ้างอิงของวัตถุที่ไม่เคลื่อนไหว ($\mathcal{F}_E$) รูปที่ 3.1 แสดงคาร์ทีเซียนเฟรมอ้างอิงของวัตถุทั้งสองชนิด



รูปที่ 3.1 คาร์ทีเซียนเฟรมอ้างอิงของกล่อง ($\mathcal{F}_E$) และของหุ่นยนต์ ($\mathcal{F}_R$)

Configuration space สามารถแบ่งออกได้เป็น 2 space นั่นคือ space ที่วัตถุสามารถเคลื่อนที่ผ่านได้ซึ่งถูกเรียกว่า free space และ space ที่วัตถุไม่สามารถผ่านได้ถูกเรียกว่า occupied space ดังนั้นเส้นทางที่ต้องการหาโดยปกติจะอยู่ใน free space นั่นเอง

3.1 การแบ่งเซลย่อย (Cell Decomposition)

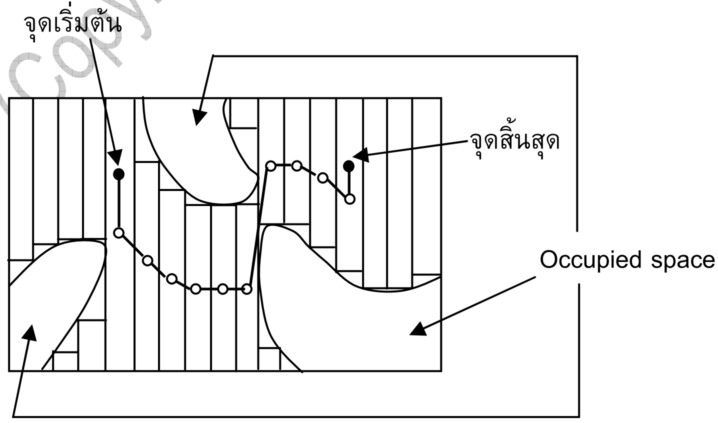
วิธีการนี้เป็นการแบ่ง free space เป็นพื้นที่ย่อยที่เชื่อมต่อกัน และพื้นที่ย่อยเหล่านี้มีเพียงจำนวนจำกัด ซึ่งพื้นที่เหล่านี้ถูกเรียกว่า cell นั่นเอง คุณสมบัติสคัญของพื้นที่ย่อยเหล่านี้คือ ในการหาเส้นทางในพื้นที่ย่อยเดียวกันเป็นการหาส้นทางที่ง่าย เช่นเดินทาเป็นเส้นตรงนั่นเอง และการหาเส้นทางจะกลายเป็นการหากราฟ (graph search) ซึ่งจะกล่าวถึงในหัวข้อต่อไป

อัลกอริทึมของการแบ่งเซลย่อยที่ง่ายที่สุดเป็นดังรูปที่ 3.2 แต่วิธีการนี้จะทำได้ยากขึ้นถ้า configuration space มีลักษณะที่ซับซ้อนและมีขอบที่โค้ง ดังนั้นโดยปกติในการแบ่งพื้นที่ย่อยจะทำโดยการประมาณให้สิ่งกีดขวางมีลักษณะเป็นสี่เหลี่ยมเพื่อให้ง่ายในการแบ่งพื้นที่ย่อน และตัวอย่างของการหาเส้นทางโดยการใช้การแบ่งพื้นที่ย่อยแสดงในตัวอย่างที่ 3.1

1. แบ่ง free space เป็นพื้นที่ย่อยที่ติดกัน (cell)
2. กำหนดว่า cell ไหนต่อกับ cell ไหน และสร้าง adjacency graph โดยที่จุดแต่ละจุดเป็น cell และเส้นเชื่อมเป็นเส้นที่เชื่อม cell ที่ติดกัน
3. กำหนดว่าจุดเริ่มต้น และจุดสิ้นสุดของการเดินทางอยู่ใน cell ใด และหาเส้นทางใน adjacency graph ระหว่าง cell ทั้งสอง
4. จากลำดับของ cell ที่ได้ในข้อ 3 คำนวณหาส้นทางภายในแต่ละ cell จากตำแหน่งที่ขอบที่ติดกับ cell ก่อนหน้า (ถ้าใช้สี่เหลี่ยมเป็นพื้นที่ย่อย ให้เชื่อมจุด 2 จุดใน cell ด้วยเส้นตรง

รูปที่ 3.2 อัลกอริทึมของการแบ่งพื้นที่ย่อยอย่างง่าย

ตัวอย่างที่ 3.1 สมมุติให้ configuration space สำหรับการหาเส้นทางเป็นดังรูปที่ 3.2 และสมมุติว่าเราเลือกใช้การแบ่งพื้นที่ย่อยให้พื้นที่ย่อยมีลักษณะเป็นสี่เหลี่ยม



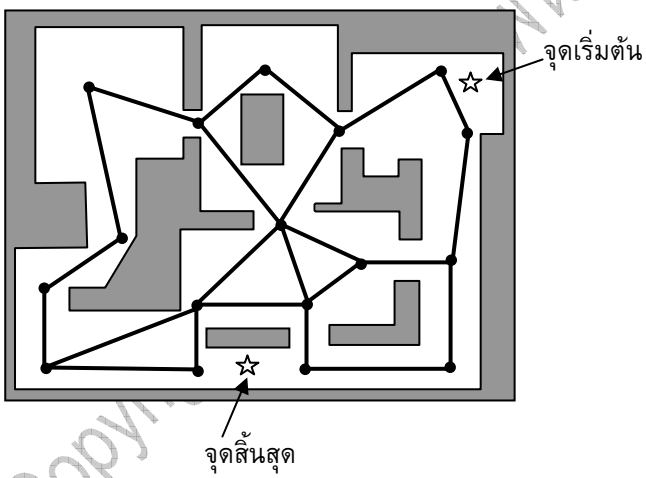
รูปที่ 3.2 การแบ่งพื้นที่ย่อย

เส้นทางจะถูกสร้างโดยการเชื่อมจุดกึ่งกลางของพื้นที่ย่อยที่เป็นพื้นที่ย่อยในเส้นทางที่ได้จาก adjacency graph ซึ่งมีจุดใน adjacency graph เป็นจุดกึ่งกลางของแต่ละ cell นั่นเอง และที่จุดเริ่มต้นต้องเดินทางจากจุดเริ่มต้นมาที่จุดกึ่งกลางในพื้นที่ย่อยที่มีตุดเริ่มต้นก่อน และในพื้นที่ย่อยที่มีจุดสิ้นสุดก็ต้องเดินทางจากจุดกึ่งกลางของพื้นที่ย่อยนั้นไปยังจุดสิ้นสุดเช่นกัน ■

3.2 การหากราฟ (Graph Search)

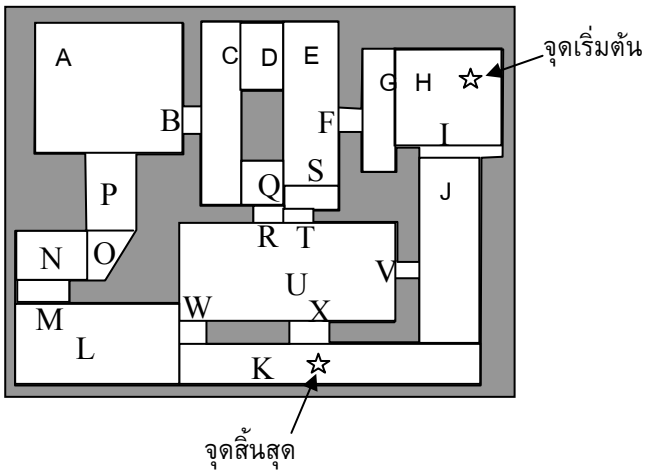
3.2.1 การสร้าง navgraph และ navmesh

การหาเส้นทางใน adjacency graph ในหัวข้อที่แล้วหรือแม้กระทั่งการหาเส้นทางอิสระ (free path) ใน configuration space สามารถใช้เทคนิคการหากราฟได้ โดยที่สร้างกราฟจาก navigation graph หรือ navgraph ซึ่งเป็นกราฟที่เป็นเหมือนนามธรรมของตำแหน่งต่างๆในสภาวะแวดล้อมของเกมส์ ที่ตัวแทนสามารถเดินทางผ่านได้โดยใช้เส้นเชื่อมต่อระหว่างจุดเหล่านี้ แต่ละจุดใน navgraph แทนตำแหน่งของพื้นที่สำคัญภายในสภาวะแวดล้อมนั้น และแต่ละเส้นเชื่อมแทนการเชื่อมโยงระหว่างจุดเหล่านี้ ยิ่งไปกว่านั้นแต่ละเส้นเชื่อมมีราคาติดอยู่ด้วย ซึ่งโดยปกติคือระยะทางระหว่างจุดที่เส้นเชื่อมนี้เชื่อมถึง รูปที่ 3.3 แสดง navigation graph ที่สร้างขึ้นมาในสภาวะแวดล้อมที่เป็นกำแพงกั้น โดยปกติเราสามารถสร้าง navigation graph โดยการเลือกจุดในสภาวะแวดล้อมเองด้วยมือ ซึ่งจุดเหล่านี้ถูกเรียกว่า point of visibility (POV) ที่ปกติจะเป็นจุดที่สำคัญในสภาวะแวดล้อมนั้น นั่นคือแต่ละจุดมีเส้นสายตา (line of sight) ไปยังจุดอื่นอย่างน้อย 1 จุด



รูปที่ 3.3 ตัวอย่างของ navgraph โดยที่ POV ถูกสร้างด้วยมือ

โดยปกติตัวแทนสามารเดินทางออกนอกเส้นเชื่อมได้โดยเดินทางใน free space แต่ตัวแทนใช้ navgraph ในการหาเส้นทางเท่านั้น



รูปที่ 3.4 navmesh ของสภาวะแวดล้อมเดียวกับรูป 3.3

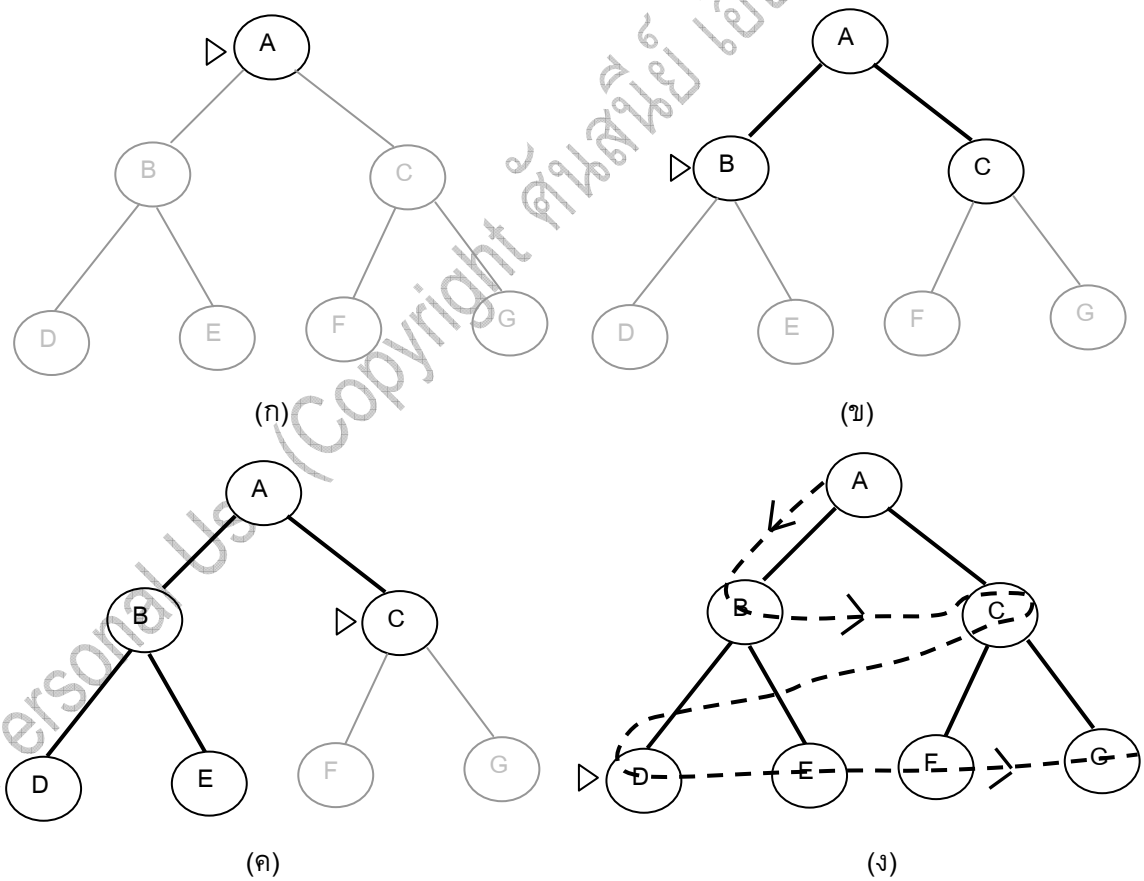
ในปัจจุบันวิธีที่เป็นที่นิยมคือการสร้าง network ของ convex polygon ที่เรียกว่า navmesh เพื่ออธิบายพื้นที่ที่ตัวแทนเดินทางผ่านได้ในสภาวะแวดล้อมของเกมส์นั้น ดังนั้นเราสามารถใช้กราฟแทนสภาวะแวดล้อมได้โดยที่แต่ละจุดในกราฟแทน convex space ดังแสดงในรูปที่ 3.4

3.2.2 อัลกอริทึมในการหาเส้นทาง

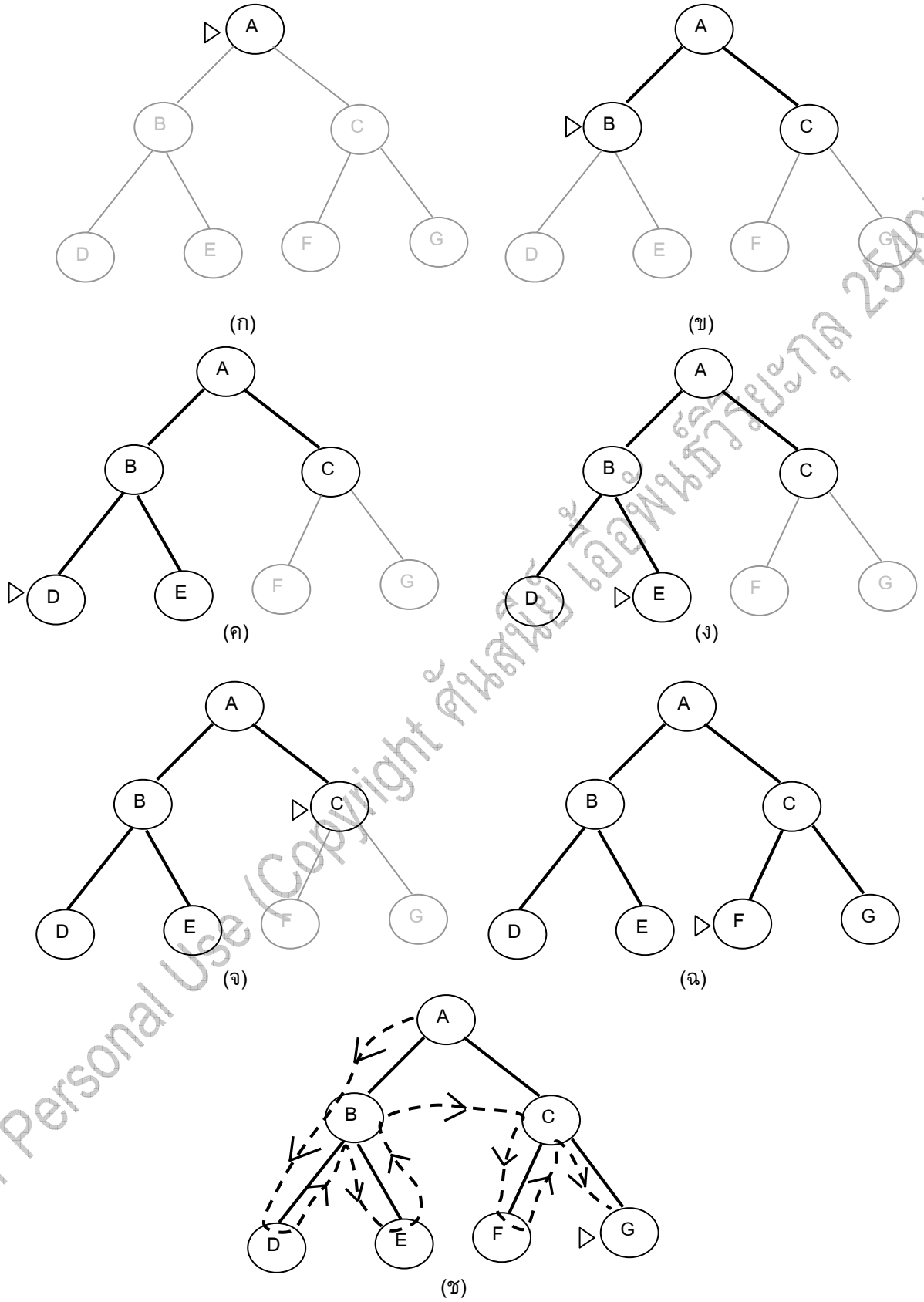
วิธีการหาเส้นทางมีด้วยกันหลายวิธีรวมทั้ง Breadth-first search Depth-first search Greedy Best-first search A* search และ Recursive best-first search ซึ่งจะอธิบายในหัวข้อต่อไป

3.2.2.1 Breadth-first search

Breadth-first search เป็นวิธีการง่ายๆ เริ่มจากขยาย root node และหลังจากนั้นตัวตามหลัง (successor) ของ root จะขยายตามมา และหลังจากนั้นตัวตามหลังของตัวตามหลังของ root ก็จะขยายอีก และเป็นเช่นนี้ไปเรื่อยๆ โดยปกติทุกจุดที่อยู่ในความลึกเดียวกันจะถูกขยายพร้อมกัน และจะถูกขยายก่อนจุดที่อยู่ในความลึกที่ลึกลงไป กระบวนการนี้แสดงในรูปที่ 3.5 เส้นทางในการหาเส้นทางเป็นดังเส้นประในรูป 3.5(ง) node ที่จะอยู่ในเส้นทางเรียงลำดับดังนี้ {A, B, C, D, E, F, G}



รูปที่ 3.5 Breadth-first search สำหรับ binary tree (ก) root node เตรียม ขยาย (ข) node B เตรียมขยาย (ค) node C เรียมขยาย (ง) node D เตรียมขยาย



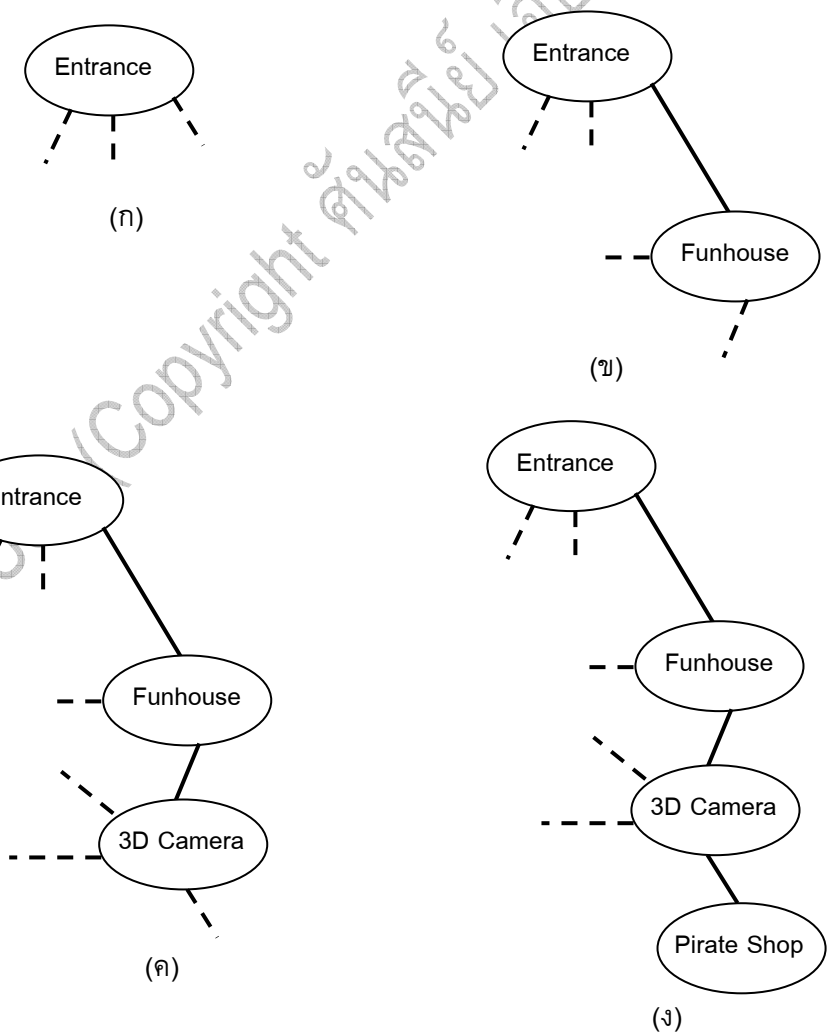
รูปที่ 3.6 Depth-first search ของ binary tree (ก) root node เตรียมขยาย (ข) node B เตรียม ขยาย (ค) node D เตรียมขยาย (ง) node E เตรียมขยาย (จ) node C เตรียมขยาย (ฉ) node F เตรียมขยาย (ช) node G เตรียมขยาย

3.2.2.2 Depth-first search

Depth-first search เป็นการหากราฟที่จะขยายทางด้านลึกก่อน นั่นคือการหาจะหาไปจนถึงจุดที่ลึกที่สุดของ search tree (ที่จุดนี้ไม่มีตัวตามหลัง) และการหาจะย้อนกลับไปที่ตัวตามหลังอันต่อไป วิธีการนี้แสดงในรูปที่ 3.6 เส้นประที่แสดงในรูป 3.6 (ช) เป็นเส้นทางที่หาได้ ดังนั้น node ที่อยู่ในเส้นทางที่เป็นคำตอบคือ {A, B, D, E, C, F, G} ตัวอย่างที่ 3.2 เป็นตัวอย่างของการนำ Depth-first search มาใช้ในการสร้างแผนที่ของสถานที่ที่หนึ่ง

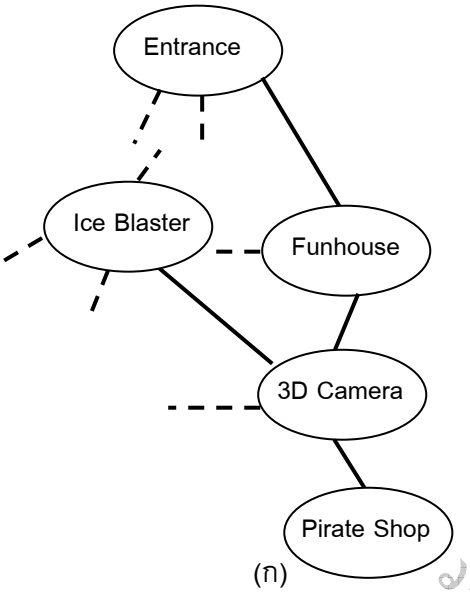
ตัวอย่างที่ 3.2 สมมุติให้นายแมวอยู่ที่ประตูทางเข้าของสวนสนุกแห่งหนึ่ง และเขาไม่มีแผนที่ แต่เขาทราบว่าของเล่นและจุดต่าง ๆในสวนสนุกเชื่อมต่อถึงกันได้ทั้งหมด ดังนั้นสิ่งที่นายแมวทำคือนายแมวแทนจุดต่าง ๆในสวนสนุกด้วยจุดในกราฟ และเส้นทางที่เชื่อมระหว่างจุดเหล่านี้แทนด้วยเส้นเชื่อมในกราฟ และนายแมวจะใช้ Depth-first search ในการเดินทางให้ทั่วสวนสนุก

จากประตูทางเข้านายแมวเลือกเดินทางทางหนึ่งในการเดิน ซึ่งเป็นทางที่เขายังไม่ได้เดิน ซึ่งการเดินตามทางนี้เขาจะเดินตามทางนั้นจนกระทั่งไม่สามารถไปต่อได้ดังรูปที่ 3.7 เส้นประคือทางที่ยังไม่ได้เดินในขณะที่เส้นทึบคือทางเดินไป

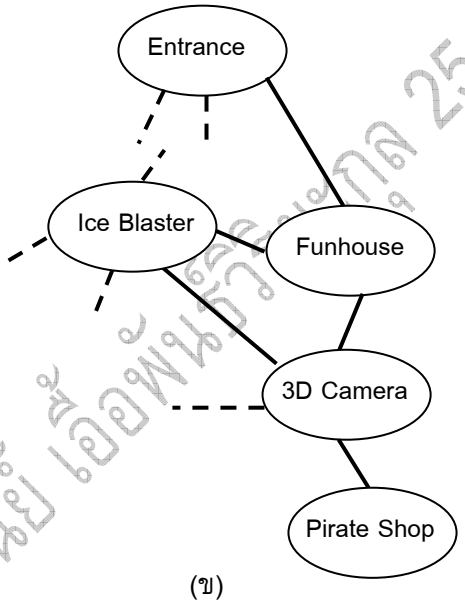


รูปที่ 3.7 การเดินทางของนายแมว (ก) อยู่ที่ Entrance (ข) เดินไป Funhouse (ค) เดินไป 3D Camera (ง) เดินไป Pirate Shop

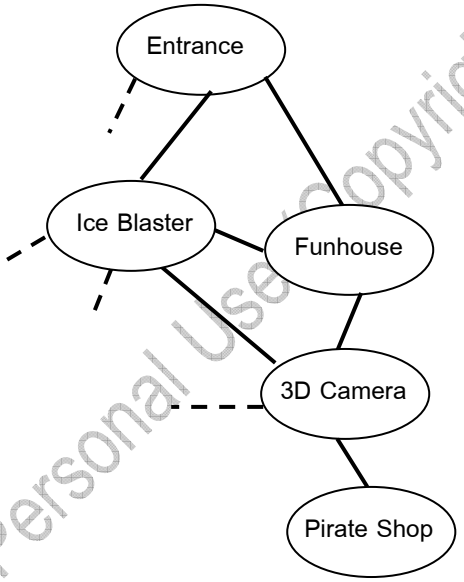
เมื่อนายแมวเดินไปถึง Pirate Shop เขาพบว่าไม่มีทางเดินต่อไปแล้ว ดังนั้นเขาจึงเดินย้อนกลับมาที่ 3D Camera และเดินไปทางอื่นที่ยังไม่ได้เดินไปดังรูปที่ 3.8 (ก) และเมื่อเขาไปถึง Ice Blaster เขาพบมีเส้นทางทั้งหมด 4 เส้นทางแต่เส้นทาง 2 เส้นทางในนั้นจะย้อนกลับไปในที่ที่เขาไปมาแล้ว (Entrance และ Funhouse) นายแมวจึงเลือกทางที่เหลือ ซึ่งเขาเดินมาถึง Slot Machines ดังรูป 3.8 (ข) (ค) และ 3.8 (ง) และนายแมวทำแบบนี้ไปเรื่อยๆ ดังรูป 3.8 (จ) (ฉ) (ช) และ 3.8 (ซ) และรูปที่ 3.9 แสดงแผนที่ของสวนสนุกที่ได้



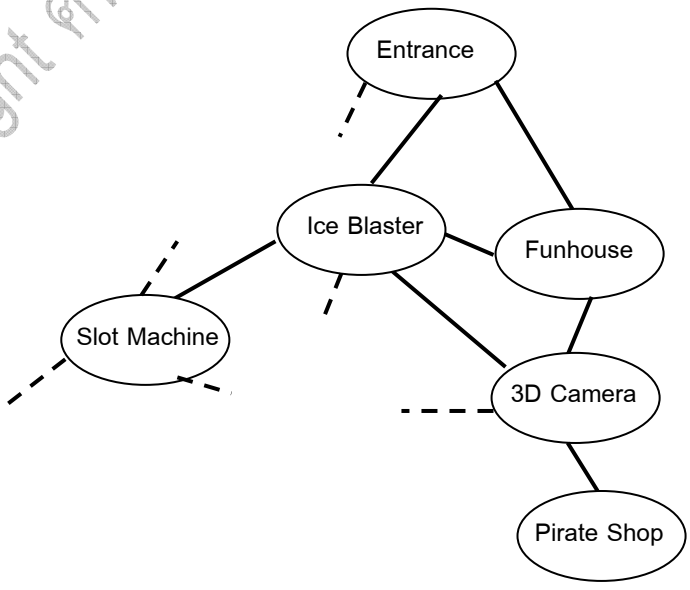
(ก)



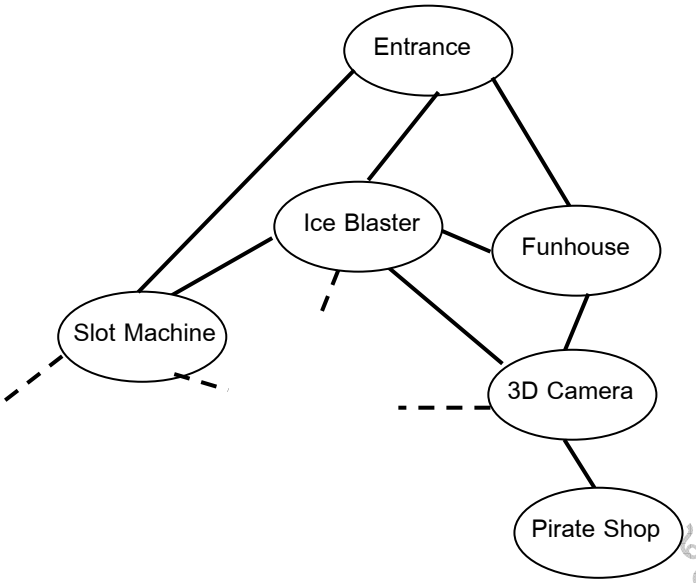
(ข)



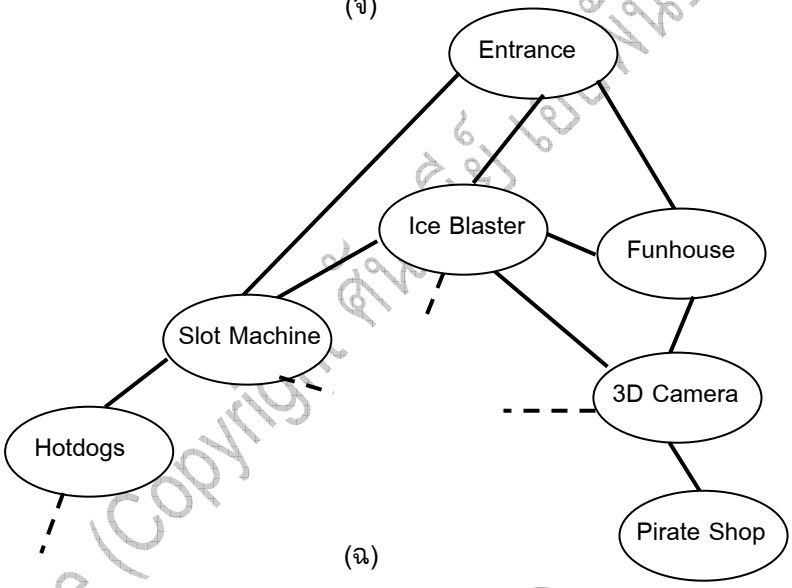
(ค)



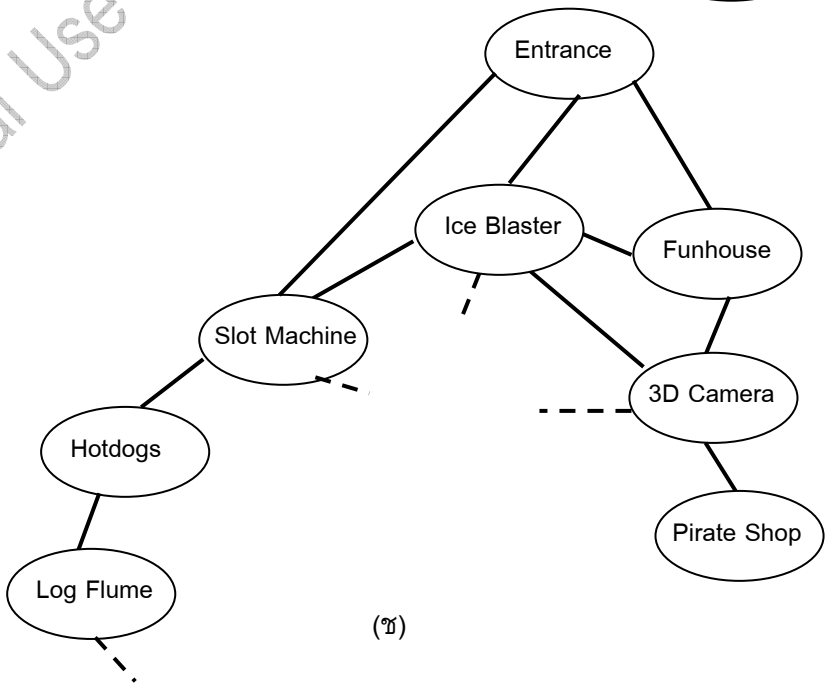
(ง)



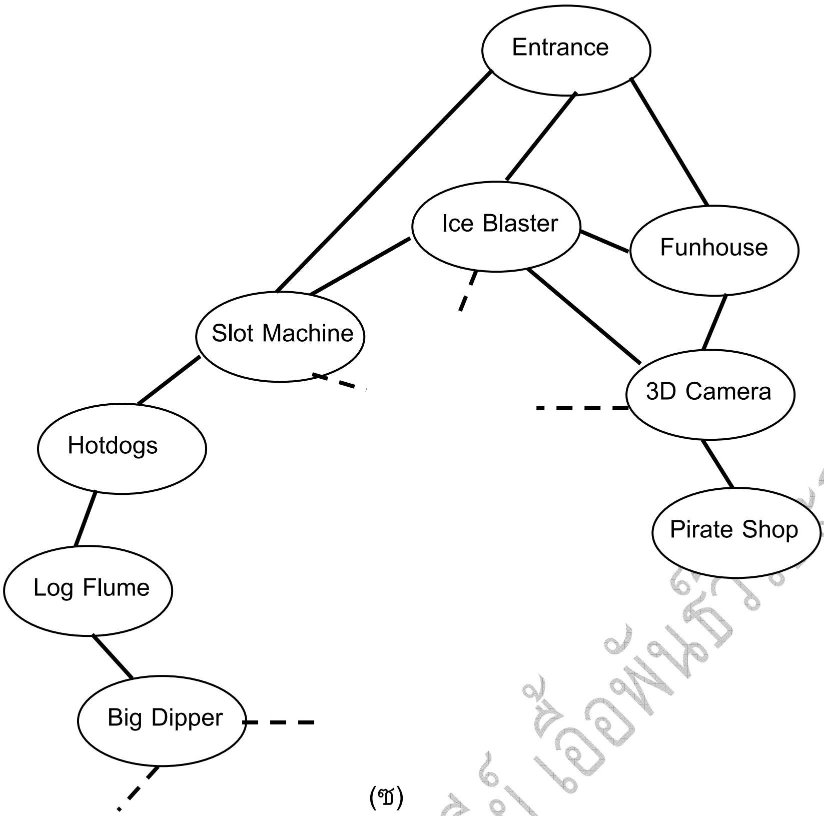
(จ)



(ฉ)

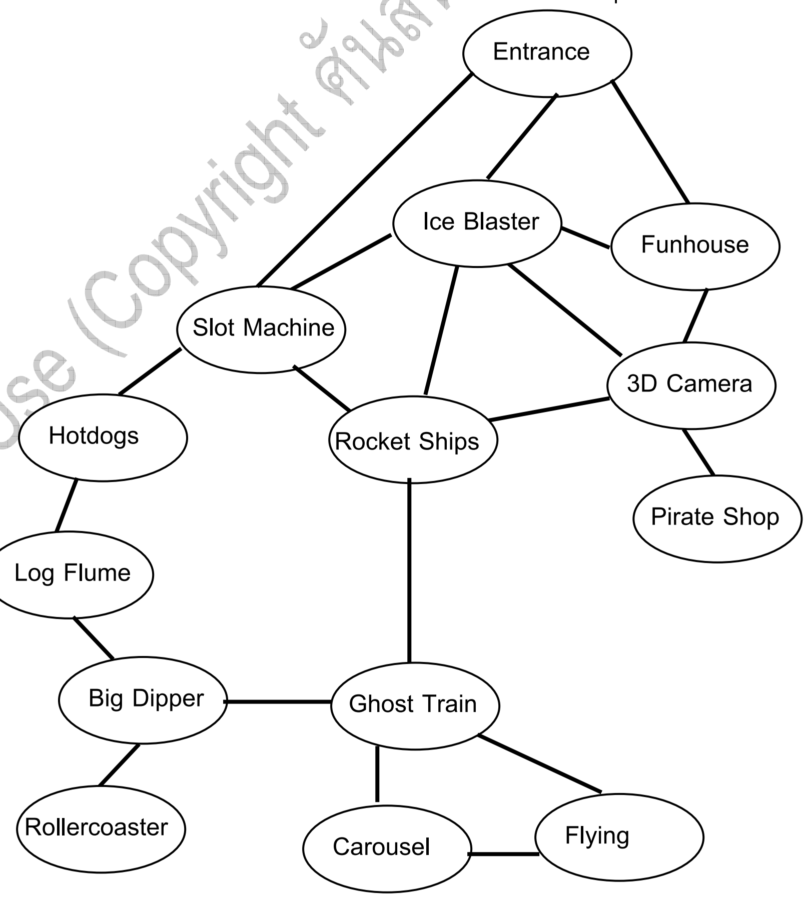


(ช)



(ซ)

รูปที่ 3.8 การใช้ Depth-first search ของการหาเส้นทางในสวนสนุกของนายแมว



รูปที่ 3.9 แผนที่ของสวนสนุกที่ได้ ■

3.2.2.3 Greedy Best-first search

Greedy Best-first search เป็นการหา ที่ใช้ข้อมูลที่เกี่ยวกับโครงสร้างของปริภูมิการหา (search space) นั่นคือ node จะถูกให้ขยายหรือไม่ขึ้นอยู่กับฟังก์ชันประเมิณค่า (evaluation function) ($f(n)$) ซึ่งโดยปกติในกรณีของ Greedy Best-first search node ที่มี $f(n)$ น้อยจะถูกเลือกให้ขยาย เพราะ $f(n)$ เป็นฟังก์ชันที่ขึ้นกับระยะห่างของ node จากจุดหมาย (goal) เท่านั้น ดังนั้น $f(n) = h(n)$ โดยที่ $h(n)$ เป็นฟังก์ชันสำนึก (heuristic function) ที่เป็นฟังก์ชันการประมาณค่าของทางที่ถูกที่สุดจาก node ไปยัง node ที่เป็นจุดหมาย

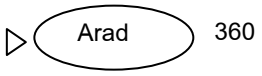
แต่ทั้งนี้ทั้งนั้นการหาโดยใช้ Greedy Best-first search คล้ายคลึงกับการหาแบบ Depth-first search นั่นคือพยายามที่จะเดินไปในทางเดียวจนถึงจุดหมาย แต่จะถอยหลังถ้าเจอทางตัน

ตัวอย่างที่ 3.3 สมมุติต้องการหาเส้นทางใน Romania โดยให้ฟังก์ชันสำนึกเป็นระยะที่เป็นเส้นตรง (straight-line distance heuristic) (h_{SLD}) ถ้าต้องการเดินทางไป Bucharest ระยะเส้นตรงไปยัง Bucharest จากเมืองต่างๆแสดงดังรูปที่ 3.10

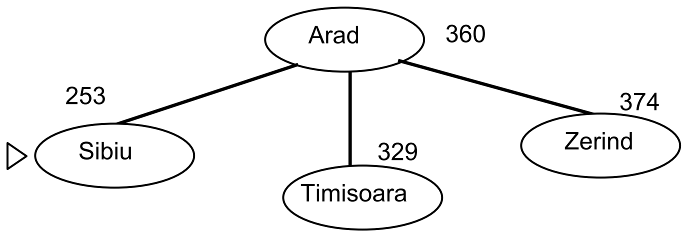
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

รูปที่ 3.10 ค่า h_{SLD} ของแต่ละเมืองไปยัง Bucharest

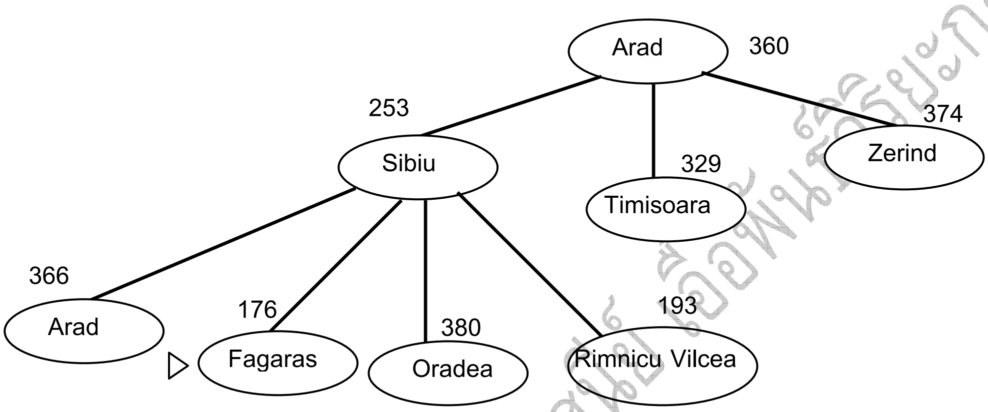
สมมุติต้องการเดินทางจาก Arad ไปยัง Bucharest ต้องการหาเส้นทางที่ดีที่สุดโดยใช้การหาแบบ Best-first search ดังนั้นเริ่มจาก node แรกคือ Arad ที่จะถูกขยาย เมื่อขยายแล้วพบว่า Sibiu มี h_{SLD} น้อยที่สุดก็ขยาย Sibiu และพบว่า Fagaras มี h_{SLD} น้อยที่สุดจึงขยาย Fagaras ก็จะเจอ Bucharest นั่นเอง ดังนั้นเส้นทางจาก Arad ถึง Bucharest คือ {Arad, Sibiu, Fagaras, Bucharest} กระบวนการหานี้แสดงในรูปที่ 3.11 ในรูปนี้จะมีตัวเลขซึ่งเป็นค่า h_{SLD} เขียนข้างแต่ละ node



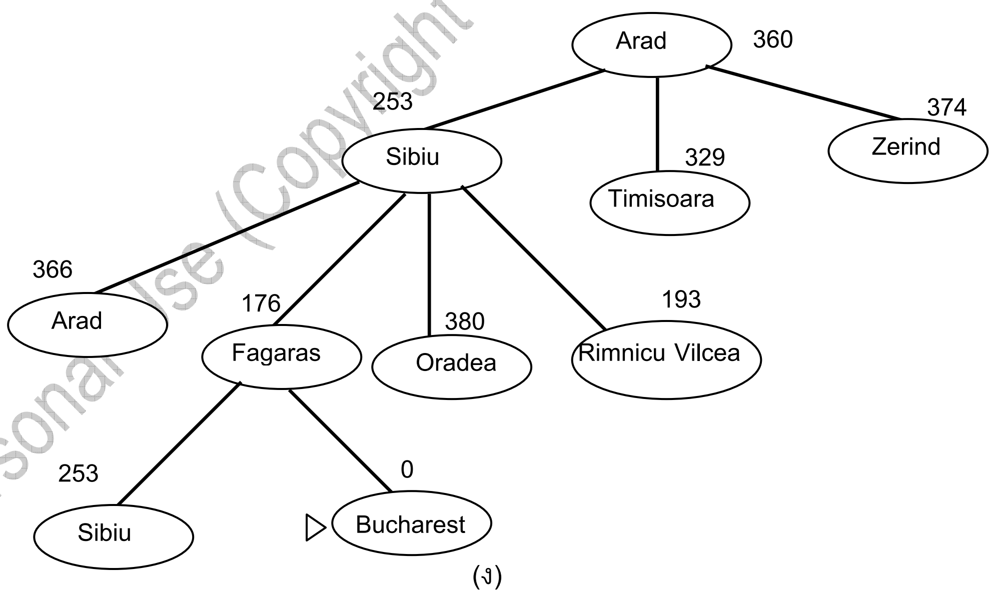
(ก)



(ข)



(ค)



(ง)

รูปที่ 3.11 (ก)initial state (ข) หลังจากขยาย Arad (ค) หลังจากขยาย Sibiu (ง) หลังจากขยาย Fagaras

■

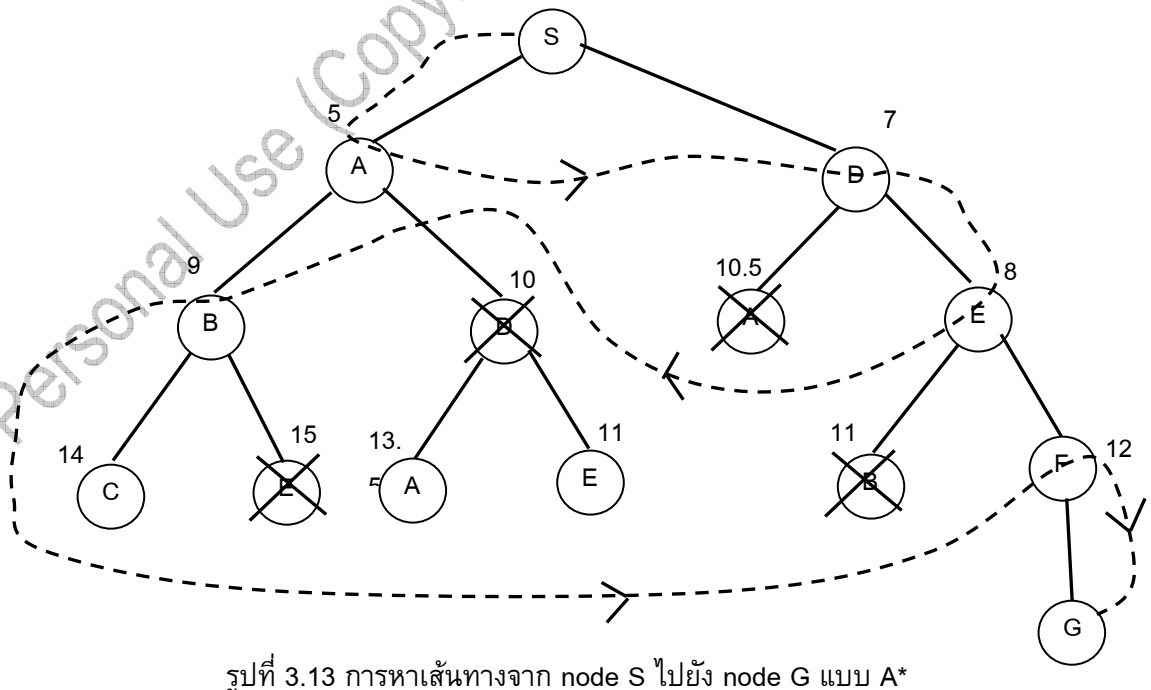
3.2.2.4 A* search

ในการหาแบบ A* ก็ยังคงอ้างอิง ฟังก์ชันประเมินค่า $f(n)$ ในการเลือก node ขยาย แต่ในการหาแบบ A* ฟังก์ชันประเมินค่ามีค่าขึ้นกับฟังก์ชันสำนึก (ฟังก์ชันการประมาณค่าของทางที่ถูกที่สุดจาก node ไปยัง node ที่เป็นจุดหมาย) และการประมาณค่าของการเดินทางมาถึง node นี้จาก node เริ่มต้น ($g(n)$) ดังนั้น $f(n) = g(n) + h(n)$ นั่นเอง

ในการหาแบบ A* มี list ที่ต้องใช้อีก 2 list คือ Open list และ Close list โดยที่ Open list คือ list ที่เก็บ node ที่ยังไม่ถูกขยาย และ Close list คือ list ที่เก็บ node ที่ถูกขยายแล้ว และเมื่อ node ใดที่ถูกขยาย ค่า $f(n)$ ของตัวตามหลังของ node นี้จะถูกคำนวณและตัวตามหลังเหล่านี้จะถูกเก็บใน Open list อัลกอริทึมของ A* search แสดงในรูปที่ 3.12

1. ใส่ starting node ใน Open list
2. ถ้า Open list ไม่มีสมาชิก ออกจากการหา (ไม่สามารถหาทางได้)
3. เลือก node ใน Open list ที่มีค่า *f* น้อยที่สุด ถ้า node นี้เป็นจุดหมาย หยุดการหา (หาทางได้)
4. ขยาย node ที่ได้ในข้อ 3 และใส่ node นี้ใน Close list ตรวจสอบตัวตามหลังของ node นี้ทุกตัวว่าอยู่ใน Open list หรือ Close list มาก่อนหรือไม่ ถ้าไม่ ก็นำตัวตามลหังทุกตัวใส่ใน Open list
5. ถ้าตัวตามหลังตัวใดอยู่ใน Open list หรือ Close list ให้ตรวจสอบว่าทางใหม่มีประสิทธิภาพกว่าหรือไม่ (ค่า *f* น้อยกว่า) ถ้าใช่ให้ปรับทาง
6. กลับไปทำข้อ 2

รูปที่ 3.12 อัลกอริทึมในการหาแบบ A*

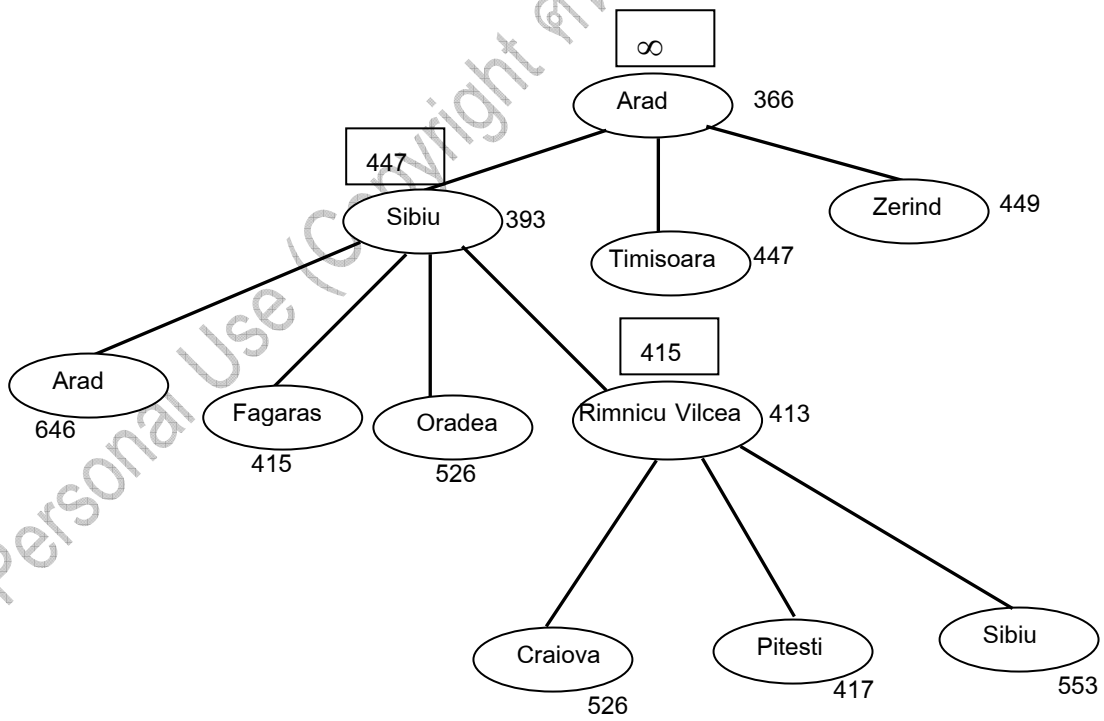


รูปที่ 3.13 การหาเส้นทางจาก node S ไปยัง node G แบบ A*

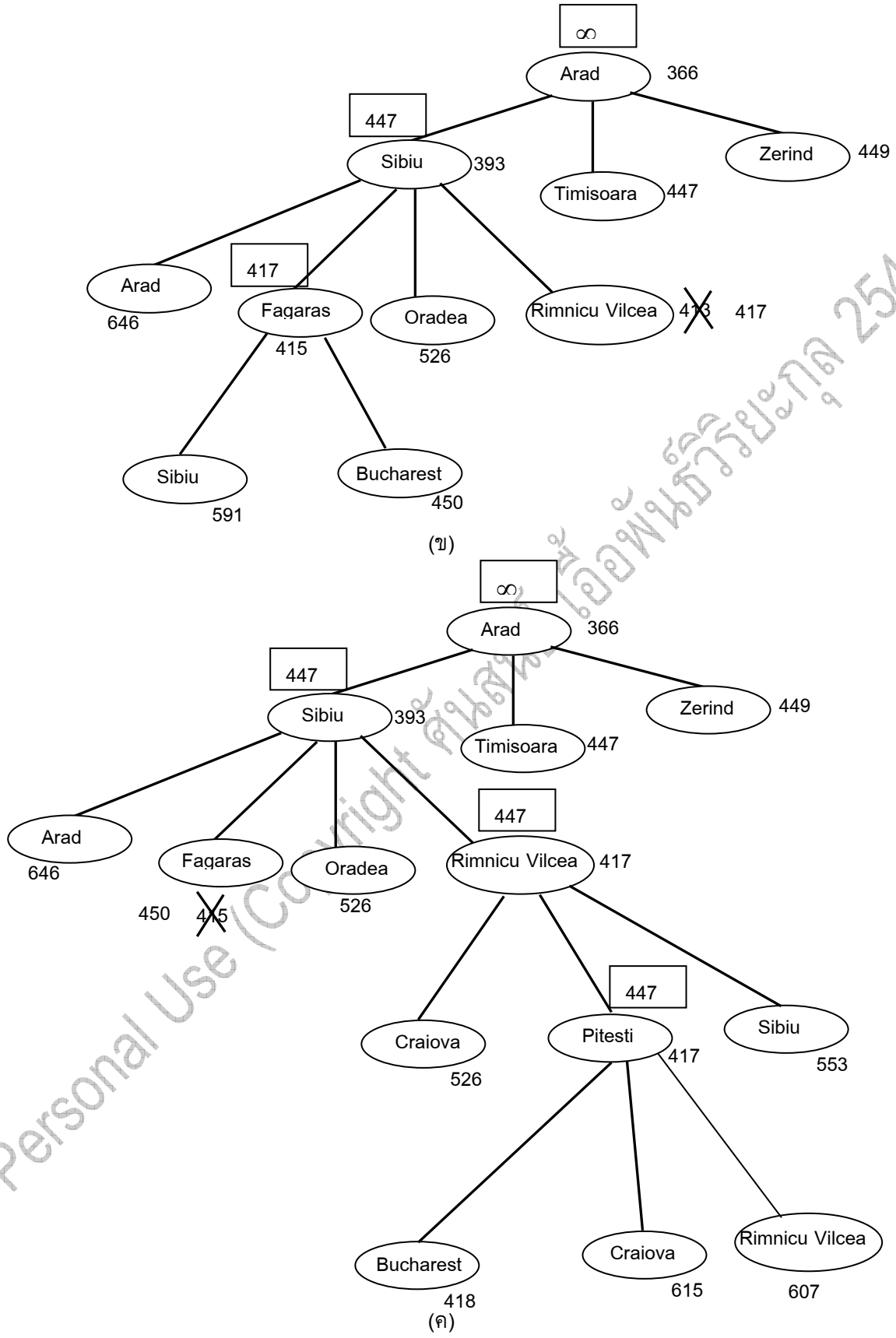
รูปที่ 3.13 แสดงการหาแบบ A* โดยที่ค่าที่เขียนข้างแต่ละ node คือค่า *f* โดยที่เริ่มจาก S ถูกขยายแล้วได้ A_5 และ D_7 ซึ่งทั้งสอง node ไม่ได้อยู่ในทั้ง Open list และ Close list และ A_{15} ถูกเลือกให้ขยายได้ B_9 และ D_{10} และ node D อยู่ใน Open list อยู่แล้วและมีค่า *f* ที่น้อยกว่า node D ที่ได้มาใหม่ทำให้ D_{10} ถูกทำให้สิ้นสุด และทำแบบนี้ไปจนกว่าจะเจอ node G หรือ ไม่สามารถหาทางได้ ซึ่งจากรูป node D_{10}, $A_{10.5}$ B_{11} และ E_{15} คือ node ที่ถูกทำให้สิ้นสุด และเส้นทางที่หาได้คือ {S, A_5, D_7, E_8, B_9, F_{12}, G}

3.2.2.5 Recursive best-first search (RBFS)

วิธีการหาแบบนี้เป็นการหาแบบเวียนเกิด (recursive) วิธีการนี้จะพยายามเดินทางไปตามเส้นเดิมลงไปเรื่อยๆ โดยที่จะเก็บค่า *f* ของเส้นทางสำรองของ node บรรพบุรุษ (ancesteor) ของ node ปัจจุบันที่ดีที่สุดไว้ ถ้า node ปัจจุบันมีค่า *f* มากกว่าจะเดินย้อนกลับไปที่ทางสำรอง ซึ่งในการเดินย้อนกลับ RBFS จะแทนค่า *f* ของแต่ละ node ในเส้นทางเดิมด้วยค่า *f* ที่ดีที่สุดของ node ลูก ของ node นั้น ในการทำเช่นนี้ RBFS จะสามารถจำค่า *f* ของใบ (leaf) ที่ดีที่สุดของซับทรี (subtree) ที่ถูกลืม และเพื่อใช้ในการตัดสินใจว่าจะขยายซับทรีนี้หรือไม่ รูปที่ 3.14 แสดงการหาเส้นทางโดยใช้ RBFS จากเมือง Arad ไปยังเมือง Bucharest เริ่มแรก RBFS เดินทางจาก Arad ไปยัง Rimmicu Vilcea และเปลี่ยนแปลงเส้นทางไปทาง Fagaras และเปลี่ยนเส้นทางอีกครั้ง ซึ่งการเปลี่ยนเส้นทางนี้เกิดขึ้นเนื่องจากทุกครั้งที่มีการขยายเส้นทางที่ดีที่สุด มีโอกาสที่ค่า *f* จะเพิ่ม ซึ่งโดยปกติถ้าเหตุการณ์นี้เกิดเส้นทางที่ดีเป็นอันดับ 2 อาจกลายเป็นเส้นทางที่ดีที่สุดได้ ดังนั้นการหาจึงต้องเดินทางย้อนกลับ



(ก)



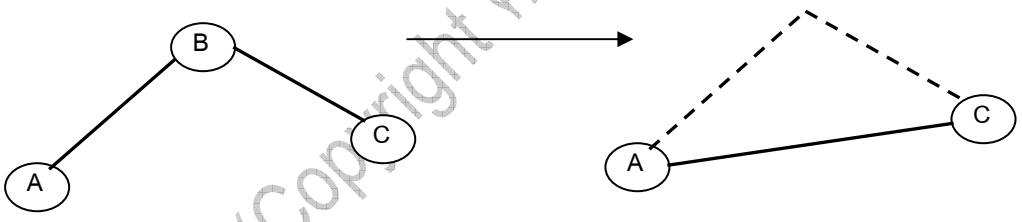
รูปที่ 3.14 การหาเส้นทางจาก Arad ไป Bucharest โดยใช้ RBFS (ก) หลังจากขยาย Arad Sibiu และ RImnicu Vilcea (ข) หลังจากย้อนกลับไปที่ Sibiu และขยาย Fagaras (ค) หลังจากกลับมาที่ Rimnicu Vilcea และขยาย Pitesti

ในรูปที่ 3.14 ค่า *f* ที่เป็นค่า *f*-limit แสดงข้างบน node ปัจจุบัน ในรูป 3.14 (ก) เส้นทางที่ดีที่สุดจาก Rimmicu Vilcea เป็นทางที่ลงไปถึง Pitesti แต่ปรากฏว่าค่า *f* ที่ node นี้มากกว่าค่า *f* ของเส้นทางสำรอง (Fagarus) จึงเดินย้อนกลับไปที่ Fagaras และค่า *f* ของ Rimmicu Vilcea จะเป็นค่า *f* ที่ดีที่สุดของซับทรีของ Rimmicu Vilcea คือ 417 และค่า *f*-limit ที่ Fagaras ก็ถูกตั้งเป็น 417 เช่นกัน หลังจากนั้น Fagaras ถูกขยาย และเห็นว่าค่าที่ดีที่สุดของซับทรีนี้คือ 450 ซึ่งมากกว่าค่า *f*-limit จึงเดินทางย้อนกลับมาที่ Rimmicu Vilcea และ Rimmicu Vilcea ถูกขยาย แต่ในครั้งนี้ ทางสำรองที่ดีที่สุดเป็นทางที่ผ่าน Timisoara ค่า *f*-limit ของ Rimmicu Vilcea จึงเป็น 447 และในการขยายครั้งนี้จึงทำให้การเดินทางไปสิ้นสุดที่ Bucharest ได้โดยที่ค่า *f* ไม่เกินค่า *f*-limit

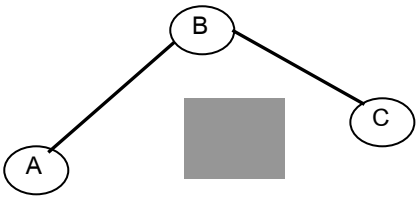
3.3 การปรับการเดินทางให้เรียบ (Path Smoothing)

ถ้าเกมส์ใช้การหาเส้นทางดังที่กล่าวมาข้างต้นจะได้เส้นทางการเดินที่มีลักษณะที่ไม่เรียบดังรูปที่ 3.2 ซึ่งเป็นเส้นทางที่ไม่เป็นไปตามธรรมชาติตามสายตาของมนุษย์ ดังนั้นในการแก้ปัญหานี้จึงต้องทำหลังจากที่ได้เส้นทางมาแล้ว ซึ่งวิธีที่จะกล่าวถึงนี้จะเป็นวิธีที่หยาบแต่การประมาวลผลเร็ว

วิธีนี้เป็นวิธีทีค่อนข้างรวดเร็ว ซึ่งทำได้โดยการตรวจสอบการผ่านได้ (possibility) ระหว่างจุด 2 จุด ถ้า 1 ในจุดนั้นไม่จำเป็นก็แทนที่จุด ทั้ง 2 จุดด้วยจุดจุดเดียวดังรูปที่ 3.15 ในรูปนี้แต่เดิมต้องเดินทางจากจุด A ไปจุด B แล้วจึงไป C แต่เนื่องจากการเอาจุด B ออกไม่ทำให้เกิดความเสียหายเส้นทางการเดินจึงเป็นจากจุด A ไปยังจุด C ได้ทันที แต่ถ้าสถานการณ์เป็นดังรูปที่ 3.16 ไม่สามารถเอาจุดใดออกได้เลย



รูปที่ 3.15 การปรับเส้นทางให้เรียบ



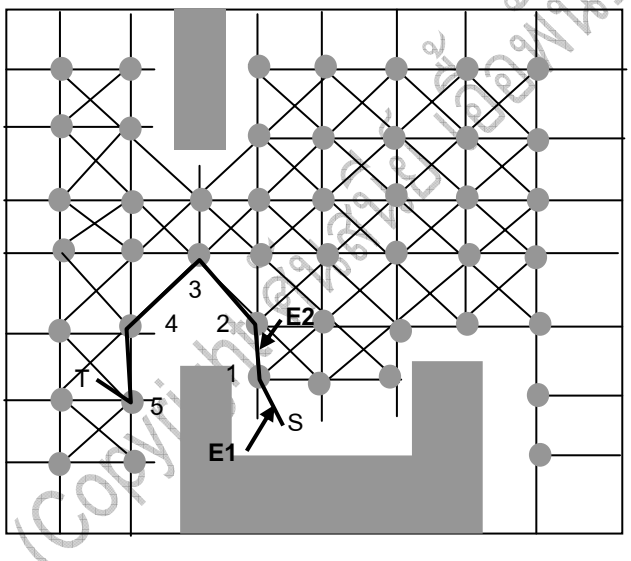
รูปที่ 3.16 เมื่อมีสิ่งกีดขวาง

อัลกอริทึมของการปรับเส้นทางแบบหยาบแสดงในรูปที่ 3.17 และรูป 3.18 แสดงการทำงานของวิธีการนี้ ซึ่งจากเดิม E1 เป็นเส้นเชื่อม S — 1 และ E2 เป็นเส้นเชื่อม 1 — 2 จะเห็นว่าตัวแทนสามารถเดินทางจากต้นทางของ E1 ไปยังปลายทางของ E2 ได้โดยที่ไม่มีสิ่งกีดขวางดังนั้น node 2 จึงถูกใส่เป็นปลายทางของ E1 แทน และเส้นเชื่อม 1 — 2 จึงถูกเอาออกจากเส้นทางเดิน และ E2 จะเก็บต้นทางของ E1 กลายเป็นเส้นเชื่อม 2 — 3 ดังรูป 3.18(ข) และเมื่อพบว่าสามารถเดินทางจากต้นทาง E1 ไปยังปลายทาง E2 ได้โดยไม่ผ่านสิ่งกีดขวาง node 3 จึงกลายเป็ยปลายทางของ E1 และ E2

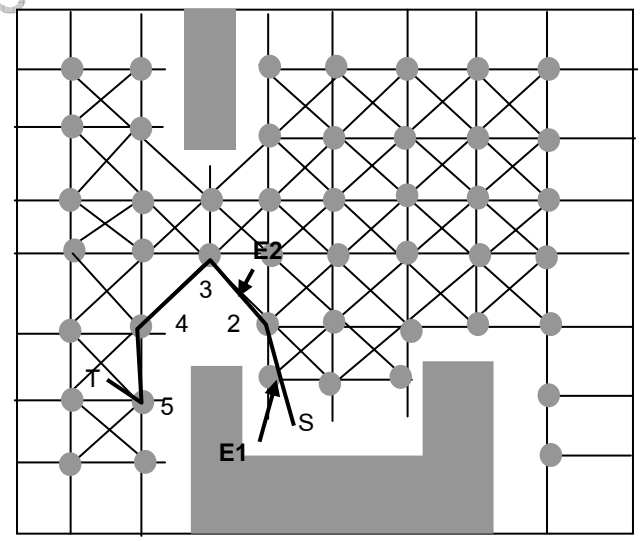
เป็นเส้นเชื่อม 3 — 4 ดังรูปที่ 3.18 (ค) แต่ครั้งนี้มีสิ่งกีดขวางจึงไม่สามารถทำแบบเดิมได้จึงต้องย้าง E1 ให้เป็นเส้นเชื่อม 3 — 4 และ E2 เป็นเส้นเชื่อม 4 — 5 ดังรูป 3.18 (ง) และครั้งนี้ก็มีสิ่งกีดขวางจึงต้องขยับ E1 เป็นเส้นเชื่อม 4 — 5 และ E2 เป็นเส้นเชื่อม 5 — T และครั้งไม่มีสิ่งกีดขวางจึงปรับเส้นโดยการนำ node ออกจะได้เส้นทางสุดท้ายดังรูป 3.18 (จ)

1. เก็บปลายทางของ E2
2. ถ้าตัวแทนสามารถเดินทางระหว่างทั้งสองจุดได้โดยไม่มีสิ่งกีดขวางให้ตั้งค่าปลายทางของ E1 ให้เป็นปลายทางของ E2 และ E2 จะเป็นเส้นเชื่อมเส้นถัดไป
3. ถ้าตัวแทนไม่สามารถเดินผ่านทั้ง 2 จุดได้กำหนดให้ E2 เป็น E1 และขยับ E2 ไปเป็นเส้นเชื่อมถัดไป
4. ทำซ้ำจนกระทั่งปลายทางของ E2 เป็นปลายทางของเส้นทาง

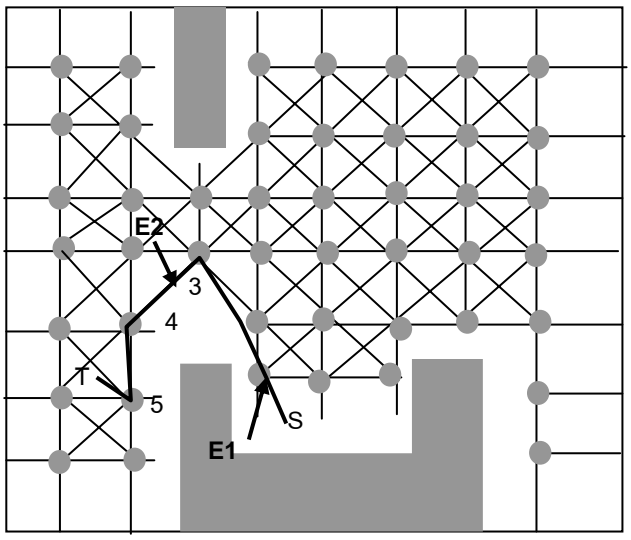
รูปที่ 3.17 อัลกอริทึมในการปรับเส้นทางให้เรียบแบบหยาบ



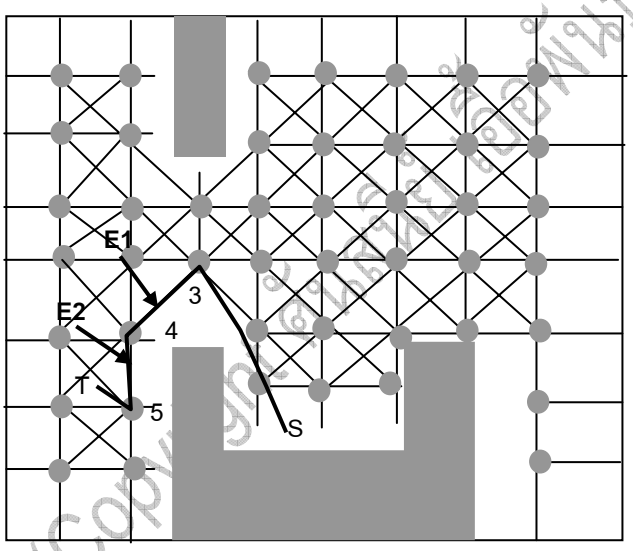
(ก)



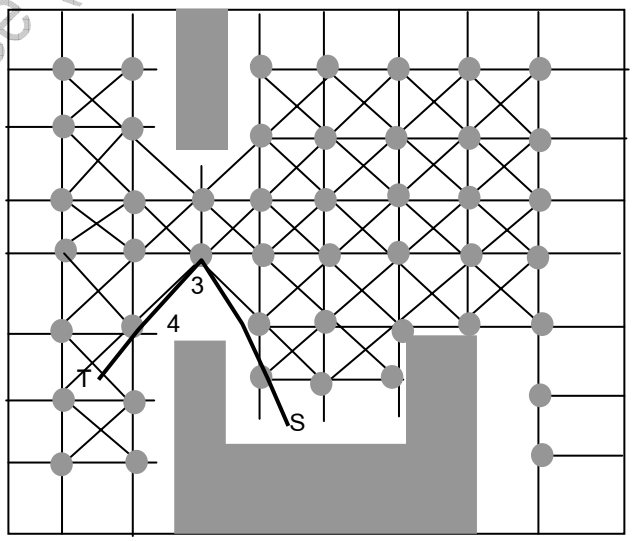
(ข)



(ค)



(ง)

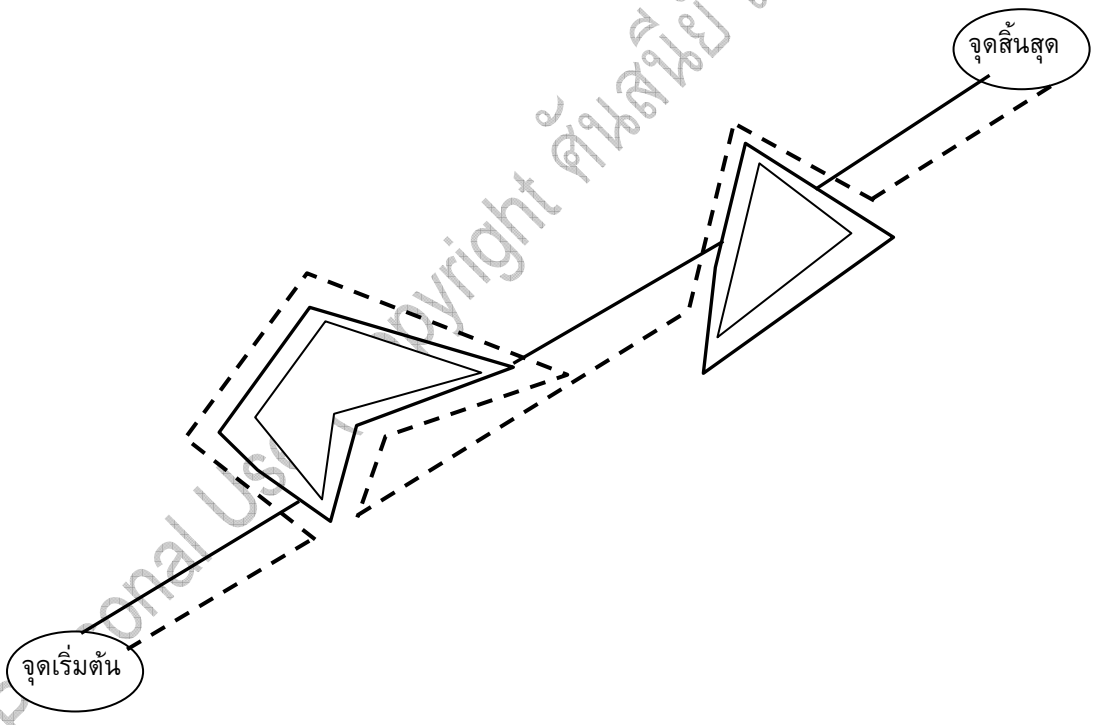


(จ)

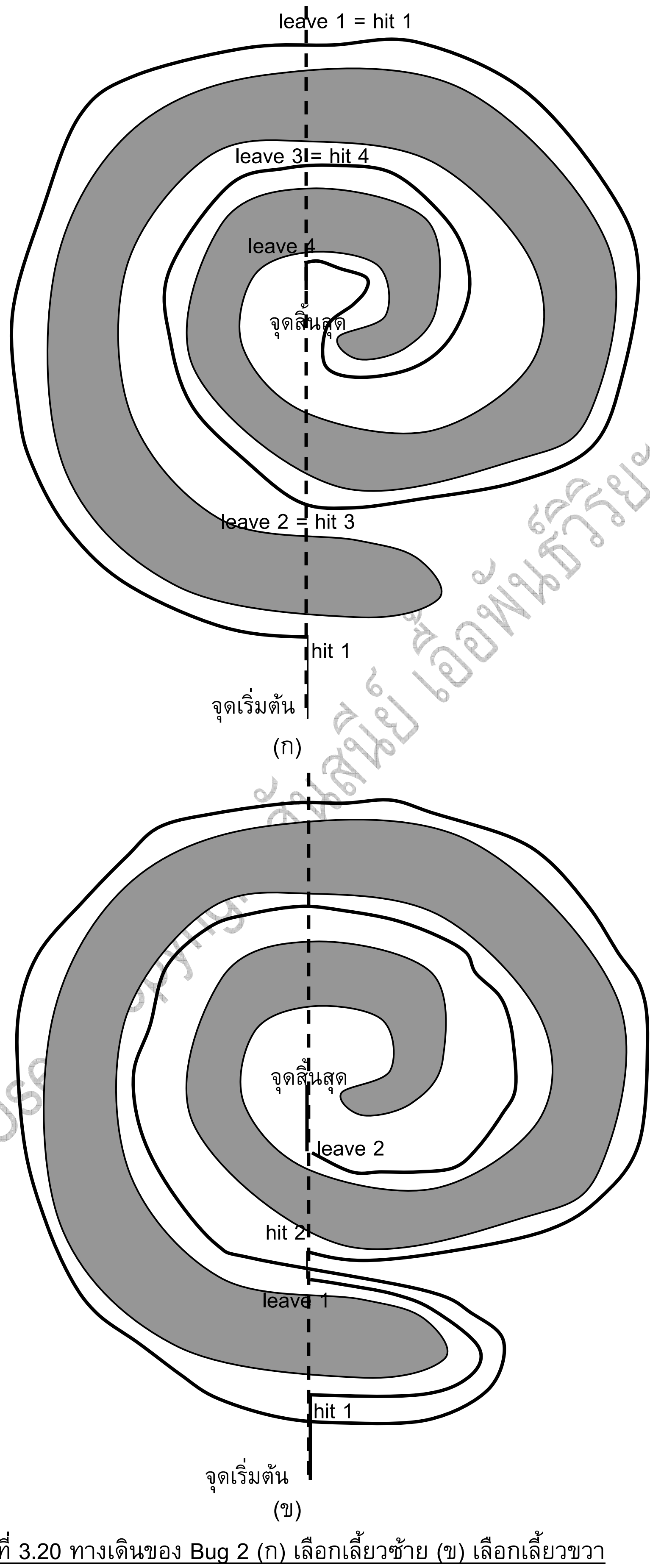
รูปที่ 3.18 การทำงานของการปรับเส้นทางให้เรียบแบบหยาบ

3.4 Lumelsky Bug อัลกอริทึม

การเดินทางหลบสิ่งกีดขวางที่ไม่ต้องสร้างกราฟและใช้การหาเส้นทางดังที่กล่าวมา นั่นคือการเดินทางโดยใช้ Lumelsky Bug อัลกอริทึม ซึ่งอัลกอริทึมตั้งสมมุติฐานว่าตัวแทนรู้จุดเริ่มต้นและจุดสิ้นสุด และอัลกอริทึมนี้มี 2 ชนิดนั่นคือ Bug 1 และ Bug 2 ซึ่งการเดินทางของอัลกอริทึม Bug จะเริ่มจากการเลือกสุ่มการเลี้ยว (เลี้ยวซ้าย/เลี้ยวขวา) เพื่อเป็นการเลี้ยวหลักตลอดการเดินทาง (local direction) ดังนั้นถ้าตัวแทนเจอสิ่งกีดขวางตังแทนจะเดินรอบสิ่งกีดขวางนั้น โดยจะจดจำจุดที่อยู่ใกล้จุดสิ้นสุด แล้วเดินรอบอีกครั้งเพื่อให้หลุดออกจากสิ่งกีดขวางไปที่จุดนั้น ส่วนใน Bug 2 ตัวแทนจะเดินรอบสิ่งกีดขวางจนกว่าจะเจอเส้นทางที่อยู่ในทิศทางเดิมเพื่อเดินไปหาจุดสิ้นสุด นั่นคือจะเดินตรงไปหาจุดสิ้นสุดเป็นเส้นตรง ถ้าเจอสิ่งกีดขวางจะเดินรอบสิ่งกีดขวางจนกระทั่งเจอเส้นตรงเส้นเดิม ถ้าจุดที่จะหลุดออกจากสิ่งกีดขวาง (leave point) อยู่ใกล้กว่าจุดที่ชนสิ่งกีดขวาง (hit point) ให้ออกจากตรงจุดนั้นและเดินไปบนแนวเดิม ถ้าไม่เช่นนั้นให้เดินรอบสิ่งกีดขวาง ถ้าเจอจุดที่ชนสิ่งกีดขวางจุดเมโดยที่ไม่เจอจุดที่หลุดออกจากสิ่งกีดขวาง แสดงว่าไม่สามารถเดินถึงจุดสิ้นสุดได้นั่นเอง อัลกอริทึมทั้งสองแสดงในรูปที่ 3.19 โดยที่ทางเดินจาก Bug 1 ใช้เส้นทึบ และทางเดินจาก Bug 2 คือเส้นประ และ Bug 2 เลือกที่จะเลี้ยวซ้าย และ 3.20 โดยที่ 3.20(ก) Bug 2 ที่เลือกเลี้ยวซ้ายตลอด และ 3.20(ข) คือทางเดินของ Bug 2 ทีเลือกเลี้ยวขวาตลอด



รูปที่ 3.19 ทางเดินของ Bug 1 (เส้นทึบ) และทางเดินของ Bug 2 (เส้นประ)

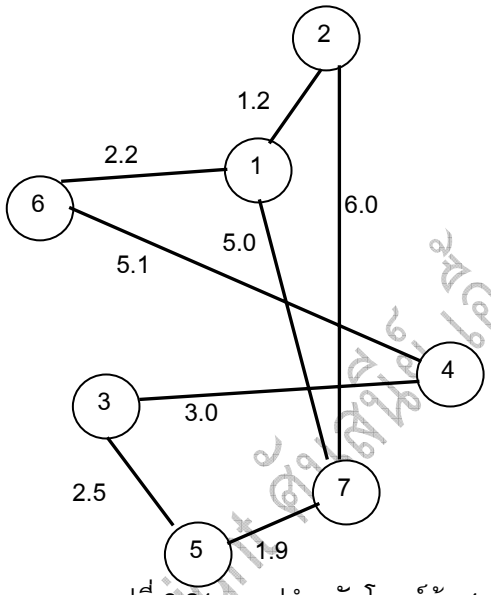


รูปที่ 3.20 ทางเดินของ Bug 2 (ก) เลือกเลี้ยวซ้าย (ข) เลือกเลี้ยวขวา

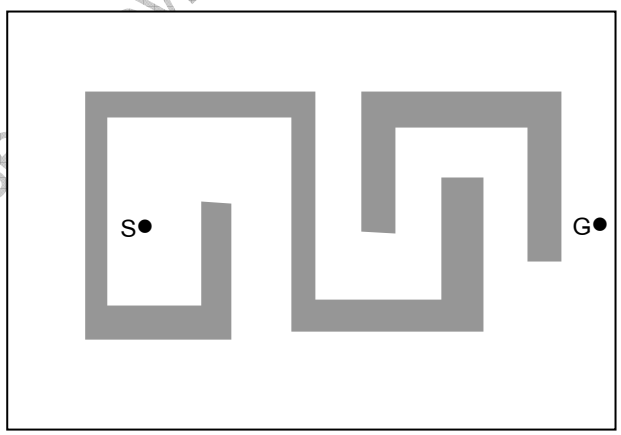
ในบทนี้ได้กล่าวถึงวิธีการสร้างกราฟ และการหาเส้นทางโดยใช้วิธี Depth-first search Breadth-first search Greedy Best-first search และ A* search ซึ่งวิธีเหล่านี้มีทั้งที่ใช้ข้อมูลของโครงสร้าง และไม่ใช้ข้อมูลของโครงสร้าง และในบทที่ 4 จะกล่าวถึงการเคลื่อนที่เป็นกลุ่ม

คำถามท้ายบทที่ 3

1. ให้ใช้รูปที่ 3.21 ในการหาเส้นทางโดยที่ใช้ node ใดก็ได้เป็น starting node และ จุดหมาย โดยใช้วิธี Depth-first search Breadth-first search และ Greedy Best-first search ซึ่งในกรณีของ Greedy Best-first search ให้สมมุติว่าตัวเลขที่เขียนบนเส้นเชื่อมคือระยะระหว่างทั้งสอง node ที่เส้นเชื่อมนั้นเชื่อมอยู่



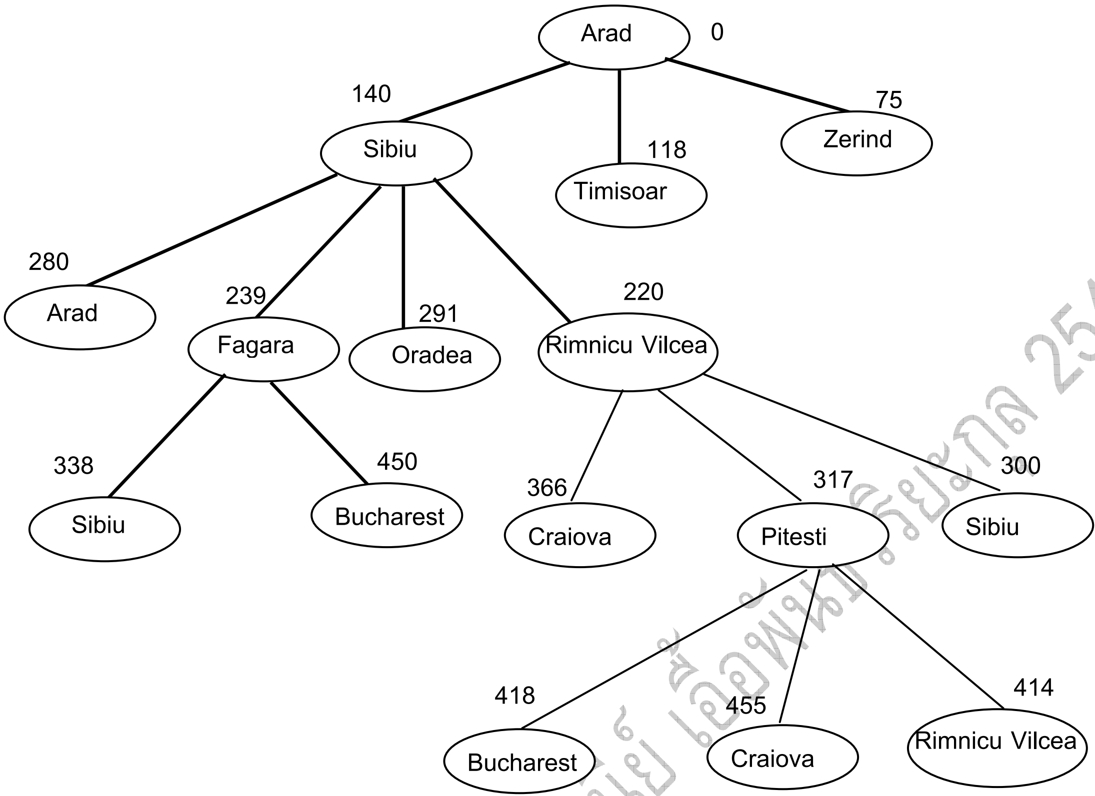
รูปที่ 3.21 กราฟสำหรับโจทย์ข้อ 1



รูปที่ 3.22 รูปสำหรับโจทย์ข้อ /

2. จากรูปที่ 3.22 ถ้าต้องการเดินทางจากจุด S ไปยังจุด G จะต้องทำอย่างไรบ้างและทำได้อย่างไร อย่าลืมว่าจะต้องสร้าง Navgraph ก่อนที่จะใช้วิธีการหา

3. จากรูปที่ 3.23 จงใช้การหาเส้นทางแบบ A* ในการเส้นทางจาก Arad ไปยัง Bucharest ให้ค่าของ *g* ของแต่ละ node แสดงในรูปที่ 3.23 และค่า *h* แสดงในรูปที่ 3.10



รูปที่ 3.23 รูปสำหรับโจทย์ข้อ 3

การเคลื่อนไหวแบบกลุ่มที่ฉลาด
Intelligent Group Movement

4

ในบทนี้จะกล่าวถึงการเคลื่อนที่เป็นกลุ่มของตัวแทนหลายตัวโดยเป็นการเคลื่อนที่แบบฉลาด ซึ่งการเคลื่อนที่แบบนี้จะใช้หลักการของ swarm intelligence และการนำ swarm อย่างง่ายไปใช้ในเกมส์

4.1 Swarm Intelligence

สมมุติว่ามีกลุ่มคนกลุ่มหนึ่งออกเดินทางเพื่อค้นหาสมบัติ แต่ละคนมีอุปกรณ์ในการค้นหาเหมือนกันและสามารถคุยกันในกลุ่มเพื่อนรอบข้าง *n* คนได้โดยผ่านอุปกรณ์สื่อสาร ดังนั้นแต่ละคนจะรู้ว่าคนอื่นที่อยู่ในกลุ่มเพื่อนรอบข้างคนใดอยู่ใกล้สมบัติ ซึ่งถ้าแต่ละคนเคลื่อนที่เข้าใกล้คนนั้น ก็จะมีโอกาสหาสมบัติเจอมากขึ้น เหตุการณ์ที่กล่าวข้างต้นเป็นการแสดงถึง swarm behavior ที่แต่ละคนใน swarm สามารถคุยกันได้ในการแก้จุดประสงค์รวม (global objective) ซึ่งมีประสิทธิภาพมากกว่าการแก้ปัญหาโดยคนคนเดียว

swarm เป็นโครงสร้างที่ประกอบไปด้วยสิ่งมีชีวิต (organism) หรือตัวแทนที่โต้ตอบกันได้ ซึ่งสิ่งมีชีวิตเหล่านี้รวมถึง มด ผึ้ง ฝูงปลา และฝูงนก โดยปกติตัวแทนใน swarm จะมีโครงสร้างที่ง่าย แต่เมื่อมารวมกลุ่มกันจะกลายเป็นสิ่งที่ซับซ้อนขึ้น

พฤติกรรมโดยรวมของ swarm ของสิ่งมีชีวิตในสังคมเป็นไปในลักษณะที่ไม่ใช่เส้นตรงซึ่งพฤติกรรมนี้มาจากพฤติกรรมของแต่ละตัวตนใน swarm ดังนั้นพฤติกรรมของแต่ละตัวตนกับพฤติกรรมของ swarm มีความสัมพันธ์กันอย่างมั่นคง นั่นคือพฤติกรรมของแต่ละตัวตนเมื่อรวมกันจะสามารถกำหนดรูปร่างและพฤติกรรมของ swarm ได้ในขณะที่พฤติกรรมของ swarm จะกำหนดเงื่อนไขที่ทำให้แต่ละตัวตนกระทำการใด ซึ่งการกระทำเหล่านี้เปลี่ยนสภาวะแวดล้อม และพฤติกรรมของแต่ละตัวตนและเพื่อนจะเปลี่ยนไปด้วย ซึ่งเงื่อนไขเหล่านี้ถูกกำหนดโดยพฤติกรรมของ swarm รวมทั้งที่เป็นรูปแบบเชิงพื้นที่และเชิงเวลา

พฤติกรรมของ swarm ไม่ได้ถูกกำหนดโดยพฤติกรรมของแต่ละตัวตนเท่านั้นยังรวมถึงการโต้ตอบระหว่างแต่ละตัวตนด้วย การโต้ตอบระหว่างแต่ละตัวตนช่วยในการพัฒนาความรู้เกี่ยวกับสิ่งแวดล้อมและเพิ่มสมรรถนะของ swarm ไปสู่คำตอบที่เหมาะสม

การโต้ตอบแบ่งออกเป็น 2 ประเภทคือการโต้ตอบทางตรง และทางอ้อม ตัวอย่างของการโต้ตอบทางตรงคือเป็นการโต้ตอบที่ผ่านการมองเห็น การได้ยิน และการสัมผัสทางเคมี ส่วนการโต้ตอบทางอ้อมเกิดขึ้นเมื่อมีตัวแทน 1 ตัวเปลี่ยนสิ่งแวดล้อมและตัวแทนตัวอื่นตอบสนองต่อสิ่งแวดล้อมที่เปลี่ยนไปนั้นนั่นเอง ซึ่งการโต้ตอบแบบนี้ถูกเรียกว่า stigmergy

โครงสร้างเครือข่ายทางสังคมของ swarm ให้ช่องทางการสื่อสารสำหรับแต่ละตัวตนเพื่อแลกเปลี่ยนความรู้เชิงประสบการณ์ระหว่างกัน และสิ่งที่น่าสนใจคือโครงสร้างเครือข่ายทางสังคมของ swarm มีความสามารถในการ self-organize เพื่อสร้างโครงสร้างที่เหมาะสม การกระจายงาน การสะสมอาหาร และอื่นๆ

Self-organization เป็นเซตของกลไกพลวัต (dynamical mechanism) ที่โครงสร้างที่ปรากฏที่ระดับในวงกว้างของระบบมาจากการโต้ตอบขององค์ประกอบในระดับล่าง กฎที่ใช้ในการโต้ตอบเป็นการกระทำโดยใช้ข้อมูลเฉพาะที่เท่านั้น ไม่ได้ใช้รูปแบบส่วนกลางเลย ซึ่งนี่เป็นคุณสมบัติของระบบที่เกิดขึ้นเองไม่ได้เกิดขึ้นเนื่องจากผลกระทบภายนอกที่ใส่เข้าไป self-organization มี ส่วนประกอบ 4 ส่วนคือ

1. Positive feedback (amplification) เป็นพฤติกรรมพื้นฐานที่ต้องการส่งเสริมการสร้างโครงสร้าง ซึ่งรวมถึงการสรรหา (recruitment) และการเสริมกำลัง (reinforcement) ตัวอย่างของการสรรหาเช่น การสรรหาแหล่งอาหารเป็น positive feedback ที่ขึ้นกับการวางทางเดินและการตามทางเดินในมดบางขนิดเป็นต้น
2. Negative feedback เป็นสิ่งที่ตรงกันข้ามกับ positive feedback เพื่อช่วยทำให้รูปแบบที่สะสมไว้สมดุลย์กัน negative feedback เกิดจากแหล่งอาหารมีตัวตนหนาแน่น เป็นต้น
3. Fluctuation เช่น random walk ความผิดพลาด random task-switching และอื่นๆ ไม่เพียงแต่โครงสร้างเกิดจากการสุ่ม (randomness) แต่การสุ่มยังเป็นสิ่งที่สำคัญเนื่องจากสามารถค้นพบคำตอบใหม่ได้ และ fluctuation สามารถเป็นจุดกำเนิดของการเติบโตของโครงสร้างได้
4. Multiple interaction แต่ละตัวตนสามารถสร้างโครงสร้างที่เป็น self-organize ได้เช่น สร้างทางเดินโดยใช้ pheromone ซึ่งการเดินตามทางสามารถทำได้โดยดารโต้ตอบจากการวางทางนั่นเอง

4.1.1 Particle swarm optimization (PSO)

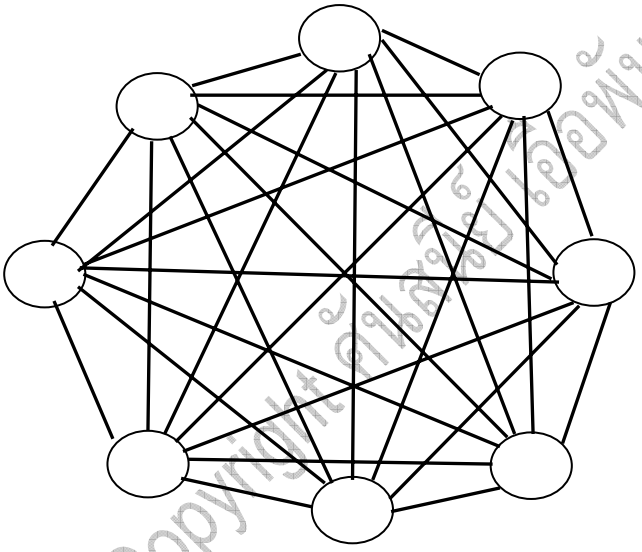
Particle swarm optimization (PSO) เป็นอัลกอริทึมในการหาโดยมีประชากรเป็นพื้นฐาน ซึ่งกระบวนการนี้ลอกเลียนแบบมาจากพฤติกรรมทางสังคมของนกในฝูงนก อัลกอริทึมนี้ถูกสร้างขึ้นมาครั้งแรกเพื่อค้นหารูปแบบที่ควบคุมนกให้บินแบบซิงโครนัส (synchronous) และเมื่อมีการเปลี่ยนทิศทางก็สามารถกลับมารวมกลุ่มกันในรูปแบบที่เหมาะสม [Kennedy95]

ใน PSO แต่ละตัวตนถูกเรียกว่าอนุภาค จะเคลื่อนที่ในปริภูมิการหา (search space) แบบ hyperdimension การเปลี่ยนตำแหน่งของอนุภาคในปริภูมิการหาเกิดจากการที่แต่ละตัวตนพยายามเลียนแบบความสำเร็จของตัวตนอื่น ดังนั้นประสบการณ์และความรู้ของเพื่อนบ้านมีผลกับการเปลี่ยนของอนุภาคใน swarm

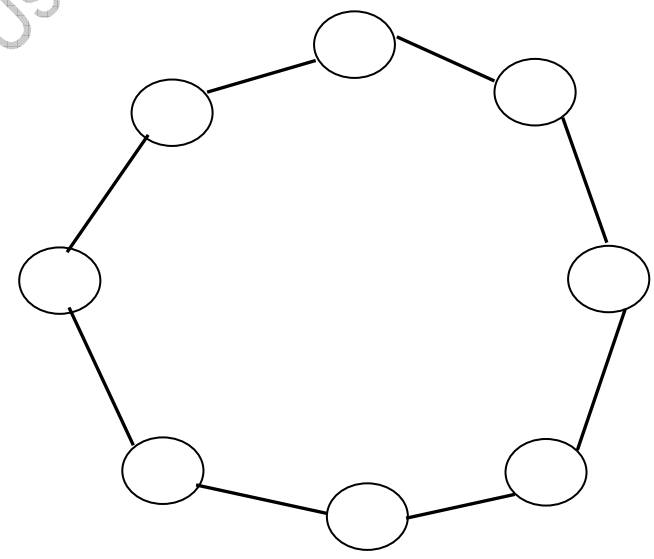
แต่ละตัวตนใน swarm เรียนจากตัวตนอื่นตามความรู้ที่ตัวตนนั้นสั่งสม และพยายามปรับตัวเองให้คล้ายคลึงกับเพื่อนบ้าน (neighbor) ที่ดีที่สุด โครงสร้างทางสังคมของ PSO ถูกกำหนดจากลักษณะการก่อตัวของรอบบ้าน (neighborhood) แต่ละตนในรอบบ้านเดียวกันสามารถโต้ตอบกันได้ ลักษณะของรอบบ้านที่จะกล่าวถึงมี 3 ชนิด ซึ่งลักษณะเหล่านี้กำหนดให้อนุภาคใดอยู่ที่ใดตามป้ายที่ติดกับแต่ละอนุภาค ไม่ใช่ข้อมูลทางทอพอโลยี (topology) เช่นระยะยูคริเดียน

1. Star topology แต่ละอนุภาคสามารถคุยได้กับทุกอนุภาค เนื่องจากเครือข่ายทางสังคมก่อตัวในลักษณะที่เป็นการเชื่อมต่อแบบสมบูรณ์ ดังรูปที่ 4.1(ก) ในกรณีนี้แต่ละอนุภาคจะพยายามเคลื่อนที่ไปหาอนุภาคที่ดีที่สุด (คำตอบที่ดีที่สุด) ในสมาชิกทั้งหมดของ swarm ดังนั้น PSO รุ่น (version) นี้จะเป็นอัลกอริทึมที่ถูกเรียกว่า gbest

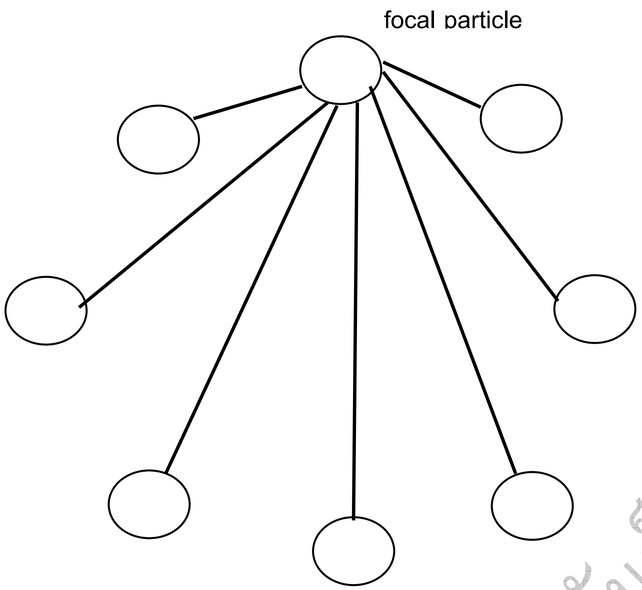
2. Ring topology ในกรณีนี้แต่ละอนุภาคจะสื่อสารกับเพื่อนบ้านที่ติดกัน *n* เพื่อนบ้าน ดังนั้นถ้า *n* เท่ากับ 2 แต่ละอนุภาคจะติดต่อกับเพื่อนบ้านที่ติดกันเท่านั้น ดังแสดงในรูปที่ 4.1 (ข) แต่ละอนุภาคจะพยายามเคลื่อนที่เข้าใกล้อนุภาคที่ดีที่สุดในเพื่อนบ้าน PSO รุ่นนี้จะเป็นอัลกอริทึมที่ถูกเรียกว่า lbest ซึ่งอัลกอริทึมนี้สามารถปรับตัวเป็นทั้งการเคบื่อนที่เข้าหาเพื่อนบ้านที่ดีที่สุดในขณะเดียวกันเคลื่อนที่เข้าใกล้อนุภาคที่ดีที่สุดใน swarm ซึ่งโครงสร้างแบบวงกลมนี้มีข้อดีคือสามารถเดินทางหาคำตอบได้ในพื้นที่ที่กว้าง แต่การลู่เข้าจะช้าลง
3. Wheel topology ในกรณีนี้มีอนุภาคเพียงอนุภาคเดียวที่เชื่อมต่อกับอนุภาคอื่น ดังแสดงในรูปที่ 4.1(ค) ซึ่งอนุภาคนี้เรียกว่า focal particle และอนุภาคนี้จะปรับตำแหน่งเข้าหาอนุภาคที่ดีที่สุด และถ้าการเปลี่ยนแปลงนี้ทำให้ประสิทธภาพเพิ่มขึ้น การเพิ่มนี้จะถูกสื่อสารไปให้ยังอนุภาคอื่น



(ก)



(ข)



(ค)

รูปที่ 4.1 (ก) Star topology (ข) Ring topology ที่มี n = 2 (ค) Wheel topology

Swarm ประกอบด้วยเซตของอนุภาค และแต่ละอนุภาคแทนคำตอบที่เป็นไปได้ อนุภาคจะเคลื่อนที่ไปในปริภูมิไฮเพอร์ (hyperspace) โดยที่ตำแหน่งของอนุภาคจะเปลี่ยนตามประสบการณ์ของนุภาคและของเพื่อนบ้าน ให้ $\vec{x}_i(t)$ แทนตำแหน่งของอนุภาค P_i ในปริภูมิไฮเพอร์ที่เวลา t ตำแหน่งของ P_i จะเปลี่ยนโดยใช้ความเร็ว $\vec{v}_i(t)$ ดังนี้

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{4.1}$$

เวกเตอร์ความเร็วเป็นสิ่งที่ผลักดันกระบวนการหาคำตอบที่เหมาะสมที่สะท้อนถึงการแลกเปลี่ยนข้อมูลทางสังคม อัลกอริทึม PSO ที่จะกล่าวถึงต่อไปนี้มีด้วยกีน 3 อัลกอริทึมดังนี้

Individual Best

ในอัลกอริทึมนี้แต่ละอนุภาคจะเปลี่ยนตำแหน่ง โดยใช้ประสบการณ์ของอนุภาคเองเท่านั้น (*pbest*) และไม่ใช้ข้อมูลจากอนุภาคอื่นดังแสดงในรูปที่ 4.2 ซึ่งในกรณีนี้การปรับความเร็วของแต่ละอนุภาคเป็นไปดังนี้

$$\vec{v}_i(t) = \vec{v}_i(t-1) + \rho\left(\vec{x}_{pbest_i}(t) - \vec{x}_i(t)\right) \tag{4.2}$$

โดยที่ ρ เป็ตัวเลขบวกที่สุ่มมา

ถ้าอนุภาคเคลื่อนที่ออกห่างจากจุดที่เป็นคำตอบที่ดีที่สุดที่เคยเจอแล้วจะมีการเปลี่ยนแปลงความเร็วมากทำให้อนุภาคเคลื่อนเข้าใกล้คำตอบที่ดีที่สุด ค่าสูงสุดของ ρ โดยปกติจะเป็นคนที่ใช้อัลกอริทึมนี้เป็นคนกำหนด แต่ถ้าค่า ρ มากเกินไปทำให้การเคลื่อนที่ของอนุภาคแกว่งได้ แต่ถ้าค่า ρ น้อยจะทำให้การเคลื่อนที่ค่อนข้างเรียบ

1. Initialize swarm ($P(t)$) โดยที่ $\vec{x}_i(t)$ ของแต่ละอนุภาค $P_i \in P(t)$ ถูกสุ่มมาในปริภูมิไฮเพอร์ และให้ $t = 0$
2. คำนวณค่าประสิทธิภาพ $\mathcal{F}$ ของแต่ละอนุภาค โดยใช้ตำแหน่งปัจจุบัน
3. เปรียบเทียบค่าประสิทธิภาพของแต่ละอนุภาคกับประสิทธิภาพที่ดีที่สุด

   If $\mathcal{F}(\vec{x}_i(t)) < pbest_i$ then

   $pbest_i = \mathcal{F}(\vec{x}_i(t))$

   $\vec{x}_{pbest_i}(t) = \vec{x}_i(t)$
4. ปรับความเร็วของแต่ละอนุภาคโดยใช้สมการที่ 4.2
5. แต่ละอนุภาคเคลื่อนที่ไปที่ตำแหน่งใหม่

   $\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$

   $t = t + 1$
6. ไปที่ข้อ 2 และทำซ้ำจนกว่าจะลู่เข้า (converge)

รูปที่ 4.2 *pbest* อัลกอริทึม

Global Best

อัลกอริทึมที่เป็นเครือข่ายของเพื่อนบ้านแบบ star topology นั่นคืออนุภาคจะเคลื่อนเข้าหาอนุภาคที่ดีที่สุดใน swarm (*gbest*) ในขณะที่ยังคงใช้ตำแหน่งที่ดีที่สุดของตัวอนุภาคเองด้วย ดังแสดงดังรูปที่ 4.3 ซึ่งในกรณีนี้การปรับความเร็วของแต่ละอนุภาคคือ

$$\vec{v}_i(t) = \vec{v}_i(t-1) + \rho_1\left(\vec{x}_{pbest_i}(t) - \vec{x}_i(t)\right) + \rho_2\left(\vec{x}_{gbest}(t) - \vec{x}_i(t)\right) \quad (4.3)$$

โดยที่ ρ_1 และ ρ_2 เป็นตัวเลขบวกที่สุ่มมา เทอมที่ 2 ในสมการ (4.3) เกี่ยวข้องกับองค์ประกอบความรู้ของตัวอนุภาค และเทอมที่ 3 ในสมการ (4.3) เกี่ยวข้องกับองค์ประกอบทางสังคม

ถ้าอนุภาคเคลื่อนที่ไปไกลกว่าตำแหน่งที่ดีที่สุดของทั้ง swarm และตำแหน่งที่ดีที่สุดของตัวอนุภาค ความเร็วของอนุภาคจะเปลี่ยนมาก เพื่อเคลื่อนที่อนุภาคกลับมาตำแหน่งที่ดีที่สุด ค่า ρ_1 และ ρ_2 ถูกนิยามโดยที่ $\rho_1 = r_1 c_1$ และ $\rho_2 = r_2 c_2$ โดยที่ r_1 และ r_2 ถูกสุ่มจาก uniform distribution $U(0,1)$ และค่า c_1 และ c_2 เป็นค่าความเร่งบวกคงที่ และได้มีการพิสูจน์ว่าเพื่อให้ได้การเคลื่อนที่ที่ดีค่า c_1 และ c_2 ควรจะเป็นดังนี้

$$c_1 + c_2 \leq 4 \quad (4.4)$$

ถ้า $c_1 + c_2 > 4$ จะทำให้ความเร็วและตำแหน่งจะพุ่งเข้าหาอนันต์ (infinity) [Kennedy98]

Local Best

อัลกอริทึมนี้ถูกเรียกว่า *lbest* ซึ่งเป็นเครือข่ายเพื่อนบ้านแบบวงกลม อนุภาคจะเคลื่อนที่ตามเพื่อนบ้านที่ดีที่สุด และประสบการณ์ของตัวอนุภาคเอง ดังนั้นอัลกอริทึมจะคล้ายกับ *gbest* เพียงแต่เปลี่ยนข้อที่ 4 และ 5 ในรูปที่ 4.3 จาก *gbest* เป็น *lbest* โดยพิจารณาเฉพาะเพื่อนบ้านตามโครงสร้าง

เพื่อนบ้านเท่านั้น อัลกอริทึมนี้จะลู่เข้าช้ากว่า *gbest* แต่ให้คำตอบที่ดีกว่าและเคลื่อนที่ได้ในพื้นที่ที่กว้างในปริภูมิการหา

1. Initialize swarm ($P(t)$) โดยที่ $\vec{x}_i(t)$ ของแต่ละอนุภาค $P_i \in P(t)$ ถูกสุ่มมาในปริภูมิไฮเพอร์ และให้ $t = 0$
2. คำนวณค่าประสิทธิภาพ $\mathcal{F}$ ของแต่ละอนุภาค โดยใช้ตำแหน่งปัจจุบัน
3. เปรียบเทียบค่าประสิทธิภาพของแต่ละอนุภาคกับประสิทธิภาพที่ดีที่สุด

   If $\mathcal{F}(\vec{x}_i(t)) <$ $pbest_i$ then

   $pbest_i = \mathcal{F}(\vec{x}_i(t))$

   $\vec{x}_{pbest_i}(t) = \vec{x}_i(t)$
4. เปรียบเทียบค่าประสิทธิภาพของแต่ละอนุภาคกับประสิทธิภาพที่ดีที่สุดของทั้ง swarm

   If $\mathcal{F}(\vec{x}_i(t)) <$ *gbest* then

   $gbest = \mathcal{F}(\vec{x}_i(t))$

   $\vec{x}_{gbest}(t) = \vec{x}_i(t)$
5. ปรับความเร็วตามสมการที่ 4.3
6. แต่ละอนุภาคเคลื่อนที่ไปที่ตำแหน่งใหม่

   $\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$

   $t = t + 1$
7. ไปที่ข้อ 2 และทำซ้ำจนกว่าจะลู่เข้า (converge)

รูปที่ 4.3 *gbest* อัลกอริทึม

ไม่ว่าจะเป็นอัลกอริทึมแบบใดสิ่งที่ต้องคำนวณคือประสิทธิภาพ ซึ่งที่จริงเป็นการคำนวณหาค่าความเหมาะสม (fitness value) ซึ่งค่านี้เป็นการวัดว่าคำตอบนี้เป็นคำตอบที่ใกล้กับคำตอบที่เหมาะสมเท่าไร เช่นต้องการหาค่า x_1 และ x_2 ที่ทำให้ฟังก์ชัน $f(x_1, x_2) = \sin x_1 \sin x_2 \sqrt{x_1 x_2}$ มีค่าน้อยที่สุด ดังนั้นฟังก์ชันค่าความเหมาะสม (fitness function) ก็คือฟังก์ชัน $f(x_1, x_2)$ นั่นเอง

ส่วนคำถามที่ว่าเมื่อไร PSO ควรจะถูกพิจารณาว่าเป็นการลู่เข้าแล้ว โดยปกติอัลกอริทึมจะทำงานต่อไปจนกว่าจำนวนครั้งที่ทำซ้ำ (iteration) เท่ากับจำนวนครั้งที่ทำซ้ำที่มากที่สุดที่ตั้งไว้ตั้งแต่ทีแรก หรือได้ค่าความเหมาะสมที่ดีที่สุดแล้ว หรือความเร็วของทุกอนุภาคมีการเปลี่ยนแปลงน้อยมากนั่นเอง

จากอัลกอริทึมที่กล่าวถึงข้างต้นจะเห็นได้ว่า *gbest* เป็น *lbest* ประเภทหนึ่งโดยที่ทั้ง swarm เป็นเอนบ้าน แต่ *gbest* มีข้อเสียคืออาจจะให้คำตอบที่เป็นเพียงค่าต่ำสุดเฉพาะที่ (local minima) เพราะทุกอนุภาคจะถูกดึงเข้าหาคำตอบนั้นทั้งหมด แต่ถ้าเพื่อนบ้านมีจำนวนน้อย จะทำให้มีจำนวนกลุ่มเพื่อนบ้านมากทำให้พื้นที่ในการค้นหาเป็นไปได้กว้างขวางขึ้น ทำให้ PSO มีโอกาสที่จะติดที่ค่าต่ำสุดเฉพาะที่ น้อย แต่ข้อเสียคือจะลู่เข้าช้า

ในบางครั้งการตั้งค่าสูงสุดของความเร็ว (V_{max}) ในทุกมิติจะช่วยป้องกันไม่ให้อนุภาคเคลื่อนที่จากพื้นที่หนึ่งไปยังอีกพื้นที่หนึ่งในปริภูมิการหาเร็วเกินไป นั่นคือ ถ้า $v_{ij}(t) > V_{max}$ ตั้งให้ $v_{ij}(t) = V_{max}$

หรือถ้า $v_{ij}(t) < -V_{max}$ ตั้งให้ $v_{ij}(t) = -V_{max}$ โดยที่ $v_{ij}(t)$ คือความเร็วของอนุภาค P_i ที่มิติ j ที่เวลา t และโดยปกติ V_{max} จะเป็นฟังก์ชันของเรนจ์ของปัญหาเช่น x_{ij} อยู่ในช่วง [-50, 50] V_{max} จะแปรผันตรงกับ 50 นั่นเอง แต่ได้มีงานวิจัยอื่น [Clerc02] ที่กล่าวว่าไม่จำเป็นต้องใช้ v_{max} ถ้า

$$\vec{v}_i(t) = \kappa\left[\vec{v}_i(t-1) + \rho_1\left(\vec{x}_{pbest_i}(t) - \vec{x}_i(t)\right) + \rho_2\left(\vec{x}_{gbest}(t) - \vec{x}_i(t)\right)\right] \quad (4.5)$$

โดยที่

$$\kappa = 1 - \frac{1}{\rho} + \frac{\sqrt{\left|\rho^2 - 4\rho\right|}}{2} \quad (4.6)$$

ซึ่ง $\rho = \rho_1 + \rho_2 > 4.0$

ในบางครั้งในการเพิ่มประสิทธิภาพอาจมาจากการที่ควบคุมผลกระทบจากความเร็วเมื่อรอบที่แล้ว ทำได้โดยการเพิ่มค่า Inertia weight ดังนี้

$$\vec{v}_i(t) = \phi\vec{v}_i(t-1) + \rho_1\left(\vec{x}_{pbest_i}(t) - \vec{x}_i(t)\right) + \rho_2\left(\vec{x}_{gbest}(t) - \vec{x}_i(t)\right) \quad (4.7)$$

โดยที่ ϕ คือ inertia weight ซึ่งเป็นค่าที่ควบคุมอิทธิพลของความเร็วเมื่อครั้งที่แล้วที่จะมีผลต่อความเร็วปัจจุบัน ถ้าค่า inertia weight มีค่ามากจะทำให้มีการค้นหาในพื้นที่ที่กว้าง ในขณะที่ถ้าค่าน้อยจะมีพื้นที่ในการค้นหาน้อย และจากการศึกษาพบว่าเพื่อเป็นการประกันการลู่เข้าเงื่อนไขต่อไปนี้ควรจะเป็นจริง

$$\phi > \frac{1}{2}(c_1 + c_2) - 1 \quad (4.8)$$

โดยที่ $\phi \leq 1$ [Clerc02]

ในบางครั้งได้มีการปรับอัลกอริทึม PSO เพื่อพัฒนาประสิทธิภาพของ PSO เช่นการใช้กระบวนการ selection หรือ breeding ซึ่งเป็นกระบวนการที่คล้ายกับกระบวนการใน Genetic Algorithms กระบวนการ selection แสดงในรูปที่ 4.4

1. สำหรับแต่ละอนุภาคใน swarm ให้คำนวณค่าประสิทธิภาพของอนุภาคตามกลุ่มที่ถูกเลือกมาแบบสุ่ม ซึ่งในกลุ่มมี *k* อนุภาค
2. จัดอันดับอนุภาคตามคะแนนที่ได้ในข้อ 1
3. เลือกอนุภาคในครึ่งบนและสำเนา (copy) ตำแหน่งของอนุภาคเหล่านี้ไปยังอนุภาคในครึ่งหลังของ swarm โดยไม่มีการเปลี่ยน *pbest* ของอนุภาคในครึ่ง

รูปที่ 4.4 กระบวนการ selection

ซึ่งกระบวนการนี้ทำก่อนที่จะมีการคำนวรหาการปรับความเร็ว ส่วนกระบวนการ breeding หรือ reproduction ทำได้โดยการเลือกอนุภาค 2 อนุภาค (P_a และ P_b) ด้วยกระบวนการเลือกคล้ายใน Genetic Algorithms (ซึ่งจะไม่ขอกล่าวในที่นี้) เพื่อสร้างอนุภาคลูก 2 อนุภาค ดังนี้

$$\vec{x}_a(t+1) = r_1\vec{x}_a(t) + (1-r_1)\vec{x}_b(t) \quad (4.9ก)$$

$$\vec{x}_b(t+1) = r_1\vec{x}_b(t) + (1-r_1)\vec{x}_a(t) \quad (4.9ข)$$

$$\vec{v}_a(t+1) = \frac{\vec{v}_a(t) + \vec{v}_b(t)}{\left\|\vec{v}_a(t) + \vec{v}_b(t)\right\|}\left\|\vec{v}_a(t)\right\| \quad (4.9ค)$$

$$\vec{v}_b(t+1) = \frac{\vec{v}_a(t) + \vec{v}_b(t)}{\left\|\vec{v}_a(t) + \vec{v}_b(t)\right\|}\left\|\vec{v}_b(t)\right\| \tag{4.9ง}$$

โดยที่ r_1 ถูกสุ่มจาก uniform distribution $U(0,1)$ อัลกอริทึมของการ breeding แสดงในรูปที่ 4.5

1. คำนวณหาตำแหน่งและความเร็วของอนุภาค
2. แต่ละอนุภาคจะถูกกำนดค่า breeding probability (p_b)
3. เลือกอนุภาค 2 อนุภาค (P_a และ P_b) เพื่อสร้างอนุภาคลูก 2 อนุภาคโดยใช้สมการที่ 4.9
4. ตั้งค่า *pbest* ของแต่ละอนุภาคที่เกี่ยวข้องในกระบวนการนี้ด้วยค่าตำแหน่งปัจจุบัน

รูปที่ 4.5 breeding อัลกอริทึม

ในกระบวนการ breeding นี้ป้องกันไม่ให้อนุภาคที่ดีที่สุดมีบทบาทในกระบวนการนี้มากเกินไป และเป็นการป้องกันไม่ให้ลู่เข้าเร็วเกินไป เพราะไม่ได้ใช้ค่าความเหมาะสมเลย

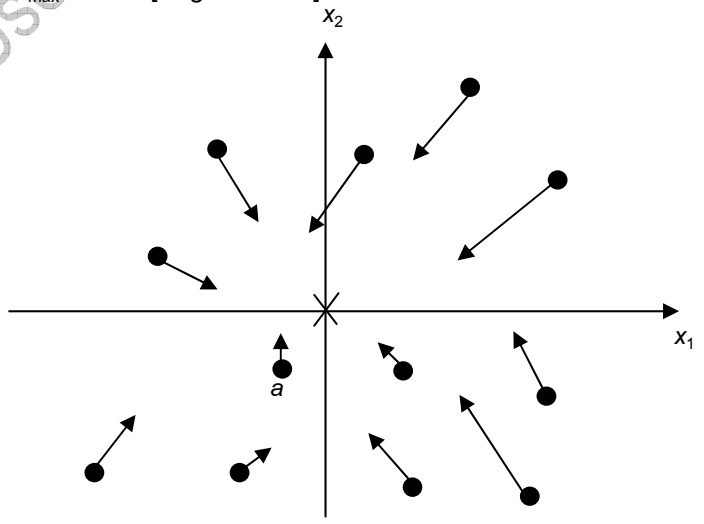
ใน topology ที่ใช้มีการเลือกตำแหน่งของอนุภาคในเพื่อนบ้านตามค่าดรรชนี (index) แต่ก็มีบางงานวิจัยที่ใช้ระยะระหว่างอนุภาคในการกำหนดตำแหน่ง นั่นคืออนุภาค P_b เป็นเพื่อนบ้านของ P_a ถ้า

$$\frac{\left\|\vec{x}_a - \vec{x}_b\right\|}{d_{\max}} < \xi \tag{4.10}$$

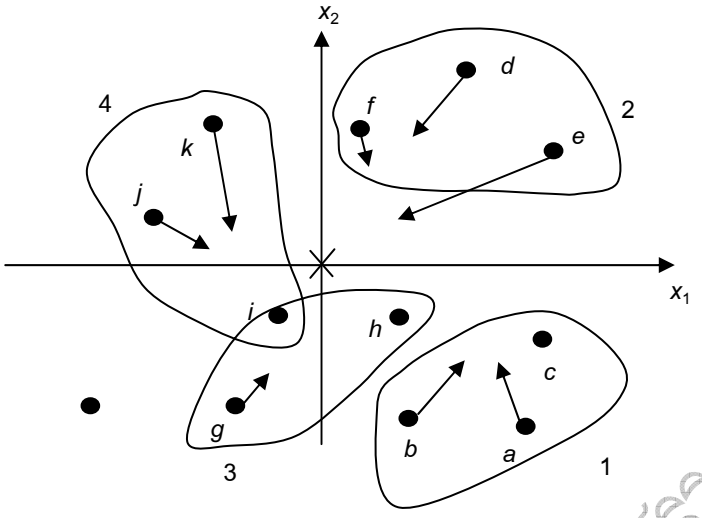
โดยที่ $d_{\max}$ คือระยะที่ไกลที่สุดระหว่าง 2 อนุภาค และ

$$\xi = \frac{3t + 0.6t_{\max}}{t_{\max}} \tag{4.11}$$

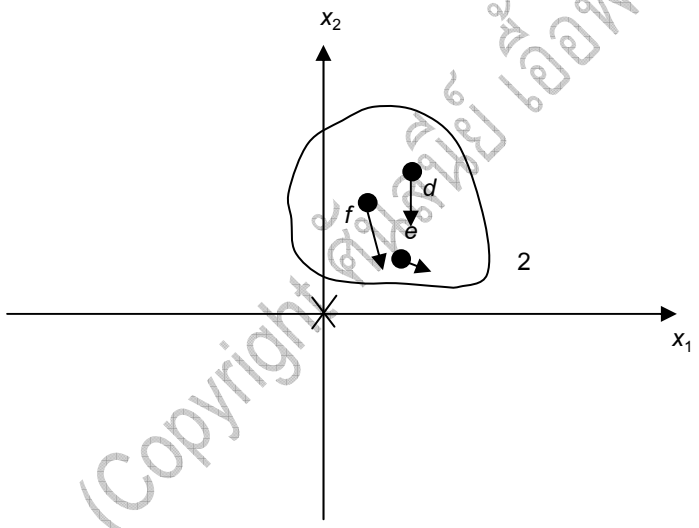
โดยที่ t คือจำนวนรอบ (iteration number)ปัจจุบัน และ $t_{\max}$ เป็นจำนวนรอบที่มากที่สุด ดังนั้นจะเห็นว่าเริ่มแรกจะมีเพื่อนบ้านน้อย และเมื่อเวลาผ่านไปจำนวนเพื่อนบ้านจะเยอะขึ้นจนกระทั่งกลายเป็น *gbest* ในขณะที่ $t \rightarrow t_{\max}$ นั่นเอง [Suganthan99]



(ก)



(ข)



(ค)

รูปที่ 4.6 (ก) *gbest* (ข) รอบแรกของ *lbest* (ค) รอบที่สองของ *lbest*

ตัวอย่างที่ 4.1 ต้องการหาค่า x_1 และ x_2 ที่ทำให้ $f\left(x_1, x_2\right) = x_1^2 + x_2^2$ น้อยที่สุดซึ่งคำตอบควรจะเป็น $x_1 = 0$ และ $x_2 = 0$ ถ้าใช้ *gbest* แสดงในรูปที่ 4.6 (ก) จะเห็นว่าอนุภาค *a* เป็นอนุภาคที่ดีที่สุดและในรอบแรก *pbest* ของแต่ละอนุภาคคือตำแหน่งปัจจุบัน ดังนั้นอนุภาค *a* เท่านั้นที่มีผลต่อการเคลื่อนที่ของอนุภาคอื่น นั่นคืออนุภาคอื่นเคลื่อนที่เข้าหา *a* นั่นเอง ถ้าใช้ *lbest* แสดงในรูปที่ 4.6 (ข) จะเห็นว่า ในกลุ่มที่ 1 อนุภาค *a* และ *b* เคลื่อนที่เข้าหาอนุภาค *c* ซึ่งเป็นคำตอบที่ดีที่สุดในกลุ่มนี้ ส่วนกลุ่มที่ 2 อนุภาค *d* และ e เคลื่อนที่เข้าหา *f* แต่ในรอบถัดไป อนุภาค *e* เป็นคำตอบที่ดีที่สุด ดังนั้น *d* และ *f* จะเคลื่อนที่เข้าหา *e* ดังแสดงในรูปที่ 4.6 (ค) ■

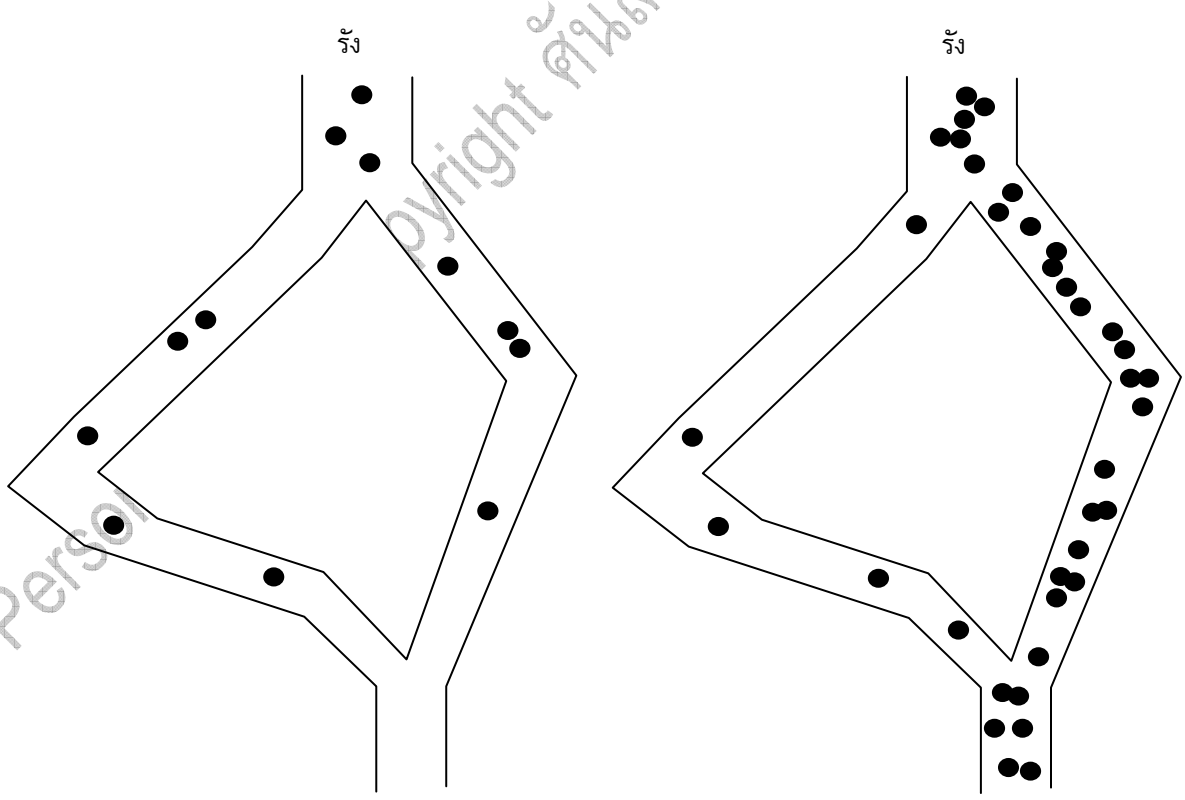
4.1.2 Ant Colony Optimization (ACO)

กระบวนการทำงานใน ant colony เกี่ยวข้องกับงานหลายประเภทดังเช่น

1. การสืบพันธุ์ (reproduction): งานของนางพญามด
2. การป้องกัน (defense): งานของมดทหาร
3. การหาอาหาร (food collection): งานของมดงานที่มีหน้าที่เฉพาะ
4. การดูแลตัวอ่อน (brood care) : งานของมดงานที่มีหน้าที่เฉพาะ
5. การดูแลรังรวมทั้งดูแลสุสาน (nest brooming including cemetery maintenance): : งานของมดงานที่มีหน้าที่เฉพาะ
6. การสร้างและซ่อมแซมรัง (nest building and maintenance): : งานของมดงานที่มีหน้าที่เฉพาะ

การกระจายและการทำงานขึ้นอยู่กับความแตกต่างทางกายวิภาค (anatomy) และ stignergy พฤติกรรมการกระจายภายใน colony ถูกเรียกว่า stignergy และ stigmergy ในธรรมชาติมีลักษณะดังต่อไปนี้

1. การขาดการประสานงานตรงกลาง
2. การสื่อสารและการประสานงานระหว่างแต่ละตัวตนใน colony ขึ้นอยู่กับการปรับสภาวะแวดล้อมเฉพาะที่
3. positive feedback ซึ่งเป็นการเสริมการกระทำ เช่นการเดินตามทางเพื่อเก็บอาหาร



รูปที่ 4.7 มดเอินตามทางที่มี pheromone

มดมีความสามารถในการหาเส้นทางที่สั้นที่สุดระหว่างรังและแหล่งอาหาร ได้มีการทดลองหลายครั้งโดยการสร้างเส้นทางระหว่างรังและอาหารหลายเส้นทางดังเช่นรูปที่ 4.7 ซึ่งเมื่อเวลาผ่านไปมดจะเดินไปในทางที่สั้นที่สุดเสมอ พฤติกรรมนี้สามารถอธิบายได้คือ มดจะปล่อย pheromone ไปตามทางระหว่างทางไปหาอาหาร และระหว่างเดินกลับรัง และในการเลือกเส้นทางมดจะเลือกเดินทางที่มี pheromone มากกว่าเสมอ ซึ่งทางที่สั้นกว่าจะมี pheromone มากกว่าเนื่องจากมดเดินกลับมาที่รังได้เร็วกว่ามดที่เดินไปตามทางที่ยาวกว่า และ pheromone จะระเหยไปตามเวลาทำให้ pheromone ในทางที่ยาวกว่าน้อยลงไปตามเวลาและไม่มีเลยในที่สุด

การประยุกต์ใช้งานของ ant colony optimization (ACO) คือใช้ในการแก้ปัญหา Traveling salesman problem (TSP) ซึ่งสาเหตุที่ ACO ถูกใช้ในการแก้ปัญหา TSP คือ

1. ปัญหานี้เป็นการหาเส้นทางที่สั้นที่สุดซึ่งกระบวนการทำงานของ ant colony สามารถปรับให้แก้ปัญหานี้ได้ง่าย
2. ปัญหานี้เป็น NP-hard problem
3. มีการศึกษาเกี่ยวกับการแก้ปัญหานี้มากจนกระทั่งปัญหานี้กลายเป็นปัญหาที่ใช้เทียบวัดประสิทธิภาพ (benchmark problem)
4. การอธิบายพฤติกรรมของอัลกอริทึมในการแก้ปัญหานี้ทำได้ง่าย

ก่อนที่จะอธิบายถึงการนำ ACO ไปใช้ในการแก้ปัญหานี้จะกล่าวถึงความคิดพื้นฐานของอัลกอริทึมที่เป็น ant-based นั่นคืออัลกอริทึมเหล่านี้ใช้ positive feedback เช่นพฤติกรรมการวางทาง (trail-laying) การตามทาง (trail-following) ของมดบางชนิด เพื่อเป็นการเสริมการหาคำตอบที่ดี ความคิดต่อมาคือ virtual pheromone ใช้ในการเสริม นั่นคือคำตอบที่ดีจะถูกในความจำเพื่อที่อาจจะถูกใช้เพื่อกลายเป็นคำตอบที่ดีขึ้น และเนื่องจากคำตอบที่ดีอาจจะไม่ใช่คำตอบที่ดีที่สุด ดังนั้นจึงต้องใช้ negative feedback หรือการทำให้ pheromone ระเหยไป เพื่อป้องกันไม่ให้เกิดการลู่เข้าก่อนเวลาอันควร ซึ่งในกระบวนการนี้จะต้องใช้ มาตราเวลา (time scale) แต่ถ้ามาตราเวลามีค่ามากเกินไปจะทำให้เกิดการลู่เข้าก่อนเวลาอันควรและลู่เข้าหาคำตอบที่เหมาะสมเฉพาะที่ และถ้ามาตราเวลามีค่าน้อยเกินไปอาจจะไม่มีพฤติกรรมการทำงานร่วมกันเกิดขึ้น ซึ่งพฤติกรรมนี้ใช้การสำรวจคำตอบที่ต่างกันของมดที่เกิดขึ้นพร้อมกัน มดที่ทำงานดีจะมีอิทธิพลกับการสำรวจของมดในรอบถัดไป และเนื่องจากมดสำรวจคำตอบที่ต่างกัน ทาง pheromone จะเป็นสิ่งที่เกิดจากมุมมองที่ต่างใน search space และเมื่อมดที่ทำงานดีที่สุดเป็นสิ่งที่เสริมคำตอบ ก็ยังคงมีการทำงานร่วมกันในส่วนของเวลาเนื่องจากมดในรอบถัดไปจะใช้ทาง pheromone นำทางการสำรวจ

Ant System (AS)

เป้าหมายใน TSP คือต้องการหาเส้นทางเดินที่สั้นที่สุดในการไปยังเมืองต่างๆ n เมืองและแต่ละเมืองจะต้องผ่านแค่ 1 ครั้งเท่านั้น ปัญหานี้อาจถูกนิยามในมิติยูคริเดียน (Euclidean space) ซึ่งสามารถหาระยะจากเมือง i ไปยังเมือง j ดังสมการที่ 4.12

$$d_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{4.12}$$

โดยที่ (x_i, y_i) และ (x_j, y_j) เป็นพิกัด (coordinate) ของเมือง i และเมือง j ตามลำดับ แต่ในบางกรณีอาจจะการเดินทางจากเมือง i ไปยังเมือง j ใช้ระยะที่ไม่เท่ากับการเดินทางจากเมือง j และเมือง i (d_{ij}

$\neq d_{ji}$) ซึ่งในกรณีนี้ปัญหานี้จะกลายเป็น asymmetric TSP แต่ไม่ว่าจะเป็น symmetric หรือ asymmetric TSP ก็ไม่ทำให้การแก้ปัญหาของ AS เปลี่ยนไป

ให้กระบวนการนี้มีมด m ตัว และในแต่ละรอบมดแต่ละตัวต้องเดินทางให้ครบทุกเมืองโดยต้องทำงาน $n = |N|$ ขั้นและกระบวนกานี้จะสิ้นสุดถ้าทำงานทั้งหมด t_{max} รอบ ในการที่มดตัดสินใจว่าจะเดินทางจากเมือง i ไปยังเมือง j ในรอบที่ t ขึ้นอยู่กับ

1. เมืองนั้นถูกเดินผ่านไปหรือยัง มดแต่ละตัวจะมีความจำ (เรียกว่า tabu list) ซึ่งความจำนี้จะโตภายในการเดินทางแต่จะว่างระหว่างการเดินทางครั้งหนึ่งกับครั้งอื่น ดังนั้นเซต J_i^k ซึ่งเป็นเซตที่เก็บเมืองที่มดตัวที่ k ยังไม่ได้ไปในขณะที่อยู่ที่เมือง i (ในตอนแรกของรอบเซตนี้จะประกอบด้วยเมืองทุกเมืองยกเว้นเมือง i)
2. ส่วนกลับของระยะทางดังสมการที่ 4.13 เรียกว่าทัศนวิสัย (visibility) ซึ่งเป็นค่าที่ขึ้นอยู่กับข้อมูลเฉพาะที่ (local information) และเป็นตัวแทนของความต้องการสำนึก (heuristic desirability) ที่ต้องการเลือกเมือง j จากเมือง i แต่ค่านี้จะไม่มีการเปลี่ยนตลอดการหาคำตอบ

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{4.13}$$

3. จำนวนของทาง virtual pheromone $\tau_{ij}(t)$ ที่อยู่บนเส้นเชื่อมระหว่างเมือง i ไปยังเมือง j ค่านี้เป็นตัวแทนของความต้องการเลือกเมือง j จากเมือง i ที่เรียนได้ ซึ่งค่านี้เป็นข้อมูลส่วนรวม (global information) ค่านี้จะเปลี่ยนระหว่างการแก้ปัญหาโดยสะท้อนถึงประสบการณ์ของมด

โอกาสที่มดตัวที่ k เลือกเมือง j จากเมือง j ในระหว่างการเดินในรอบที่ t เป็นไปตาม กฎการแทรนซิชัน (transition rule หรือ random proportional transition rule) ซึ่งเป็นดังสมการที่ 4.14

$$p_{ij}^k(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^\alpha \left[\eta_{ij}\right]^\beta}{\sum_{l \in J_i^k} \left[\tau_{il}(t)\right]^\alpha \left[\eta_{il}\right]^\beta} & \text{if } j \in J_i^k \\ 0 & \text{else} \end{cases} \tag{4.14}$$

โดยที่ α และ β เป็นตัวแปรที่ปรับได้ที่จะควบคุมความเข้มของทาง (trail intensity)และทัศนวิสัย ถ้า $\alpha = 0$ เมืองที่ใกล้ที่สุดจะมีโอกาสถูกเลือกมากกว่า แต่ถ้า $\beta = 0$ จะเลือกสิ่งที่มาจาก pheromone ซึ่งอาจจะได้ทางเดินที่ไม่เหมาะสมได้ หลังจากมดเดินจนครบทุกเมืองในรอบที่ t มดตัวที่ k จะปรับค่า pheromone ของเส้นเชื่อม (i, j) ดังสมการที่ 4.15 ซึ่งค่านี้จะขึ้นกับประสิทธิภาพของมดตัวนั้น

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & \text{if } (i,j) \in T^k(t) \\ 0 & \text{if } (i,j) \notin T^k(t) \end{cases} \tag{4.15}$$

โดยที่ $T^k(t)$ คือทางเดินจนครบทุกเมืองในรอบที่ t และ $L^k(t)$ คือความยาวของทางเดินนี้ ส่วน Q คือตัวแปรที่ตั้งไว้แต่แรก

เพื่อให้มีการสำรวจ solution space อย่างมีประสิทธิภาพจึงต้องทำให้ วิธีการนี้ต้องทำให้มีการระเหยของ pheromone ไม่เช่นนั้นมดจะเดินไปในทางเดินเดิมตลอกเนื่องจากการปรับค่า pheromone นั่นเอง ดังนั้นการปรับค่า pheromone ของเส้นเชื่อม (*i*, *j*) เป็นดังสมการที่ 4.16

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{4.16}$$

โดยที่ $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)$ ซึ่ง m คือจำนวนมดทั้งหมดและ ρ เป็นสัมประสิทธิ์ของความเสื่อม (decay coefficient) จำนวนมดทั้งหมดถ้ามีทากเกินไปจะทำให้ได้เส้นทางที่เหมาะสมเฉพาะที่ซึ่งทำให้เกิดการลู่เข้าที่เร็วเกินไปและอาจจะได้คำตอบที่ไม่ดี แต่ถ้ามีน้อยเกินไปอาจจะไม่ทำให้เกิดการทำงานร่วมกัน ซึ่งได้มีงานวิจัยสรุปว่าให้ตั้งจำนวนมดเท่ากับจำนวนเมือง [Dorigo96] และงานวิจัยเดียวกันนี้ได้ใช้มดที่เป็น elitist มาช่วยในการเพิ่มประสิทธิภาพของ AS ซึ่งมดที่เป็น elitist เป็นมดที่จะเสริมเส้นทาง T^+ ซึ่งเป็นเส้นทางที่ดีที่สุดดังนั้นสมการปรับ pheromone จะเป็น

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) + e\Delta\tau_{ij}^{e}(t) \tag{4.17}$$

โดยที่

$$\Delta\tau_{ij}^{e}(t) = \begin{cases} \dfrac{Q}{L^+} & \text{if } (i,j) \in T^+ \\ 0 & \text{if } (i,j) \notin T^+ \end{cases} \tag{4.18}$$

โดยที่ L^+ เป็นความยาวของเส้นทาง T^+ ซึ่งเป็นการเพิ่ม pheromone ให้กับเส้นทางที่อยู่ในเส้นทางที่ดีที่สุดนั่นเองรูปที่ 4.8 แสดงอัลกอริทึมของ AS

1. ตั้งค่า $\tau_{ij}(0) = \tau_0$ สำหรับทุกเส้นเชื่อม (*i*, *j*)
2. For k =1 to m
   นำมดตัวที่ k ไปไว้ที่เมืองที่สุ่มมา
   End
3. ตั้ง T^+ และคำนวณ L^+
4. For t = 1 to t_{max}
   For k = 1 to m
   สร้างทาง $T^k(t)$ โดยที่ทำขั้นตอนต่อไปนี้ n-1 ครั้ง
   เลือกเมืองถัดไปจาก เมือง i คือเมือง j จากความน่าจะเป็นจากสมการที่ 4.14
   End
   คำนวณ $L^k(t)$ สำหรับทุกเส้นทาง $T^k(t)$ สำหรับ $1 \le k \le m$
   ถ้าเส้นทางที่ได้ดีกว่า T^+ ตั้ง T^+ และคำนวณ L^+ ใหม่
   ปรับค่า pheromone ตามสมการ 4.17 สำหรับทุกเส้นเชื่อม (*i*, *j*)
   ตั้งค่า $\tau_{ij}(t+1) = \tau_{ij}(t)$ สำหรับทุกเส้นเชื่อม (*i*, *j*)
   End

รูปที่ 4.8 AS อัลกอริทึม

Ant Colony System (ACS)

อัลกอริทึมนี้ถูกพัฒนาขึ้นมาเพื่อพัฒนาประสิทธิภาพของ AS นั่นคือสามารถหาคำตอบที่ดีที่สุดในเวลาอันสมควรสำหรับปัญหาที่มีขนาดใหญ่ไม่มากนัก ซึ่งอัลกอริทึมนี้แตกต่างจาก AS ที่ กฎแทรนซิชัน การปรับทางที่มี pheromone การใช้การปรับ pheromone เฉพาะที่เพื่อประโยชน์ในการสำรวจ และการใช้รายการของตัวเลือก (candidate list) กฎแทรนซิชันสำหรับกรณีนี้คือเลือกเมือง j จากเมือง i ตามสมการที่ 4.19

$$j = \begin{cases} \underset{u \in J_i^k}{\arg\max} \left\{ \left[\tau_{iu}(t) \right] \left[\eta_{iu} \right]^{\beta} \right\} & \text{if } q \le q_0 \\ J & \text{if } q > q_0 \end{cases} \tag{4.19}$$

โดยที่ q เป็นตัวแปรที่สุ่มมาจาก uniform distribution $U(0,1)$และ q_0 เป็นตัวแปรปรับได้ที่มีค่าอยู่ในช่วง $0 \le q_0 \le 1$ และ $J \in J_i^k$ เป็นเมืองที่ถูกเลือกแบบสุ่มโดยใช้สมการที่ 4.20

$$p_{iJ}^k(t) = \frac{\left[\tau_{iJ}(t) \right] \left[\eta_{iJ} \right]^{\beta}}{\sum_{l \in J_i^k} \left[\tau_{il}(t) \right] \left[\eta_{il} \right]^{\beta}} \tag{4.20}$$

ซึ่งจะเห็นได้ว่ากฎแทรยซิชันของ ACS จะคล้ายกับ AS ถ้า $q > q_0$ ซึ่งเป็นการเลือกการสำรวจมากกว่าแต่ถ้า $q \le q_0$ จะเป็นการเลือกที่ใชประโยชน์จากความรู้เกี่ยวกับปัญหา ซึ่งเป็นความรู้เกี่ยวกับระยะและความจำเกี่ยวกับทางที่มี pheromone นั่นเอง และถ้าตั้งค่า q_0 ให้เข้าใกล้ 1 คำตอบเฉพาะที่จะถูกเลือก และถ้าตั้งค่า q_0 เข้าใกล้ 0 คำตอบเฉพาะที่ทุกคำตอบจะถูกสำรวจ ดังนั้นโดยปกติค่า q_0 จะถูกตั้งให้เข้าใกล้ 0 ตั้งแต่แรกและค่อยๆปรับจนเข้าใกล้ 1 ในตอนท้าย

ส่วนสมการปรับทาง pheromone จะปรับเฉพาะทางที่ดีที่สุดที่ได้ตั้งแต่รอบแรกซึ่งคือ

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho \Delta \tau_{ij}(t) \tag{4.21}$$

สำหรับทุกเส้นเชื่อม (i, j) ในเส้นทาง T^+ และ

$$\Delta \tau_{ij}(t) = \frac{1}{L^+} \tag{4.22}$$

โดยที่ L^+ เป็นความยาวของเส้นทาง T^+

ACS จะมีการปรับโดยใช้ทาง pheromone เฉพาะที่ด้วย นั่นคือมดตัวที่ k เลือกเมือง $j \in J_i^k$ จากเมือง i ทำให้มีการสะสม pheromone ในเส้นเชื่อม (i, j) ดังนี้

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho \tau_0 \tag{4.23}$$

โดยที่ τ_0 คือค่าเริ่มต้นของทาง pheromone ซึ่งโดยปกติ $\tau_0 = (n\ L_{nn})^{-1}$ โดยที่ n คือจำนวนเมืองทั้งหมด และ L_{nn} เป็นความยาวสำนึกของเส้นทางที่ดีที่สุดที่ได้จากการหาเพื่อนบ้านข้างเคียง ซึ่งการปรับแบบนี้จะทำให้เส้นเชื่อมที่มดเดินผ่านจะมีค่าลดลงเรื่อยๆ จนกระทั่งมดจะเดินลู่เข้าหาเส้นทางที่ดีที่สุดในที่สุด

รายการตัวเลือกจะเป็นรายการของเมืองที่อยากให้มดเดินไปผ่านจากเมืองที่อยู่ โดยที่ไม่ต้องหาจากเมืองทั้งหมด โดยปกติในรายการนี้จะมีเมืองอยู่ cl เมือง และเมืองในรายการตัวเลือกจะถูกเรียงตามระยะจากน้อยไปหามาก และมดจะเลือกเมืองในรายการนี้ก่อน นั่นคือถ้ามีเมืองที่ยังไม่ได้เดินไปใน

รายการนี้จะเลือกเมืองต่อไปโดยใช้สมการ 4.19 แต่ถ้าไม่มีเมืองที่ยังไม่ได้ไปในรายการนี้แล้วจะเลือกเมืองที่ใกล้ที่สุดจากเมืองนอกรายการที่ยังไม่ได้ไปนั่นเอง อัลกอริทึมของ ACS แสดงในรูปที่ 4.9

```
1.  ตั้งค่า τ_ij(0) = τ_0 สำหรับทุกเส้นเชื่อม (i, j)
2.  For k =1 to m
        นำมดตัวที่ k ไปไว้ที่เมืองที่สุ่มมา
    End
3.  ตั้ง T+ และคำนวณ L+
4.  For t = 1 to t_max
        For k = 1 to m
            สร้างทาง T^k(t) โดยที่ทำขั้นตอนต่อไปนี้ n-1 ครั้ง
                If มีเมืองอย่างน้อย 1 เมืองในรายการตัวเลือก
                    เลือกเมือง j จากเมือง i โดย j ∈ J_i^k และเป็นเมืองในรายการ
ตามสมการที่ 4.19
                Else เลือกเมือง j ∈ J_i^k ที่ใกล้ที่สุด
                หลังจากแต่ละทรานซิชันปรับค่าทาง pheromone ตามสมการที่
4.23
        End
        คำนวณ L^k(t) สำหรับทุกเส้นทาง T^k(t) สำหรับ 1 ≤ k ≤ m
        ถ้าเส้นทางที่ได้ดีกว่า T+ ตั้ง T+ และคำนวณ L+ ใหม่
        สำหรับทุกเส้นเชื่อม (i, j) ใน T+ ปรับค่าทาง pheromone ตามสมการที่
4.21
        ตั้งค่า τ_ij(t + 1) = τ_ij(t) สำหรับทุกเส้นเชื่อม (i, j)
    End
```

รูปที่ 4.9 ACS อัลกอริทึม

Ant Colony Optimization ถูกนำไปใช้ในงานหลายประเภทรวมทั้งการหาเส้นทางที่สั้นที่สุดด้วย

4.2 การนำ swarm อย่างง่ายไปใช้ในเกมส์

ในกรณีที่มีตัวแทนในเกมส์มากการคำนวณโดยใช้ swarm ปกติอาจช้าเกินไปและไม่ทำให้เหมือนจริง ดังนั้นการเคลื่อนที่ของตัวแทนอาจทำได้ดังนี้ แต่ละตัวแทนจะถูกพิจารณาว่าอยู่ในพื้นที่ (inner zone) หรืออยู่นอกพื้นที่ (outer zone) โดยพิจารณาจาก abs(dx) + abs(dz) น้อยกว่า swarm_range ที่ตั้งไว้ ถ้าตัวแทนอยู่นอกพื้นที่ (ไกลจากเป้าหมาย) เปลี่ยนแปลงความเร็วและทิศทางเพียงเล็กน้อยเพื่อไม่ให้ตัวแทนแยกออกจากกัน และถ้าทิศทางของตัวแทนอยู่ห่างจากเป้าหมาย ให้ดึงตัวแทนกลับมา

ถ้าตัวแทนอยู่ในพื้นที่ความเร็วและทิศทางถูกตั้งโดยการแปรผันต่อกันนั่นคือ การตั้งทิศทางขึ้นกับความเร็ว และการตั้งความเร็วขึ้นกับทิศทาง ซึ่งอาจทำให้ตัวแทนเคลื่อนที่รอบๆเป้าหมายนั่นเอง

ตัวแทนแต่ละตัวแทนจะเก็บตำแหน่งปัจจุบัน และปรับค่าตำแหน่ง และคำนวณหาความแตกต่างระหว่างตำแหน่งของตัวแทนและเป้าหมาย ถ้าเป้าหมายอยู่ในระยะของตัวแทน ให้โจมตีได้

ในการเพิ่มความเร็วของตัวแทนที่อยู่นอกพื้นที่โดยให้น้อยกว่าค่าที่มากที่สุด ซึ่งความเร็วสูงสุดของแต่ละตัวแทนควรจะแตกต่างกันระหว่างตัวแทน ถ้าตัวแทนเดินทางถูกทิศทางให้ปรับทิศทางเพียงเล็กน้อย แต่ถ้าเดินห่างจากทิศทางที่ควรจะเป็นมากให้ปรับทิศทางค่อนข้างมาก

คำถามท้ายบทที่ 4

1. ให้เขียนโปรแกรมจำลองการโจมตีศัตรูที่เข้ามาในกำแพงเมืองโดยใช้ PSO โดยให้ใช้ Star topology
2. ให้เขียนโปรแกรมแบบเดียวกับข้อ 1 แต่ให้ใช้ PSO ที่เป็น Ring topology ที่มีค่า $n = 4$
3. สมมุติมีมด 2 ตัวคือมด A และมด B โดยที่มด A เดินไปหาอาหารตามทางที่สั้นที่สุด แต่มด B เดินไปหาอาหารตามทางที่ยาวที่สุด หลังจากมด B เดินถึงแหล่งอาหาร เส้นทางใดที่เป็นเส้นทางกลับถึงรังที่มีความน่าจะเป็นสูงในการที่มด B จะเลือกเดิน

สถาปัตยกรรมการตัดสินใจ
Decision-making Architecture

5

วิธีการตัดสินใจมีหลายวิธีการแต่ที่จะกล่าถึงในบทนี้มีเพียง Bayesian networks ทฤษฎี Dempster-shafer และ Decision networksเท่านั้น

5.1 Bayesian networks

ตัวแทนที่อยู่ในเกมส์ อยู่ในสภาวะแวดล้อมที่นักพัฒนาโปรแกรมสร้างขึ้นดังนั้นตัวแทนจึงมีความรู้เกี่ยวกับโลกในเกมส์ทั้งหมด และ Bayesian networks จะทำให้ตัวแทนทำงานได้คล้ายกับมนุษย์มากกว่า ซึ่งวิธีการนี้อยู่บนพื้นฐานของความสัมพันธ์ที่เป็นเหตุเป็นผลระหว่างปรากฏการณ์ต่างๆ และถูกอธิบายในรูปของกราฟ เมื่องสร้างกราฟได้แล้วใช้ความน่าจะเป็นในการทำนายผลลัพธ์ของการกระทำบางอย่าง และในขณะเดียวกันทำนายพฤติกรรมของผู้เล่นได้เช่นกัน

แต่ก่อนที่จะกล่าวถึง Bayesian network จะขอกล่าวถึงทฤษฎีความน่าจะเป็นอย่างย่อก่อน โดยเริ่มจาก atomic event เป็นข้อกำหนดสมบูรณ์ของสถานะของโลกที่ตัวแทนมีความไม่แน่นอน เช่น สมมุติโลกประกอบไปด้วย Cavity และ Toothache ซึ่งทั้งสองเป็นตัวแปรสุ่มแบบบูลีน (Boolean randon variavle) ที่มีโดเมนเป็น ⟨จริง, เท็จ⟩ ดังนั้นจะมี atomic event ที่ไม่เหมือนกันอยู่ 4 เหตุการณ์คือ Cavity เป็นจริงและ Toothache เป็นจริง Cavity เป็นจริงและ Toothache เป็นเท็จ Cavity เป็นเท็จและ Toothache เป็นจริง Cavity เป็นเท็จและ Toothache เป็นเท็จนั่นเอง

a priori probability หรือความน่าจะเป็นแบบไม่มีเงื่อนไข (unconditional probability) เป็นความน่าจะเป็นของเหตุการณ์นั้นเช่น P(Cavity=จริง) = P(Cavity) = 0.1 หรือ P(Cavity=เท็จ) = P(~Cavity) = 0.9 เป็นต้น แต่ถ้าตัวแปรนั้นเป็นตัวแปรสุ่มแบบไม่ต่อเนื่อง (discrete random variable) ซึ่งเป็นตัวแปรที่มีโดเมนที่นับได้เช่นตัวแปร Weather มีโดเมนเป็น ⟨sunny, rainy, cloudy, snow⟩ อาจจะมี apriori probability เป็น P(Weather=sunny) =0.7, P(Weather=rain)=0.2, P(Weather=cloudy)=0.08 และ P(Weather=snow)=0.02 เป็นต้น ถ้ามีการหาความน่าจะเป็นของ 2 ตัวแปรเช่น P(Weather, Cavity) การหาความน่าจะเป็นแบบนี้เรียกว่า joint probability distributiion ส่วน conditional probability $P(A|B)$ คือความน่าจะเป็นของเหตุการณ์ A ถ้าเรารู้เหตุการณ์ B ซึ่งถูกนิยามโดย

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{5.1}$$

และ $P(A, B) = P(A|B)\ P(B)$ หรือ $P(A, B) = P(B|A)\ P(A)$ นั่นเอง โดยปกติสัจพจน์ (axiom) ของความน่าจะเป็นคือ

1. $0 \leq P(A) \leq 1$
2. ความจำเป็นจริง (necessarily true) มีความน่าจะเป็นเป็น 1 และความจำเป็นเท็จ (necessarily false) มีความจำเป็นเป็น 0 นั่นคือ P(จริง) = 1 และ P(เท็จ) = 0
3. $P(A \vee B) = P(A) + P(B) - P(A,B)$

พื้นฐานของการหาเหตุผลเชิงน่าจะเป็น (probabilistic reasoning) คือ ทฤษฎีของ Bayes เช่น สมมุติเราอยากรู้ว่า มีโอกาสเท่าใดที่ฝนจะตกเมื่อวานถ้าเราเจอว่าสนามหญ้าหน้าบ้านเปียก ทฤษฎีของ Bayes ทำให้เราคำนวณความน่าจะเป็นนี้จาก ความน่าจะเป็นที่สนามหญ้าหน้าบ้านเปียกเป็นเท่าใด ถ้าฝนตกเมื่อวานจริง นั่นคือ

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{5.2}$$

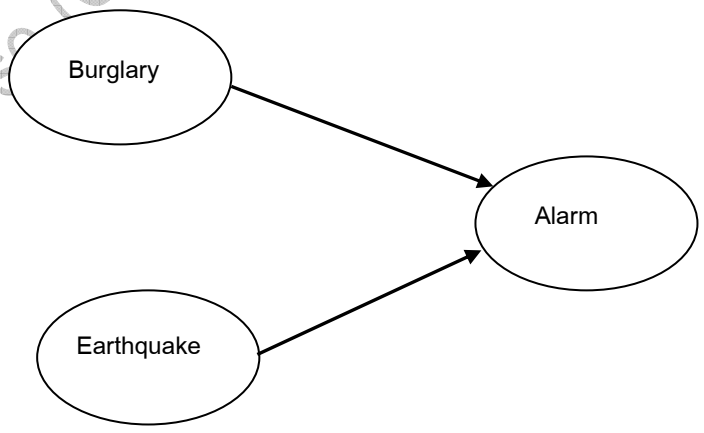
ซึ่งในเหตุการณ์ที่กล่าวข้างต้น $P(A|B)$ คือความน่าจะเป็นที่ฝนตกเมื่อวานเป็นเท่าใดถ้าสนามหญ้าหน้าบ้านเปียก ส่วน $P(B|A)$ คือความน่าจะเป็นของสนามหญ้าหน้าบ้านเปียกถ้าฝนตกเมื่อวานจริง และ $P(A)$ คือโอกาสที่ฝนตก และ $P(B)$ คือโอกาสที่สนามหญ้าหน้าบ้านเปียกนั่นเอง

ถ้าเราสามารถเขียนประพจน์ (proposition) และนิยามความสัมพันธ์เป็นเหตุเป็นผลระหว่างประพจน์ได้เช่น *A* ทำให้เกิด *B* ทำให้นำมาสร้างเป็นกราฟได้ และเมื่อได้กราฟสามารถใช้การหาเหตุผลเชิงน่าจะเป็นมาหาเหตุผลในกราฟได้เช่นกัน

Bayesian network เป็นกราฟระบุทิศทาง (directed graph) ที่แต่ละ node แทนข้อมูลความน่าจะเป็นเชิงตัวเลขนั่นเอง คุณลักษณะของ Bayesian network มีดังนี้

1. ตัวแปรสุ่ม (random variable) ที่ต้องการถูกนำมาใช้เป็น node ใน Bayesian network ซึ่งตัวแปรเหล่านี้เป็นได้ทั้งตัวแปรไม่ต่อเนื่อง และตัวแปรต่อเนื่อง
2. เส้นเชื่อมจาก node *X* ไปยัง node *Y* ถูกเรียกว่า *X* เป็นพ่อแม่ (parent) ของ node *Y*
3. แต่ละ node X_i มี conditional probability distribution $P(X_i \mid Parents(X_i))$ ที่บ่งบอกถึงผลกระทบของ node ที่เป็นพ่อแม่ต่อ node นี้
4. กราฟจะไม่มีรอบที่ระบุทิศทาง (directed cycle) ดังนั้นจะเป็นกราฟระบุทิศทางที่ไม่มีวง (directed, acyclic graph, DAG)

รูปที่ 5.1 คือ Bayesian network ที่มีประพจน์ 3 ประพจน์ที่สามารถอ่านได้ว่า Burglary และ Earthquake สามารถทำให้เกิด Alarm ได้เป็นจริง



รูปที่ 5.1 Bayesian network ของการเกิด Alarm

การคำนวณเริ่มจากคำนวณหาโอการที่จะเกิด Earthquake และ Burglary เช่นสมมุติให้โอกาสที่จะเกิด Earthquake ($P(E)$) เป็น 0.002 และโอกาสที่จะเกิด Burglary ($P(B)$) เป็น 0.001 หลังจากนั้นสามารถหา conditional probability ของการเกิด Alarm ได้ ซึ่งสมมุติว่าจากการทดลองหลายครั้งหรือจากการอ่านสถิติเก่าทำให้สรุปได้ดังตารางที่ 5.1

ตารางที่ 5.1 โอกาสที่จะเกิด Alarm ในสถานการณ์ต่างๆ $P(A|B, E)$

Burglary	Earthquake	Alarm
จริง	จริง	0.95
จริง	เท็จ	0.94
เท็จ	จริง	0.29
เท็จ	เท็จ	0.001

ซึ่งโอกาสที่จะเกิด Alarm ในตาราง 5.1 คือ conditional probability ($P(A|B, E)$) ดังนั้นโอกาสที่จะเกิด Alarm ในแต่ละกรณีต่างๆจริงคือ $P(A,B,E) = P(A) = P(B) \times P(E) \times P(A|B, E)$ นั่นเองซึ่งแสดงในตารางที่ 5.2

ตารางที่ 5.2 โอกาสที่จะเกิด Alarm ในกรณีต่างๆจริง $P(A)$

$P(B)$	$P(E)$	$P(A|B, E)$	$P(A)$	$\alpha P(A)$
T=0.001	T=0.002	0.95	0.000002	0.000795
T=0.001	F=0.998	0.94	0.000938	0.372814
F=0.999	T=0.002	0.29	0.000579	0.230127
F=0.999	F=0.998	0.001	0.000997	0.396264

ในการหา $P(A,B,E)$ ที่จริงมาจากกฎลูกโซ่ซึ่งคือ

$$P(x_1,...,x_n) = P(x_n|x_{n-1},...,x_1)P(x_{n-1}|x_{n-2},...,x_1)\cdots P(x_2|x_1)P(x_1) \quad (5.3)$$

ซึ่งสมการนี้บ่งบอกว่า Bayesian network จะเป็นตัวแทนที่ดีของปัญหาถ้าแต่ละ node เป็นอิสระอย่างมีเงื่อนไข (conditionally independent) ของตัวนำหน้า (predecessor) นั่นคือ node พ่อแม่ (parent node) ของ x_i ควรจะมี node $x_1,...,x_{i-1}$ นั่นเอง

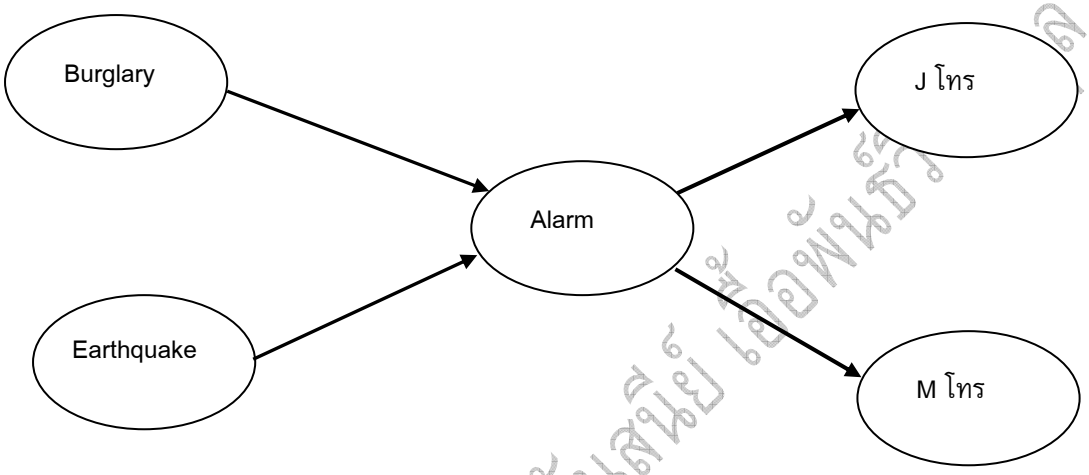
จากตารางที่ 5.2 จะเห็นว่าผลรวมของ $P(A)$ เท่ากับ 0.002516 ซึ่งหมายถึงโอกาสที่จะเกิด Alarm ในวันหนึ่งเป็น 0.25% นั่นเอง และในคอลัมภ์ขวาสุดคือความน่าจะเป็นที่ทำให้เป็นบรรทัดฐาน (normalized probability) เพื่อที่จะให้ผลรวมเท่ากับ 1 จากตารางผลรวมของคอลัมภ์นี้เท่ากับ 1 ซึ่งในกรณีนี้ α มีค่าเท่ากับ.1/002516 ซึ่งความน่าจะเป็นในคอลัมภ์นี้บอกได้ว่าในกรณีที่มีการเกิด Alarm เป็นไปได้ว่า 37% เกิดจาก Burglary และ 23% เกิดจาก Earthquake และ 39% ไม่ได้เกิดจากอะไรเลย และ 0.0795% เกิดจากทั้ง Burglary และ Earthquake แต่ถ้าเรารู้แน่นอนว่าเกิด Earthquake แถวบ้านทำให้ $P(E)$ เท่ากับ 1 ถ้าเป็นจริงและเท่ากับ 0 ถ้าเป็นเท็จ ทำให้ค่าต่างๆเป็นดังตารางที่ 5.3

ตารางที่ 5.2 โอกาสที่จะเกิด Alarm ในกรณีต่างๆจริง $P(A)$ ในกรณีที่เกิด Earthquake

$P(B)$	$P(E)$	$P(A|B, E)$	$P(A)$	$\alpha P(A)$
T=0.001	T=1	0.95	0.00095	0.003269
T=0.001	F=0	0.94	0	0
F=0.999	T=1	0.29	0.28971	0.996731
F=0.999	F=0	0.001	0	0

ผลรวมของ $P(A)$ คือ 0.29066 ทำให้วันนี้มีโอกาสเกิด Alarm เป็น 29% และ 99.67% ในนั้นเกิดจาก Earthquake และ 33% เกิดจาก Burglary นั้นเอง

ถ้าเหตุการณ์มีความซับซ้อนมากขึ้นโดยการที่มีเพื่อนบ้าน 2 คนคือนาย J และนางสาว M ที่สัญญาว่าจะโทรศัพท์มาบอกทุกครั้งที่ได้ยินเสียง Alarm ซึ่งนาย J จะโทรมาทุกครั้งที่ได้ยินแต่มีบางครั้งที่นาย J สับสนระหว่างเสียง Alarm กับเสียงโทรศัพท์ ส่วนนางสาว M ชอบฟังเพลงเสียงดังและบางครั้งจะไม่ได้ยินเสียง Alarm ซึ่ง Bayesian network ของเหตุการณ์นี้แสดงในรูปที่ 5.2 และโอกาสที่จะเกิด $P(B)$ $P(E)$ และ $P(A|\ B, E)$ เป็นดังตารางที่ 5.1 และโอกาสที่นาย J และนางสาว M จะโทร $P(J)$ และ $P(M)$ เป็นดังตารางที่ 5.3



รูปที่ 5.3 Bayesian network ของการเกิด Alarm และการโทรศัพท์ของนาย J และนางสาว M

ตารางที่ 5.3 โอกาสที่นาย ก และนางสาว ข จะโทรศัพท์

$P(J)$	$P(M)$
T=0.90	T=0.70
F=0.05	F=0.01

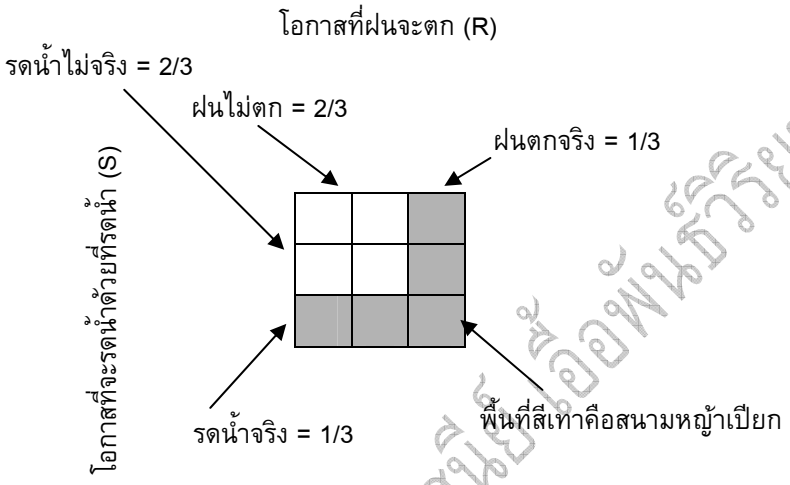
จากรูปที่ 5.3 จะเห็นว่า Burglary และ Earthquake มีผลโดยตรงกับ Alarm แต่การที่นาย J และนางสาว M จะโทรศัพท์ขึ้นอยู่กับ Alarm อย่างเดียวเท่านั้น ดังนั้น $P(J)$ ในตารางที่ 5.3 คือ $P(J|A)$ และเช่นเดียวกัน $P(M) = P(M|A)$ นั่นเอง ถ้าต้องการรู้ว่าโอกาสที่นาย J และนางสาว M จะโทรศัพท์ในขณะที่เกิด Alarm และไม่ได้เกิด Burglary และไม่มี Earthquake คือ $P(J,M,A, \sim B, \sim E) = P(J|A) \times P(M|A) \times P(A|\ \sim B, \sim E) \times P(\sim B) \times P(\sim E) = 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 = 0.00062$ และในกรณีการที่นาย J โทรศัพท์จะไม่มีผลกระทนโดยตรงต่อนางสาว M หรือการที่นางสาว M โทรก็ไม่ได้มีผลกระทบกับนาย J เช่นกัน ดังนั้น $P(M|\ J,A,E,B) = P(M|\ A)$ นั่นเอง

โดยปกติการอนุมาน (inference) มีอยู่ 3 ประเภทคือ

1. การอนุมานเชิงเหตุ (Causal inference) เช่นเหตุการณ์ A ทำให้เกิด B และถ้ารู้ค่าของ A สามารถหาความน่าจะเป็นของ B wfh
2. การอนุมานเชิงวินิจฉัย (Diagnostic inferece) เช่นเหตุการณ์ A ทำให้เกิด B และถ้ารู้ค่าของ B สามารถหาความน่าจะเป็นของ A ได้

3. การอนุมานเชิงเหตุระหว่างกัน (Intercausal inference) บางครั้งเรียก explaining away เช่นเหตุการณ์ A และ A ทำให้เกิด C และรู้ค่า C สามารถคำนวณว่าการเปลี่ยนในความน่าจะเป็นของ A จะมีผลกับความน่าจะเป็นของ B เท่าใดถึงแม้ว่าทั้ง A และ B เป็นตัวแปรที่อิสระต่อกัน

สมมุติให้การรดน้ำด้วยที่รดน้ำแทนด้วย S และฝนตกเมื่อคืนแทนด้วย R และการที่สนามหญ้าหน้าบ้านเปียกแทนด้วย W และให้โอกาสที่ S และ R เป็นจริงเป็น 1 ใน 3 และทั้ง S และ R เป็นตัวแปรที่มีอิสระต่อกัน ดังแสดงในรูปที่ 5.4

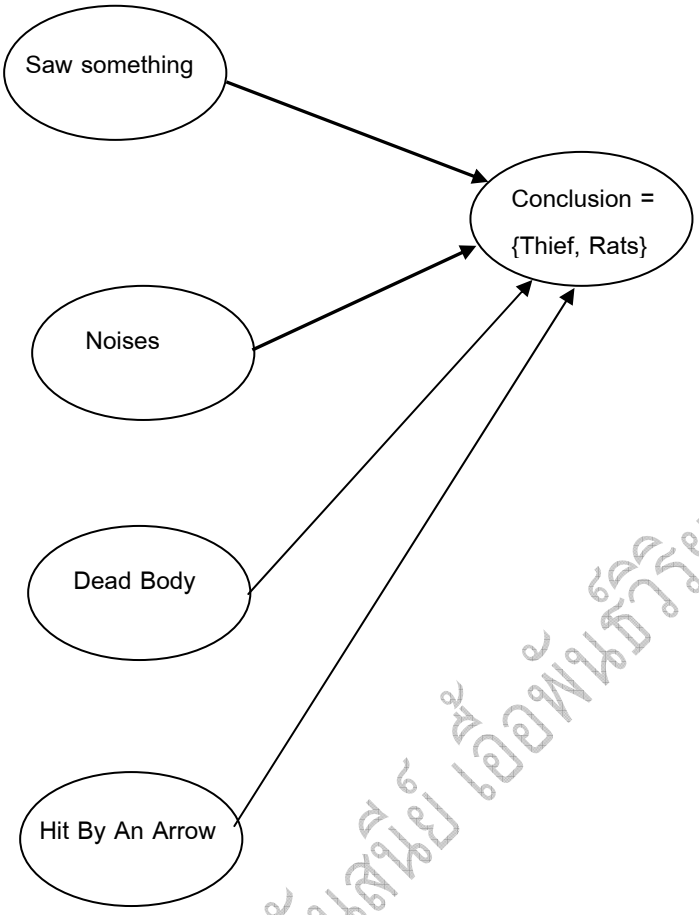


รูปที่ 5.4 joint probability ของฝนตก (R) และรดน้ำ (S)

จากรูปที่ 5.4 มีโอกาส 33% ที่ฝนตกและ 33% ที่จะรดน้ำ และโอกาสที่ w เป็นจริงคือ 55% (5 ใน 9) ซึ่งในนี้มี 3 ใน 5 ที่ S เป็นจริง และ 3 ใน 5 ที่ R เป็นจริง ถ้าสมมุติเพื่อนโทรมาบอกว่าเมื่อคืนฝนตก ทำให้โอกาสการเปียกของสนามหญ้ามีเพียงคอลัมภ์ขวาสุดในรูปที่ 5.4 เท่านั้นถึงแม้ว่าโอกาสเกิด S ยังคงเป็น 33% และถ้าเพื่อนบอกว่าฝนไม่ได้ตกทำให้รู้ว่าสนามหญ้าเปียกเนื่องจากการรดน้ำมีความเป็นไปได้ถึง 100% เช่นกัน เหตุการณ์ที่กล่าวนี้เป็น explaining away นั่นเอง

ตัวอย่างที่ 5.1 ต้องการสร้าง first-person sneaker เกมส์ซึ่งเหมือนกับ first-person shooter เกมส์เพียงแต่ใช้อาวุธอื่นแทนปืน สมมุติให้ตัวแทนเป็นยามในเกมส์นี้และถ้ายามเกิดความสงสัย ผู้เล่นจะปล่อยยามไว้เพื่อให้ยามเลิกกังวล ซึ่งอาจแสดงด้วยคำพูดว่าสงสัยจะเป็นหนู ซึ่งในกรณีนี้ยามทำตัวเป็น Bayesian inference ซึ่ง explaining away จากหลักฐานที่ได้ และผู้เล่นจะส่งหลักฐานต่างๆให้ยามเพื่อให้ยามคิดว่าเป็นหนู ซึ่งเหตุการณ์นี้สามารถใช้ Bayesian network มาช่วยยามตัดสินใจได้ดังแสดงในรูปที่ 5.5

สิ่งที่ต้องการนอกจากนี้คือเมตริกของเหตุการณ์ต่างๆที่จะมีผลต่อการตัดสินใจของยาม ■



รูปที่ 5.5 Bayesian network ของยาม

5.2 ทฤษฎี Dempster-Shafer

การรวมกันของของแหล่งของหลักฐานสามารถทำได้ใน Bayesian network แต่สถานการณ์เหล่านั้นต้องเป็น conditionally independent หรือเชื่อมกันด้วย conditional probability แต่ในความเป็นจริงการพิสูจน์ความเป็นอิสระทำได้ยากโดยเฉพาะในเกมส์ที่มีตัวแปรมาก ซึ่งอาจมีผลกระทบต่อกันในทางที่ไม่สามารถทำนายได้ ดังนั้นบางสถานการณ์ในเกมส์สามารถใช้ Bayesian network ได้แต่ในบางสถานการณ์อาจต้องใช้วิธีการที่มีข้อจำกัดของความเชื่อ (Belief) น้อยลง

สมมุติต้องการประมาณว่าทีม A จะชนะการแข่งขันเบสบอลวันนี้ด้วยสถิติที่ผ่านมาใน 1 ปีที่ผ่านมาคือชนะ 60 ครั้งแพ้ 40 ครั้งและปีนี้ที่ผ่านมาทีมนี้มีสถิติชนะ 17 จาก 20 ครั้งด้วยนาย S เป็นผู้เล่นคนแรก แต่อย่างไรก็ตามเราไม่ทราบว่านาย S จะถูกกำหนดให้เริ่มเล่นเป็นคนแรกในวันนี้หรือไม่ ดังนั้นการประมาณของเราอาจจะขึ้นอยู่กับแนวคิดดังต่อไปนี้

1. ความน่าเชื่อถือ (Credibility) ซึ่งเป็นสิ่งที่วัดหลักฐานที่มีสนับสนุนความเชื่อมากน้อยแค่ไหน ในกรณีนี้คือสถิติของทีมนี้ทำให้มีความน่าเชื่อถือว่าจะชนะ 60% ซึ่งภายหลังจะเรียกว่าตัววัดความเชื่อ (belief measure)
2. ความเป็นไปได้ (Plausibility) ซึ่งเป็นสิ่งที่วัดว่าหลักฐานที่มีล้มเหลวในการทำให้ความเชื่อหายไปเช่นเราไม่รู้ว่านาย S จะพิตช์ (pitch) วันนี้แต่เราหวังว่านาย S จะพิตช์และจากสถิติของทีมที่มีนาย S เป็นคนพิตช์ทำให้รู้ว่าความเป็นไปได้ที่จะชนะวันนี้คือ 85%

3. ความเชื่อ (belief) เป็นมาตราวัดที่อยู่ระหว่างความน่าเชื่อถือและความเป็นไปได้ การประมาณทางลบจะเข้าใกล้ความน่าเชื่อถือ และการประมาณทางบวกจะเข้าใกล้ความเป็นไปได้ ซึ่งความเชื่อจะขึ้นกับหลักฐานที่มีอยู่ ซึ่งในที่นี้ก็คือตัววัดความเชื่อ (belief measure) นั่นเอง

กฎของ Dempster ทำให้แหล่งของหลักฐานหลายแหล่งสามารถรวมกันได้เพื่อสนับสนุนการตัดสินใจ แต่ก่อนที่จะกล่าวถึงกฎของ Dempster จำเป็นที่จะต้องกล่าวถึงตัววัดความเชื่อ (belef measure) ก่อน ให้เซตสากล (universal set) X ตัววัดความเชื่อเป็นฟังก์ชันดังนี้

$$\text{Bel: } \mathscr{P}(X) \rightarrow [0,1] \quad (5.4)$$

โดยที่ $\text{Bel}(\varnothing) = 0$ และ $\text{Bel}(X) = 1$ และ

$$\text{Bel}\left(A_1 \bigcup A_2 \bigcup \ldots \bigcup A_n\right) \geq \sum_j \text{Bel}\left(A_j\right) - \sum_{j<k} \text{Bel}\left(A_j \bigcap A_k\right) + \ldots + (-1)^{n+1} \text{Bel}\left(A_1 \bigcap A_2 \bigcap \ldots \bigcap A_n\right) \quad (5.5)$$

สำหรับทุกแฟมิลี่ของเซตย่อยของ X สำหรับแต่ละเซต $A \in \mathscr{P}(X)$ ค่า Bel(A) คือระดับความเชื่อ (degree of belief) ว่าสมาชิกตัวนั้นใน X อยู่ในเซต A โดยค่านี้จะวัดขึ้นอยู่กับหลักฐานที่มี ซึ่งเราอาจมองเห็นว่าเซตย่อยของ X เป็นคำตอบสำหรับคำถามใดคำถามหนึ่ง เราอาจคิดว่าคำตอบนั้นถูกแต่ไม่ทราบว่าคำตอบนั้นเป็นอันไหน และสำหรับเซต $A_1, A_2, \ldots, A_n$ เป็น pair-wise disjoint และจากสมการที่ 5.5 จะเห็นได้ว่าความเชื่อที่เกี่ยวข้องกับการยูเนียนของเซตเหล่านี้ต้องไม่น้อยกว่าผลรวมของของความเชื่อของแต่ละเซต

สมมุติให้เซต $A \subseteq B$ ($A, B \in \mathscr{P}(X)$) และให้ $C = B - A$ ดังนั้น $A \cup C = B$ และ $A \cap C = \varnothing$ ดังนั้น

$$\text{Bel}(A \cup C) = \text{Bel}(B) \geq \text{Bel}(A) + \text{Bel}(C) - \text{Bel}(A \cap C) \quad (5.6)$$

ซึ่งคือ

$$\text{Bel}(B) \geq \text{Bel}(A) + \text{Bel}(C) \quad (5.7)$$

หรือ $\text{Bel}(B) \geq \text{Bel}(A)$ นั่นเอง และถ้า $A_1 = A$ และ $A_2 = {\sim}A$ จากสมการที่ 5.5 จะได้ว่า

$$\text{Bel}(A) + \text{Bel}({\sim}A) \leq 1 \quad (5.8)$$

สิ่งที่เกี่ยวข้องกับตัววัดความเชื่อคือตัววัดความเป็นไปได้ (plausibility measure, Pl) ซึ่งถูกนิยามดังนี้

$$\text{Pl}(A) = 1 - \text{Bel}({\sim}A) \quad (5.9)$$

สำหรับทุก $A \in \mathscr{P}(X)$ ซึ่งตัววัดความเป็นไปได้เป็นฟังก์ชันที่เป็นดังนี้

$$\text{Pl: } \mathscr{P}(X) \rightarrow [0,1] \quad (5.10)$$

โดยที่ $\text{Pl}(\varnothing) = 0$ และ $\text{Pl}(X) = 1$ และ

$$\text{Pl}\left(A_1 \bigcap A_2 \bigcap \ldots \bigcap A_n\right) \leq \sum_j \text{Pl}\left(A_j\right) - \sum_{j<k} \text{Pl}\left(A_j \bigcup A_k\right) + \ldots + (-1)^{n+1} \text{Pl}\left(A_1 \bigcup A_2 \bigcup \ldots \bigcup A_n\right) \quad (5.11)$$

สำหรับทุกแฟมิลี่ของเซตย่อยของ X และจากสมการที่ 5.11 ทำให้สรุปได้ว่า

$$\text{Pl}(A) + \text{Pl}({\sim}A) \geq 1 \quad (5.12)$$

ทั้งตัววัดความเชื่อและตัววัดความเป็นไปได้ถูกบ่งบอกลักษณะได้ด้วยฟังก์ชัน

$$m: \mathcal{P}(X) \rightarrow [0,1] \tag{5.13}$$

โดยที่ $m(\varnothing) = 0$ และ

$$\sum_{A \in \mathcal{P}(X)} m(A) = 1 \tag{5.14}$$

ซึ่งฟังก์ชันนี้ถูกเรียกว่าการกำหนดค่าความน่าจะเป็นพื้นฐาน (basic probability assignment (bpa)) แต่ละเซต $A \in \mathcal{P}(X)$ ค่า $m(A)$ จะบ่งบอกถึงสัดส่วนที่หลักฐานที่มีสนับสนุนคำกล่าวที่ว่าสมาชิกตัวนั้นของ X อยู่ในเซต A ค่านี้จะไม่ได้บ่งบอกถึงอะไรที่เกี่ยวกับเซตย่อยของ A แม้แต่น้อย แต่จะเกี่ยวกับกับเซต A เท่านั้น ค่า bpa นี้ไม่ใช่ความน่าจะเป็นดังนั้นจึงไม่มีคุณสมบัติของความน่าจะเป็นเลยคือ

1. ไม่จำเป็นที่ $m(X) = 1$
2. ไม่จำเป็นที่ $m(A) \leq m(B)$ เมื่อ $A \subseteq B$
3. ไม่มีความสัมพันธ์ใดๆระหว่าง $m(A)$ และ $m(\sim A)$

ความสัมพันธ์ระหว่างตัววัดความเชื่อและ bpa แสดงในสมการที่ 5.16 และ 5.17 ในขณะที่สมการที่ 5.15 คือความสัมพันธ์ระหว่างตัววัดความเป็นไปได้และ bpa

$$\text{Bel}(A) = \sum_{B|B \subseteq A} m(B) \tag{5.15}$$

$$m(A) = \sum_{B|B \subseteq A} (-1)^{|A-B|} \text{Bel}(B) \tag{5.16}$$

$$\text{Pl}(A) = \sum_{B|A \cap B \neq \varnothing} m(B) \tag{5.17}$$

ซึ่งความสัมพันธ์เหล่านี้มีความหมายคือ $m(A)$ บ่งบองถึงระดับของหลักฐานหรือความเชื่อที่เกี่ยวกับสมกาชิกที่เกี่ยวข้องกับคำถามอยู่ในเซต A ส่วน Bel(A) บ่งบอกถึงหลักฐานหรือความเชื่อทั้งหมดว่าสมาชิกอยู่ในเซต A รวมทั้งเซตย่อยของ A ด้วย ส่วน Pl(A) คือไม่เพียงแต่บ่งบอกถึงหลักฐานหรือความเชื่อทั้งหมดว่าสมาชิกอยู่ในเซต A รวมทั้งเซตย่อยของ A เท่านั้นแต่ยังรวมไปถึงหลัดฐานหรือความเชื่อเพิ่มเติมที่เกี่ยวข้องกับเซตที่คาบเกี่ยวกับเซต A ด้วยนั่นเอง ดังนั้นสำหรับทุกเซต $A \in \mathcal{P}(X)$

$$\text{Pl}(A) \geq \text{Bel}(A) \tag{5.18}$$

สำหรับเซต $A \in \mathcal{P}(X)$ ที่มีค่า $m(A) > 0$ ถูกรียกว่า focal element ของ m ซึ่งคือเวตย่อยของ X ที่มีหลักฐานบ่งชี้อยู่ เมื่อเซต X เป็นเซตจำกัด สามารถอธิบาย m ด้วย fical element A พร้อมค่า $m(A)$ ได้ ซึ่งคือ $\langle \mathcal{F}, m \rangle$ หรือ body of evidence โดยที่ $\mathcal{F}$ และ m แทนเซตของ focal element และค่า bpa ที่เกี่ยวข้อง สำหรับความไม่รู้โดยสิ้นเชิง (total ignorance) จะถูกบ่งชี้โดยที่ $m(X) = 1$ และ $m(A) = 0$ สำหรับทุกเซต $A \neq X$ นั่นคือเรารู้ว่าสมาชิกตัวนั้นอยู่ในเซตสากล แต่ไม่มีหลักฐานใดที่บ่งบอกถึงว่าอยู่ในเซตย่อยไหนเลย ซึ่งในกรณีนี้ทำให้ Bel(X) = 1 และ Bel(A) = 0 สำหรับทุกเซต $A \neq X$ เช่นกัน แต่สำหรับ Pl($\varnothing$)= 0 และ Pl(A) = 1 สำหรับทุกเซต $A \neq \varnothing$

สำหรับการรวมกันของหลักฐานโดยใช้ทฤษฎีของ Dempster-Shafer ทำได้ดังนี้ ให้หลักฐานมาจาก 2 แหล่ง เช่นคำตอบที่ได้จากผู้เชี่ยวชาญ 2 คน และแทนด้วย bpa m_1 และ m_2 ดังแสดงในรูปที่ 5.6 ซึ่งแหล่งที่ 1 มีหลักฐานสำหรับเซต $A_1, A_2, ..., A_i, ..., A_k$ ดังนั้น body of evidence สำหรับแหล่งที่ 1

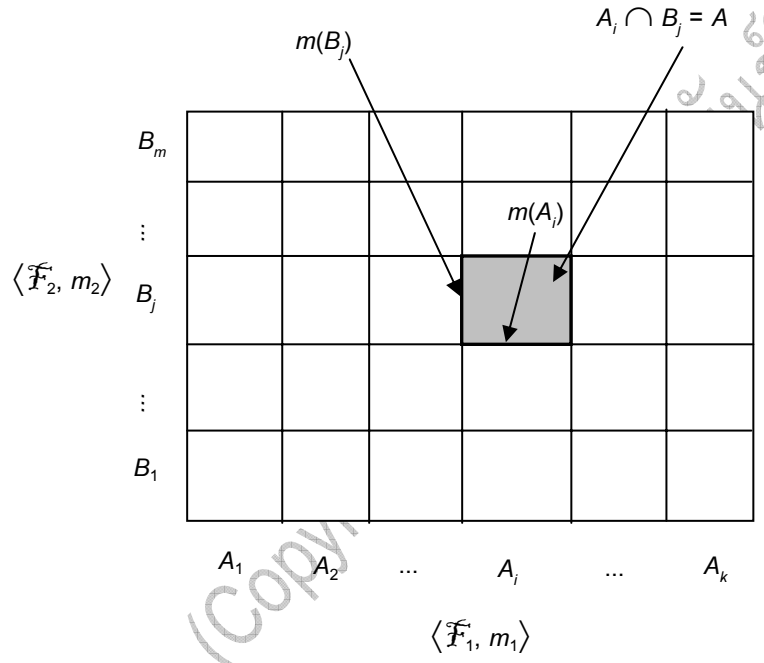
คือ $\langle \mathcal{F}_1, m_1 \rangle$ และแหล่งที่ 2 มีหลักฐานสำหรับเซต B_1, B_2,...,B_j,...,B_m ดังนั้น body of evidence สำหรับแหล่งที่ 2 คือ $\langle \mathcal{F}_2, m_2 \rangle$ ดังนั้นหลักฐานของเหตุการณ์ A คือพื้นที่สีเทาในรูปที่ 5.6 นั่นเอง ซึ่งเกิดจากการอินเตอร์เซกชันระหว่างเซต A_i และเซต B_j นั่นเอง สมการในการรวมหลักฐานจาก 2 แหล่ง คือ

$$m(A) = \frac{\sum_{A_i \cap B_j = A} m_1(A_i) m_2(B_j)}{1 - K} \tag{5.19}$$

โดยที่

$$K = \sum_{A_i \cap B_j = \varnothing} m_1(A_i) m_2(B_j) \tag{5.20}$$

ซึ่งการรวมกันของทั้ง 2 แหล่งจะได้ body of evidence $\langle \mathcal{F}, m \rangle$ นั่นเอง



รูปที่ 5.6 body of evidence สำหรับแหล่งที่ 1 และแหล่งที่ 2

ตัวอย่างที่ 5.2 สมมุติให้มีการค้นพบภาพเก่าที่มีความคล้ายกับภาพของ Raphael ตั้งแต่มีการค้นพบจะมีคำถามเหล่านี้เกิดขึ้นเสมอ ซึ่งคำถามเหล่านี้คือ

1. ภาพที่เจอเป็นภาพของ Raphel จริงหรือไม่
2. ภาพที่เจอเป็นภาพที่วาดนักเรียนของ Raphel หรือไม่
3. ภาพที่เจอเป็นภาพปลอมหรือไม่

ให้ R (แทนภาพของ Raphel), D (แทนภาพที่วาดโดยนักเรียนของ Raphel) และ C (แทนภาพปลอม) เป็นเซตย่อยของเซตสากล X (เซตของรูปภาพทั้งหมด) และสมมุติให้ผู้เชี่ยวชาญ 2 คนมาตรวจภาพนี้และให้ m_1 และ m_2 กลับมาดังตารางที่ 5.4

ตารางที่ 5.4 หลักฐานจากผู้เชี่ยวชาญ 1 และ 2 และการรวมหลักฐาน

Focal elements	ผู้เชี่ยวชาญคนที่ 1		ผู้เชี่ยวชาญคนที่ 2		หลักฐานรวม	
	m_1	Bel_1	m_2	Bel_2	m	Bel
R	0.05	0.05	0.15	0.15	0.21	0.21
D	0	0	0	0	0.01	0.01
C	0.05	0.05	0.05	0.05	0.09	0.09
R∪D	0.15	0.20	0.05	0.20	0.12	0.34
R∪C	0.10	0.20	0.20	0.40	0.20	0.50
D∪C	0.05	0.10	0.05	0.10	0.06	0.16
R∪D∪C	0.60	1.00	0.50	1.00	0.31	1.00

ตัวอย่างเช่นผู้เชี่ยวชาญ 1 ให้ระดับของหลักฐานว่าภาพวาดโดย Raphel หรือวาดโดยนักเรียนของ Raphel คือ $m_1(R \cup D) = 0.15$ และเมื่อได้ body of evidence จากทั้ง 2 ผู้เชี่ยวชาญ สามารถคำนวณหา Bel ของทั้งสองผู้เชี่ยวชาญจากค่า bpa ได้โดยใช้สมการที่ 5.13 ในการคำนวณหาค่า m รวมใช้สมการที่ 5.17 และต้องคำนวณหา K ก่อนซึ่งคือ

$$\begin{aligned} K = {} & m_1(R)\, m_2(D) + m_1(R)\, m_2(C) + m_1(R)\, m_2(D \cup C) + m_1(D)\, m_2(R) + m_1(D)\, m_2(C) \\ & + m_1(D)\, m_2(R \cup C) + m_1(C)\, m_2(R) + m_1(C)\, m_2(D) + m_1(C)\, m_2(R \cup D) \\ & + m_1(R \cup D)\, m_2(C) + m_1(R \cup C)\, m_2(D) + m_1(D \cup C)\, m_2(R) \\ = {} & 0.03 \end{aligned}$$

ดังนั้น $1 - K = 0.97$ ค่า $m(R)$ หาได้โดย

$$\begin{aligned} m(R) = {} & \big[\, m_1(R)\, m_2(R) + m_1(R)\, m_2(R \cup D) + m_1(R)\, m_2(R \cup C) + m_1(R)\, m_2(R \cup D \cup C) \\ & + m_1(R \cup D)\, m_2(R) + m_1(R \cup D)\, m_2(R \cup C) + m_1(R \cup C)\, m_2(R) \\ & + m_1(R \cup C)\, m_2(R \cup D) + m_1(R \cup D \cup C)\, m_2(R) \big] / 0.97 \\ = {} & 0.21 \end{aligned}$$

ส่วนค่า m ของอย่างอื่นจะคำนวณในทำนองเดียวกัน ซึ่งคือค่าที่แสดงในตารางที่ 5.4 นั่นเอง ■

5.3 Decision Networks

Decision networks เป็นการรวม Bayesion networks และ node ที่เป็น การกระทำ (action) และ utility เข้าด้วยกัน โดยปกติ decision network แทนข้อมูลเกี่ยวกับสถานะปัจจุบันของตัวแทน สถานะที่เป็นผลจากการกระทำของตัวแทน และ utility ของสถานะ ตัวอย่างของ decision network ของปัญหาที่ตั้งของสนามบิน ซึ่งแสดงในรูปที่ 5.7 โดยที่ node ที่อยู่ในรูปมี 3 ประเภทคือ

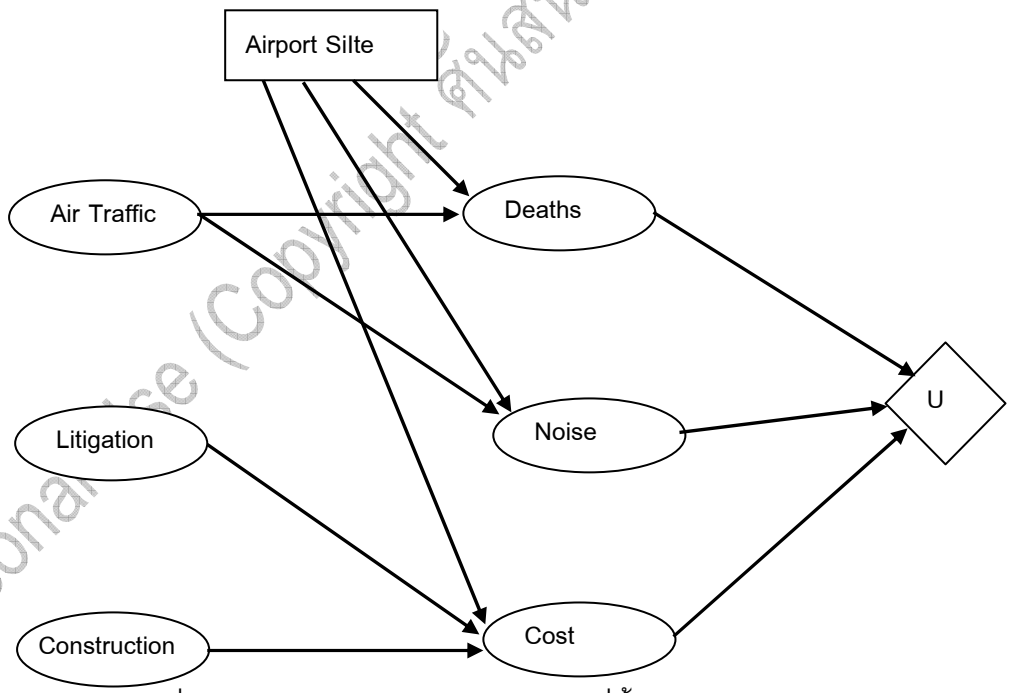
1. Chance node (วงกลม) เป็น node ที่เป็นตัวแทนของตัวแปรสุ่ม ซึ่งจะเหมือนกับใน Bayesian network ซึ่งในกรณีนี้ตัวแทนอาจมีความไม่แน่นอนต่อ ราคาก่อสร้าง

(construction cost) ระดับของการจราจรทางอากาศ (level of air traffic) และความเป็นไปได้ในการถูกฟ้อง (litigation potential)

2. Decision node (สี่เหลี่ยม) เป็น node ที่ตัวตัดสินใจมีตัวเลือกของการกระทำ ซึ่งในกรณีนี้คือ Airport Site ที่รับค่าของแต่ละสถานที่ ซึ่งการเลือกจะมีผลกับ Cost Noise และ Safety
3. Utility node (สี่เหลี่ยมข้าวหลามตัด) เป็น node ที่แทนฟังก์ชัน utility ของตัวแทน node
4. เหล่านี้มี node พ่อแม่ที่เป็นตัวแปรที่อธิบายผลลัพธ์ที่มีผลโดยตรงกับ utility node

การกระทำถูกเลือกโดยการประเมิณ decision network สำหระบแต่ละสถานะที่เป็นไปได้ที่ decision node หลังจาก decision node ตั้งค่าแล้ว node นี้จะมีพฤติกรรมคล้าย chance node และทำการประเมินโดย

1. ตั้งค่าตัวแปรของหลักฐาน (evidence variable) สำหรับสถานปัจจุบัน
2. สำหรับแต่ละค่าที่เป็นไปได้ของ decision node
   - ตั้งค่าใน decision ศำหรับค่านั้น
   - คำนวณ conditional probability สำหรับ node พ่อแม่ของ utility node
   - คำนวณ utility ผลลัพธ์ของการกระทำนั้น
3. เลือกการกระทำที่มีค่า utility สูงที่สุด



รูปที่ 5.7 decision network สำหรับปัญหาที่ตั้งของสนามบิน

กระบวนการทำงานเของระบบการตัดสินใจของตัวแทนอัตโนมัติเป็นดังรูปที่ 5.8 นั่นคือตัวแทนรับข้อมูลจากสิ่งแวดล้อม (1) ซึ่งข้อมูลเหล่านี้จะเปลี่ยนความจำของตัวแทน (2) และตัวแทนจะทำการตัดสินใจซึ่งมีการตั้งเวลาของการตัดสินใจไว้ (3) ซึ่งกระบวนการตัดสินใจสามารถเข้าถึงความจำได้

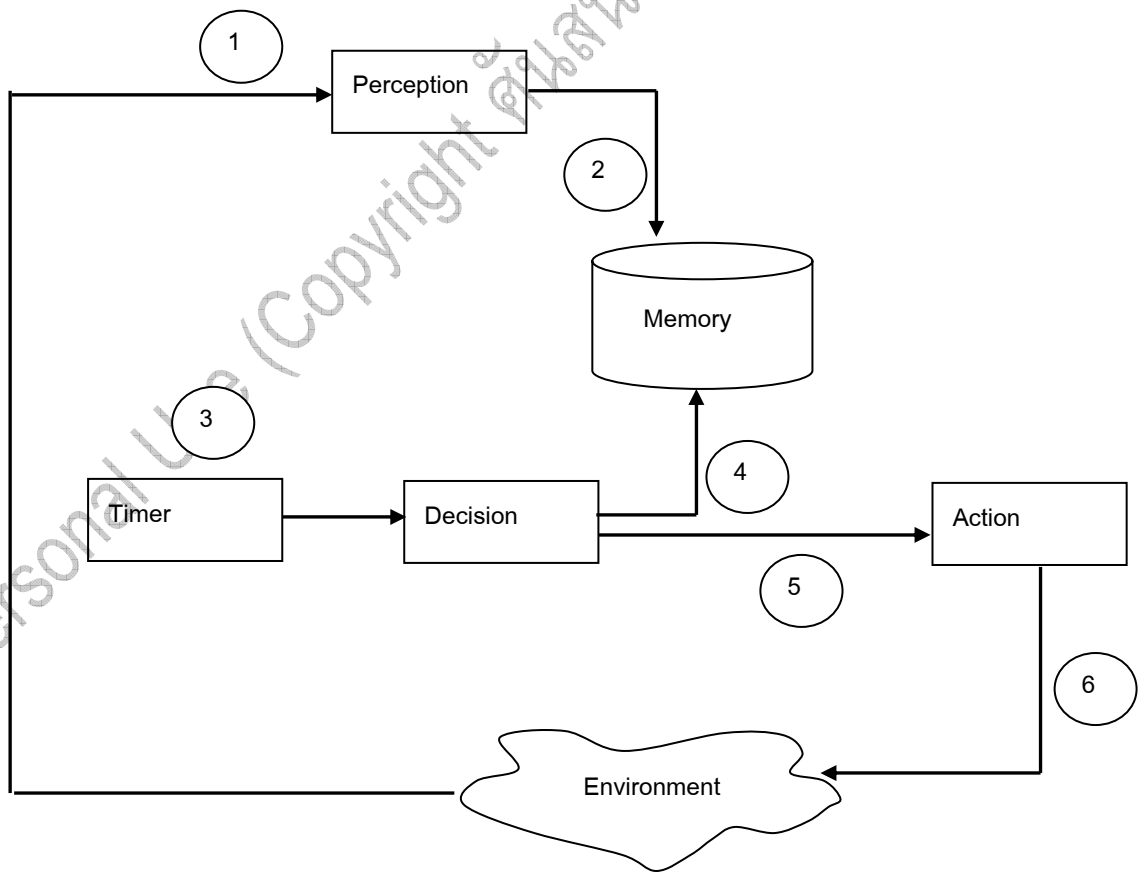
(4) และจะส่งการตัดสินใจเพื่อบอกให้ตัวแทนว่าควรจะทำอะไร (5) และการกระทำของตัวแทนจะเปลี่ยนสิ่งแวดล้อม (6)

นอกเหนือจากวิธีการตัดสินใจที่ได้กล่าวถึงไปแล้วทั้งสองวิธีนั้นยังมีวิธีการอื่นอีกมากมาย รวมทั้งการนำการหาเหตุผลโดยประมาณซึ่งได้กล่าวไปแล้วในบทที่ 2 ตัวอย่างของการนำการหาเหตุผลโดยประมาณมาใช้ในการตัดสินใจคือ สมมุติในเกมส์มีตัวแทนอัตโนมัติชื่อ Ian และได้เดินเข้าไปในคุกซึ่งจะไปเจอ Wally ซึ่งเป็นศัตรูกับ Ian ที่เผ็นตัวละครที่ถูกผู้เล่นควบคุม Ian จะได้รับข้อมูลว่า Wally เข้ามาอยู่ในระยะสายตาแล้ว Ian ต้องค้นหาในความจำเพื่อหาข้อมูลอ้างอิงเกี่ยวกับ Wally สมมุติให้ความจำมีความเกลียด (Hate) และ Ian คำนวณหาระยะห่างด้วย เช่นความเกลียดเป็น 0.77 และระยะเป็น 0.75 ดังนั้นความเกลียดและระยะเป็นตัวแปรภาษาของอินพุตให้กับระบบ โดยที่พจน์ภาษาของอินพุตความเกลียดคือ “Not hated” “Somewhat hated” “Hated” และ “Very hated” ส่วนพจน์ภาษาของระยะเป็น “At” “Very close” “Close” “Pretty far” และ “Very far” ส่วนตัวแปรภาษาของเอาต์พุตคือพฤติกรรมการต่อสู้ (combat behavior) ซึ่งมีพจน์ภาษาคือ “None” “Melee” “Crossbow” และ “Fireball” และกฎที่มีอาจจะเป็น

If ระยะเป็น “Pretty far” และ ความเกลียดเป็น “Hated” แล้วใช้ “Crossbow”

If ระยะเป็น “Pretty far” และ ความเกลียดเป็น “Very hated” แล้วใช้ “Fireball”

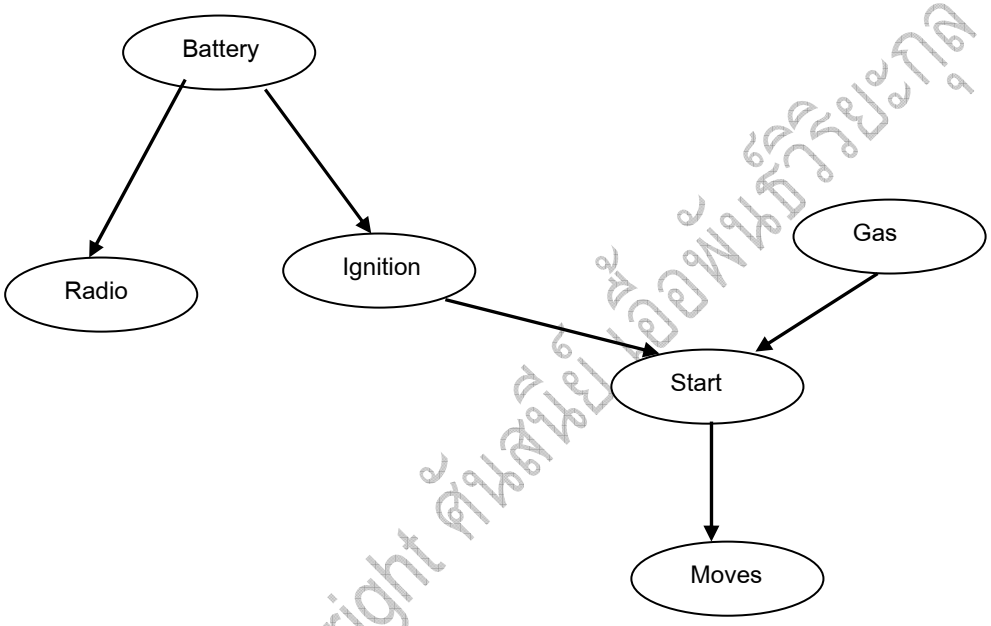
ซึ่งในกระบวนการคำนวณสามารถใช้การคำนวณที่ได้อธิบายในบทที่ 2 นั่นเอง



รูปที่ 5.8 ระบบการตัดสินใจของตัวแทนอัตโนมัติ

คำถามท้ายบทที่ 5

1. จากรูปที่ 5.9 ซึ่งเป็น network สำหรับการวินิจฉัยรถยนต์ ซึ่งแต่ละตัวแปรในรูปเป็นตัวแปรบูลีน (Boolean variable) ที่มีค่าจริงที่บ่งบอกว่าชิ้นส่วนนั้นทำงาน จงเติมตัวแปรบูลีน IcyWeather และ StarterMotor และจงหา conditional probability ที่เหมาะสม



รูปที่ 5.9 Bayesian network ที่อธิบายบางลักษณะของระบบไฟฟ้าของรถยนต์

2. ให้ $X = \{a, b, c, d\}$ ถ้ามี $m(\{a, b, c\}) = 0.5$, $m(\{a, b, d\}) = 0.2$ และ $m(x) = 0.3$ จงหาตัววัดความเชื่อ และตัววัดความเป็นไปได้

3. จงหา m รวมที่ไม่ได้แสดงในตัวอย่างที่ 5.2 พร้อมทั้งคำนวณหาตัววัดความเชื่อด้วย

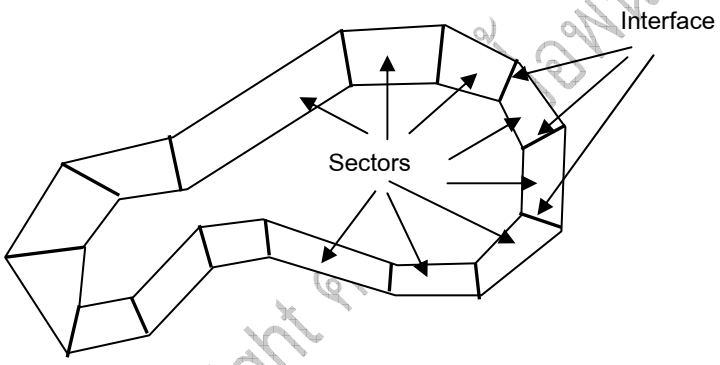
ปัญญาประดิษฐ์ในเกมส์การแข่งและกีฬา
Racing and Sports Artificial Intelligence

6

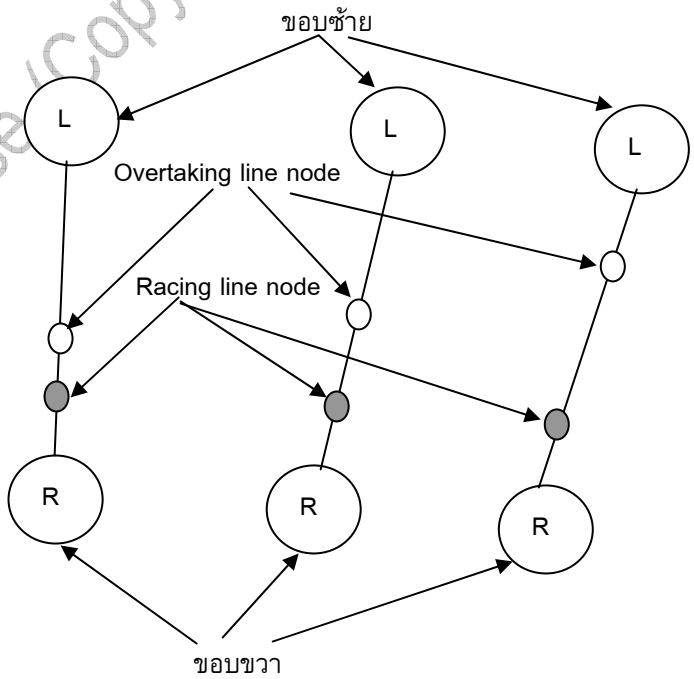
ในบทนี้จะกล่าวถึงการใช้ปัญญาประดิษฐ์ในเกมส์การแข่งและกีฬา ซึ่งประกอบไปด้วยการสร้างทางแข่ง (racetrack) และการทำให้ระบบปัญญาประดิษฐ์ทำการแข่งได้ รวมทั้งการใช้ปัญญาประดิษฐ์ในเกมส์กีฬาโดยใช้ตัวอย่างของการแข่งเบสบอล

6.1 การแทนทางแข่งเพื่อปัญญาประดิษฐ์

ทางแข่งจะถูกนิยามเป็นห่วงโซ่ของเซกเตอร์ (sector) ที่บ่งบอกว่าพื้นที่ใดเป็นพื้นที่ที่สามารถผ่านได้ แต่ทั้งนี้ทั้งนั้นอาจวิ่งอยู่ข้างนอกเซกเตอร์ได้ถ้าสิ่งแวดล้อมให้ทำได้ แต่ละเซกเตอร์จะเชื่อมกันด้วยอินเตอร์เฟส (interface) ดังแสดงในรูปที่ 6.1

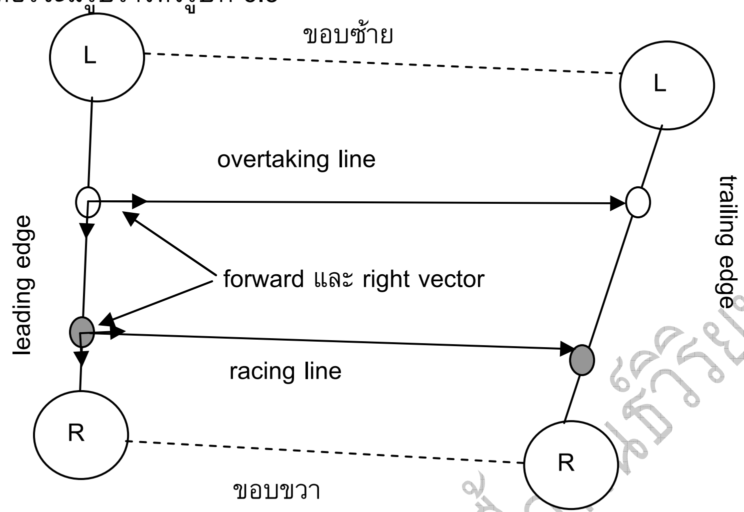


รูปที่ 6.1 ทางแข่งที่ถูกแบ่งเป็นเซกเตอร์และอินเตอร์เฟส



รูปที่ 6.2 อินเตอร์เฟศ racing line node และ overtaking line node

อินเตอร์เฟสจะนิยามขอบซ้ายและขอบขวาของถนนและเส้นขับ (driving line) ซึ่งแบ่งเป็น racing line และ overtaking line ซึ่ง node เส้นขับทั้งสองจะอยู่แถวขอบซ้ายและขอบขวานั่นเอง ซึ่งแสดงในรูปที่ 6.2 การเชื่อมระหว่าง node เหล่านี้จากอินเตอร์เฟสไปยังอีกอินเตอร์เฟสหนึ่งคือ driving lline นั่นเอง ซึ่งเซกเตอร์จะมีรูปร่างดังรูปที่ 6.3



<u>รูปที่ 6.3 การสร้างเซกเตอร์</u>

แต่ละเซกเตอร์จะเก็บระยะของทางแข่งเพื่อให้ระบบปัญญาประดิษฐ์รู้ว่าต้องเดินทางเท่าไรถึงจะถึงเส้นชัย และเพื่อการเปรียบเทียบระยะกับคู่แข่ง ซึ่งวิธีที่ง่ายที่สุดคือนับระยะตาม racing line นั่นเอง

driving line นิยามเส้นที่เหมาะสมระหว่างอินเตอร์เฟส ซึ่งแต่ละโครงสร้างของเส้นจะมีตำแหน่งของจุดเริ่มต้น ความยาว และเวกเตอร์ไปข้างหน้า (forward direction vector) และเวกเตอร์ไปทางขวา (right direction vector) เวกเตอร์ไปข้างหน้าคือเวกเตอร์เวกเตอร์จากจุดเริ่มต้นของเส้นจนถึงจุดสุดท้ายที่ถูกทำให้เป็นบรรทัดฐานหลังจากที่ทำให้ *Y* (ความสูง)เป็น 0 หรือการฉายไปยังระนาบ *XZ* ส่วนเวกเตอร์ไปทางขวาเป็นเวกเตอร์ที่ตั้งฉากกับเวกเตอร์ไปข้างหน้าซึ่งเป็นเวกเตอร์ที่วัดว่ารถยนต์อยู่ห่างจาก driving line เท่าใด

ระนาบที่ใช้ในการทำเครื่องหมายของเส้นขอบทั้ง 4 ของแต่ละเซกเตอร์ที่ถูกสร้างใน 2 มิติ เพื่อให้ง่ายต่อการทดสอบว่าจุดอยู่ข้างในเซกเตอร์หรือไม่ เซตกเตอร์ควรจะเป็นคอนเวกซ์ และปัญญาประดิษฐ์ควรจะรู้ว่ารถแต่ละคันอยู่ในเซกเตอร์ใด การทดสอบว่าตำแหน่งของรถอยู่ในด้านที่เป็นบวกของจุดทั้ง 4 ของระนาบขอบแสดงว่ารถอยู่ข้างในเซกเตอร์

เพื่อให้เป็นการง่ายต่อปัญญญาประดิษฐ์ข้อมูลต่อไปนี้ควรจะเก็ยในแต่บะเซกเตอร์

1. ชนิดของทาง ซึ่งจะบอกว่าเป็นเส้นทางชนิดใด เช่น normal shortcut long route weapon pick-up route และอื่นๆ ซึ่งปัญญาประดิษฐ์สามารถข้อมูลเหล่านี้ในการเลือกทางเช่นถ้าต้องการอาวุธอาจเลือกเดินไปทาง weapon pick-ip route เป็นต้น
2. ชนิดของลักษณะภูมิประเทศ (terrain type) เช่นปัญญาประดิษฐ์ต้องการทางลัด แต่ถ้าทางลัดเป็นทางที่ขรุขระ แล้วจะมีแต่รถที่ออกแบบมาเพื่อการนี้เท่านั้นที่จะเลือกทางนี้
3. กำแพง ทางบางทางอยู่ในพื้นที่ปิดเช่นมีช่องแคบ หรือมีกำแพงที่ขอบถนน ถ้ามีข้อมูลเหล่านี้ทำให้ปัญญาประดิษฐ์สามารถรักษาระยะได้

4. Hairpin Turn ซึ่งตัวบ่งชี้ของ hairpin left หรือ hairpin right จะเป็นสิ่วที่บอกถึงทางเลี้ยวฉับพลัน
5. brake/throttle ค่านี้จะอยู่ในช่วง −1.0 ถึง +1.0 ซึ่งเป็นตัวบ่งบอกว่าให้ทำ full brake หรือ full throttle

6.2 ลอจิกการแข่งของปัญญาประดิษฐ์

ถึงแม้ว่าปัญญาประดิษฐ์จะเดินตาม driving line แต่ไม่ได้เดินตามแบบรถไฟวิ่งบนราง แต่ใช้ driving line เป็นแนวทางเท่านั้น ซึ่งโครงของปัญญาประดิษฐ์ในกรณีมีดังนี้

1. ไฟไนต์สเตทแมชชีน ใช้เพื่อเก็บข้อมูลการแข่งในปัจจุบัน
2. ช่วงเวลาที่คงที่ (fixed time step) การใช้ช่วงเวลาที่คงที่เพื่อป้องกันไม่ให้ผลที่ได้แตกต่างกัน แต่ก็ยังอาจจะมีข้อผิดพลาดอื่นเกิดขึ้นเช่นปัญญาประดิษฐ์ต้องทำการตัดสินใจหลายอย่างในช่วงเวลาที่กำหนดซึ่งอาจทำให้ปัญญาประดิษฐ์พลาด braking point หรือตอบสนองช้าเกินไปต่อสิ่งกีดขวาง หรือไม่สอดคล้องกันเมื่อเป็นเกมส์ที่เล่นผ่าน network แต่การที่ตั้งช่วงเวลาคงที่ทำให้ลอจิกของปัญญาประดิษฐ์ทำงานเหมือนเดิมในแพลตฟลร์ม (platform) ที่ต่างกันได้
3. การควขคุมรถยนต์ ในการควบคุมรถยนต์ควรประกอบด้วยการควบคุมการเลี้ยว การเร่ง การเบรค เช่นค่าควบคุมการเลี้ยว dx จาก −1.0 ถึง +1.0 คือเลี้ยวซ้ายเต็มที่จนถึงเลี้ยวขวาเต็มที่ หรือถ้าเป็น dy ที่มีค่าตั้งแต่−1.0 ถึง +1.0 ถ้าเป็นค่าลบจะหมายถึงค่าเบรคจาก 0 ถึง 100% แต่ถ้าเป็นค่าบวกจะหมายถึงความเร่งตั้งแต่ 0 ถึง 100% เช่นกัน
4. ทำให้เป็น 2 มิติ สามารถฉาย 3 มิติให้เป็น 2 มิติในระนาบ *XZ* ได้โดยการทำให้ค่าแกน *Y* เป็น 0 และทำให้เป็นบรรทัดฐานค่าเวกเตอร์ที่จำเป็นแสดงในตารางที่ 6.1

ตารางที่ 6.1 เวกเตอร์ในระนาบ *XZ* ที่ถูกฉายมา

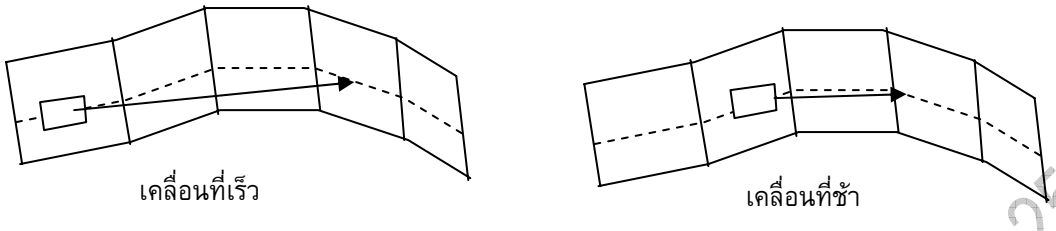
ตัวแปร	ความหมาย
Forward	ทิศทางไปข้างหน้าของรถยนต์
DestForward	ทิศทางของปลายทาง
DestRight	ทิศทางที่ตั้งฉากกับ DestForward

ปัญญาปะดิษฐ์ถูกตั้งค่าเริ่มต้นด้วยเซกเตอร์ที่รถอยู่ในตอนแรก และในกรณีที่ถ้ารถไถลออกนอกเส้นทางและต้องกลับมาที่เดิมได้ควรจะมีการเก็บเซกเตอณ์สุดท้ายที่รถอยู่ไว้ด้วย

สำหรับปัญญาประดิษฐ์ที่ต้องทำการตัดสินใจจำเป็นที่จะต้องรู้ว่าตอนนี้อยู่ที่ใดในสภาวะแวดล้อมนั้น ซึ่งในเกมสการแข่งก็คือตำแหน่งในทางแข่งนั่นเอง แต่การทดสอบว่ารถอยู่ที่ใดจ้องหาเซกเตอร์ปัจจุบันที่รถอยู่ ซึ่งเป็นการเสียเวลา ดังนั้นเพื่อป้องกันการเสียเวลาจึงต้องเก็บเซกเตอร์สุดท้ายที่รถอยู่ไว้ และถ้าต้องการรู้ว่ารถอยู่ที่ใดให้ไปตรวจสอบกับเซกเตอร์นั้นถ้ารถไม่ได้อยู่ในเซกเตอร์นั้นแล้วให้หาไปตามรายการเซกเตอร์ต่อไปและเซกเตอร์ก่อนหน้าจนกระทั่งเจอรถ

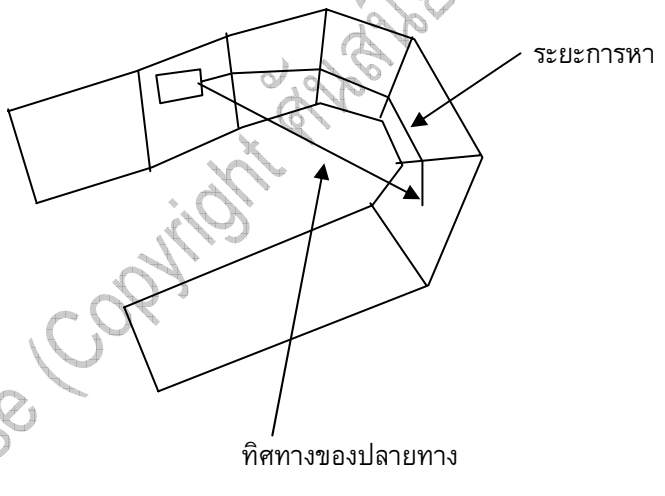
ถ้าปัญญาประดิษฐ์มาถึงทางแยกให้เลือกทาที่ดีที่สุดที่ขึ้นกับความต้องการ ซึ่งตรงนี้สามารถใช้ระบบการตัดสินใจมาช่วยได้ ปัญญาประดิษฐ์ที่ดีควรจะมีการมองไปข้างหน้าด้วยเช่นรู้ว่าจะถึงทาง

เลี้ยวฉับพลัน ควรจะให้เวลาเพียงพอสำหรับการเบรคเพื่อให้ถึงทางเลี้ยวด้วยความเร็วที่เหมาะสม การมองไปข้างหน้าทำได้โดยการเดินตามเซกเตอร์จากตำแหน่งปัจจุบันดังรูปที่ 6.4

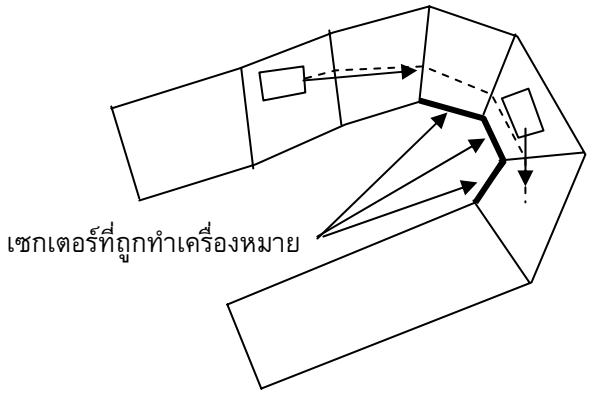


รูปที่ 6.4 การมองไปข้างหน้าของรถ

การปรับความเร็วของรถให้แปรปันกับระยะข้างหน้าข้อดีหลายอย่างเช่น เมื่อเคลื่อนที่ช้าใช้การมองเฉพาะที่ ทำให้ได้ค่าที่แน่นอน แต่ถ้าเคลื่อนที่เร็วทำให้ได้เส้นทางที่เรียบขึ้นและทำให้มีเวลาในการตอยสนองที่ดี แต่ข้อเสียของการมองไปข้างหน้าคือถ้ามีทางเลี้ยวฉับพลันทำให้รถตัดโค้งไปเลยแทนที่จะวิ่งในทางวิ่งดังแสดงในรูปที่ 6.5 (ก) สามารถแก้ได้โดยการทำเครื่องหมายเซกเตอร์ที่อยู่รอบทางเลี้ยวฉับพลัน และการหาจะถูกจำกัดลง เมื่อการหาเจอเซกเตอร์ที่ถูกทำเครื่งหมายเหล่านี้ ปัญญาประดิษฐ์จะมองหาเซกเตอร์แรก และเมื่อวิ่งจนถึงระยะที่ถูกจำกัดก็เริ่มมองหาทางต่อไปดังรูป 6.5 (ข)



(ก)

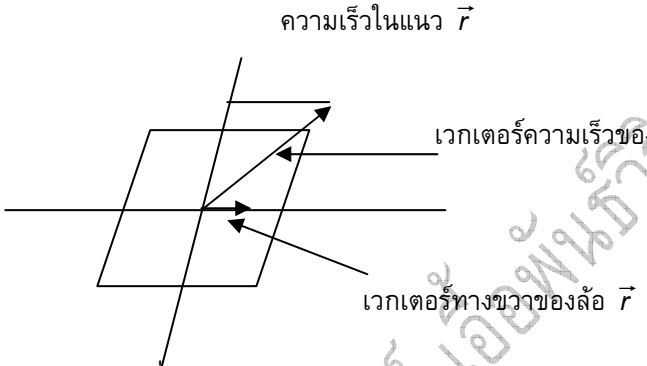


(ข)

รูปที่ 6.5 (ก) การหาเส้นทางปกติ (ข) การหาเส้นทางโดยที่มีระยะจำกัด

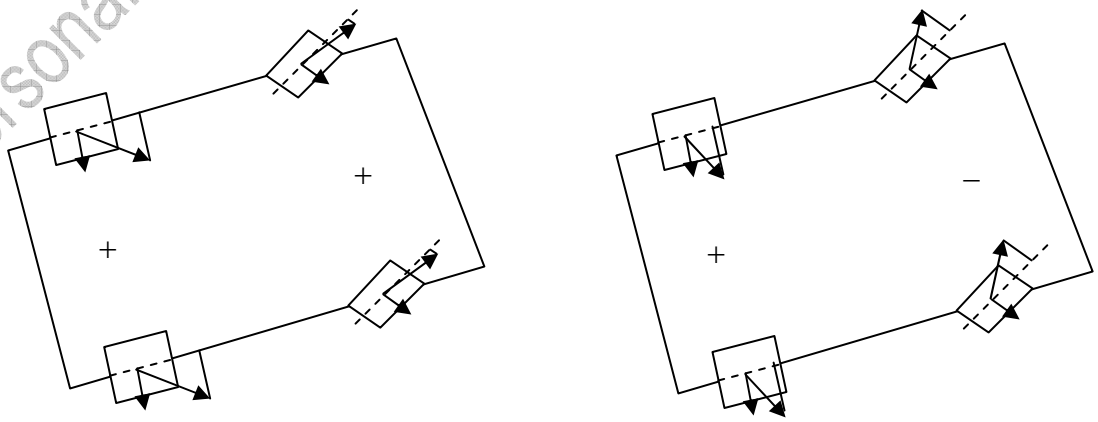
เมื่อปัญญาประดิษฐ์วิ่งเข้าใกล้คู่แข่ง จะต้องหาทางวิ่งอื่นที่จะไม่ชน สามารถทำได้โดยวิ่งบน overtaking line แทนที่จะวิ่งบน racing line และเพื่อเป็นการเพิ่มประสิทธิภาพสามารถมี overtaking line ได้มากกว่า 1 เส้น

นอกเหนือจากนี้ต้องสามารถแก้ไขการหมุนล้อของรถ ความเร่ง และหรือการเบรค ซึ่งสามารถทำได้โดยตรวจสอบความเสถียรภาพของรถโดยการเปรียบเทียบกับความเร็วของล้อแต่ละข้าง ซึ่งความเร็วของล้อแต่ละข้างคำนวณจาก dot product ระหว่างความเร็วล้อและเวกเตอร์ทางขวาของล้อ ดังรูปที่ 6.6

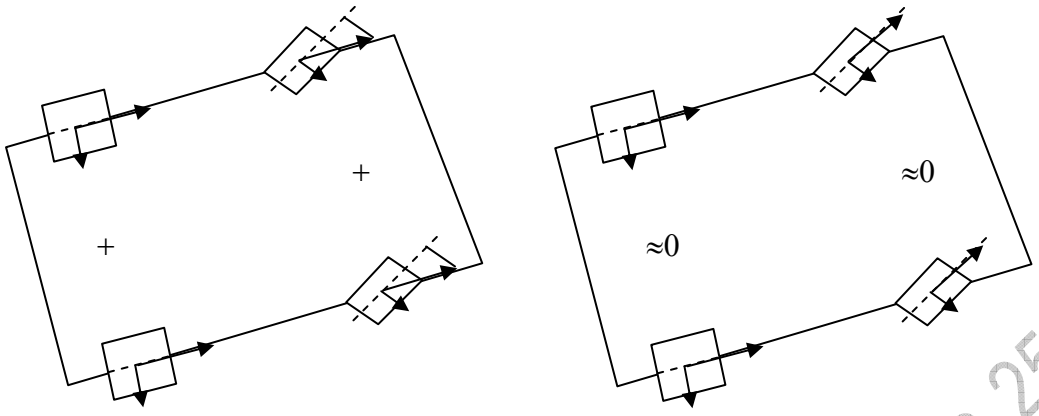


รูปที่ 6.6 การหาความเร็วของล้อ

เมื่อรถอยู่ในสถานะเสถียร คือรถจะไม่อยู่ในสถานะ under-steering หรือ over-steering ซึ่งการตรวจสอบทำได้จากการหาความเร็วของล้อ ซึ่งถ้ารถอยู่ในสถานะเสถียร ไม่จำเป็นต้องแก้ไข และการตรวจสอบ under-steeringme ซึ่งจะเกิดในกรณีที่รถพยายามเลี้ยวในขณะที่วิ่งไปข้างหน้าซึ่งตรวจสอบได้จากความเร็วของล้อหน้าและล้อหลังมีเครื่องหมายเดียวกัน และถ้าขนาดของความเร็วของล้อหน้ามากกว่าล้อหลัง ยิ่งมากเท่าไร แสดงว่าอยู่ใน under-steering มากเท่านั้น ส่วน over-steering เกิดในกรณีที่รถเลี้ยวมากเกินไป ซึ่งอาจทำให้รถหมุนได้ และกรณีนี้มี 2 กรณีด้วยกันคือ กรณีจะเกิดเมื่อ ความเร็วล้อหน้าและล้อหลังมีเครื่องหมายเดียวกัน และความเร็วล้อหลังมีขนาดมากกว่าล้อหน้า ซึ่งยิ่งมากเท่าไรยิ่ง over-steering เท่านั้น ส่วนกรณีที่ 2 คือรถหมุนไปแล้ว ซึ่งในกรณีนี้ความเร็วล้อหน้าและล้อหลังมีเครื่องหมายที่ต่างกัน ถ้าขนาดของความเร็วทั้งสองรวมกันมีค่ามากเท่าไร ยิ่งเป็น over-steering เท่านั้น ทั้ง over-steering และ under-steering แสดงในรูปที่ 6.7



(ก) (ข)



(ค) (ง)

รูปที่ 6.7 สถานะทั้ง 3 ของรถ (ก) over-steering เมื่อเครื่องหมายของความเร็วล้อหน้าและหลังเหมือนกัน (ข) over-steering เมื่อเครื่องหมายของความเร็วล้อหน้าและหลังต่างกัน (ค) under-steering เมื่อเครื่องหมายเมื่อเครื่องหมายของความเร็วล้อหน้าและหลังเหมือนกัน (ง) สถานะปกติ

ในการแก้ไข under-steering และ over-steering ทำได้โดยการหาค่าแก้ไข ซึ่งค่าแก้ไขนี้จะถูกบวกเพิ่มหรือลบออกขึ้นอยู่กับความเหมาะสม ค่าแก้ไขของ uner-steering คือ

$$correction\_understeering = \frac{\left|FrontVel + \mathrm{Re}\, arVel\right|}{UndersteerRange} \tag{6.1}$$

ส่วนค่าแก้ไข over-steering ในกรณีที่เครื่องหมายเหมือนกันคือ

$$correction\_oversteering = \frac{\left|FrontVel - \mathrm{Re}\, arVel\right|}{UndersteerRange} \tag{6.2}$$

และค่าแก้ไข over-steering ในกรณีที่เครื่องหมายต่างกันคือ

$$correction\_oversteering = \frac{\left|\mathrm{Re}\, arVel\right| - \left|FrontVel\right|}{UndersteerRange} \tag{6.3}$$

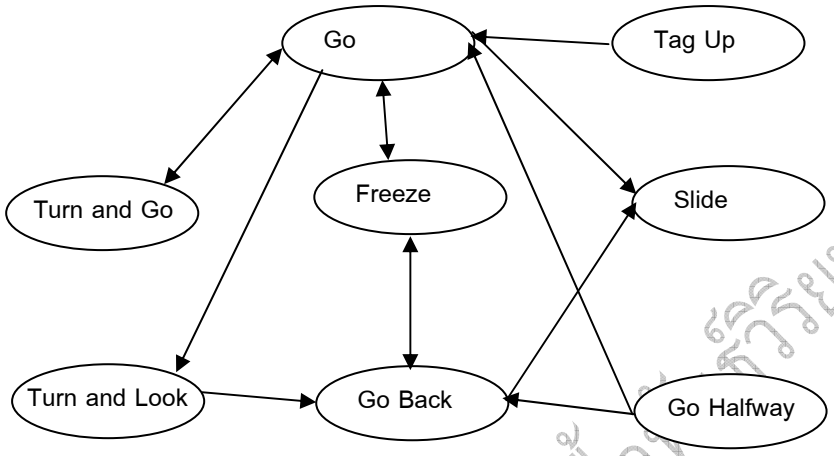
หลังจากที่ตำแหน่งที่ทำนายไว้ถูกคำนวณแล้วโดยการบวกความเร็วปัจจุบันกับจำนวนเวลาในอนาคตที่ทำนายไว้ และบวกเพิ่มตำแหน่งปัจจุบันจะได้ตำแหน่งใหม่ ซึ่งตำแหน่งใหม่นี้จะถูกตรวจสอบกับขอบซ้ายและขอบขวาเพื่อหาว่ารถจะเข้าใกล้ขอบด้านใดหรือไม่ ถ้าใกล้เดินไปก็ต้องทำการแก้ไข ซึ่งค่าแก้ไขคำนวณจาก ระยะหารด้วยขีดแบ่ง ซึ่งค่านี้จะถูกบวกเข้าหรือลบออกจะขึ้นอยู่กับขอบด้านใดที่รถจะชน ถ้ารถจะหลุดออกจากทางวิ่ง ก็ควรจะป้องกันด้วยการเบรค ซึ่งค่า dy จะถูกตั้งเป็น −1.0 และเพื่อป้องกันไม่ให้รถหมุนควรบังคับล้อให้ไปข้างหน้า จนกระทั่งความเร็วรถลดลงและรถอยู่ในเซกเตอร์

6.3 ความร่วมมือของตัวแทนในไฟไนต์สเตทแมชชีนในกีฬาเบสบอล

ในการออกแบบเกมส์กีฬาต้องนิยามว่าคัวแทนจะทำอะไรบ้าง ในกรณีนี้จะใช้พฤติกรรมเป็นสถานะในไฟไนต์สเตทแมชชีน สำหรับเบสบบอลในหัวข้อนี้เราจะใช้เพียง trackling the fielding และ baserunning เท่านั้น

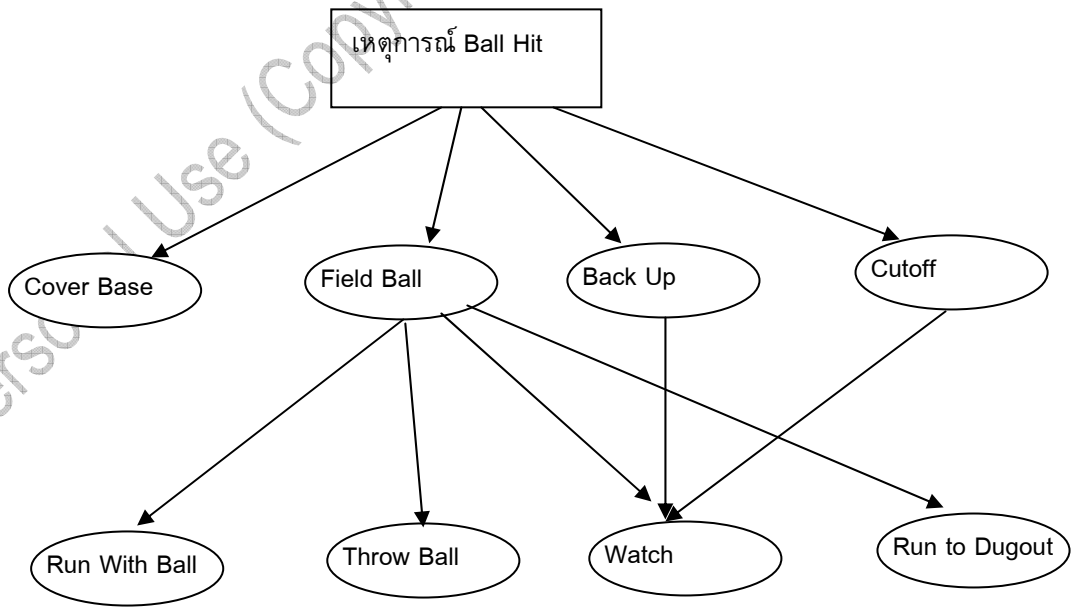
พฤติกรรมของ baserunning มีเพียง 2 อย่างที่เป็นพื้นฐานคือ วิ่งหรือไม่วิ่ง ซึ่งไฟไนค์สเตทแมชชีนของ baserunning แสดงในรูปที่ 6.8 จะเห็นว่ามีพฤติกรรม Go และสิ่งตรงข้ามคือ Go Back

ซึ่งจะทำพฤติกรรมนี้เมื่อผู้เล่นให้อินพุตหรือรับลูกบอลได้ ส่วนพฤติกรรม Freeze คือพฤติกรรมที่อยู่เฉย และพฤติกรรม Go Halfway เป็นส่วนผสมระหว่าง Go และ Freeze ส่วนพฤติกรรม Tag Up จะทำเมื่อเกิด fly ball ส่วนพฤติกรรมเกี่ยวกับแอนนิเมชันเป็นพฤติกรรม Slide Turn and Go และ Turn and Look



รูปที่ 6.8 ไฟไนต์สเตทแมชชีนของ baserunning

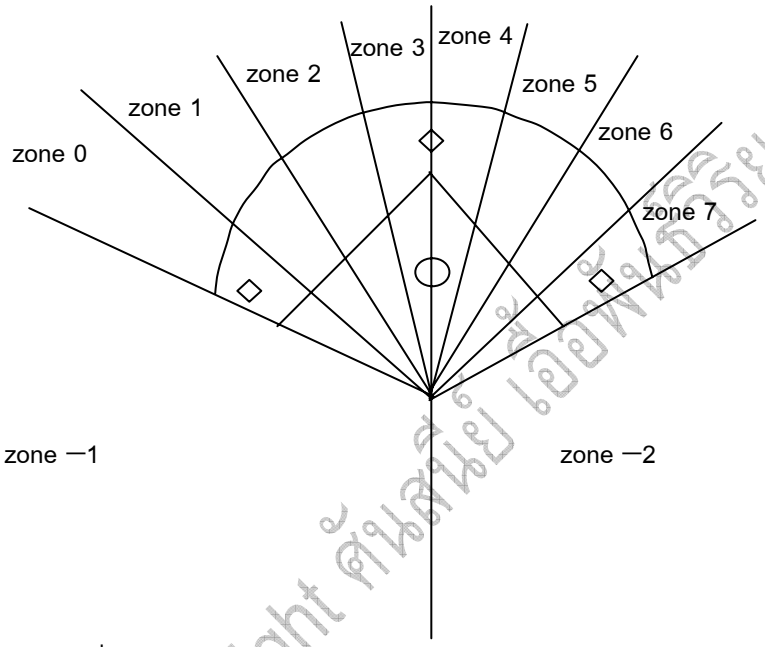
ส่วนพฤติกรรมของ fielding ขึ้นอยู่กับเหตุการณ์ 2 ประเภทคือ Ball Hit และ Ball Fielded ในเหตุการณ์ของ Ball Hit จะมีพฤติกรรม 4 อย่างคือ Cover Base Field Ball Cut Off และ Back Up ซึ่งพฤติกรรมเหล่านี้จะถูกตั้งค่าโดยดูจากที่ที่ล๔กบอลถูกตี และความเร็วของลูกบอลตอนที่ออกจากไม้ตี ซึ่งพฤติกรรมเหล่านี้แสดงในรูปที่ 6.9 และตัวแทนที่อยู่ใน Field Ball ที่ไม่มีลูกจะกลับไปอยู่ที่พฤติกรรม Watch ส่วนตัวแทนที่มีลูกจะเลือกพฤติกรรม Run With Ball Throw Ball หรือ Run to Dugout ส่วนตัวแทนที่ไม่ได้อยู่ใน Field Ball จะทำพฤติกรรม Watch เช่นกัน



รูปที่ 6.9 ไฟไนต์สเตทแมชชีนของ Fielding สำหรับเหตุการณ์ Ball Hit

ในเหตุการณ์ Ball Hit ควรจะเก็บข้อมูลที่เพียงพอต่อการให้ค่าพฤติกรรมของสมาชิกทุกคนในทีม และเราสามารถคำนวณเวกเตอร์ความเร็วต้นและเส้นทางของลูกบอล ซึ่งค่าเหล่านี้ควรจะสามารถคำนวณได้ตั้งแต่ตอนที่ตีลูกและควรคำนวณตนกระทั่งลูกบอลถูกรับ

เราสามารถเก็บตำแหน่งของลูกบอล และสามารถหา hit type และ hit zone ได้จากความเร็วต้นของลูกบอลตอนที่ถูกตี ซึ่ง zone ถูกแบ่งออกเป็น 10 zone ดังรูปที่ 6.10 ในการคำนวณหา hit zone ทำได้โดยใช้การคำนวณปกติ แต่การหา hit type หาได้จากการใช้กระบวนการตัดสินใจต่างๆ ในรูปที่ 6.10 zone ที่เป็นลบทั้งสองเป็น zone ที่เป็นเขตฟาวล์ (foul zone)



รูปที่ 6.10 การหา zone สำหรับความเร็วต้นของลูกบอล

การหา hit type จะเป็นสิ่งที่ซับซ้อนกว่าซึ่งสามารถเป็นไปได้ในหลายแนวทาง แต่ในที่นี้จะทำให้ง่ายโดยการกำหนดให้มี 3 ชนิดคือ hit type ground ball hit type fly ball และ hit type line drive ซึ่งการหา hit type เหล่านี้สามารถใช้ระบบกฎมาช่วยในการตัดสินใจได้ เมื่อคำนวณ hit type และ zone ได้แล้วสามารถหาพฤติกรรมที่เหมาะสมได้เช่น hit type ground ball ใน zone 7 ซึ่งจะเห็นว่า baseman คนแรกและ field ด้านขวาควรจะมีพฤติกรรม Field Ball ส่วน base คนที่สาม shortstop catcher และ pitcher ควรจะเป็น Cover Base ส่วน baseman คนที่สองจะทำ Cutoff หรือ Field Ball ขึ้นอยู่กับความเร็วต้นของลูกบอล และ fielder ด้านซ้ายและตรงกลางจะทำ Back Up เพื่อมองหา fielder และ/หรือ base เพื่อระวังหลังเป็นต้น แต่ถ้าเป็น hit type fly ball ที่ zone 7 ทุกตัวแทนทำพฤติกรรมเดิมยกเว้น baseman คนแรกที่จะทำ Cover Base แทนที่จะทำ Field Ball และ Pitcher จะทำ Back Up แทนที่ Cover Base

ถ้าเป็น hit type grounder ใน zone −1 หรือ zone −2 ทุกตัวแทนจะทำ Watch แต่ถ้าเป็น hit type popup จะเป็น baseman คนแรก คนที่สามและ catcher ที่จะทำ Field Ball ส่วน hit type drive หรือ hit type fly ball จะเป็น fielder ด้านซ้ายและขวา รวมทั้ง baseman คนแรก และคนที่สามที่จะทำ Field Ball

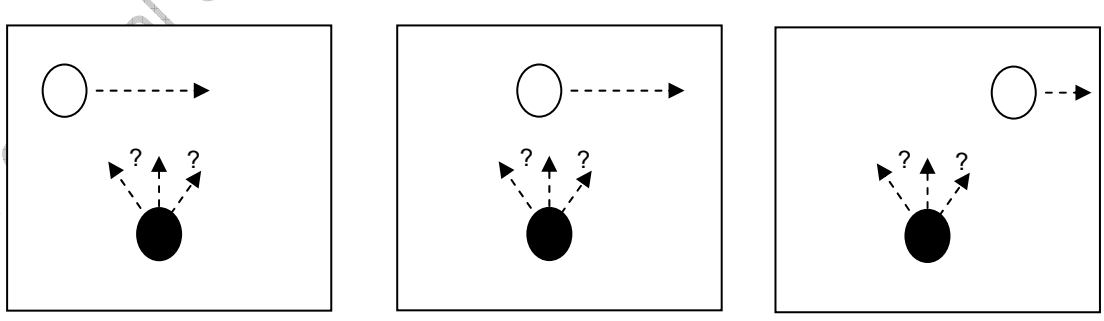
สำหรับ baserunning พฤติกรรมค่อนข้างจะไม่ซับซ้อนเนื่องจากต้องการวิ่งเพื่อให้ได้คะแนน ดังนั้นการให้ค่าพฤติกรรมจึงค่อนข้างง่าย ในการพิจารณาพฤติกรรมในส่วนนี้ยังคงใช้ hit type และ

zone เช่นกัน เช่น hit type ground ball จะมี 2 พฤติกรรมคือ Go และ Go Back ตัวแทนที่เป็น runner ที่มีโอกาสจะทำ Go ส่วนตัวแทนที่ไม่มีโอกาสมากเท่าจะทำ Go ถ้าลูกบอลถูกตีไปที่ zone 4 ถึง 7 ในทำนองเดียวกัน hit type fly ball ก็จะทำให้เกิด Go Halfway หรือ Tag Up แต่ถ้าเป็น hit type line drive ให้ทำ Freeze สำหรับทุกตัวแทน แต่การทำงานร่วมกันของทุกตัวแทนเป็นสิ่งที่ซับซ้อนกว่า fielder นั่นคือถ้า runner วิ่งไปจนถึงประมาณ 20 ฟุตหน้า base ซึ่งจะทำสิ่งที่คล้ายกับมนุษย์นั่นคือ ตรวจสอบ fielder คนที่อยู่ใกล้ลูกบอลที่สุด หรือถ้าลูกบอลถูกรับได้และถูกโยนกลับมา ลูกบอลนั้นถูกโยนไป base ใดหลังจากนั้นค่าเหล่านี้จะผ่านระบบการตัดสินใจว่าจะทำ Turn and Go หรือ Turn and Look หรือ Slide

ส่วนเหตุการณ์ Ball Field เมื่อลูกบอลอยู่ใน field ทุกตัวแทนที่เป็น field ยกเว้นตัวแทนที่มีลูกบอลต้องทำให้แน่ใจว่าทุก base มีคนอยู่ซึ่งส่วนนี้ไม่ซับซ้อนแต่ส่วนที่ยากกว่าน่าจะเป็นต้องตัดสินใจว่าจะทำอย่างไรกับลูกบอล ซึ่งสามารถใช้การคำนวณหาระยะว่าอยู่ไกลจาก base ที่จะขว้างไปเท่าใดและสถานการณ์ของเกมส์เป็นอย่างไร นั่นคือถ้าเป็น fly ball 95% ของการโยนจะเป็นการโยนกลับไปที่ home หรือ base แรก

6.4 การรับลูกบอล

ในหัวข้อนี้อาจนำไปใช้ในการรับลูกบาสเกตบอลหรือในการเล่นเบสบอลหรือการเล่นกีฬาแบบอื่นที่มีการรับลูกก็ได้ สมมุติให้วัตถุชนิดหนึ่งอยู่ที่ตำแหน่ง P_b และเดินทางด้วยความเร็วคงที่ (V_b) เป็นเส้นตรง และวัตถุอีกอันอยู่ที่ตำแหน่ง P_p ต้องการรับลูกบอลแต่ไม่สามารถเคลื่อนที่เร็วกว่าความเร็ว S จากข้อมูลเหล่านี้ต้องการรับลูกบอลด้วยความเร็ว V_p แต่โดยปกติลูกบอลไม่ได้เคลื่อนที่เป็นเส้นตรง ดังนั้นจึงต้องแบ่งแบบจำลองออกเป็น 2 ส่วนคือ ระดับความสูงของลูกบอล และการเคลื่อนที่ในระนาบพื้น ซึ่งแกนทั้งสองในระนาบนี้จะตั้งฉากกับระดับความสูงของลูกบอล และเราทำให้การพิจารณาง่ายขึ้นด้วยการทำให้วัตถุไม่มีการหมุน และเคลื่อนที่ด้วยความเร็วสูงสุด แต่การทำเช่นนี้ทำให้ผู้เล่นจริงตัดสินสถานการณ์ได้ลำบากขึ้น และต้องใช้วิธีการอื่นที่สามารถชดเชยการเปลี่ยนทิศได้ ดังนั้นในแบบจำลองนี้มีตัวแปร 4 ชนิดคือ ตำแหน่งของลูกบอล ความเร็วของลูกบอล ตำแหน่งของผู้รับบอล และความเร็วสูงสุดที่ผู้รับบอลเคลื่อนที่ได้ สถานการณ์ที่กล่าวถึงนี้แสดงในรูปที่ 6.11



รูปที่ 6.11

สำหรับการรับบอลที่จะเกิดขึ้นได้ ตำแหน่งของลูกบอลและผู้เล่นต้องเป็นตำแหน่งเดียวกันที่เวลา t ดังนั้นถ้าเรารู้ V_p ก่อนสมการในการตัดลูกบอลเป็นดังนี้

$$P_b + V_b t = P_p + V_p t \qquad (6.4)$$

แต่ในความเป็นจริง V_p ก็เป็นอีกตัวแปรหนึ่งที่ต้องคำนวณหาเช่นกัน ดังนั้นการแก้ปัญหาต้องเปลี่ยนเป็นวิธีอื่น เราทราบว่าตำแหน่งของผู้เล่นและลูกบอลเป็นตำแหน่งเดียวกันดังนั้นระยะระหว่างผู้เล่นที่เวลาเริ่มต้นและลูกบอลที่เวลา t เป็น $|(P_b - P_p) + V_p t|$ ซึ่งสามารถมองได้เป็น 2 ส่วนคือระยะระหว่างลูกบอลและผู้เล่นในตอนแรก และการเคลื่อนที่ของลูกบอลเนื่องจากเวกเตอร์ความเร็ว ถ้าผู้เล่นสามารถเคลื่อนที่ด้วยระยะที่เท่ากับระยะห่างจากลูกบอลผู้เล่นจะตัดลูกบอลได้นั่นคือ

$$|(P_b - P_p) + V_p t| = s_p t \tag{6.5}$$

เนื่องจากตำแหน่งเริ่มต้นไม่เปลี่ยนและเพื่อความง่าย ให้ $P = (P_b - P_p)$ และตัวห้อยของ s และ V สามารถเอาออกได้ ดังนั้นสมการที่ 6.5 เป็น

$$|P + Vt| = st \tag{6.6}$$

ซึ่ง
$$\sqrt{(P + Vt)\bullet(P + Vt)} = st \tag{6.7}$$

ทำให้
$$(P + Vt)\cdot(P + Vt) = (st)^2 \tag{6.8}$$

$$P\cdot P + 2P\cdot Vt + V\cdot Vt^2 = s^2t^2 \tag{6.9}$$

$$(V\cdot V - s^2)t^2 + (2P\cdot V)t + (P\cdot P) = 0 \tag{6.10}$$

ซึ่

สมการนี้สามารถหาคำตอบได้โดยใช้การแก้สมการกำลัง 2 ซึ่งคือ $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ ซึ่งในกรณีนี้

$$\begin{aligned} b^2 - 4ac &= (2P\cdot V)^2 - 4(V\cdot V - s^2)(P\cdot P) \\ &= 4(P\cdot V)^2 - 4(V\cdot V - s^2)(P\cdot P) \\ &= (P\cdot V)^2 - (V\cdot V - s^2)(P\cdot P) \\ &= (P\cdot V)^2 + (s^2 - V\cdot V)(P\cdot P) \end{aligned} \tag{6.11}$$

ดังนั้นคำตอบจะมี 3 ลักษณะคือ ไม่มีคำตอบที่เป็นจำนวนจริง นั้นคือ $(s^2 - V\cdot V)$ เป็นลบแสดงว่า $s<|V|$ ซึ่งจะเกิดขึ้นเมื่อลูกบอลเคลื่อนที่ด้วยความเร็วที่เร็วกว่า ความเร็วสูงสุดของผู้เล่น ส่วนอีกลักษณะคือมีคำตอบที่เป็นจำนวนจริง ซึ่งจะเกิดขึ้นเมื่อผู้เล่นรับลูกบอลได้ และในกรณีนี้สมการที่ 6.11 ต้องเป็น 0 ทำให้

$$\frac{-b}{2a} = \frac{-(P\bullet V)}{2\left(V\bullet V - s^2\right)} \tag{6.12}$$

และจากสมการที่ 6.11 จะได้ว่า

$$(P\cdot V)^2 + (s^2 - V\cdot V)(P\cdot P) = 0$$

$$(P\cdot V)^2 = -(s^2 - V\cdot V)(P\cdot P) \tag{6.13}$$

จะเห็นได้ว่าด้านซ้ายของสมการที่ 6.13 ต้องมีค่าเป็นบวกทำให้เทอมด้านขวาต้องเป็นลบทำให้รู้ว่าผู้เล่นตองวิ่งเร็วกว่าลูกบอลทำให้ตัวหารในสมการที่ 6.12 เป็นบวกนั่นเอง และเราสามารถอธิบายสมการที่ 6.12 ได้ว่า

1. $(P\cdot V) < 0$ ทำให้ตัวตั้งในสมการที่ 6.12 เป็นบวกแสดงว่าจะรับลูกบอลได้ในอนาคตได้เพียงเวลาเดียวเท่านั้น

2. $(P \cdot V) > 0$ ทำให้สมการที่ 6.12 เป็นลบแสดงว่าการรับลูกบอลเกิดขึ้นในอดีตซึ่งเป็นไปไม่ได้

ส่วนลักษณะสุดท้ายคือมีคำตอบเป็นจำนวนจริง 2 ค่าซึ่งในกรณีนี้ผู้เล่นไม่จำเป้นต้องวิ่งเร็วมากถ้าผู้เล่นอยู่ไกลจากตำแหน่งเริ่มต้นของลูกบอล และอยู่ใกล้จุดที่เป็นคำตอบก็รับลูกบอลได้ ซึ่งในกรณีนี้มีคำตอบที่เป็น

1. บวกทั้งคู่ ซึ่งสามารถรับลูกบอลที่เวลาใดก็ได้ใน 2 คำตอบนี้ ในกรณีนี้ลูกบอลวิ่งเร็วกว่าผู้เล่น แต่ผู้เล่นอยู่ในเส้นทางของลูกบอกจึงรับลูกบอลได้
2. ลบทั้งคู่ กรณีก็เป็นเช่นเดียวกับกรณีแรก แต่เวลาทั้งสองเกิดในอดีตจึงไม่สามารถรับลูกบอลได้
3. บวก 1 ค่าและลบ 1 ค่า ส่วนในกรณีนี้ผู้เล่นวิ่งเร็วกว่าลูกบอกจึงสามารถรับลูกบอลได้ แต่คำตอบที่เป็นลบไม่สามารถใช้ได้เพราะเป็นเวลาในอดีต

หลังจากได้เวลามาแล้วสามารถคำนวณหาตำแหน่งที่จะวิ่งไปรับลูกบอลได้ แต่ในกรณีของคำตอบที่เป็นจำนวนจริง 2 คำตอบ ควรจะเลือกเวลาที่อยู่ประมาณตรงกลางของช่วง ซึ่งถูกเรียกว่า จุด lazy แต่เรารู้ว่า ความเณ้วของการรับควรจะน้อยที่สุดทเที่เป็นไปได้ และที่ความเร็วต่ำสุดของการรับที่ยอมรับได้มีจุดรับลูกเพียง 1 จุด ดังนั้นคำตอบควรจะมีแค่ 1 เท่านั้น สำหรับคำตอบจริง 1 คำตอบได้ว่า

$$(P \cdot V)^2 + (s^2 - V \cdot V)(P \cdot P) = 0$$

$$(P \cdot V)^2 + s^2(P \cdot P) - (V \cdot V)(P \cdot P) = 0$$

$$s^2(P \cdot P) = (V \cdot V)(P \cdot P) - (P \cdot V)^2$$

$$s^2 = \frac{(V \bullet V)(P \bullet P) - (P \bullet V)^2}{(P \bullet P)}$$

$$s = \sqrt{\frac{(V \bullet V)(P \bullet P) - (P \bullet V)^2}{(P \bullet P)}} \qquad (6.14)$$

หลังจากที่คำนวณหา s ได้สามารถหา t ได้และสามารถหาตำแหน่งที่รับลูกบอลได้ในที่สุด

วิธีการที่พูดถึงในบทนี้เป็นเพียงวิธีการง่ายๆ และนอกเหนือจากนี้ยังมีสถานการณ์อื่นที่อาจจะซับซ้อนกว่านี้ ซึ่งอาจจะจำเป็นที่จะต้องใช้ปัญญาประดิษฐ์ที่ซับซ้อนขึ้นในการแก้ปัญหา

คำถามท้ายบทที่ 6

1. จงเขียนโปรแกรมอย่างง่ายในการแข่งรถ
2. จงเขียนโปรแกรมอย่างง่ายในการแข่งบาสเกตบอล

การเรียนรู้ Learning 7

แนวคิดเบื้องลักของการเรียนรู้คือไม่เพียงแต่การรับรู้ถูกใช้ในการกระทำของตัวแทนแต่เอาไว้พัฒนาหรือเพิ่มความสามารถของตัวแทนในอนาคต การเรียนรู้เป็นได้ตั้งแต่ความจำจากประสบปารณ์จนถึงการสร้างทฤษฎีทางวิทยาศาสตร์ ซึ่งในบทนี้จะกล่าวถึงการเรียนรู้แบบอุปนัย (inductive learning)

7.1 รูปแบบของการเรียนรู้ (Forms of Learning)

โดยปกติการเรียนรู้จะถูกแบ่งออกเป็น 3 ประเภทใหญ่คือ supervised unsupervised และ reinforcement learning ซึ่งปัญหาของ supervised learning เกี่ยวกับการเรียนรู้ฟังก์ชันของอินพุตและเอาต์พุตจากตัวอย่าง ตัวอย่างของ supervised learning เช่น ตัวแทนเรียนขับรถโดยสารประจำทางและทุกครั้งที่ครูบอกให้เบรค ตัวแทนสามารถเรียนรู้เงื่อนไขของการกระทำเบรคว่าต้องเบรคเมื่อไร ซึ่งในกรณีนี้ตัวแทนเรียนเงื่อนไขของการเบรคโดยที่เป็นฟังก์ชันที่ส่งทอดสถานะไปยังเอาต์พุตที่เป็นบูลีน (เบรคหรือไม่เบรค) หรือการมองรูปที่มีรถบัสหลายรูป ตัวแทนสามารถเรียนที่จะจำรถบัสได้ ซึ่งในกรณีนี้ตัวแทนเรียนฟังก์ชันที่ส่งทอดรูปภาพไปยังเอาต์พุตที่เป็นบูลีน (มีหรือไม่มีรถบัส) หรือการลองทำการกระทำและสังเกตุผลลัพธ์ เช่นการเบรคแรงบนถนนที่เปียก ตัวแทนสามารถเรียนผลกระทบของการกระทำเหล่านั้นได้ แต่ในกรณีนี้ทฤษฎีการเบรคเป็นฟังก์ชันจากสถานะไปยังการรกะทำการเบรค (ระยะหยุดเท่าไร) ซึ่งในกรณีนี้จะเห็นว่าค่าของเอาต์พุตได้จากการรับรู้ของตัวแทน สำหรับสภาวะแวดล้อมที่มีการสังเกตุการณ์เต็มรูปแบบ ตัวแทนจะสามารถรับรู้ผลกระทบของการกระทำและสามารถใช้ supervised learning ในการเรียนรู้และสามารถทำนายได้ แต่สำหรับสภาวะแวดล้อมที่มีการสังเกตุการณ์ไม่เต็มรูปแบบ ปัญหาจะค่อนข้างซับซ้อนเพราะผลกระทบฉับพลันไม่สามารถมองเห็นได้

ปัญหาของ unsurpervised learning จะเกี่ยวกับการเรียนรู้รูปแบบของอินพุตโดยไม่มีค่าเอาต์พุตเกี่ยวข้อง ตัวอย่างเช่นตัวแทนที่ขับรถยนต์โดยสารประจำทางอาจจะค่อย ๆพัฒนาแนวคิดเรื่องวันที่การจราจรดี และวันที่การจราจรไม่ดีได้เองโดยที่ไม่ต้องมีตัวอย่างของแต่ละวันมาให้ก่อน การเรียนรู้แบบ unsurpervised learning อย่างเต็มรูปแบบไม่สามารถเรียนว่าจะทำอะไรเพราะว่าไม่มีข้อมูลเกี่ยวกับอะไรคือการกระทำที่ถูกต้องหรือสถานะที่ต้องการ

ปัญหาของ reinforcement learning เป็นหัวข้อที่ทั่วไปที่สุดในการเรียนรู้ทั้งสามแบบ คือแทนที่จะเรียนว่าจะทำอะไรจากครู จะเป็นการเรียนจากการให้รางวัล ตัวอย่างเช่นการได้คะแนนในการแข่งปิงปองบ่งบอกว่าเป็นการกระทำที่ดี

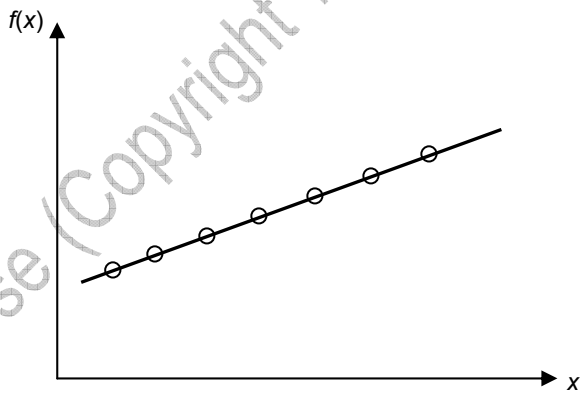
7.2 การเรียนรู้แบบอุปนัย (Inductive Learning)

อัลกอริทึมสำหรับ supervised learning แบบกำหนด (deterministic) เป็นการเรียนรู้ที่มีตัวอย่างอินพุตและเอาต์พุต (x, $f(x)$) และต้องการฟังก์ชัน f หรือฟังก์ชันที่ใกล้เคียงกับฟังก์ชัน f กระบวนการของการอนุมานแบบอุปนัย (inductive inference) หรือการอุปนัยคือ ให้ตัวอย่างของ f

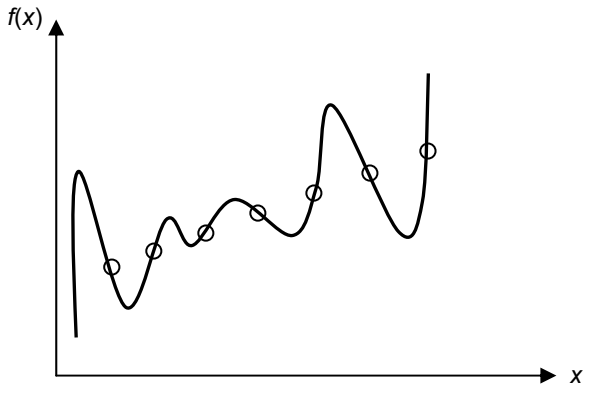
หลายตัวอย่างและพยายามหาฟังก์ชัน h ที่ประมาณฟังก์ชัน f โดยที่ฟังก์ชัน h ถูกเรียกเป็นสมมุติฐาน สาเหตุที่การเรียนรู้แบบนี้ทำได้ยากเนื่องจากไม่สามารถบอกได้ว่า h ที่ได้เป็นการประมาณ f ที่ดีหรือไม่ สมมุติฐานที่ดีจะต้อง generalize ดีด้วยนั่นคือสามารถทำนายเอาต์พุตของตัวอย่างที่ไม่เคยเห็นได้อย่างถูกต้อง ซึ่งนี่คือปัญหาพื้นฐานของการอุปนัย

รูปที่ 7.1 แสดงถึงตัวอย่างของการหาฟังก์ชันของตัวอย่าง $(x, f(x))$ โดยที่ x และ $f(x)$ เป็นค่าจริง สมมุติฐานที่ถูกเลือกคือให้เป็นเซตของฟังก์ชันพหุนาม (polynomial function) ที่มีระดับ (degree) อย่างมาก k เช่น $3x^2 + 2$, $x^{17} - 4x^3$ เป็นต้น ซึ่งในรูปที่ 18.1(ก) ฟังก์ชันที่ได้จะเป็นฟังก์ชันเส้นตรงที่เหมาะกับข้อมูลชุดนี้พอดี นั่นคือเส้นตรงที่ได้เปฯสมมุติฐานต้องกัน (consistent hypothesis) เพราะสามารถได้เอาต์พุตเช่นเดียวกับข้อมูลทุกอย่าง ส่วนรูปที่ 7.1(ข) เป็นฟังก์ชันพหุนามที่มีระดับสูงขึ้นและต้องกันกับข้อมูลเช่นกัน ซึ่งนี้แดสงให้เห็นถึงประเด็ยแรกของการเรียนรู้แบบอุปนัยว่า เมื่อมีสมมุติฐานต้องกันมากกว่า 1 สมมุติฐานอันใดควรจะถูกเลือก ซึ่งหนึ่งในคำตอบคือ ให้เลือกสมมุติฐานที่ง่ายที่สุด (Ockham) นั่นคือสมมุติฐานใดที่ยากกว่าตัวข้อมูลเองจะไม่สามารถดึงรูปแบบออกจากข้อมูลได้ ซึ่งในกรณีนี้คือเส้นตรง

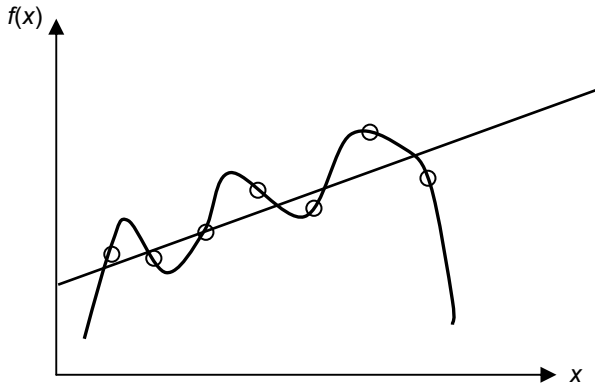
รูปที่ 7.1(ค) เป็นชุดข้อมูลอีกชุดหนึ่งซึ่งในชุดนี้ไม่มีสมมุติฐานต้องกันที่เป็นเส้นตรง และสมมุติฐานตองกันที่ได้จะเป็นฟังก์ชันพหุนามที่มีระดับ 6 (7 ตัวแปร) ซึ่งในกรณีนี้ชุดข้อมูลมีตัวอย่าง 7 ตัวอย่างและฟังก์ชันที่ได้มีตัวแปรเท่ากับจำนวนตัวอย่าง ซึ่งไม่น่าจะเป็นฟังก์ชันที่ดี สางที่ดีกว่าน่าจะเป็นเส้นตรงเพราะถึงแม้ว่าจะไม่เหมาะสมกับชุดข้อมูลมากนักแต่อย่างน้อยน่าจะให้คำตอบที่สมเหตุสมผล



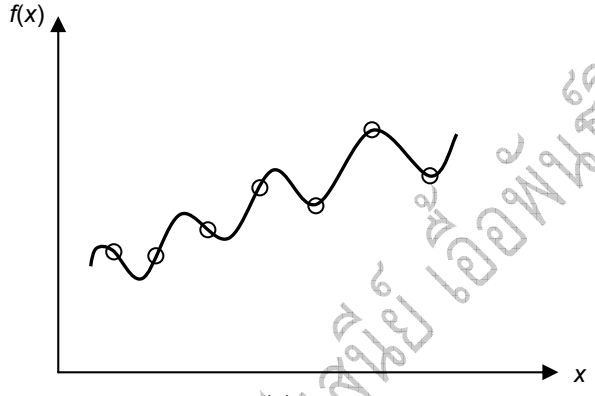
(ก)



(ข)



(ค)



(ง)

รูปที่ 7.1 (ก) ตัวอย่าง (x, $f(x)$) และสมมุติฐานต้องกันที่เป็นเส้นตรง (ข) สมมุติต้องกันที่เป็นฟังก์ชันพหุนามที่มีระดับ 7 (ค) ชุดข้อมูลอีกชุดและสมมุติฐานต้องกันที่มีระดับ 6 และเส้นตรงที่ประมาณว่าจะต้องกัน (ง) ฟังก์ชันรูปไซน์ที่ต้องกันกับชุดข้อมูลที่ 2

จากรูปที่ 7.1(ค) จะเห็นได้ว่าสมมุติฐานที่อยู่ในรูปของ $ax + b + c\sin(x)$ เป็นฟังก์ชันที่ต้องกันกับชุดข้อมูลอันที่ 2 ซึ่งแสดงให้เห็นว่าการเลือกสมมุติฐานเป็นสิ่งที่สำคัญ และถ้าในมิติของสมมุติฐานมีฟังก์ชันจริงเราเรียกปัญหาการเรียนรู้นั้นว่าเป็นปัญหาที่ realizable แต่ถ้าในมิติของสมมุติฐานไม่มีฟังก์ชันจริงเราเรียกปัญหาการเรียนรู้นั้นว่าเป็นปัญหาที่ unrealizable ซึ่งไม่สามารถบอกได้ว่าปัญหานั้นจะ realizable หรือ unrealizable ดังนั้นเราจำเป็นต้องใช้ความรู้เบื้องต้น (prior knowledge) มาสร้างสมมุติฐานซึ่งเรารู้ว่าฟังก์ชันต้องมีอยู่ในนี้แน่นอน

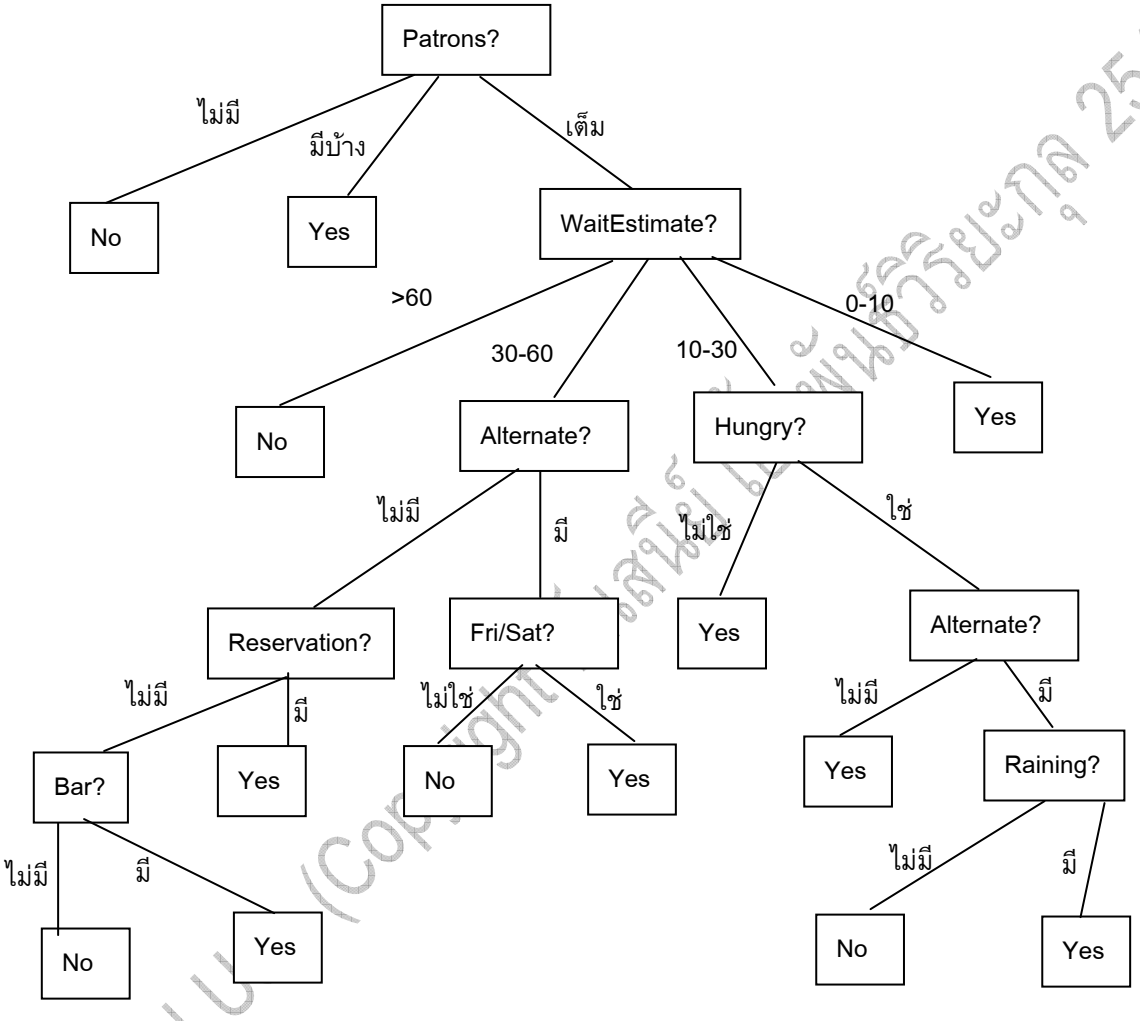
ทางแก้อีกทางหนึ่งคือใช้มิติของสมมุติฐานที่ใหญ่ที่สุด แต่การใช้มิติของสมมุติฐานที่ใหญ่จะต้องแลกกับการคำนวณที่ใช้เวลานาน ดังนั้นในงานส่วนใหญ่จะเป็นการเรียนรู้ที่เกี่ยวข้องกับการแทนที่ง่าย

7.3 ต้นไม้ตัดสินใจ (Decision Tree)

ต้นไม้ตัดสินใจรับอินพุตที่เป็นวัตถุหรือสถานการณ์ที่ถูกอธิบายด้วยเซตของลักษณะประจำ (attribute) และให้เอาต์พุตเป็นการตัดสินใจ (ค่าที่ทำนายได้จากอินพุต) ลักษณะประจำของอินพุตเป็นได้ทั้งค่าที่ไม่ต่อเนื่องและต่อเนื่อง และค่าเอาต์พุตเป็นได้ทั้งค่าที่ไม่ต่อเนื่องและต่อเนื่อง การเรียนรู้ฟังก์ชันที่ใช้ค่าไม่ต่อเนื่องเรียกว่า การจำแนก ส่วนการเรียนรู้ฟังก์ชันที่ใช้ค่าต่อเนื่องเรียกว่า การ

ถดถอย (regression) แต่ในบทนี้เราจะกล่าวถึงการจำแนกแบบบูลีน (Boolean classification) ที่แต่ละตัวอย่างถูกจำแนกเป็น จริงหรือบวก (true or positive) และ เท็จหรือลบ (false or negative)

ต้นไม้ตัดสินใจทำการตัดสินใจโดยทำการทดสอบตามลำดับ แต่ละ node ในต้นไม้จะเกี่ยวข้องกับการทดสอบค่าของคุณสมบัติ 1 อย่าง และกิ่งที่ออกไปจาก node จะถูกติดป้ายด้วยค่าที่เป็นเป็นไปได้ของการทดสอบนั้น แต่ละบัพใบ (leaf node) จะบ่งบอกถึงค่าที่จะถูกคืนกลับมาถ้ามาถึงที่บัพใบนั้น



รูปที่ 7.2 ต้นไม้ตัดสินใจในการตัดสินใจว่าจะรอโต๊ะที่ร้านอาหารหรือไม่

ในการอธิบายหัวข้อนี้จะใช้การอธิบายจากตัวอย่างดังต่อไปนี้ สมมุติปัญหาคือจะรอโต๊ะที่ร้านอาหารหรือไม่ ในการเรียนรู้แบบนี้เราต้องสร้างลักษณะประจำก่อน สมมุติเราเลือกลักษณะประจำดังนี้

1. Alternate: มีร้านอาหารอื่นที่อยู่บริเวณเดียวกันหรือไม่
2. Bar: ที่ร้านอาหารนี้มีส่วนที่เป็นบาร์ที่มีที่นั่งสบายๆให้รอหรือไม่
3. Fri/Sat: เป็นวันศุกร์หรือวันเสาร์
4. Hungry: เราหิวหรือไม่
5. Patrons: มีคนอยู่ในร้านอาหารมากน้อยแค่ไหน (มีค่าเป็น ไม่มี มีบ้าง เต็ม)
6. Price: ราคา (มีค่าเป็น $(น้อย) $$(ปานกลาง) $$$(มาก))
7. Raining: ข้างนอกฝนตกหรือไม่

8. Reservation: เราได้จองไว้ก่อนหรือไม่
9. Type: ชนิดของร้านอาหาร (มีค่าเป็น ฝรั่งเศส อิตาเลียน ไทย หรือเบอร์เกอร์)
10. WaitEstimate: จะต้องรอนานแค่ไหน (มีค่าเป็น 0-10 นาที 10-30 นาที 30-60 นาที และ >60 นาที)

รูปร่างของต้นไม้ตัดสินใจที่อาจเป็นไปได้ แสดงในรูปที่ 7.2 สังเกตุได้ว่าไม่ได้ใช้ลักษณะประจำ Price และ Type แสดงว่าที่งสองลักษณะถือว่าไม่สำคัญ จากรูปจะเห็นได้ว่า Patrons=Full และ WaitEstimate=0-10 คำตอบจะออกมาเป็น บวก (นั่นคือจะรอ)

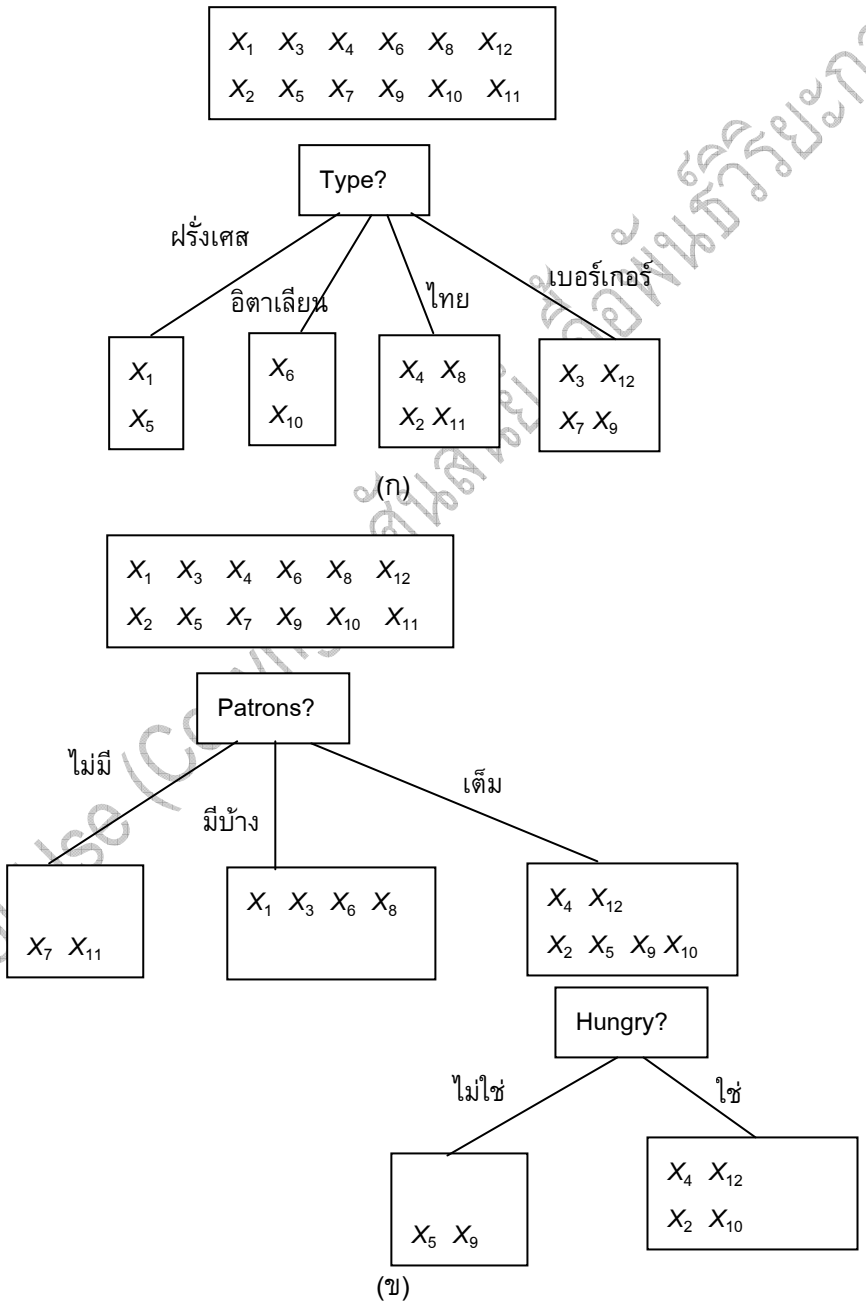
7.3.1 กระบวนการสร้างต้นไม้ตัดสินใจจากตัวอย่าง

ตัวอย่างของต้นไม้ตัดสินใจแบบบูลีนประกอบด้วยเซตของลักษณะประจำของอินพุต X และค่าเอาต์พุตที่เป็นบูลีน y สมมุติให้มีตัวอย่าง 12 ตัวอย่างดังแสดงในตารางที่ 7.1

ตารางที่ 7.1 ตัวอย่างสำหรับปัญหาการรอร้านอาหาร

ตัวอย่างที่	ลักษณะประจำ										เป้าหมาย
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
X_1	มี	ไม่มี	ไม่ใช่	ใช่	มีบ้าง	$$$	ไม่มี	มี	ฝรั่งเศส	0-10	Yes
X_2	มี	ไม่มี	ไม่ใช่	ใช่	เต็ม	$	ไม่มี	ไม่มี	ไทย	30-60	No
X_3	ไม่มี	มี	ไม่ใช่	ไม่ใช่	มีบ้าง	$	ไม่มี	ไม่มี	เบอร์เกอร์	0-10	Yes
X_4	มี	ไม่มี	ใช่	ใช่	เต็ม	$	มี	ไม่มี	ไทย	10-30	Yes
X_5	มี	ไม่มี	ใช่	ไม่ใช่	เต็ม	$$$	ไม่มี	มี	ฝรั่งเศส	>60	No
X_6	ไม่มี	มี	ไม่ใช่	ใช่	มีบ้าง	$$	มี	มี	อิตาเลียน	0-10	Yes
X_7	ไม่มี	มี	ไม่ใช่	ไม่ใช่	ไม่มี	$	มี	ไม่มี	เบอร์เกอร์	0-10	No
X_8	ไม่มี	ไม่มี	ไม่ใช่	ใช่	มีบ้าง	$$	มี	มี	ไทย	0-10	Yes
X_9	ไม่มี	มี	ใช่	ไม่ใช่	เต็ม	$	มี	ไม่มี	เบอร์เกอร์	>60	No
X_{10}	มี	มี	ใช่	ใช่	เต็ม	$$$	ไม่มี	มี	อิตาเลียน	10-30	No
X_{11}	ไม่มี	ไม่มี	ไม่ใช่	ไม่ใช่	ไม่มี	$	ไม่มี	ไม่มี	ไทย	0-10	No
X_{12}	มี	มี	ใช่	ใช่	เต็ม	$	ไม่มี	ไม่มี	เบอร์เกอร์	30-60	Yes

ตัวอย่างที่เป็นบวกเป็นตัวอย่างที่มีเป้าหมายเป็นบวกนั่นคือ (X_1, X_3, X_4, X_6, X_8, X_{12}) ส่วนตัวอย่างที่เป็นลบคือ (X_2, X_5, X_7, X_9, X_{10}, X_{11}) ชุดตัวอย่างที่อยู่ในตารางที่ 7.1 คือชุดที่ใช้ในการสอนการเรียนรู้ (training data set) ในการสร้างต้นไม้ตัดสินใจอาจจะสร้างได้โยการให้มี 1 เส้นทางสำหรับ 1 ตัวอย่าง โดยที่จะทดสอบแต่ละลักษณะประจำและให้ค่าตามค่าของแต่ละลักษณะประจำในตัวอย่างนั้นๆ แต่การสร้างต้นไม้ลักษณะจะให้คำตอบที่ถูกต้องถ้าอินพุตที่เข้ามามีลักษณะเหมือนกับชุดตัวอย่างนี้เท่านั้น แต่สำหรับอินพุตแบบอื่นอาจจะให้คำตอบที่ถูกเลย นั่นคือต้นไม้ที่ได้จะจำลักษณะของชุดตัวอย่างที่ใช้ในการสอนการเรียนรู้นั่นเอง



<u>รูปที่ 7.3 การแบ่งตัวอย่างโดยการทดสอบลักษณะประจำ (ก) ใช้ Type (ข) ใช้ Patrons</u>

ในการสร้างต้นไม้ตัดสินใจเพื่อให้ได้ต้นไม้ที่สามารถจำแนกได้ถูกด้วยการทดสอบจำนวนน้อยครั้ง นั่นคือทุกเส้นทางในต้นไม้สั้น และต้นไม้ทั้งต้นมีขนาดเล็ก ดังนั้นเริ่มด้วยการทดสอบลักษณะ

ประจำที่สำคัญที่สุดก่อน นั่นคือเราแบ่งชุดข้อมูลที่เป็นบวกและลบออกจากกัน หลังจากนั้นเลือกว่าจะทดสอบลักษณะประจำอันใดก่อน ดังเช่นรูปที่ 7.3(ก) เลือกที่จะทดสอบ Type ซึ่งจะเห็นว่าลักษณะประจำนี้จะให้ผลผลัพธ์ 4 กลุ่ม โดยที่แต่ละกลุ่มมีจำนวนของตัวอย่างที่เป็นบวกและลบเท่ากัน แสดงให้เห็นว่า Type ไม่ใช่ลักษณะประจำที่ดี ส่วนรูปที่ 7.3(ข) จะเห็นว่า Patrons เป็นลักษณะประจำที่ดี เนื่องจาก ทางเลือก ไม่มี และมีบ้าง ให้กลุ่มของตัวอย่างที่เป็นลบ หรือ บวก อย่างเดียว แต่กลุ่มที่ได้จากทางเลือกเต็มมีตัวอย่างที่เป็นทั้งบวกและลบปนกัน ซึ่งโดยปกติหลักจากลักษณะประจำอันที่หนึ่งแบ่งกลุ่มตัวอย่างออกจากกันแล้ว แต่กละกลุ่มจะเป็นปัญหาในการเรียนรู้ต้นไม้ตัดสินใจต่อไปด้วยจำนวนตัวอย่างและลักษณะประจำที่น้อยลง ในการเรียกซ้ำ (recursive) นี้มีประเด็นที่น่าสนใจอยู่ 4 ประเด็นคือ

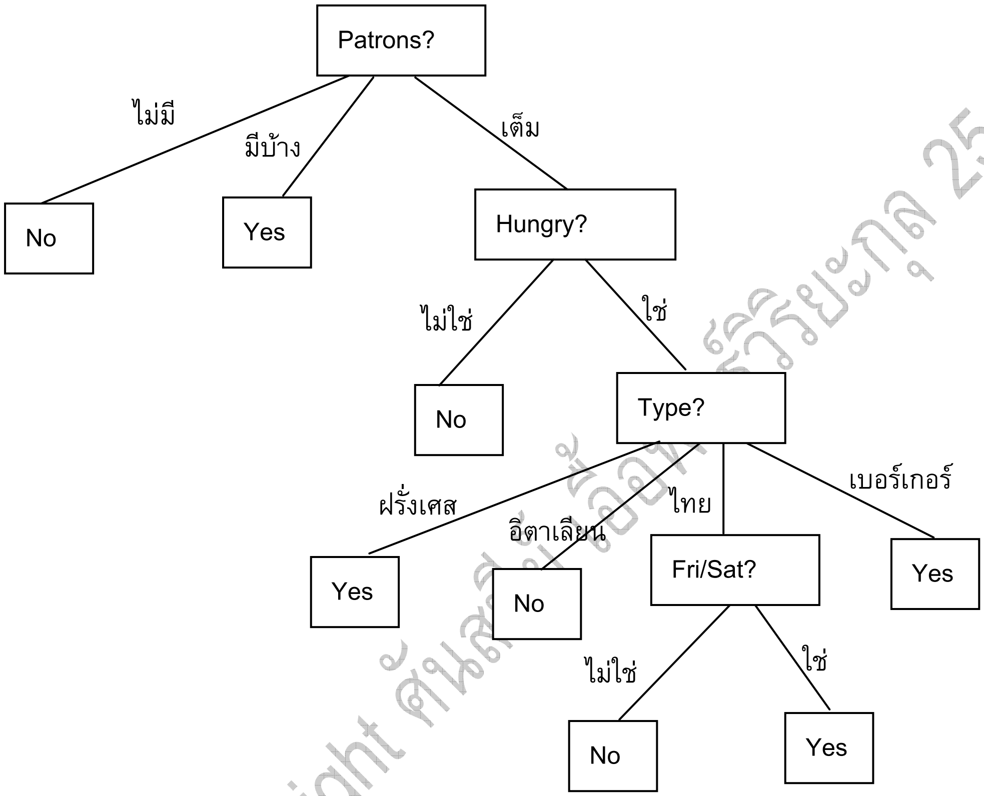
1. ถ้ามีตัวอย่างที่เป็นบวกและตัวอย่างที่เป็นลบ ให้เลือกลักษณะประจำที่ดีที่สุดเพื่อแบ่งกลุ่มอีกครั้ง
2. ถ้าตัวอย่างที่อยู่ในกลุ่มเป็นบวกทั้งหมด (ลบทั้งหมด) แสดงว่าทำการเรียนรู้เสร็จแล้ว ดังเช่นรูปที่ 7.3(ข) ในทางเลือกไม่มี และ มีบ้าง
3. ถ้าไม่มีตัวอย่างเหลือ แสดงว่าไม่มีตัวอย่างที่มีลักษณะเช่นนั้น ให้ตัดสินใจตามการจำแนกส่วนใหญ่ของพ่อแม่ของ node นั้น
4. ถ้าไม่มีลักษณะประจำเหลือ แต่ยังมีตัวอย่างทั้งบวก และ ลบ แสดงว่าตัวอย่างมีรายละเอียดเช่นเดียวกันแต่ถูกจำแนกต่างกัน ซึ่งเหตุการณ์จะเกิดขึ้นในกรณีที่ข้อมูลที่มีไม่ถูก หรือที่เรียกว่าสิ่งรบกวน (noise) ในข้อมูล และอาจจะเกิดในกรณีที่ลักษณะประจำที่มีไม่ให้ข้อมูลที่เพียงพอในการอธิบายอย่างเต็มที่ หรือโดเมนไม่กำหนด (nondeterministic) ทางแก้ทำได้โดยใช้เสียงส่วนใหญ่ (majority vote)

```
1.  If ตัวอย่างไม่มี return Default
2.  else if ทุกตัวอย่างมีการจำแนกเหมือนกัน return การจำแนก
3.  else if ลักษณะประจำไม่มี return majority-value(examples)
4.  else
        เลือกลักษณะประจำที่ดีที่สุด (best)
        ต้นไม้ตัดสินใจอันใหม่ด้วย root ทดสอบ best
        m = majority-value(examples)
        for แต่ละ Value v_i best
            examples_i ← {elements of examples with best = v_i}
            subtree ← decision-tree-learning(examples, attribs – best, m)
            เพิ่มกิ่งของต้นไม้ด้วยป้าย v_i และ subtree
5.  return tree
```

รูปที่ 7.4 อัลกอริทึมสำหรับการเรียนต้นไม้ตัดสินใจ

อัลกอริทึมสำหรับการเรียนต้นไม้ตัดสินใจแสดงในรูปที่ 7.4 ส่วนวิธีการเลือกลักษณะประจำจะอธิบายในหัวข้อต่อไป ส่วนรูปร่างต้นไม้ตัดสินใจสุดท้านที่ได้จากอัลกอริทึมนี้แสดงในรูปที่ 7.5 ซึ่งจะเห็นได้ว่าไม่เหมือนกับต้นไม้ตัดสินใจในรูปที่ 7.2 แต่ไม่ได้หมายความว่าต้นไม้ที่ได้จะผิดเนื่องจากการ

เรียนรู้นี้เป็นการเรียนรู้จากตัวอย่าง และให้คำตอบที่เหมือนกับตัวอย่างในขณะที่ต้นไม้ที่ได้มีขนาดเล็กกว่ารูปที่ 7.2 และจากอัลกอริทึมแสดงให้เห็นว่าไม่มีความจำเป็นที่ต้องใช้ Raining และ Reservation เพราะสามารถจำแนกตัวอย่างได้ถูกต้องโดยไม่ต้องใช้ลักษณะประจำนี้ และยังสามารถที่จะบอกได้ว่าเราจะรอที่ร้านอาหารไทยในวันหยุดสุดสัปดาห์



รูปที่ 7.5 ต้นไม้ตัดสินใจที่ได้จากการเรียนรู้จากตัวอย่างทั้ง 12 ตัวอย่าง

ถ้ามีตัวอย่างมากกว่านี้เราอาจจะได้ต้นไม้ตัดสินใจที่ใกล้เคียงกับต้นไม้ในรูปที่ 7.2 และต้นไม้ตัดสินใจที่ได้ในรูป 7.4 จะมีความผิดพลาดเกิดขึ้น เช่นในต้นไม้นี้ไม่เคยเห็นเคสที่มีการรอ 0-10 นาที แต่ร้านอาหารเต็ม และสำหรับเคสที่มี Hungry เป็นเท็จต้นไม้บอกว่าไม่ต้องรอ แต่ในความเป็นจริงเราจะรอ ดังนั้นจึงมีคำถามตามมาว่าอัลกอริทึมให้ต้นไม้ที่ต้องกันแต่ไม่ถูกต้องหรือไม่ และถ้าไม่ถูกต้องจะไม่ถูกต้องขนาดใด

7.3.2 การเลือกลักษณะประจำในการทดสอบ

เราต้องการเลือกลักษณะประจำที่ดีที่สามารถแบ่งตัวอย่างเป็นบวก หรือ ลบทั้งหมดได้ ซึ่ง Patrons ในหัวข้อที่แล้วเป็นลักษณะประจำที่ค่อนข้างดีถึงแม้จะไม่ดีที่สุด ดังนั้นเราจึงต้องมีตัววัดเพื่อที่เราจะสามารถเลือกลักษณะประจำได้ และตัววัดนี้ควรจะมีค่ามากที่สุดถ้าลักษณะประจำนั้นดีที่สุด แต่ควรจะมีค่าน้อยที่สุดถ้าลักษณะประจำนั้นไม่ควรถูกนำมาใช้เลย หนึ่งในตัววัดที่สามารถนำมาใช้ได้คือจำนวนที่คาดหมายของข้อมูลที่มีให้โดยลักษณะประจำนั้น ซึ่ง Shannon และ Weaver ได้ให้คำจำกัดในรูปของสมการทางคณิตศาสตร์ไว้ตั้งแต่ปี คศ. 1949 เพื่อให้เข้าใจง่ายขึ้นให้มองว่าเป็นการให้คำตอบกับคำถาม เช่นเหรียญจะออกหัวหรือไม่ จำนวนของข้อมูลที่มีอยู่ในคำตอบจะขึ้นอยู่กับความรู้เบื้องต้นของคนคนนั้น ซึ่งตัววัดในทฤษฎีข้อมูลจะมีหน่วยเป็น bits โดยที่ 1 bits ของข้อมูลเพียงพอสำหรับ

คำตอบ yes/no ของคำถามเกี่ยวกับที่ไม่รู้คำตอบ เช่นการโยนเหรียญ โดยปกติสำหรับคำตอบใดๆ v_i จะมีความน่าจะเป็นของการเกิดของคำตอบนี้เป็น $P(v_i)$ ดังนั้นเนื้อหาข้อมูล I ของคำตอบจริงเป็น

$$I\left(P\left(v_1\right),...,P\left(v_n\right)\right)=\sum_{i=1}^{n}-P\left(v_i\right)\log_2 P\left(v_i\right) \tag{7.1}$$

ดังนั้นในตัวอย่างของการโยนเหรียญเราจะได้

$$I\left(\frac{1}{2},\frac{1}{2}\right)=-\frac{1}{2}\log_2\frac{1}{2}-\frac{1}{2}\log_2\frac{1}{2}=1 \tag{7.2}$$

ถ้าในการโยนเหรียญออกหัว 99% จะได้ $I(1/100,\ 99/100)$=0.08 Bits และถ้าความน่าจะเป็นของหัวเข้าใกล้ 1 ข้อมูลของคำตอบจริงจะเป็น 0

สำหรับการเรียนรู้ต้นไม้ตัดสินใจ คำถามที่ควรจะถามคือ สำหรับชุดตัวอย่างนี้มีการจำแนกที่ถูกต้องเท่าไร ต้นไม้ตัดสินใจที่ถูกต้องจะตอบคำถามนี้ การประมาณของความน่าจะเป็นของคำตอบที่เป็นไปได้ก่อนที่ลักษณะประจำจะถูกทดสอบคือสัดส่วนของตัวอย่างที่เป็นบวก และลบในชุดตัวอย่างการสอนนั้น สมมุติให้มี p ตัวอย่างบวกและ n ตัวอย่างลบ การประมาณของข้อมูลในคำตอบที่ถูกคือ

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right)=-\frac{p}{p+n}\log_2\frac{p}{p+n}-\frac{n}{p+n}\log_2\frac{n}{p+n} \tag{7.3}$$

ซึ่งในตัวอย่างของร้านอาหารเรามี $p=n=6$ ดังนั้นเราต้องการ 1 bit ของข้อมูล

สำหรับการทดสอบลักษณะประจำ A จะไม่ให้ข้อมูลทากนัก แต่จะให้ข้อมูลอะไรเราบางอย่าง เราสามารถวัดได้อย่างแน่นอนจากการมองว่าต้องการข้อมูลเท่าใดหลังจากการทดสอบลักษณะประจำ ลักษณะประจำ A แบ่งเซตของชุดตัวอย่างการสอน E ให้เป็นเซตย่อย $E_1, E_2,..., E_v$ ตามค่าของ A ที่มีค่า v ค่า สมมุติแต่ละเซตย่อย E_i มี p_i ตัวอย่างบวกและ n_i ตัวอย่างลบ ดังนั้นถ้าเราเดินตามเส้นทางของกิ่งนั้นเราต้องการ $I(p_i/(p_i+n_i),\ n_i/(p_i+n_i))$ bit ของข้อมูลเพิ่มเติมเพื่อที่จะตอบคำถาม การสุ่มเลือกตัวอย่างจากชุดข้อมูลการสอนที่มีค่า i ของลักษณะประจำมีความน่าจะเป็นเป็น $(p_i+n_i)/(p+n)$ โดยเฉลี่ย ดังนั้นหลังจากทดสอบลัปษณะประจำ A เราต้องการ

$$\text{Re}\,mainder\left(A\right)=\sum_{i=1}^{v}\frac{p_i+n_i}{p+n}I\left(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right) \tag{7.4}$$

bit ของข้อมูลเพื่อที่จะจำแนกตัวอย่าง ข้อมูลขยาย (information gain) จากการทดสอบลัปษณะประจำคือความแตกต่างระหว่างความต้องการข้อมูลเดิมและความต้องการใหม่ดังนี้

$$Gain\left(A\right)=I\left(\frac{p}{p+n},\frac{n}{p+n}\right)-\text{Re}\,mainder\left(A\right) \tag{7.5}$$

ซึ่งในการเลือกลักษณะประจำที่ดีที่สุดจะเลือกลักษณะประจำที่มีค่าขยายที่เยอะที่สุด จากตัวอย่างร้านอาหารจะได้ว่า

$$Gain\left(Patrons\right)=1-\left[\frac{2}{12}I(0,1)+\frac{4}{12}I(1,0)+\frac{6}{12}I\left(\frac{2}{6},\frac{4}{6}\right)\right]\approx 0.541\ \text{bits} \tag{7.6}$$

$$Gain\left(Type\right)=1-\left[\frac{2}{12}I\left(\frac{1}{2},\frac{1}{2}\right)+\frac{2}{12}I\left(\frac{1}{2},\frac{1}{2}\right)+\frac{4}{12}I\left(\frac{2}{4},\frac{2}{4}\right)+\frac{4}{12}I\left(\frac{2}{4},\frac{2}{4}\right)\right]=0\ \text{bits} \tag{7.7}$$

จะเห็นได้ว่าค่าขยายของ Patrons มากกว่าค่าขยายของ Type ซึ่งแสดงว่า Patrons เป็นลักษณะประจำที่ดีกว่าดังที่แสดงในรูปที่ 7.3

เราสามารถประเมินประสิทธิภาพของสมมุติฐานได้โดยการตรวจการทำนายที่ได้กับคำตอบที่แท้จริง โดยการประเมินจะทำจากการวัดเปอร์เซนต์ของการจำแนกที่ถูกต้องนั่นเอง ซึ่งการประเมินทำนองนี้จำเป็นที่จะต้องใช้ชุดข้อมูลที่ใช้สำหรับทดสอบ (testing data set)

เราได้กล่าวถึงกรณีที่มีตัวอย่างมากกว่า 1 ตัวอย่างที่มีการอธิบายลักษณะที่เหมือนกันแต่มีการจำแนกที่ต่างกัน ซึ่งต้นไม้ตัดสินใจจะไม่สามารถหาคำตอบที่ต้องกันกับทุกตัวอย่างได้ ทางแก้ที่เคยกล่าวไว้คือให้ใช้การจำแนกส่วนใหญ่ถ้าจำเป็นต้องใช้สมมุติฐานไม่ต่อเนื่อง หรือใช้การประมาณความน่าจะเป็นของแต่ละการจำแนกโดยใช้ความถี่ที่จำเป็น แต่ในความเป็นจริงมีความเป็นไปได้ที่ต้นไม้ตัดสินใจจะให้คำตอบที่ต้องกันกับทุกตัวอย่าง เนื่องจากอัลกอริทึมใช้ลักษณะประจำที่ไม่ตรงประเด็น ตัวอย่างเช่นปัญหาการพยายามโยนลูกเต๋า สมมุติว่าทำการทดลองเป็นช่วงเวลาหยึ่งด้วยลูกเต๋าหลายแบบ และให้ลักษณะประจำที่อธิบายการทดลองเป็น

1. Day: วันที่ทำการโยนลูกเต๋า (วันจันทร์ วันอังคาร วันพุธ วันพฤหัสบดี)
2. Month: เดือนที่โยนลูกเต๋า (มกราคมหรือกุมภาพันธ์)
3. Color: สีของลูกเต๋า (แดงหรือน้ำเงิน)

ตราบใดที่ไม่มีตัวอย่างที่มีคำอธิบายที่เหมือนกันมากกว่า 1 ตัวอย่างต้นไม้ตัดสินใจจะหาสมมุติฐานที่ตรงกับการเรียนรู้ ยิ่งมีลักษณะประจำมากเท่าไร โอกาสที่จะได้สมมุติฐานที่ตรงกับการเรียนรู้ยิ่งมีมาก ซึ่งสมมุติฐานที่ได้จะเป็นสมมุติฐานที่ไม่ถูกต้อง เพราะสิ่งที่อยากได้คือค่าที่ได้จากบัฟใบ (leaf node) ควรจะเข้าใกล้ 1/6 สำหรับการโยนแต่ละครั้ง หลังจากที่ได้ตัวอย่างที่เพียงพอ

ในการหาสมมุติฐานที่ดีเราจะต้องระวังเรื่องการหาความหมายในข้อมูลที่ไม่จำเป็น ซึ่งถูกเรียกว่า overfitting ซึ่งเหตุการณ์นี้จะเกิดขึ้นเมื่อฟังก์ชันเป้าหมายไม่ได้ถูกสุ่มอย่างแท้จริง ซึ่งเหตุการณ์นี้อาจจะเกิดขึ้นกับอัลกอริทึมการเรียนรู้ทั่วไปไม่เฉพาะกับต้นไม้ตัดสินใจ

วิธีที่จะแก้ไขปัญหานี้อย่างง่ายคือการทำ decision tree pruning ซึ่งเป็นการทำที่ป้องกันการแยกซ้ำของลักษณะประจำที่ไม่แน่ชัดว่าตรงประเด็น ถึงแม้ว่าข้อมูลที่ node ในต้นไม้จะไม่ถูกจำแนกอย่างเป็นเอกรูป (uniform) ซึ่งเราสามารถใช้ค่าขยายในการวัดได้นั่นคือยิ่งค่าต่ำเท่าไรยิ่งไม่ตรงประเด็นเท่านั้น แต่ค่าขยายจะต้องมีค่าเป็นเท่าใดจึงจะแยกได้

เราสามารถใช้การทดสอบทางสถิติ statistical significance test ซึ่งในการทดสอบนี้ตั้งสมมุติฐานว่าไม่มีรูปแบบรองรับ (underlying pattern) หรือที่ถูกเรียกว่า null hypothesis แล้วข้อมูลจริงจะถูกวิเคราะห์เพื่อคำนวณหาความเบี่ยงเบนของการหายไปของรูปแบบ ถ้าระดับความเบี่ยงเบนไม่น่าจะมีโอกาสเกิด (โดยปกติมีค่าเฉลี่ยที่ประมาณ 5% หรือน้อยกว่า) แสดงว่าเป็นหลักฐานที่ดีว่ามีรูปแบบที่สำคัญในข้อมูล ความน่าจะเป็นถูกคำนวณจากการแจกแจงมาตรฐานของจำนวนของความเบี่ยงเบนที่เราคาดหวังว่าจะเจอในการสุ่มตัวอย่าง

ในกรณีนี้ null hypothesis คือลักษณะประจำที่ไม่ตรงประเด็น ดังนั้นค่าขยายสำหรับตัวอย่างที่มีจำนวนมากควรจะมีค่าเป็น 0 เราจำเป็นที่จะต้องหาความน่าจะเป็นของ จำนวนตัวอย่างขนาด v จะแสดงออกถึงความเบี่ยงเบนจากการแจกแจงที่คาดหวังของตัวอย่างบวกและลบภายใต้ null hypothesis เราสามารถวัดความเบี่ยงเบนนี้โดยการเปรียบเทียบจำนวนจริงของตัวอย่างบวกและลบในเซตย่อย p_i และ n_i ด้วยจำนวนคาดหวัง $\hat{p}_i$ และ $\hat{n}_i$ โดยสมมุติการไม่ตรงประเด็นอย่างแท้จริงดังนี้

$$\hat{p}_i = p \times \frac{p_i + n_i}{p + n} \quad (7.8ก)$$

และ

$$\hat{n}_i = n \times \frac{p_i + n_i}{p + n} \quad (7.8ข)$$

การวัดความเบี่ยงเบนทั้งหมดคือ

$$D = \sum_{i=1}^{v} \frac{\left(p_i - \hat{p}_i\right)^2}{\hat{p}_i} + \frac{\left(n_i - \hat{n}_i\right)^2}{\hat{n}_i} \quad (7.9)$$

ภายใต้ null hypothesis ค่า D จะต่ำที่ถูกกระจายตาม χ^2 (chi-squared) distribution ด้วยค่าระดับความอิสระ (degree of freedom) เป็น $v - 1$ สามารถใช้ตารางมาตรฐานของ χ^2 ในการหาความน่าจะเป็นของลักษณะประจำที่ไม่ตรงประเด็นได้ ซึ่งจากค่านี้สามารถนำมาสร้าง χ^2 pruning ได้ ซึ่งการสร้างต้นไม้ตัดสินใจที่ใช้ pruning จะได้ต้นไม้ที่เล็กกว่าและง่ายกว่า ในขณะที่มีประสิทธิภาพมากกว่า

นอกหนือจากนี้ยังสามารถใช้ cross-validation ในการลดปัญหา overfitting ได้ ซึ่งกระบวนการนี้สามารถใช้ได้กับอัลกอริทึมการเรียนรู้ทั่วไปไม่จำเป็นต้องเป็นต้นไม้ตัดสินใจ ซึ่งความคิดพื้นฐานของวิธีการนี้คือวัดว่าอัลกอริทึมที่ได้ทำงานดีแค่ไหนสำหรับข้อมูลที่ไม่เคยเห็นมาก่อน สามารถทำได้โดย แบ่งส่วนหนึ่งของชุดข้อมูลเพื่อเอาไว้สำหรับการทดสอบของอัลกอริทึมที่ได้จากข้อมูลที่เหลือ *K*-fold cross validation คือการที่เราทำการทดลองทั้งหมด *k* ครั้ง แต่ละครั้งแบ่งชุดข้อมูลไว้ 1/*k* สำหรับการทดสอบ และหาค่าเฉลี่ยในที่สุด

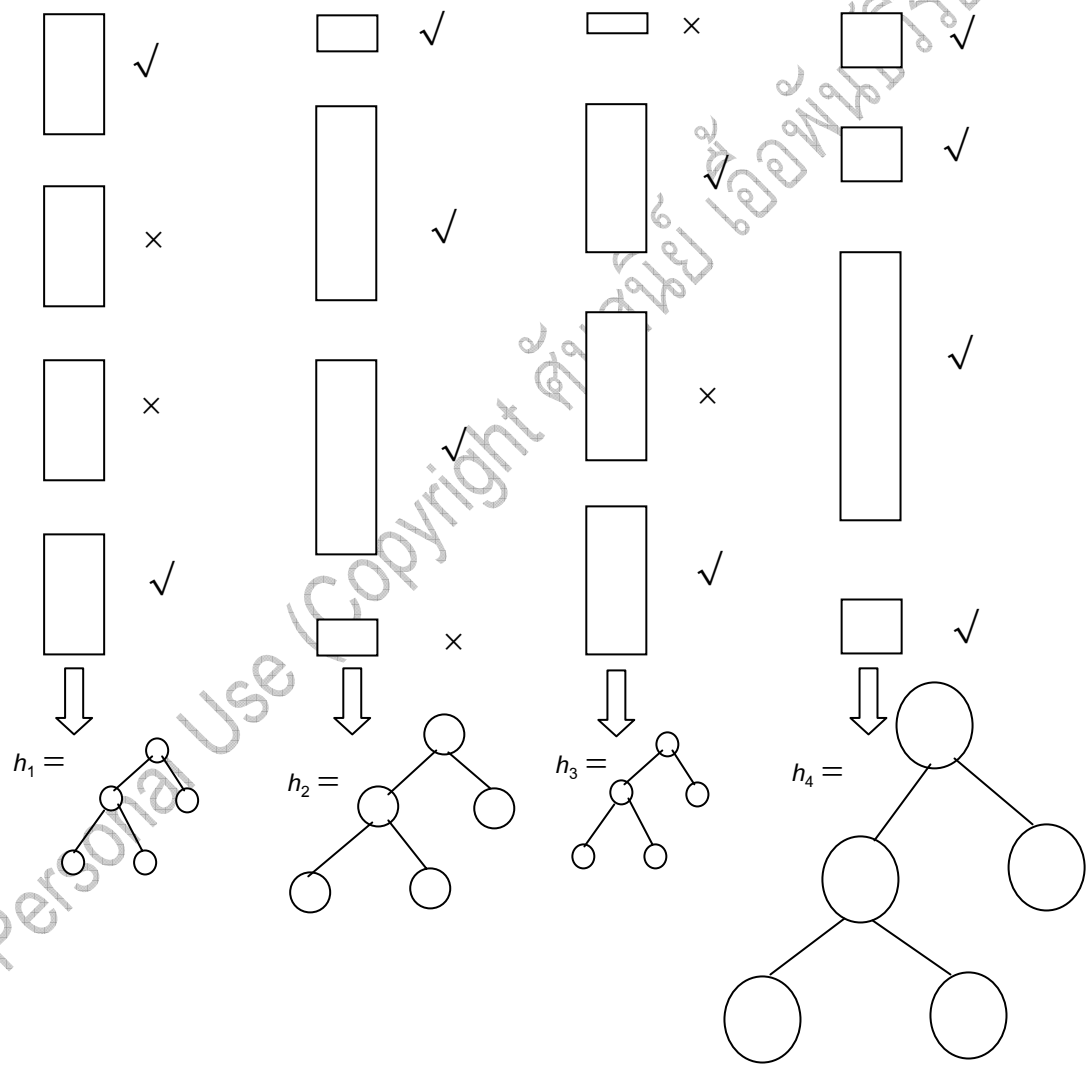
7.4 การเรียนรู้ที่นำมาเข้าชุดกัน (Ensemble learning)

ความคิดของการเรียนรู้แบบนี้คือ เลือกพวก รือ ensemble ของสมมุติฐานจากมิติสมมุติฐานและรวมเข้าด้วยกันเพื่อใช้ในการทำนาย ตัวอย่างเช่นการสร้างต้นไม้ตัดสิสใจหลายต้นจากชุดข้อมูลการสอนชุดเดียวกัน และใช้วิธีการลงคะแนนในการเลือกการจำแนกที่ดีที่สุดสำหรับตัวอย่างใหม่ที่เข้ามา

ให้มี ensemble ของ $M = 5$ สมมุติฐาน และสมมุติเรารวมการทำนายโดยใช้เสียงส่วนใหญ่ สำหรับการที่ ensemble จะจำแนกผิด อย่างน้อยต้องมี 3 จาก 5 สมมุติฐานที่จำแนกผิด ดังนั้นคาดหวังว่าเหตุการณ์นี้น่าจะเกิดน้อยกว่าการที่ 1 สมมุติฐานจะจำแนกผิด สมมุติให้แต่ละสมมุติฐาน h_i ใน ensemble มีความผิดพลาด p (ความน่าจะเป็นที่ตัวอย่างที่สุ่มมาถูกจำแนกผิดด้วยสมมุติฐาน h_i เป็น p) สมมุติให้ความผิดพลาดที่แต่ละสมมุติฐานเป็นอิสระต่อกัน ถ้า p มีค่าน้อยแล้วโอกาสที่จะมีการจำแนกผิดมากจะน้อยมาก แต่ในความเป็นจริงสมมุติฐานที่ว่าความผิดพลาดจะเป็นอิสระกันนั้นไม่ค่อยจะมีเหตุผลเนื่องจากสมมุติฐานจะถูกชักนำให้ไปในทางเดียวกันซึ่งอาจจะผิดโดยชุดข้อมูลการสอน แต่ถ้าสมมุติฐานมีความแตกต่างกันเบ็กน้อย ก็อาจจะลดสหสัมพันธ์ (correlation) ระหว่างความผิดพลาดได้ ดังนั้น ensemble learning น่าจะเป็นสิ่งที่เป็นประโยชน์

วิธีการ ensemble ที่ใช้กันมากที่สุดเรียกว่า boosting ในการทำความเข้าใจจะอธิบายความคิดของ weighted training set ก่อน ในชุดข้อมูลการสอนแต่ละตัวอย่างมีน้ำหนักมาด้วย $w_j \geq 0$ ถ้าตัวอย่างมีน้ำหนักมากแสดงว่าตัวอย่างนั้นมีความสำคัญในการเรียนรู้ของสมมุติฐานมาก วิธีการ

boosting จะเริ่มจากให้ทุกตัวอย่างมีค่าน้ำหนัก $w_j = 1$ จากนั้นสร้างสมมุติฐานแรก h_1 ซึ่งสมมุติฐานนี้จะจำแนกตัวอย่างบางตัวอย่างถูก และบางตัวอย่างจะถูกจำแนกผิด ซึ่งเราต้องการให้สมมุติฐานทำได้ดีขึ้นในตัวอย่างที่ถูกจำแนกผิด ดังนั้นจึงเพิ่มค่านำหนักให้กับตัวอย่างเหล่านั้นและลดค่าน้ำหนักของตัวอย่างที่ถูกจำแนกถูก และใช้ตัวอย่างเหล่านี้ในการสร้างสมมุติฐาน h_2 ทำกระบวนการนี้ต่อไปเรื่อยๆ จนกระทั่งสร้างสมมุติฐานได้ M สมมุติฐาน โดยที่ M คือค่าที่ตั้งไว้ตั้งแต่แรก สมมุติฐานที่เป็น ensemble สุดท้ายคือการรวมแบบ weighted-majority ของทุก M สมมุติฐาน โดยที่ค่าน้ำหนักจะบ่งบอกถึงการมำประสิทธิภาพของแต่ละสมมุติฐานในชุดข้อมูลการสอน รูปที่ 7.6 แสดงถึงการทำงานของ boosting ที่ได้กล่าวไปโดยที่กล่องสี่เหลี่ยมคือตัวอย่าง ความสูงของแต่ละกล่องคือน้ำหนักที่ติดกับตัวอย่างนั้น เครางหมายถูกและผิดคือถูกจำแนกถูก ถูกจำแนกผิดตามลำดับ ขนาดของต้นไม้ตัดสินใจบ่งบอกถึงน้ำหนักของแต่ละสมมุติฐานใน ensemble สุดท้าย



รูปที่ 7.6 การทำงานของ boosting

7.5 การเรียนรู้ทางสถิติ

ในหัวข้อก่อนหน้าแนวคิดจะเป็นข้อมูลและสมมุติฐาน แต่ในหัวข้อนี้ข้อมูลเป็นหลักฐาน (evidence) นั่นคือการสร้างกรณีตัวอย่าง (instantiation) ของตัวแปรสุ่มที่อธิบายโดเมน ส่วนสมมุติฐานเป็นทฤษฎีความน่าจะเป็นของการที่โดเมนทำงานอย่างไร รวมถึงทฤษฎีทางลอจิกซึ่งเป็นกรณีพิเศษ

ตัวอย่างต่อไปนี้ถูกใช้ในการอธิบายหัวข้อนี้ สมมุติให้มีลูกอม 2 รสคือรสเชอร์รี และรสส้ม ลูกอมทั้งสองถูกห่อด้วยกระดาษทึบที่มีสีเหมือนกัน ลูกอมจะถูกขายในถุงใหญ่โดยที่มีอยู่ 5 ประเภทคือ h_1 มีลูกอมรสเชอร์รี 100% h_2 มีลูกอมรสเชอร์รี 75% และลูกอมรสส้ม 25% h_3 มีลูกอมรสสเชอร์รี 50% และรสส้ม 50% h_4 มีลูกอมรสเชอร์รี 25% และรสส้ม 75% ส่วน h_5 มีลูกอมรสส้ม 100% ถ้าให้ถุงลูกอมถุงใหม่ ตัวแปรสุ่ม H (สำหรับสมมุติฐาน) แทนชนิดของถุงด้วยค่าที่เป็นไปได้ h_1 ถึง h_5 ในที่นี้ H ไม่ใช่การสังเกตุโดยตรงแน่นอน และราคาของลูกอมเป็นที่เปิดเผยนั่นคือ D_1 D_2 ... D_N โดยที่แต่ละ D_i คือตัวแปรสุ่มด้วยค่าที่เป็นไปได้คือลูกอมรสเชอร์รี และลูกอมรสส้ม สิ่งที่ตัวแทนต้องทำคือต้องทำนายว่าลูปอมรสต่อไปเป็นรสอะไร

Bayesian learning เป็นการคำนวณความน่าจะเป็นของแต่ละสมมุติฐานที่มีข้อมูลให้ และทำการทำนายตามนั้น นั่นคือการทำนายหาจากการใช้สมมุติฐานทุกสมมุติฐาน และให้น้ำหนักด้วยความน่าจะเป็นของสมมุติฐาน นอกเหนือจากการใช้สมมุติฐานที่ดีที่สุดอันเดียว ในกรณีนี้การเรียนรู้จะเป็นการอนุมานความน่าจะเป็น ให้ **D** แทนข้อมูลทั้งหมด ด้วยค่าที่ถูกสังเกตุ **d** แล้วความน่าจะเป็นของแต่ละสมมุติฐานสามารถหาได้จากกฎของ Bayes

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i) \tag{7.10}$$

สมมุติต้องการทำนายเกี่ยวกับปริมาณที่ไม่รู้ X จะได้

$$\mathbf{P}(X|\mathbf{d}) = \sum_i \mathbf{P}(X|\mathbf{d},h_i)\mathbf{P}(h_i|\mathbf{d}) = \sum_i \mathbf{P}(X|h_i)P(h_i|\mathbf{d}) \tag{7.11}$$

โดยที่สมมุติว่าแต่ละสมมุติฐานหาการแจกแจงความน่าจะเป็น (probability distribution) บน X สมการที่ 7.11 แสดงให้เห็นว่าการทำนายเป็นการหาน้ำหนักโดยเฉลี่ยของการทำนายแต่ละสมมุติฐาน ปริมาณที่สำคัญสำหรับวิธีการนี้คือ hypothesis prior $P(h_i)$ และ likelihood ของข้อมูลภายใต้แต่ละสมมุติฐาน $P(\mathbf{d}|h_i)$

สำหรับตัวอย่างเรื่องลูกอม สมมุติให้การแจกแจงก่อน บน $h_1,..,h_5$ คือ $\langle 0.1, 0.2, 0.4, 0.2, 0.1\rangle$ likelihood ของข้อมูลหาได้จากการคำนวณภายใต้การสังเกตุ ซึ่งเป็น independently และ identically distributed ดังนั้น

$$P(\mathbf{d}|h_i) = \prod_j P(d_j|h_i) \tag{7.12}$$

ดังนั้นถ้าต้องการรู้ว่าถุงที่หยิบมาใหม่เป็นถุงประเภทใดก็สามารถคำนวณได้จากสมการที่ 7.10 นั่นเอง

7.6 การเรียนรู้ด้วยข้อมูลที่สมบูรณ์

การเรียนรู้นี้เป็นการเรียนรู้ตัวแปรด้วยข้อมูลที่สมบูรณ์ ซึ่งการเรียนรู้นี้เป็นการหาตัวแปรสำหรับแบบจำลองความน่าจะเป็นของโครงสร้างที่กำหนดไว้ และจะกล่าวถึงแค่การเรียนรู้ตัวแปรที่ไม่ต่อเนื่อง

สมมุติว่าซื้อถุงลูกอมที่มีทั้งรสเชอร์รีและรสส้มจากร้านใหม่ โดยมี่ไม่รู้ว่าสัดส่วนของลูกอมแต่ละรสเป็นเท่าไร ระหว่าง 0 ถึง 1 ตัวแปรในกรณีนี้ถูกเรียกว่า θ ซึ่งเป็นสัดส่วนของลูกอมรสเชอร์รี และสมมุติฐานคือ h_θ (สัดส่วนของรสส้มคือ $1-\theta$) ถ้าสมมุติว่าทุกสัดส่วนเป็น equally likely a priori แล้ววิธีการ maximum likelihood เป็นวิธีการที่เหมาะสม ถ้าแบบจำลองเป็น Bayesian network ต้องการตัวแปรสุ่ม 1 ตัวแปรคือ Flavor ซึ่งมีค่าเป็นเชอร์รี และส้ม โดยที่ความน่าจะเป็นของเชอร์รีคือ θ สมมุตอแกะห่อลูกอม N เม็ด โดยที่มี c เม็ดเป็นเชอร์รี ดังนั้น $l = N - c$ เป็นส้ม จากสมการที่ 7.12 Likelihood ของข้อมูลชุดนี้เป็น

$$P\left(\mathbf{d}\middle|h_\theta\right) = \prod_{j=1}^{N} P\left(d_j\middle|h_\theta\right) = \theta^c\left(1-\theta\right)^l \tag{7.14}$$

สมมุติฐานที่เป็น maximum likelihood หาได้จากการหาค่า θ ที่ทำให้สมการที่ 7.14 มีค่ามากที่สุด ซึ่งจะเหมือนกับการหาค่าที่มากที่สุดของ log likelihood

$$L\left(\mathbf{d}\middle|h_\theta\right) = \log P\left(\mathbf{d}\middle|h_\theta\right) = \sum_{j=1}^{N} \log P\left(d_j\middle|h_\theta\right) = c\log\theta + l\log\left(1-\theta\right) \tag{7.15}$$

ในการหา maximum likelihood ของ θ ด้วยการหาอนุพันธ์ L เทียบกับ θ และตั้งค่าให้เท่ากับ 0 จะได้

$$\frac{dL\left(\mathbf{d}\middle|h_\theta\right)}{d\theta} = \frac{c}{\theta} - \frac{l}{\left(1-\theta\right)} = 0 \tag{7.16}$$

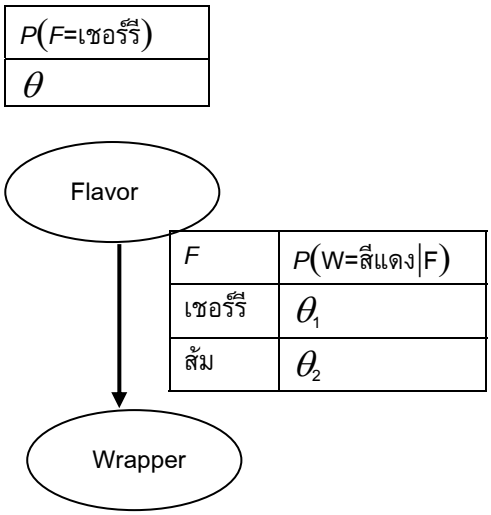
จะได้

$$\theta = \frac{c}{c+l} = \frac{c}{N} \tag{7.17}$$

จากสมการที่ 7.17 ทำให้เห็นว่า ค่าสมมุติฐานจาก maximum-likelihood h_{ML} บ่งบอกว่าสัดส่วนจริงของเชอร์รีในถุงเท่ากับสัดส่วนของเชอร์รีในถุงที่ได้จากการสังเกตุ maximum-likelihood มีข้อเสียคือ เมื่อชุดข้อมูลมีจำนวนน้อยและบางเหตุการณ์ไม่ถูกสังเกตุ เช่นไม่มีเชอร์รีในถุง สมมุติฐาน maximum likelihood จะให้ค่าความน่าจะเป็นของเหตุการณ์เหล่านั้นเป็น 0

สมมุติว่ามีร้านลูกอมร้านใหม่ต้องการให้ข้อมูลเล็กน้อยกับลูกค้าโดยใช้กระดาษห่อสีอดงและสีเขียว ตัวแปร Wrapper สำหรับแต่ละลูกอม ซึ่งถูกเลือกโดยใช้ความน่าจะเป็นโดยที่มี conditional distribution ที่ไม่รู้ที่ขึ้นกับรส แบบจำลองนี้แสดงในรูปที่ 7.7 จะเห็นว่ามีตัวแปร 3 ตัวแปรคือ θ θ_1 θ_2 จากตัวแปรเหล่านี้ likelihood ของการที่ลูกอมรสเชอร์รีในกระดาษสีเขียวหาได้จากการหา Bayesian network ปกติดังที่กล่าวมาแล้วในบทที่ 5

$$\begin{aligned} &P(\text{Flavor=เชอร์รี, Wrapper=สีเขียว}\,|\,h_{\theta,\theta_1,\theta_2}) \\ &\quad = P(\text{Flavor=เชอร์รี}\,|\,h_{\theta,\theta_1,\theta_2})P(\text{ Wrapper=สีเขียว}\,|\,\text{ Flavor=เชอร์รี, } h_{\theta,\theta_1,\theta_2}) \\ &\quad = \theta(1-\theta_1) \end{aligned} \tag{7.18}$$



รูปที่ 7.7 แบบจำลองในกรณีที่กระดาษห่อทีสีที่ขึ้นกับรสของลูกอม

สมมุติให้มีลูกอม N เม็ดและมีเชอร์รีและส้มเป็นจำนวน c และ l ตามลำดับ จำนวนกระดาษห่อเป็น r_c กระดาษสีแดงที่ห่อเชอร์รี และ g_c กระดาษสีเขียวที่ห่อเชอร์รี ในขณะที่ r_l เป็นกระดาษสีแดงที่ห่อรสส้ม และ g_l กระดาษสีเขียวที่ห่อรสส้ม ดังนั้น likelihood ของข้อมูลคือ

$$P\left(\mathbf{d}\middle|h_{\theta,\theta_1,\theta_2}\right)=\theta^c\left(1-\theta\right)^l\theta_1^{r_c}\left(1-\theta_1\right)^{g_c}\theta_2^{r_l}\left(1-\theta_2\right)^{g_l} \tag{7.19}$$

เมื่อหา log-likelihood จะได้

$$L=c\log\theta+l\log\left(1-\theta\right)+r_c\log\theta_1+g_c\log\left(1-\theta_1\right)+r_l\log\theta_2+g_l\log\left(1-\theta_2\right) \tag{7.20}$$

เมื่อหาอนุพันธ์เทียบกับตัวแปรที่ต้องการจะได้

$$\frac{\partial L}{\partial\theta}=\frac{c}{\theta}-\frac{l}{1-\theta}=0\Rightarrow\theta=\frac{c}{c+l} \tag{7.20}$$

$$\frac{\partial L}{\partial\theta_1}=\frac{r_c}{\theta_1}-\frac{g_c}{1-\theta_1}=0\Rightarrow\theta_1=\frac{r_c}{r_c+g_c} \tag{7.21}$$

$$\frac{\partial L}{\partial\theta_2}=\frac{r_l}{\theta_2}-\frac{g_l}{1-\theta_2}=0\Rightarrow\theta_2=\frac{r_l}{r_l+g_l} \tag{7.22}$$

ซึ่งคำตอบของ θ จะเหมือนกับที่เคยคำนวณได้ ส่วนคำตอบของ θ_1 ซึ่งเป็นโอกาสที่ลูกอมรสเชอร์รีถูกห่อด้วยกระดาษสีแดงเท่ากับสัดส่วนของลูกอมรสเชอร์รีที่ถูกห่อด้วยกระดาษสีแดง และในทำนองเดียวกันกับ θ_2

สิ่งที่สำคัญในที่นี้คือปัญหาการเรียนรู้ตัวแปรโดยใช้ maximum likelihood สำหรับ Bayesian network ในกรณีที่มีข้อมูลสมบูรณ์ สามารถแยกเป็นปัญหาย่อยได้ โดยที่ตัวแปร 1 ตัวสำหรับ 1 ปัญหา และอีกข้อสังแกตุหนึ่งคือ ค่าของตัวแปรเป็นความถี่ที่สังเกตุได้ของค่าตัวแปรนั้นจากค่าของพ่อแม่ที่ตั้งไว้ แต่ข้อควรระวังคือถ้าจำนวนตัวอย่างน้อยอาจจะมีค่าเป็น 0 ได้

ในกระบวนการเรียนมีการเรียนรู้อีกมากมายอาทิเช่นการเรียนโดยใช้โครงข่ายประสาทเทียม (neural networks) รวมทั้งการเรียนรู้ที่เป็นแบบจำลอง dynamics เป็นต้น

คำถามท้ายบทที่ 7

1. จงหาต้นไม้ตัดสินใจสำหรับปัญหาการตัดสินว่าจะไปข้างหน้าที่ทางแยกของถนนเมื่อสัญญาณไฟเพิ่งเปลี่ยนเป็นสีเขียว

2. จงปรับอัลกอริทึมของการสร้างต้นไม้ให้มีการ prunning โดยใช้ χ^2 prunning

3. สมมุติให้สมมุติฐาน h_5 เป็นถุงที่มีลูกอมรสส้มทั้งถุงจริง และให้ลูกอม 10 เม็ดแรกเป็นรสส้มจงหา $P(h_i|\mathbf{d})$ สำหรับ h_1 h_2 h_3 h_4 และ h_5 โดยที่ h ทั้ง 5 มีนิยามเช่นในหัวข้อ 7.5

บรรณานุกรม

[เอื้อพันธ์วิริยะกุล47] ศันสนีย์ เอื้อพันธ์วิริยะกุล “เอกสารประกอบการสอนวิชา 261494: กระบวนวิชาหัวข้อพิเศษสำหรับวิศวกรรมคอมพิวเตอร์ 1 (ทฤษฎีฟัซซีเซต)”, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเชียงใหม่, 2547

[Buckland05] M. Buckland, “ Programming Game AI by Example”, Texas, USA., Wordware Publishing Inc., 2005.

[Campbell02] M. Campbell, A. J. Jr. Hoane, and F. Hsiung Hsu, “Deep Blue”, *Artificial Intelligence*, 134, pp. 57 – 83.

[Clerc02] M. Clerc, J. Kennedy, “The Particle Swarm: Explosion, Stability, and Convergence in a Multi-dimensional Complex Space”, *IEEE Transactions on Evolutionary Computation*, Vol 6, pp 58-73, 2002.

[Dean95] T. Dean, J. Allen, and Y. Aloimonos, “ Artificial Intelligence Theory and Practice”, Menlo Park, California, USA., Addison-Wesley Publishing Company ,1995.
[Schaeffer97] J. Schaeffer, “One Jump Ahead”, Berlin, Springer, 1997.

[Dorigo96] M. Doriogo, V. Maniezzo, and A. Colorni, “The Ant System: Optimization by a Colony of Cooperating Agents”, *IEEE Trans Syst Man Cybern.*,B26, 1996, pp. 29-41.

[Engelbrecht05] A. P. Engelbrecht, “Computational Intelligence: An Introduction”, West Sussex, England, John Wiley & Sons, Ltd., 2005.

[Kennedy95] J. Kennedy and RC. Eberhart, “Particle Swarm Optimization”, *Proceedings of the IEEE International Conference on Neural Netwoorks*, Vol 4, 1995, pp. 1942-1948.

[Kennedy98] J. Kennedy, “The Behavior of Particles”, in VW. Porto, N. Saravanan, D. Waagen (eds), *Proceedings of the 7th International Conference on Evolutionary Programming*, 1998, pp. 581-589.

[Klir95] Klir, G. J. and Yuan, B., *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, New Jersey, Prentice hall, 1995.

[Klir97] Klir G. J., St.Clair, U. and Yuan, B. *Fuzzy Set Theory: Foundations and Applications*, New Jersey, Prentice hall, 1997.

[Krishnapuram03] Krishnapuram, R., "An Introduction to Fuzzy Systems", Tutorial at Faculty of Engineering, Chiang Mai University, Chiang Mai, 2003.

[Kruse95] Kruse, R., Gebhardt, J. and Klawonn, F., *Foundations of Fuzzy Systems*, England, John Wiley & Son Ltd., 1995.

[Miikkulainen06] R. Miikkulainen, B. D. Bryant, R. Cornelius, I. V. Karpov, K. O. Stanley, and C. H. Yong, "Computational Intelligence in Games", In Yen, G. and Fogel, D. B., Computational Intelligence: Principles and Practice, IEEE Computational Intelligence Society, 2006, pp. 155 – 191.

[Rabin02] S. Rabin, "AI Game Programming Wisdom", Massachusetts, USA., Charles River Media, Inc., 2002.

[Russell03] S. Russell, and P. Norvig, "Artificial Intelligence: A Modern Approach", New Jersey, USA., Pearson Education, Inc., 2003.

[Schwab04] B. Schwab, "AI Game Engine Programming", Massachusetts, USA., Charles River Media, Inc., 2004.

[Suganthan99] PN. Suganthan, "Particle Swarm Optimizer with Neighborhood Operator", *Proceedings of IEEE Congress on Evolutionary Computation*, 1999, pp. 1958-1961.

ดัชนี