การปรับปรุงความทนทานของโปรแกรมหุ่นยนต์โดยใช้การรังควาน และการสุ่มในวิธีกำหนดการเชิงพันธุกรรม

นาย ธนัท สุขกาญจนนท์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2543
ISBN 974-346-804-8

IMPROVING ROBUSTNESS OF ROBOT PROGRAMS BY USING PERTURBATION AND RANDOMNESS IN THE GENETIC PROGRAMMING

MR. THANUT SUKKARNJANONTH

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2000

ISBN 974-346-804-8

หัวข้อวิทยานิพนธ์	การปรับปรุงความทนทานของโปรแกรมหุ่นยนต์โดยใช้การรังควาน และการสุ่มในวิธีกำหนดการเชิงพันธุกรรม
โดย	นาย ธนัท สุขกาญจนนท์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…… คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(อาจารย์ ดร.สืบสกุล พิภพมงคล)

…… อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

…… กรรมการ
(อาจารย์ ดร.บุญเสริม กิจศิริกุล)

…… กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล)

ธนัท สุขกาญจนนท์ : การปรับปรุงความทนทานของโปรแกรมหุ่นยนต์โดยใช้การรังควาน และการสุ่มในวิธีกำหนดการเชิงพันธุกรรม (IMPROVING ROBUSTNESS OF ROBOT PROGRAMS BY PERTURBATION AND RANDOMNESS IN THE GENETIC PROGRAMMING) อ. ที่ปรึกษา : ผศ. ดร.ประภาส จงสถิตย์วัฒนา, 59 หน้า. ISBN 974-346-804-8.

วิทยานิพนธ์ฉบับนี้นำเสนอการเพิ่มความทนทานให้กับโปรแกรมหุ่นยนต์ที่ได้จากกำหนดการเชิงพันธุกรรม โดยวิธีการรังควานร่วมกับการสุ่ม การรังควานเป็นการปรับปรุงกระบวนการวิวัฒนาการให้เรียนรู้จากสภาพแวดล้อมที่แตกต่างกันหลาย ๆ แบบ ขณะที่การสุ่มเป็นการเพิ่มฟังก์ชันพิเศษที่มีลำดับการทำงานไม่แน่นอนขึ้นอยู่กับค่าที่สุ่มได้ ปัญหาที่ใช้ในงานวิจัยเป็นปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย ซึ่งจำลองสภาพแวดล้อมบนคอมพิวเตอร์ และกำหนดให้ความทนทาน หมายถึง การที่โปรแกรมสามารถควบคุมหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายได้ในสภาพแวดล้อมที่ไม่ได้เรียนรู้มา

ผลทดสอบความทนทานกับสภาพแวดล้อมที่ไม่ได้เรียนรู้มา 10,000 สภาพแวดล้อมพบว่า โปรแกรมหุ่นยนต์ที่ได้จากวิธีที่เสนอนี้มีความทนทานสูงถึง 90% โดยสาเหตุที่ทำให้ความทนทานเพิ่มขึ้นมาจากโปรแกรมที่ได้จากวิธีนี้ สามารถนำประสบการณ์กลับมาใช้ได้ดีที่สุด

ภาควิชา ....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต .....

สาขาวิชา ...วิทยาศาสตร์คอมพิวเตอร์..... ลายมือชื่ออาจารย์ที่ปรึกษา .....

ปีการศึกษา …2543……

4170332021 : MAJOR COMPUTER SCIENCE

KEY WORD: GENETIC PROGRAMMING / PERTURBATION / RANDOMNESS / ROBOT PROGRAMS / ROBUSTNESS.

THANUT SUKKARNJANONTH : IMPROVING ROBUSTNESS OF ROBOT PROGRAMS BY USING PERTURBATION AND RANDOMNESS IN THE GENETIC PROGRAMMING. THESIS ADVISOR: PRABHAS CHONGSTITVATANA, Ph.D. 59 pp. ISBN 974-346-804-8

This thesis proposed a method to improve the robustness of robot programs generated by genetic programming. The method combined two approaches: perturbation and randomness. Perturbation is used in the evolutionary process by learning in different multiple environments. Randomness is used in a special function which sequences the operations depended on random values. The problem that is studied in this work is the robot navigator problem in the simulated environment. Robustness is defined as an ability to reach the target in an unknown environment.

The robustness testing with 10,000 unknown environments shows that the robot programs generated by the proposed method have robustness as high as 90%. The cause of robustness improvement is the increase in the reuse of experience in the program that used perturbation and randomness.

Department …Computer Engineering….. Student's signature .....

Field of study …Computer Science……. Advisor's signature …….....

Academic year …2000……

กิตติกรรมประกาศ

ขอขอบคุณ ผู้ช่วยศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ให้คำแนะนำ ข้อคิดเห็นต่าง ๆ ตลอดการทำวิทยานิพนธ์ฉบับนี้ จนสำเร็จได้ด้วยดี และขอขอบคุณ อาจารย์ ดร.สืบสกุล พิภพมงคล อาจารย์ ดร.บุญเสริม กิจศิริกุล และ ผู้ช่วยศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล คณะกรรมการวิทยานิพนธ์ที่กรุณาเสียสละเวลาให้คำแนะนำ ตรวจทาน และแก้ไขวิทยานิพนธ์

ขอบคุณ พี่ ๆ เพื่อน ๆ น้อง ๆ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ รวมทั้งที่อื่น ๆ ที่ได้ให้คำปรึกษา ความช่วยเหลือกำลังใจอย่างดี ตลอดเวลาที่ศึกษาอยู่

ขอบคุณพิเศษ นาย วัชราชัย ศิริวัฒนากร ที่ช่วยขัดเกลาสำนวนภาษาอังกฤษในบทความวิชาการ นาย คฑา ประดิษฐ์วงศ์ ที่ช่วยตรวจทานวิทยานิพนธ์ นาย สันต์ทศน์ สุริยันต์ ที่ให้ติดรถกลับบ้านอยู่เสมอ และนางสาว ทัศนวรรณ ศูนย์กลาง ที่เอื้อเฟื้อโมเด็มให้ใช้งาน

สุดท้ายนี้ขอขอบคุณครอบครัว สุขกาญจนนท์ ญาติพี่น้องทุกคนที่ให้การดูแล สนับสนุน และกำลังใจที่ดีตลอดการทำวิทยานิพนธ์ฉบับนี้

ธนัท สุขกาญจนนท์

ตุลาคม 2543

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

รูปที่ หน้า

สารบัญภาพ (ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัญญาประดิษฐ์ (Artificial Intelligence) เป็นศาสตร์ที่มุ่งหวังให้เครื่องคอมพิวเตอร์มีความฉลาดเทียบเคียงมนุษย์ สามารถเรียนรู้ และตัดสินใจทำงานได้ด้วยตัวเอง โดยไม่ต้องอาศัยมนุษย์ช่วยในการตัดสินใจ ปัจจุบันปัญญาประดิษฐ์ได้รับการค้นคว้า วิจัยอย่างกว้างขวาง และนำไปใช้แก้ปัญหาด้านต่างๆ อาทิ งานทางด้านรู้จำตัวอักษร งานทางด้านรู้จำรูปภาพ งานสร้างโปรแกรมโดยอัตโนมัติ เป็นต้น

กำหนดการเชิงพันธุกรรม (Genetic Programming) เป็นวิธีทางการคำนวณเชิงวิวัฒน์ (Evolutionary Computation) ซึ่งเป็นแขนงหนึ่งในปัญญาประดิษฐ์ที่มีการทำงานเลียนแบบกระบวนการวิวัฒนาการทางธรรมชาติ เพื่อค้นหาคำตอบที่เป็นโปรแกรมคอมพิวเตอร์ วิธีกำหนดการเชิงพันธุกรรม ทำให้โปรแกรมคอมพิวเตอร์เรียนรู้ และวิวัฒนาการตนเอง จนกระทั่งพบโปรแกรมคำตอบที่สามารถแก้ปัญหาได้ เปรียบเสมือนกับสิ่งมีชีวิตที่จะต้องเรียนรู้ และวิวัฒนาการตนเองให้เหมาะกับสภาพแวดล้อมทางธรรมชาติเพื่ออยู่รอด จากแนวคิดการวิวัฒนาการนี้ทำให้กำหนดการเชิงพันธุกรรมเป็นวิธีที่แตกต่างจากวิธีอื่นทางปัญญาประดิษฐ์ และมีข้อได้เปรียบหลายอย่าง ซึ่งส่งผลให้กำหนดการเชิงพันธุกรรมเป็นที่นิยมในการแก้ปัญหา

ปัญหาควบคุมหุ่นยนต์เป็นปัญหาหนึ่งที่มีการนำกำหนดการเชิงพันธุกรรมเข้าไปใช้แก้ปัญหาอย่างกว้างขวาง จากการศึกษาพบว่าโปรแกรมคำตอบที่ได้จากกำหนดการเชิงพันธุกรรมนั้น เมื่อนำไปใช้แก้ปัญหาจริงแล้ว ผลปรากฏว่ามีบางคำตอบที่ไม่สามารถแก้ปัญหาได้ เนื่องจากสภาพแวดล้อมทดสอบมีความแตกต่างจากสภาพแวดล้อมเดิมที่เรียนรู้มา ซึ่งแสดงให้เห็นว่าคำตอบไม่สามารถปรับตัวให้เข้ากับสภาพแวดล้อมที่เปลี่ยนแปลงไป หรือคำตอบนั้นไม่มีความทนทาน (Robustness)

ต่อมาได้มีการศึกษา ค้นคว้า เทคนิคเพิ่มความทนทานให้กับคำตอบหลายวิธี เทคนิคแรกที่น่าสนใจ คือ การปรับปรุงกระบวนการวิวัฒนาการ ซึ่งเสนอโดย รุ่งโรจน์ นพสุวรรณชัย (2541) เทคนิคนี้เป็นการปรับปรุงกระบวนการวิวัฒนาการให้เรียนรู้จากสภาพแวดล้อมที่แตกต่างกันหลายๆแบบ และอีกเทคนิคหนึ่งที่น่าสนใจเป็นการเพิ่มฟังก์ชันพิเศษที่การปฏิบัติงานขึ้นกับการสุ่ม มีลักษณะทั่วไปใช้ได้กับทุกปัญหาเข้าไปในชุดฟังก์ชัน ซึ่งเป็นส่วนประกอบของโปรแกรม เสนอโดย มาเรีย ประทีปทองคำ (2541) จะเห็นได้ว่าสองเทคนิคนี้เป็นการใส่ความไม่แน่นอนเข้าไปในระบบการทำงานของกำหนดการเชิงพันธุกรรมเหมือนกัน แต่ลักษณะคำตอบที่ได้มีความแตกต่างกัน

วิทยานิพนธ์ฉบับนี้มีความมุ่งหมายที่จะเสนอวิธีเพิ่มความทนทานให้กับโปรแกรมคำตอบที่ได้จากกำหนดการเชิงพันธุกรรม โดยการปรับปรุงกระบวนการวิวัฒนาการ และกลุ่มฟังก์ชันที่ใช้ในกำหนดการเชิงพันธุกรรม เพื่อให้ได้โปรแกรมคำตอบที่มีความทนทาน เมื่อนำไปทดสอบกับสภาพแวดล้อมที่ไม่ได้เรียนรู้มา รวมทั้งวิเคราะห์สาเหตุที่ทำให้โปรแกรมคำตอบมีความทนทาน

1.2 วัตถุประสงค์ของงานวิจัย

เพื่อเสนอแนวทางในการเพิ่มความทนทานของคำตอบในวิธีกำหนดการเชิงพันธุกรรม โดยการปรับปรุงกระบวนการวิวัฒนาการ และฟังก์ชันของวิธีกำหนดการเชิงพันธุกรรม ในปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย

1.3 ขอบเขตของงานวิจัย

1. งานวิจัยนี้ใช้ปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางเพื่อไปหาเป้าหมายในระนาบ 2 มิติ เป็นปัญหาในการทดลอง
2. การทดลองเป็นการจำลองการทำงาน และสภาพแวดล้อมบนเครื่องคอมพิวเตอร์ โดยไม่ทำการทดลองกับสภาพแวดล้อมจริง

1.4 ขั้นตอน และวิธีดำเนินงาน

1. ศึกษาทฤษฎีของขั้นตอนวิธีเชิงพันธุกรรม และกำหนดการเชิงพันธุกรรม
2. ศึกษาวิธีการปรับปรุงฟังก์ชัน และกระบวนการวิวัฒนาการในกำหนดการเชิงพันธุกรรม
3. ศึกษาปัญหาที่เกิดขึ้น เมื่อนำโปรแกรมคำตอบที่ได้จากกำหนดการเชิงพันธุกรรมไปใช้กับสภาพแวดล้อมจริง
4. ออกแบบวิธีการทดลอง เพื่อเพิ่มความทนทานให้กับคำตอบที่ได้จากกำหนดการเชิงพันธุกรรม
5. ทำการทดลอง และเก็บผลการทดลอง
6. วิเคราะห์ผลการทดลอง
7. สรุปผลการทดลอง และจัดทำวิทยานิพนธ์

1.5 ประโยชน์ที่ได้รับจากงานวิจัย

ได้แนวทางในการเพิ่มความทนทานให้กับคำตอบที่ได้จากวิธีกำหนดการเชิงพันธุกรรม ในปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย และสาเหตุที่ทำให้คำตอบที่ได้มีความทนทาน ซึ่งอาจนำไปประยุกต์ใช้กับปัญหาอื่นๆได้

1.6 ผลงานที่ตีพิมพ์จากงานวิจัย

บทความ “Using Non-determinism to Improve the Robustness of Robot Programs Generated by Genetic Programming” โดย Thanut Sukkarnjanonth และ Prabhas Chongstitvatana ตีพิมพ์ และนำเสนอในงานประชุมวิชาการวิทยาการและวิศวกรรมคอมพิวเตอร์แห่งชาติ (NCSEC’2000) ระหว่างวันที่ 16-17 พฤศจิกายน 2543 ณ ศูนย์ประชุมแห่งชาติสิริกิติ์ กรุงเทพมหานคร

1.7 เนื้อหา และรูปแบบการนำเสนอวิทยานิพนธ์

บทที่ 2 จะเป็นการอธิบายรายละเอียดการทำงานของกำหนดการเชิงพันธุกรรม ส่วนในบทที่ 3 จะกล่าวถึงงานวิจัยต่างๆที่ผ่านมา ซึ่งได้เสนอวิธีเพิ่มความทนทานให้กับคำตอบ บทที่ 4 จะอธิบายถึงปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย และการนำกำหนดการเชิงพันธุกรรมไปใช้แก้ปัญหา ในบทที่ 5 จะอธิบายวิธีการทดลอง เพื่อเพิ่มความทนทานให้กับคำตอบ และผลลัพธ์ที่ได้จากการทดลอง ส่วนการวิเคราะห์หาสาเหตุที่ทำให้คำตอบมีความทนทานเพิ่มขึ้นจะอธิบายในบทที่ 6 และบทที่ 7 เป็นการสรุปเนื้อหาของการวิจัยนี้ และข้อเสนอแนะ

บทที่ 2

กำหนดการเชิงพันธุกรรม

บทนี้กล่าวถึงกำหนดการเชิงพันธุกรรม ซึ่งเป็นเทคนิควิธีทางปัญญาประดิษฐ์ที่ใช้แก้ปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายในงานวิจัย โดยจะอธิบายถึงความเป็นมา แนวคิดที่ใช้หาคำตอบ ขั้นตอนการทำงาน ของกำหนดการเชิงพันธุกรรม

2.1 กำหนดการเชิงพันธุกรรม

กำหนดการเชิงพันธุกรรม (Genetic Programming) สร้างโดย Koza (1992) เป็นวิธีค้นหา เพื่อสร้างโปรแกรมคอมพิวเตอร์ไปใช้แก้ปัญหา วิธีนี้ได้รับการพัฒนาต่อมาจากขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithm) ซึ่งคิดค้นโดย Holland (1975) ทั้งสองวิธีเป็นเทคนิคทางการคำนวณเชิงวิวัฒน์ (Evolutionary Computation) ที่นำเอากระบวนการวิวัฒนาการทางธรรมชาติมาใช้ในการค้นหาคำตอบ

กระบวนการวิวัฒนาการเป็นหลักความจริงตามธรรมชาติ ที่สิ่งมีชีวิตจะต้องปรับตัวให้เข้ากับสิ่งแวดล้อมเพื่ออยู่รอด สิ่งมีชีวิตที่ไม่สามารถปรับตัวเข้ากับสิ่งแวดล้อมได้ก็จะสูญพันธุ์ไป กำหนดการเชิงพันธุกรรมนำแนวคิดนี้มาสร้างโปรแกรมคอมพิวเตอร์ โดยโปรแกรมที่แก้ปัญหาได้ดี ใกล้ที่จะแก้ปัญหาได้สำเร็จจะอยู่รอด เพื่อเป็นต้นแบบในการสร้างโปรแกรมใหม่ ส่วนโปรแกรมที่แก้ปัญหาได้ไม่ดีก็จะถูกละทิ้ง โปรแกรมเหล่านี้จะถูกพัฒนาเป็นรุ่นๆ จนกระทั่งพบโปรแกรมคำตอบที่แก้ปัญหาได้สำเร็จ หรือพบเงื่อนไขที่ตั้งไว้

ข้อแตกต่างระหว่างกำหนดการเชิงพันธุกรรม และวิธีอื่นๆทางปัญญาประดิษฐ์นั้น อยู่ที่กำหนดการเชิงพันธุกรรมมีการทำงานแบบความน่าจะเป็น (Probability) และไม่ต้องมีการสร้างความรู้ (Knowledge) พื้นฐานไว้ใช้แก้ปัญหา ส่วนข้อแตกต่างของกำหนดการเชิงพันธุกรรม กับขั้นตอนเชิงพันธุกรรมนั้นอยู่ที่ลักษณะของคำตอบ โดยกำหนดการเชิงพันธุกรรมมีคำตอบเป็นโปรแกรม ส่วนขั้นตอนวิธีเชิงพันธุกรรมมีคำตอบเป็นสายอักขระ (String) ขนาดคงที่

ขั้นตอนหลักในการทำงานของกำหนดการเชิงพันธุกรรมนี้แบ่งออกเป็น การสร้างประชากรของผลเฉลยเริ่มต้น การวัดค่าความเหมาะของผลเฉลย การสร้างกลุ่มประชากรของผลเฉลยใหม่ และการหาคำตอบ

2.1.1 การสร้างประชากรผลเฉลยเริ่มต้น

ในขั้นตอนนี้จะเป็นการสร้างกลุ่มประชากร (Population) ซึ่งประกอบไปด้วยผลเฉลย(Individual) จำนวนหนึ่ง แต่ละผลเฉลยก็คือโปรแกรมคอมพิวเตอร์ที่ใช้ในการแก้ปัญหา โปรแกรมคอมพิวเตอร์เหล่านี้มีโครงสร้างเป็นต้นไม้ (Tree) ภายในต้นไม้ประกอบไปด้วยโหนด (Node) หลายๆอันซึ่งอาจเป็นฟังก์ชัน (Function) หรือเทอร์มินอล (Terminal) ก็ได้

ฟังก์ชัน เป็นสัญลักษณ์ที่ใช้แทนความหมายของตัวดำเนินการ ตัวเชื่อมต่อ เพื่อให้โปรแกรมมีลำดับขั้นในการทำงาน ยกตัวอย่างฟังก์ชัน ได้แก่

- ตัวดำเนินการทางคณิตศาสตร์ เช่น {+, -, *, /}
- ตัวดำเนินการตรรกะ เช่น {AND, OR, NOT}
- ตัวดำเนินการเงื่อนไข เช่น {If-Then-Else}
- ตัวดำเนินการวนซ้ำ เช่น {Do-Until}

หรืออาจนิยามฟังก์ชันขึ้นเองเพื่อใช้แก้ปัญหา เช่น {If-Bumped, Radar-Target}

เทอร์มินอล เป็นสัญลักษณ์ที่ใช้แทนความหมายของการทำงาน ค่าคงที่ หรือตัวแปรซึ่งเป็นอิสระต่อกัน และแบ่งย่อยไม่ได้อีก ได้แก่

- ค่าคงที่ เช่น 500 1,000
- ตัวแปร เช่น x y z
- การทำงาน เช่น เดินไปข้างหน้า (Forward) เลี้ยวซ้าย (Turnleft)

ทั้งฟังก์ชันและเทอร์มินอลจะต้องมีการคืนค่า (Return Value) เพื่อให้โปรแกรมที่สร้างจากฟังก์ชัน และเทอร์มินอลสามารถเชื่อมต่อกันและทำงานเป็นขั้นตอน รวมทั้งมีความเข้ากันได้ (Closure Property) กล่าวคือ การสร้างกลุ่มประชากรนั้นเป็นการสุ่ม (Random) เลือกเอาฟังก์ชันและเทอร์มินอลมาสร้างเป็นต้นไม้ ดังนั้นไม่ว่าจะเลือกฟังก์ชันและเทอร์มินอลใดมาต่อกัน จะต้องแปลความหมายของขั้นตอนทำงานได้ ด้วยเหตุนี้ฟังก์ชันทุกตัวสามารถใช้งานร่วมกับเทอร์มินอลทุกตัวได้

ฟังก์ชันและเทอร์มินอลนั้นมีความแตกต่างกันที่ ฟังก์ชันสามารถรับค่าส่งคืนได้ ในการทำงานจึงต้องรอรับค่าส่งคืนเพื่อทำงานต่อ ขณะที่เทอร์มินอลไม่รับค่าส่งคืน หรืออีกนัยหนึ่งการรับค่าที่ส่งคืนนั้นจะต้องมีอาร์กิวเมนต์ หรือตัวแปรมารับค่าส่งคืน ดังนั้นเทอร์มินอลก็คือฟังก์ชันที่ไม่มีอาร์กิวเมนต์

ต่อไปเป็นตัวอย่างของโปรแกรมคอมพิวเตอร์ทางคณิตศาสตร์ ซึ่งสร้างจากเซตของฟังก์ชัน

F = {+, -, *, /}

เมื่อตัวดำเนินการ + - * และ / รับอาร์กิวเมนต์ได้อย่างละสองตัว

และเซตของเทอร์มินอล

T = {a, b, c}

เมื่อ a b และ c เป็นตัวแปรแบบตัวเลข ทำหน้าที่เป็นอาร์กิวเมนต์ให้กับฟังก์ชัน

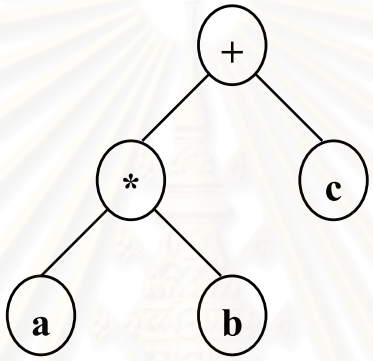
ในการเขียนสัญลักษณ์แทนผลเฉลยซึ่งเป็นโปรแกรมคอมพิวเตอร์นั้น มักนิยมเขียนอยู่ในรูปแบบของนิพจน์เชิงสัญลักษณ์ (Symbolic Expression) ตัวอย่างของผลเฉลย เช่น

(a * b) + c

เขียนในรูปของนิพจน์เชิงสัญลักษณ์ เป็น

(+ (* a b) c)

สามารถเขียนนิพจน์เชิงสัญลักษณ์ให้อยู่ในรูปของต้นไม้ ดังรูป 2.1



*รูปที่ 2.1 ต้นไม้ที่แทนโปรแกรมคอมพิวเตอร์ (+ (* a b) c)*

จากรูปที่ 2.1 จะเห็นได้ว่าต้นไม้ประกอบไปด้วยโหนดหลายๆอันเชื่อมต่อกันตามโครงสร้างของฟังก์ชัน และเทอร์มินอลแต่ละตัว ฟังก์ชันและเทอร์มินอลเหล่านี้จะถูกแทนที่ด้วยโหนดหนึ่งอัน โดยอาร์กิวเมนต์ของฟังก์ชันก็คือโหนดลูกที่อยู่ถัดลงมา ผลที่ได้จากขั้นตอนนี้ คือ ผลเฉลยในรูปแบบโปรแกรมคอมพิวเตอร์ตามจำนวนที่กำหนดไว้ ซึ่งเมื่อพิจารณาจากการสุ่มสร้างต้นไม้ ต้นไม้แต่ละต้นจะมีขนาด ความสูง ฟังก์ชัน และเทอร์มินอลที่ไม่เหมือนกัน รวมทั้งความเหมาะสมในการแก้ไขปัญหาที่แตกต่างกัน

2.1.2 การวัดค่าความเหมาะของผลเฉลย

หลังจากได้กลุ่มประชากรผลเฉลยเริ่มต้นแล้ว ขั้นต่อไปจะเป็นการนำผลเฉลยแต่ละตัวมาประมวลผล และวัดค่าความเหมาะ (Fitness Value) ในการแก้ปัญหา ขั้นตอนนี้นับเป็นขั้นตอนสำคัญในกำหนดการเชิงพันธุกรรม เนื่องจากค่าความเหมาะนี้เป็นสิ่งที่ผลักดันให้เกิดกระบวนการวิวัฒนาการเพื่อค้นหาคำตอบ

ในกระบวนการวิวัฒนาการทางธรรมชาติสิ่งมีชีวิตที่มีความเหมาะสม หรือสามารถปรับตัวกับสภาพแวดล้อมได้ดีจะดำรงอยู่ ขณะที่สิ่งมีชีวิตที่มีความเหมาะสมน้อย หรือปรับตัวเข้ากับสภาพแวดล้อมไม่ดีจะสูญหายไป กำหนดการเชิงพันธุกรรมเลียนแบบหลักการนี้ โดยใช้ฟังก์ชันความเหมาะ (Fitness function) เป็นตัววัดค่าความเหมาะของแต่ละผลเฉลย ค่าที่ได้จะเป็นเกณฑ์ในการตัดสินว่าผลเฉลยใดมีประสิทธิภาพในการแก้ปัญหาสมควรอยู่รอด และได้รับการพัฒนาเป็นรุ่นต่อไป

การกำหนดฟังก์ชันความเหมาะเป็นสิ่งที่ยาก และสำคัญต้องพิจารณาให้ดี เพื่อให้วัดค่าความเหมาะของผลเฉลยแต่ละตัวได้อย่างมีประสิทธิภาพ ยกตัวอย่างเช่น ในปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย สิ่งสำคัญของปัญหานี้ คือ ระยะทางระหว่างหุ่นยนต์กับเป้าหมาย ดังนั้นฟังก์ชันความเหมาะของปัญหานี้ควรจะใช้ระยะห่างเป็นตัววัดผลเฉลย ผลเฉลยที่ดีจะมีระยะห่างน้อย ผลเฉลยที่ไม่ดีจะมีระยะห่างมาก ดังนั้นสามารถกำหนดฟังก์ชันความเหมาะอย่างง่ายๆ ได้เป็น

Fitness function = ระยะขจัดระหว่างหุ่นยนต์กับเป้าหมาย

ฟังก์ชันความเหมาะที่ดีนั้นมีผลต่อกระบวนการวิวัฒนาการอย่างมาก เพราะค่าความเหมาะที่ได้จากฟังก์ชันนี้เป็นข้อมูลช่วยเหลือเพียงตัวเดียวที่เข้าสู่ระบบ มีผลโดยตรงต่อกระบวนการวิวัฒนาการ เช่น ทำให้ได้โปแกรมคำตอบอย่างรวดเร็ว โดยไม่ต้องเสียเวลาในการประมวลผลนาน และได้โปรแกรมคำตอบที่มีความทนทาน

หลังจากประมวลผลผลเฉลยแต่ละตัว และใช้ฟังก์ชันความเหมาะวัดค่าความเหมาะหรือประสิทธิภาพในการแก้ปัญหาของผลเฉลยแต่ละตัว ซึ่งต่อไปจะเป็นเกณฑ์ในการสร้างกลุ่มประชากรผลเฉลยใหม่

2.1.3 **การสร้างกลุ่มประชากรของผลเฉลยใหม่**

จากขั้นตอนที่แล้วผลเฉลยแต่ละตัวจะได้ค่าความเหมาะ ซึ่งจะนำมาใช้ในขั้นตอนการสร้างกลุ่มประชากรผลเฉลยใหม่นี้ ค่าความเหมาะที่ได้จะเป็นเกณฑ์ในการคัดเลือกผลเฉลย ซึ่งมีอยู่ด้วยกันหลายวิธี เช่น เรียงลำดับผลเฉลยตามค่าความเหมาะ แล้วคัดเลือกผลเฉลยที่เรียงลำดับเอาไว้ตามจำนวนที่ต้องการ เป็นต้น ผลเฉลยที่ผ่านการคัดเลือกจะได้รับการวิวัฒนาการต่อไป ส่วนผลเฉลยที่ไม่ถูกคัดเลือกก็จะทิ้งไป จากนั้นผลเฉลยที่ถูกคัดเลือกแล้วจะนำมาเป็นต้นแบบในการสร้างกลุ่มประชากรรุ่นใหม่เพื่อให้ได้ผลเฉลยที่ดีขึ้น การสร้างผลเฉลยรุ่นใหม่นี้ทำได้โดยอาศัยวิธีทางพันธุกรรม (Genetic operation) ซึ่งมีวิธีพื้นฐานอยู่ 3 วิธี ได้แก่ การสืบพันธุ์ การไขว้เปลี่ยน และการกลาย

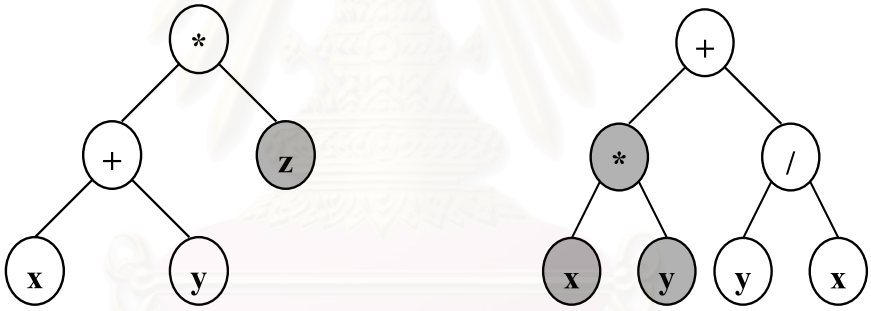
ในการอธิบายวิธีทางพันธุกรรมนี้จะใช้คำว่า ต้นไม้ แทน ผลเฉลย เพื่อสะดวกในการอธิบายทำความเข้าใจ

2.1.3.1 **การสืบพันธุ์** (Reproduction)

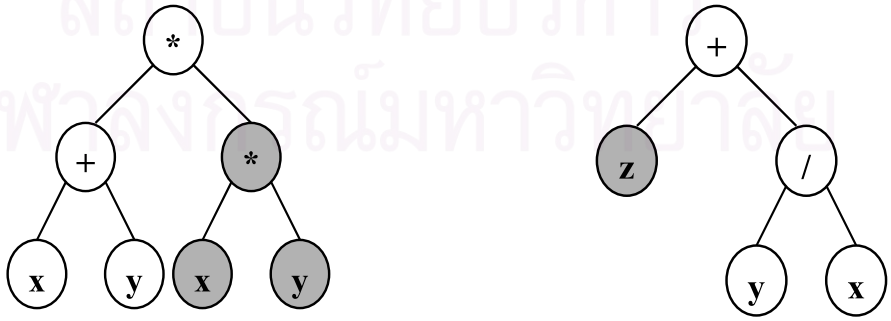
เป็นการนำต้นไม้ที่ได้รับการคัดเลือกแล้วมาเป็นต้นแบบ จากนั้นทำการคัดลอกต้นไม้ทั้งต้นนั้นไปไว้ในกลุ่มประชากรรุ่นใหม่ โดยไม่มีการเปลี่ยนแปลงภายในต้นไม้

2.1.3.2 **การไขว้เปลี่ยน** (Crossover)

เป็นวิธีทางพันธุกรรมที่สำคัญวิธีหนึ่ง กระบวนการนี้เป็นการสร้างต้นไม้ขึ้นมาใหม่ โดยการสุ่มเลือกต้นไม้จำนวน 2 ต้นจากกลุ่มต้นไม้ที่ได้คัดเลือกแล้ว เพื่อนำมาเป็น พ่อ แม่ (Parents) จากนั้นสุ่มเลือกตำแหน่งในการไขว้เปลี่ยน ซึ่งก็คือโหนดบนต้นไม้ของทั้งสองต้น แล้วนำต้นไม้ย่อย (Subtree) ที่เกิดจากโหนดที่สุ่มได้ของทั้งพ่อ และแม่มาสลับกัน ในบางกรณีโหนดที่สุ่มเลือกไม่มีต้นไม้ย่อยก็สลับเพียงโหนดเดียว จากวิธีการนี้จะได้ต้นไม้ใหม่จำนวนสองต้น (Children หรือ offspring) ที่มีลักษณะร่วมระหว่างพ่อและแม่ เปรียบเทียบได้กับการผสมพันธุ์ (Sexual combination) ในธรรมชาติ ตัวอย่างของการไขว้เปลี่ยนแสดงดังรูปที่ 2.2 และ 2.3



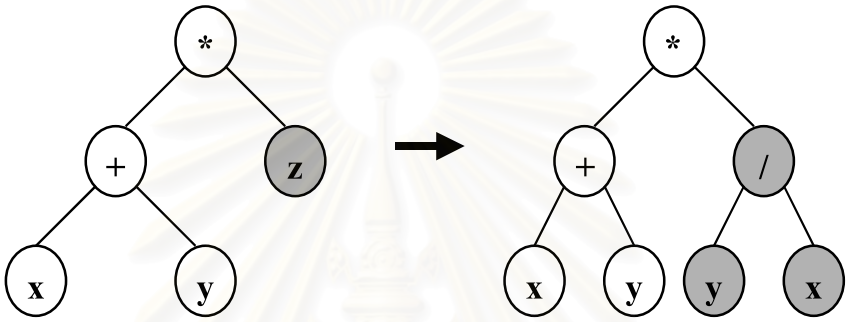
รูปที่ 2.2 ต้นไม้ที่ถูกคัดเลือกมาทำการไขว้เปลี่ยน



รูปที่ 2.3 ต้นไม้ 2 ต้นที่เกิดขึ้นหลังการไขว้เปลี่ยน

2.1.3.3 **การกลาย** (Mutation)

เป็นวิธีทางพันธุกรรมที่สำคัญอีกวิธีหนึ่ง การกลายเป็นการสุ่มเลือกต้นไม้จากกลุ่มที่ได้คัดเลือกแล้วมาทำการเปลี่ยนแปลงภายในต้นไม้ โดยสุ่มเลือกโหนดภายในต้นไม้ จากนั้นนำต้นไม้ย่อยที่สร้างใหม่ หรือโหนดที่สร้างใหม่ไปแทนที่ต้นไม้ย่อยเดิมที่เกิดจากโหนดนั้น ในกรณีที่โหนดไม่มีต้นไม้ย่อยโหนด หรือต้นไม้ย่อยที่สร้างใหม่ก็จะแทนที่โหนดเดิม วิธีการนี้เปรียบเสมือนการกลายพันธุ์ทางธรรมชาติ รูปที่ 2.4แสดงการกลายพันธุ์



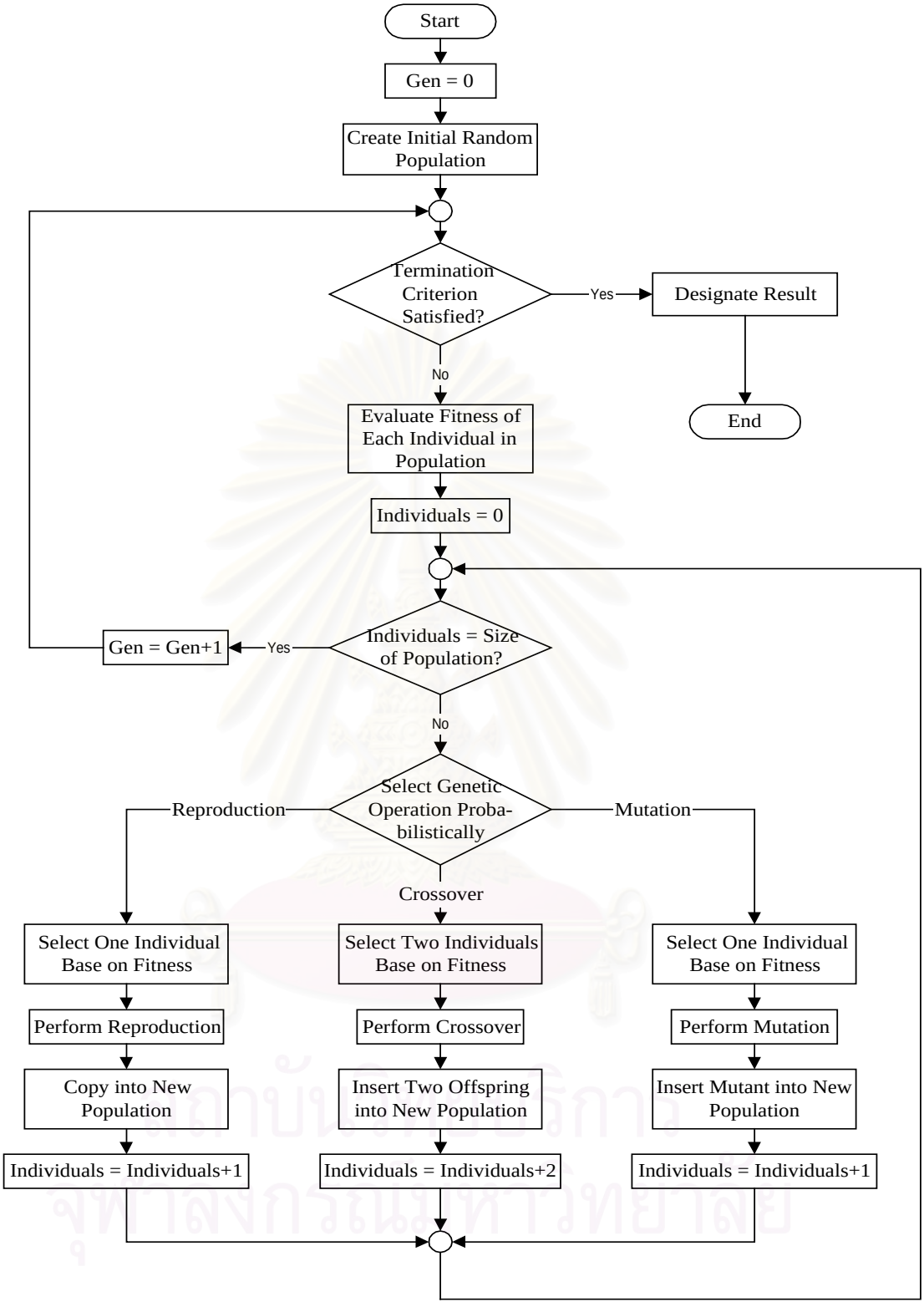
รูปที่ 2.4 ต้นไม้ก่อนการกลาย (ซ้าย) และต้นไม้ที่เกิดหลังการกลาย (ขวา)

หลังจากได้กลุ่มประชากรใหม่ครบตามจำนวนแล้ว กลุ่มประชากรนี้จะถูกนำกลับไปผ่าน ขั้นตอนการวัดค่าความเหมาะของผลเฉลย และการสร้างกลุ่มประชากรของผลเฉลยใหม่ เป็นรุ่นๆ (Generation) ต่อไป ซึ่งเรียกว่ากระบวนการวิวัฒนาการ

2.1.4 **การหาคำตอบ**

กำหนดการเชิงพันธุกรรมอาศัยกระบวนการวิวัฒนาการเพื่อหาคำตอบ (Solution) โดยในกระบวนการวิวัฒนาการนั้นเฉลี่ยแล้วผลเฉลยที่ได้ในแต่ละรุ่นจะมีค่าความเหมาะที่ดีขึ้น คือ มีความสามารถในการแก้ปัญหาได้ดีขึ้น เมื่อวิวัฒนาการเป็นรุ่นไปเป็นลำดับก็จะได้คำตอบที่สามารถแก้ปัญหาได้สำเร็จ ซึ่งกระบวนการวิวัฒนาการจะหยุดเมื่อพบคำตอบที่เหมาะสมแล้ว หรือพบเงื่อนไขที่ได้ตั้งไว้ ผลเฉลยในรุ่นสุดท้ายที่แก้ปัญหาได้สำเร็จจะเป็นคำตอบที่ได้จากกำหนดการเชิงพันธุกรรม

จากที่อธิบายการทำงานของกำหนดการเชิงพันธุกรรมมาทั้งหมดแล้วนั้น สามารถแสดงเป็นแผนภูมิสายงาน (Flow chart) ได้ดังรูปที่ 2.5



รูปที่ 2.5 แสดงแผนภูมิสายงานของกำหนดการเชิงพันธุกรรม

2.2 **สรุปท้ายบท**

กำหนดการเชิงพันธุกรรมเป็นวิธีสร้างโปรแกรมคอมพิวเตอร์ที่ไม่ต้องมีการเขียนโปรแกรมโดยตรง วิธีนี้นำหลักวิวัฒนาการทางธรรมชาติมาเป็นแนวทางเพื่อหาคำตอบ ทำให้มีความแตกต่างกับวิธีอื่นทางปัญญาประดิษฐ์ สำหรับขั้นตอนการทำงานของกำหนดการเชิงพันธุกรรมสามารถสรุปได้เป็น การสร้างฟังก์ชัน และเทอร์มินอลสำหรับแก้ปัญหา การสุ่มสร้างกลุ่มประชากรผลเฉลยเริ่มต้น การวัดประสิทธิภาพของผลเฉลย การสร้างกลุ่มประชากรผลเฉลยใหม่ และการหาคำตอบ ซึ่งในขั้นตอนเหล่านี้จะมีรายละเอียดต่างๆ เช่น จำนวนรุ่น อัตราการไขว้เปลี่ยน ที่จะต้องปรับเปลี่ยน เพื่อให้เหมาะสมกับการแก้ปัญหาต่างๆ

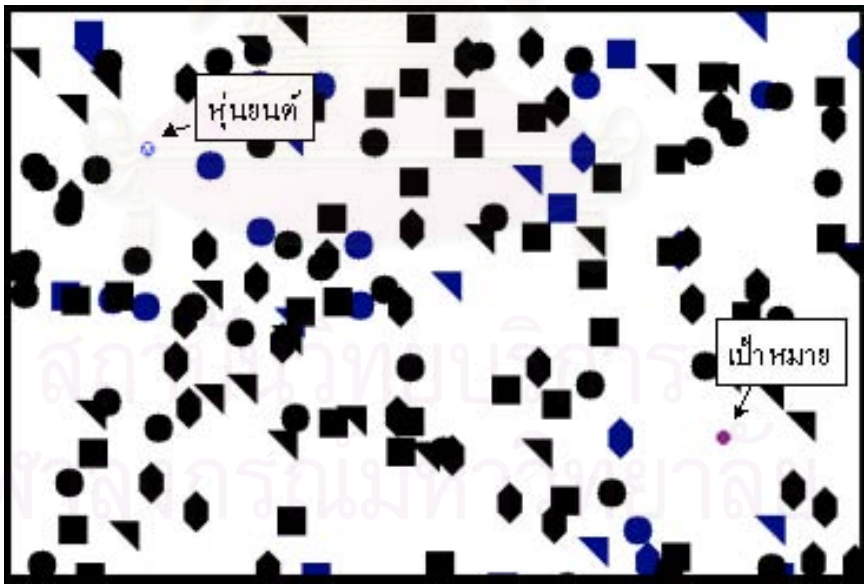
บทที่ 3

งานวิจัยที่เกี่ยวข้อง

ในบทนี้กล่าวถึงรายละเอียดของงานวิจัยต่างๆที่ผ่านมา ซึ่งเสนอแนวทางการเพิ่มความทนทานให้กับคำตอบที่ได้จากกำหนดการเชิงพันธุกรรม โดยงานวิจัยเหล่านี้จะแบ่งเป็นกลุ่มตามวิธีการปรับปรุงความทนทาน ได้แก่ การปรับปรุงโดยเรียนรู้จากสภาพแวดล้อมหลายๆแบบ การปรับปรุงโดยใช้ชุดฟังก์ชัน การปรับปรุงโดยใช้วิวัฒนาการร่วม และการปรับปรุงโดยใช้สัญญาณรบกวน

3.1 <u>การปรับปรุงความทนทานโดยเรียนรู้จากสภาพแวดล้อมหลาย ๆแบบ</u>

ธีระ โตสุโขวงศ์ และทัศสิน บัวชื่น (2540) ได้ทำการวิจัยเพื่อเพิ่มความทนทานของโปรแกรมที่ได้จากกำหนดการเชิงพันธุกรรม โดยใช้ปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายเป็นปัญหาในการทดลอง การทดลองนี้ดำเนินการในสภาพแวดล้อมจำลองบนเครื่องคอมพิวเตอร์ ซึ่งแสดงสภาพสนามที่ใช้ไว้ในรูปที่ 3.1



รูปที่ 3.1 สภาพสนามในปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย

งานวิจัยนี้นิยามความทนทาน หมายถึง การที่โปรแกรมสามารถควบคุมหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายได้ในสภาพสนามที่ไม่ได้เรียนรู้มา สำหรับวิธีเพิ่มความทนทานให้กับโปรแกรมนั้น งานวิจัยนี้ได้เสนอการปรับปรุงกระบวนการวิวัฒนาการให้เรียนรู้จากสนามหลายๆแบบ สนามที่ใช้เรียนรู้เกิดจากใส่สัญญาณรบกวนเข้าไปในสนามต้นแบบทำให้สิ่งกีดขวางเคลื่อนที่จากตำแหน่งเดิมเล็กน้อย โปรแกรมคำตอบที่ได้จากการปรับปรุงกระบวนการวิวัฒนาการนี้จะต้องสามารถเดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายได้ทุกสนามที่เรียนรู้ ผลจากการนำโปรแกรมคำตอบที่ได้ไปทดสอบกับสนามทดสอบที่ไม่ได้เรียนรู้มา ซึ่งสนามทดสอบนี้เกิดจากการใส่สัญญาณรบกวนเข้าไปในสนามต้นแบบเหมือนกับสนามที่ใช้เรียนรู้พบว่า โปรแกรมคำตอบที่ได้จากการเรียนรู้ในหลายๆสนามมีความทนทานมากกว่าโปรแกรมคำตอบที่เรียนรู้จากสนามเพียงสนามเดียว และพบว่ายิ่งใช้สนามในการเรียนรู้มากขึ้น โปรแกรมคำตอบที่ได้ก็ยิ่งมีความทนทานเพิ่มขึ้น

ต่อมา รุ่งโรจน์ นพสุวรรณชัย (2541) ได้ทำการทดลองเพิ่มเติมจากงานวิจัยของ ธีระ โตสุโขวงศ์ และทัศสิน บัวชื่น ให้ละเอียดขึ้น โดยเพิ่มจำนวนการทดลอง และจำนวนสนามที่ใช้ในส่วนของการเรียนรู้และการทดสอบ ผลการทดลองแสดงให้เห็นว่าการเพิ่มสนามเรียนรู้ให้มากขึ้นทำให้โปรแกรมคำตอบที่ได้มีความทนทานเพิ่มขึ้น อีกทั้งการเพิ่มระดับความแตกต่างของสภาพสนามที่ใช้เรียนรู้ก็มีผลทำให้ความทนทานของโปรแกรมคำตอบที่ได้มีมากขึ้นด้วย

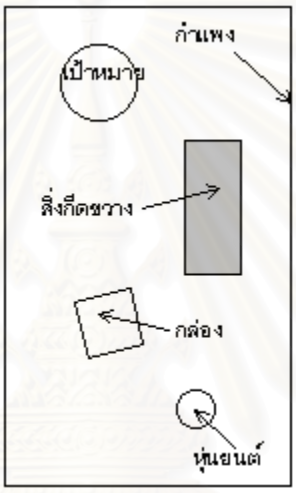
ในงานวิจัยนี้ยังได้วิเคราะห์ถึงสาเหตุที่ทำให้ความทนทานของโปรแกรมเพิ่มสูงขึ้น จากการศึกษาพบว่า "ประสบการณ์" เป็นปัจจัยที่มีผลต่อความทนทาน ซึ่งประสบการณ์ในที่นี้ หมายถึง กลุ่มของชุดคำสั่งควบคุมหุ่นยนต์ที่ได้จากตัวโปรแกรม โดยโปรแกรมที่มีอัตราส่วนการนำประสบการณ์ที่ได้ขณะเรียนรู้มาใช้กับสนามทดสอบสูง ทำให้โปรแกรมมีความทนทานเพิ่มขึ้น จากการวิเคราะห์ผลโดยใช้ประสบการณ์แสดงให้เห็นว่าการปรับปรุงกระบวนการวิวัฒนาการ โดยเรียนรู้จากสนามหลายๆแบบเป็นวิธีเพิ่มขนาดของประสบการณ์ให้กับโปรแกรม

3.2 <u>การปรับปรุงความทนทานโดยใช้ชุดฟังก์ชัน</u>

Takuya Ito Hitoshi Iba และ Masayuki Kimura (1996) ได้วิจัย และทดลองเกี่ยวกับความทนทานของโปรแกรมที่สังเคราะห์ได้จากกำหนดการเชิงพันธุกรรม โดยมีหุ่นยนต์เคลื่อนย้ายกล่องไปหาเป้าหมายเป็นปัญหาในการทดลอง ปัญหาหุ่นยนต์นี้ใช้สภาพแวดล้อมจำลองบนเครื่องคอมพิวเตอร์ดำเนินงาน ซึ่งสภาพสนามจำลองที่ใช้ประกอบไปด้วยตัวหุ่นยนต์ กล่อง สิ่งกีดขวาง และเป้าหมาย ซึ่งถูกล้อมรอบไว้ด้วยกำแพงดังรูปที่ 3.2

งานวิจัยนี้แบ่งออกเป็นสองการทดลองย่อย เพื่อสร้างโปรแกรมหุ่นยนต์ให้มีความทนทาน หรือสามารถปรับตัวเข้ากับสภาพแวดล้อมได้ดี การทดลองแรกมีการเปลี่ยนแปลงค่าเงื่อนไขเริ่มต้นของหุ่นยนต์แบบสุ่มในทุกรุ่น ส่วนการทดลองส่วนที่สองมีการใส่สัญญาณรบกวนเข้าไปในตัวเซนเซอร์ และการ

ทำงานของหุ่นยนต์ ผลลัพธ์ที่ได้จากทั้งสองการทดลองพบว่า โปรแกรมคำตอบที่ได้มีความทนทานเพิ่มขึ้นกล่าวคือ โปรแกรมนั้นยังคงสามารถใช้ได้ในสภาพแวดล้อมที่เปลี่ยนไป โดยมีค่าความเหมาะใกล้เคียงกับค่าความเหมาะเดิมก่อนที่สภาพแวดล้อมจะเปลี่ยนไป งานวิจัยนี้ยังชี้ให้เห็นว่าความทนทานของโปรแกรมคำตอบที่ได้นั้น มีสาเหตุมาจากความซ้ำซ้อน (Redundancy) ภายในโปรแกรมกล่าวคือ บางคำสั่งในตัวโปรแกรมไม่มีผลต่อสภาพแวดล้อมนี้ แต่เมื่อสภาพแวดล้อมเปลี่ยนไปกลับมีผล ดังนั้นถ้ามีฟังก์ชันที่ทำให้เกิดความซ้ำซ้อนก็จะทำให้โปรแกรมมีความทนทาน งานทดลองนี้ได้ทำการเปรียบเทียบโปรแกรมที่มีฟังก์ชันความซ้ำซ้อนกับโปรแกรมที่ไม่มี ผลปรากฏว่าโปรแกรมที่มีฟังก์ชันความซ้ำซ้อนมีความทนทานมากกว่า



รูปที่ 3.2 สภาพสนามในปัญหาหุ่นยนต์เคลื่อนย้ายกล่องไปหาเป้าหมาย

มาเรีย ประทีปทองคำ (2541) เป็นอีกงานวิจัยหนึ่งที่ปรับปรุงความทนทานโดยใช้ชุดฟังก์ชัน งานวิจัยนี้ได้เสนอแนวทางในการเพิ่มความทนทานให้กับโปรแกรมคำตอบ โดยนิยามความทนทานเป็นความสามารถในการทำงานของหุ่นยนต์ เมื่อสภาพสนามเปลี่ยนแปลงไปจากสภาพสนามที่ได้เรียนรู้มา งานวิจัยใช้ปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายเป็นปัญหาในการทดลอง ดังรูปที่ 3.1 โดยมีแนวคิดว่าถ้าให้โปรแกรมมีทางเลือก หรือความหลากหลายของการทำงานมากขึ้นแล้ว โปรแกรมที่ได้ก็จะมีความทนทานเพิ่มขึ้น ดังนั้นงานวิจัยนี้จึงสร้างฟังก์ชันพิเศษขึ้นเพื่อใส่ไปในชุดฟังก์ชัน ฟังก์ชันพิเศษนี้มีลักษณะการทำงานขึ้นอยู่กับการสุ่มทำให้โปรแกรมคำตอบที่ได้มีทางเลือกมากขึ้น ผลจากการทดลองพบว่าโปรแกรมที่มีฟังก์ชันพิเศษอยู่มีความทนทานมากกว่าโปรแกรมแบบธรรมดา หรือกล่าวได้ว่าโปรแกรมที่มีฟังก์ชันพิเศษอยู่สามารถใช้ได้กับสภาพสนามที่ไม่ได้เรียนรู้มาได้ดี และเมื่อวัดความหลากหลายของการทำงานปรากฏว่า โปรแกรมที่มีฟังก์ชันพิเศษอยู่มีความหลากหลายมากกว่าโปรแกรมแบบธรรมดา

3.3 การปรับปรุงความทนทานโดยใช้วิวัฒนาการร่วม

Gregory McNutt (1997) ใช้วิวัฒนาการร่วม (Co-Evolution) เป็นวิธีเพิ่มความทนทานให้กับโปรแกรมหุ่นยนต์ โดยวิวัฒนาการโปรแกรมหุ่นยนต์ให้มีความสามารถในการเคลื่อนที่ไปบนสนามทดสอบพร้อมกับวิวัฒนาการสนามทดสอบให้ยากขึ้น เพื่อทำให้หุ่นยนต์เคลื่อนที่ไม่สำเร็จ

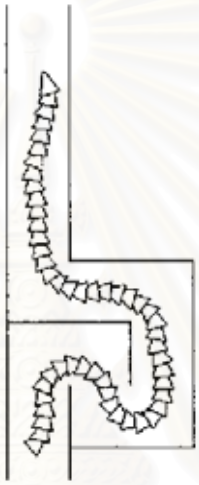
ในการวิวัฒนาการร่วมจะแบ่งเป็นสองกลุ่ม คือ โปรแกรมควบคุมหุ่นยนต์ และสนามทดสอบ ทั้งสองกลุ่มนี้มีวิวัฒนาการร่วมกัน โดยสนามที่ได้รับการวิวัฒนาการแล้วจะถูกส่งไปยังกลุ่มโปรแกรมควบคุมหุ่นยนต์เพื่อหาค่าความเหมาะ โปรแกรมที่สามารถควบคุมหุ่นยนต์ผ่านสนามทดสอบได้มากก็จะมีค่าความเหมาะที่ดี และได้รับการวิวัฒนาการต่อไป ในขณะที่สนามที่มีโปรแกรมควบคุมหุ่นยนต์ผ่านได้น้อยก็จะมีค่าความเหมาะที่ดี และได้รับการวิวัฒนาการต่อไป สภาพสนามทดสอบ และการเคลื่อนที่ของหุ่นยนต์ในงานวิจัยนี้แสดงไว้ดังรูปที่ 3.3

รูปที่ 3.3 สนามทดสอบ และการเคลื่อนที่ของหุ่นยนต์ในการทดลองของ Gregory McNutt

ผลจากการทดลองพบว่าในกรณีที่สนามทดสอบมีความยากสูง โปรแกรมคำตอบสามารถควบคุมหุ่นยนต์เคลื่อนที่บนสนามได้คิดเป็น 98% ส่วนในกรณีสนามทดสอบทั่วไป โปรแกรมคำตอบสามารถควบคุมหุ่นยนต์เคลื่อนที่ได้เป็น 100% ซึ่งชี้ให้เห็นว่าการปรับปรุงความทนทานโดยใช้วิวัฒนาการร่วมเพิ่มความทนทานให้กับโปรแกรมคำตอบได้อย่างดี

3.4 การปรับปรุงความทนทานโดยใช้สัญญาณรบกวน

Craig W. Reynolds (1994) ได้เสนอแนวความคิดเพื่อเพิ่มความทนทานให้กับโปรแกรมคำตอบที่ได้จากกำหนดการเชิงพันธุกรรม โดยการใส่สัญญาณรบกวน (noise) เข้าไปในระบบ เนื่องจากในสภาพแวดล้อมจริงนั้นสิ่งรบกวนมีผลทำให้โปรแกรมทำงานผิดพลาดได้ การใส่สัญญาณรบกวนเข้าไปจึงมุ่งหวังให้โปรแกรมมีความทนทานต่อสิ่งรบกวน สำหรับปัญหาที่ใช้ในงานวิจัยนี้ คือ การควบคุมหุ่นยนต์ให้เดินในสภาพแวดล้อมที่เป็นทางเดินแคบๆ และคดเคี้ยวได้โดยไม่ชนกำแพง สภาพสนามจำลองบนเครื่องคอมพิวเตอร์แสดงดังรูปที่ 3.4



รูปที่ 3.4 สนามจำลองที่ใช้ในงานวิจัยของ Craig W. Reynolds

ในการทดลองนี้ได้ใส่สัญญาณรบกวนเข้าไปใน ตัวเซนเซอร์ และมอเตอร์ของหุ่นยนต์ เช่น เคลื่อนที่ผิดพลาด เซนเซอร์บอกตำแหน่งผิดพลาด ผลการทดลองปรากฏว่าไม่พบโปรแกรมคำตอบที่มีความทนทาน คือ ไม่มีโปรแกรมที่สามารถควบคุมหุ่นยนต์ให้เดินผ่านสนามไปได้โดยไม่ชนกำแพง งานวิจัยนี้จึงสรุปผลได้ว่าสาเหตุที่ไม่พบโปรแกรมคำตอบนั้นอาจเนื่องมาจาก ปัญหามีความยากเกินไป หรือกำหนดค่าพารามิเตอร์ของกำหนดการเชิงพันธุกรรมไม่เหมาะสม

ต่อมา Reynolds ได้ทำการทดลองต่อจากปัญหาเดิม โดยเปลี่ยนตำแหน่งเซนเซอร์ให้อยู่คงที่เพื่อทำให้ปัญหามีความยุ่งยากน้อยลง ผลปรากฏว่าการปรับเปลี่ยนครั้งนี้ทำให้พบโปรแกรมคำตอบที่มีความทนทาน จึงสรุปได้ว่ากำหนดการเชิงพันธุกรรมสามารถสร้างโปรแกรมคำตอบที่มีความทนทานได้และการใส่สัญญาณรบกวนมีผลทำให้โปรแกรมคำตอบมีความทนทาน

3.5 **สรุปท้ายบท**

งานวิจัยที่ได้กล่าวมาในบทนี้เป็นงานวิจัยเพื่อเพิ่มความทนทานให้กับคำตอบที่ได้จากกำหนดการเชิงพันธุกรรม โดยทั้งหมดมีปัญหาเกี่ยวกับหุ่นยนต์เคลื่อนที่เป็นปัญหาในการทดลอง งานวิจัยเหล่านี้ได้เสนอวิธีเพิ่มความทนทานต่างๆ ซึ่งแบ่งออกเป็น การปรับปรุงโดยเรียนรู้จากสภาพแวดล้อมหลายๆ แบบ การปรับปรุงโดยใช้สัญญาณรบกวน การปรับปรุงโดยใช้ชุดฟังก์ชัน และการปรับปรุงโดยใช้วิวัฒนาการร่วม ซึ่งจากผลการทดลองในงานวิจัยต่างๆ แสดงให้เห็นว่า ทุกวิธีได้เพิ่มความทนทานให้กับคำตอบ

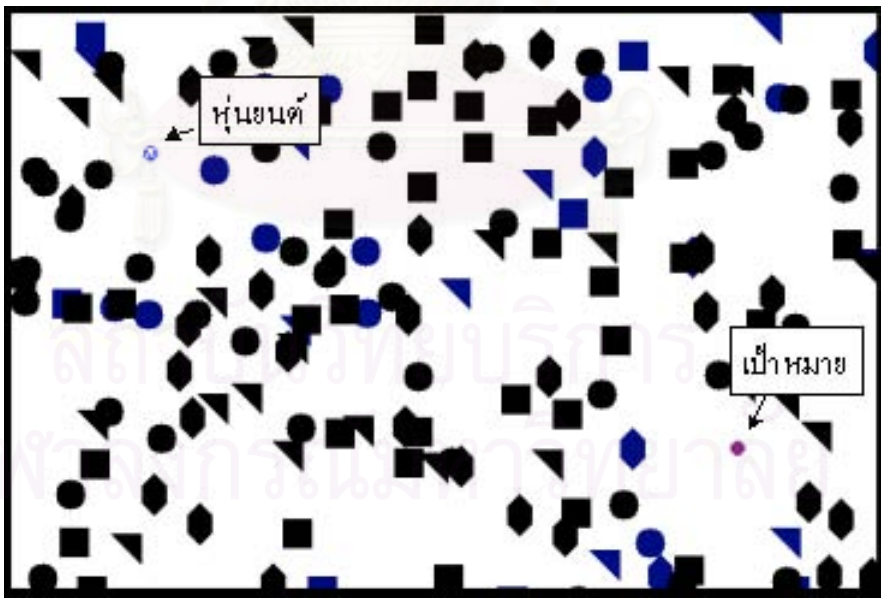
บทที่ 4

รายละเอียดการทดลอง

บทนี้อธิบายรายละเอียดต่างๆในการทดลองประกอบไปด้วย 1) ลักษณะปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย 2) รายละเอียดของกำหนดการเชิงพันธุกรรม สำหรับการแก้ปัญหา 3) การทดสอบความทนทานของโปรแกรมคำตอบ และ 4) สรุปท้ายบท

4.1 ปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย

ปัญหาที่ใช้ในงานวิจัยนี้มีลักษณะเหมือนกับปัญหาในงานวิจัยของ รุ่งโรจน์ นพสุวรรณชัย (2541) และ มาเรีย ประทีปทองคำ (2541) โดยเป้าหมายของปัญหานี้ คือ การควบคุมหุ่นยนต์ให้เคลื่อนที่ในสนามจากจุดเริ่มต้นหลบหลีกสิ่งกีดขวางไปยังเป้าหมาย และเนื่องจากการทดลองบนสภาพแวดล้อมจริงมีข้อจำกัดหลายด้าน การทดลองนี้จึงสร้างสนามทดลองเป็นสภาพแวดล้อมจำลองบนเครื่องคอมพิวเตอร์ สภาพสนามแสดงไว้ในรูปที่ 4.1



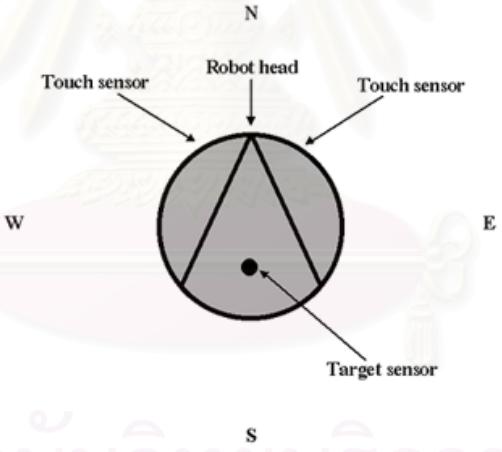
รูปที่ 4.1 สภาพสนามจำลองที่ใช้ในการทดลอง

สภาพสนามจำลองในการทดลองมีความกว้าง 550 หน่วย ความยาว 750 หน่วย ถูกล้อมรอบด้วยกำแพงหนา 5 หน่วยในทุกด้าน ภายในสนามมีสิ่งกีดขวางรูปทรงเราขาคณิต 4 กลุ่ม ได้แก่ สามเหลี่ยมมุมฉาก สี่เหลี่ยม วงกลม และหกเหลี่ยม วางอยู่กระจัดกระจาย สิ่งกีดขวางเหล่านี้มีขนาดโดยเฉลี่ยประมาณ 400 ตารางหน่วย พื้นที่รวมทั้งหมดของสิ่งกีดขวางในสนามคิดเป็น 20 % ของพื้นที่สนามทั้งหมด

สำหรับตัวหุ่นยนต์มีลักษณะดังต่อไปนี้

- เป็นวงกลมรัศมี 5 หน่วย
- สามารถหมุนได้ทีละ 22.5 องศา ทั้งตามเข็มนาฬิกา และทวนเข็มนาฬิกา
- เคลื่อนที่ไปด้านหน้าได้ครั้งละ 1.0 หน่วย ในทิศทางเดียวกับหัวหุ่นยนต์
- มีเซนเซอร์การชน (Touch sensor) ที่ส่วนหัวหุ่นยนต์ เพื่อรับรู้กับการชนสิ่งกีดขวาง หรือกำแพง
- มีเซนเซอร์ตรวจเป้าหมาย (Target sensor) สามารถวัดว่าเข้าใกล้เป้าหมาย หรือออกห่างจากเป้าหมายได้
- จุดเริ่มต้นอยู่บนพิกัด (60.0,140.0)

ลักษณะทางกายภาพของหุ่นยนต์เป็นดังรูปที่ 4.2



รูปที่ 4.2 หุ่นยนต์ที่ใช้ในการทดลอง

สำหรับเป้าหมายมีลักษณะดังนี้

- เป็นวงกลมรัศมี 5 หน่วย
- ตั้งอยู่บนพิกัด (550.0,400.0)
- ตัวเป้าหมายมีคลื่นพิเศษแผ่ออกมาทำให้เซนเซอร์ตรวจเป้าหมายของหุ่นยนต์สามารถตรวจจับได้

ในสนามจำลองนี้การที่หุ่นยนต์ถึงเป้าหมาย คือ จุดศูนย์กลางของหุ่นยนต์อยู่ในพื้นที่วงกลมของเป้าหมาย

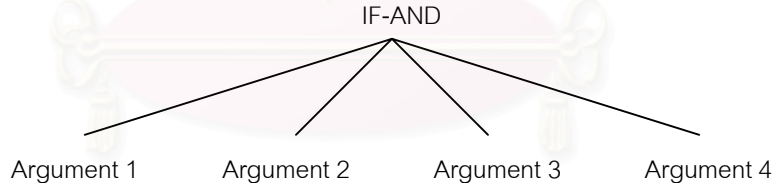
4.2 กำหนดการเชิงพันธุกรรมในการแก้ปัญหา

การทดลองนี้ใช้กำหนดการเชิงพันธุกรรมเป็นวิธีสร้างโปรแกรมคอมพิวเตอร์ เพื่อควบคุมหุ่นยนต์เคลื่อนที่หลบหลีกสิ่งกีดขวางไปหาเป้าหมาย รายละเอียดของกำหนดการเชิงพันธุกรรมในการสร้างโปรแกรมควบคุมหุ่นยนต์ เป็นดังนี้

4.2.1 ฟังก์ชัน และเทอร์มินอล

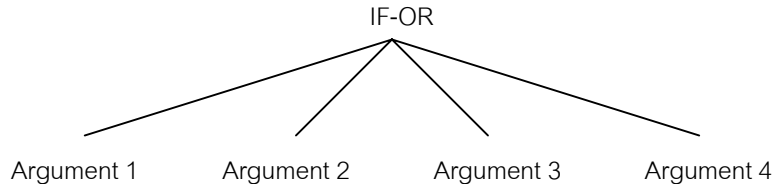
ฟังก์ชันในการสร้างโปรแกรมควบคุมนี้เป็นฟังก์ชันมาตราฐานทางตรรกศาสตร์ทั่วไปที่ใช้ในการเขียนโปรแกรม สามารถต่อเข้ากับเทอร์มินอลต่างๆ และสื่อความหมายของโปรแกรมคอมพิวเตอร์ได้ง่าย ชุดฟังก์ชันนี้ประกอบไปด้วย IF-AND IF-OR และ IF-NOT

IF-AND เป็นฟังก์ชันที่ต่อเข้ากับอาร์กิวเมนต์จำนวนสี่ตัว อาร์กิวเมนต์ทั้งสี่ตัวเป็นได้ทั้งฟังก์ชัน และเทอร์มินอล ขั้นตอนการทำงานของฟังก์ชันเริ่มจากประมวลผลในอาร์กิวเมนต์ที่หนึ่ง และสองก่อน จากนั้นนำค่าส่งคืน (Return Value) ที่ได้จากทั้งสองอาร์กิวเมนต์ข้างต้นมาประมวลผลด้วยตัวดำเนินการ AND ถ้าผลที่ได้เป็นจริงจะประมวลผลต่อในอาร์กิวเมนต์ที่สาม ถ้าผลที่ได้เป็นเท็จจะประมวลผลต่อในอาร์กิวเมนต์ที่สี่ สำหรับค่าส่งคืนของฟังก์ชันนี้เป็นค่าส่งคืนจากการประมวลผลอาร์กิวเมนต์ที่สาม หรือสี่ขึ้นอยู่กับว่าประมวลผลที่อาร์กิวเมนต์ใด ฟังก์ชัน IF-AND แสดงเป็นรูปต้นไม้ได้ดังรูปที่ 4.3



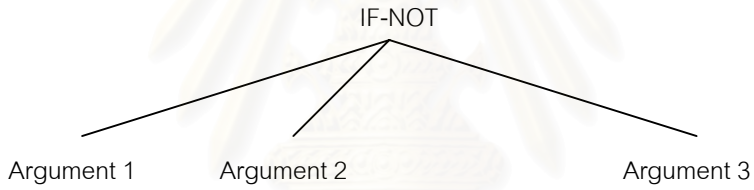
รูปที่ 4.3 ลักษณะต้นไม้ของฟังก์ชัน IF-AND

IF-OR เป็นอีกฟังก์ชันที่มีอาร์กิวเมนต์สี่ตัว ขั้นตอนการประมวลผลของฟังก์ชันนี้เริ่มจากประมวลผลในอาร์กิวเมนต์ตัวที่หนึ่ง และสองก่อน จากนั้นนำค่าส่งคืนที่ได้จากทั้งสองอาร์กิวเมนต์มาประมวลผลด้วยตัวดำเนินการ OR ถ้าผลลัพธ์ที่ได้เป็นจริงจะประมวลผลในอาร์กิวเมนต์ที่สาม แต่ถ้าผลลัพธ์ที่ได้เป็นเท็จจะประมวลผลในอาร์กิวเมนต์ที่สี่ ค่าส่งคืนของฟังก์ชันนี้เป็นค่าส่งคืนจากการประมวลผลในอาร์กิวเมนต์ที่สาม หรือสี่ ขึ้นอยู่กับว่าประมวลผลที่อาร์กิวเมนต์ใด ฟังก์ชัน IF-OR แสดงเป็นรูปต้นไม้ดังรูปที่ 4.4



รูปที่ 4.4 ลักษณะต้นไม้ของฟังก์ชัน IF-OR

IF-NOT เป็นฟังก์ชันที่มีอาร์กิวเมนต์สามตัว มีขั้นตอนการประมวลผล โดยเริ่มจากอาร์กิวเมนต์ที่หนึ่งเพียงตัวเดียว จากนั้นนำค่าส่งคืนที่ได้มาประมวลผลด้วยตัวดำเนินการ NOT ถ้าผลลัพธ์ที่ได้จากตัวดำเนินการ NOT เป็นจริงจะประมวลผลในอาร์กิวเมนต์ที่สอง แต่ถ้าผลลัพธ์ที่ได้เป็นเท็จจะประมวลผลในอาร์กิวเมนต์ที่สาม ค่าส่งคืนของฟังก์ชันนี้เป็นค่าส่งคืนจากอาร์กิวเมนต์ที่สอง หรือสาม ขึ้นอยู่กับว่าประมวลผลที่อาร์กิวเมนต์ใด ฟังก์ชัน IF-NOT แสดงเป็นรูปต้นไม้ดังรูปที่ 4.5



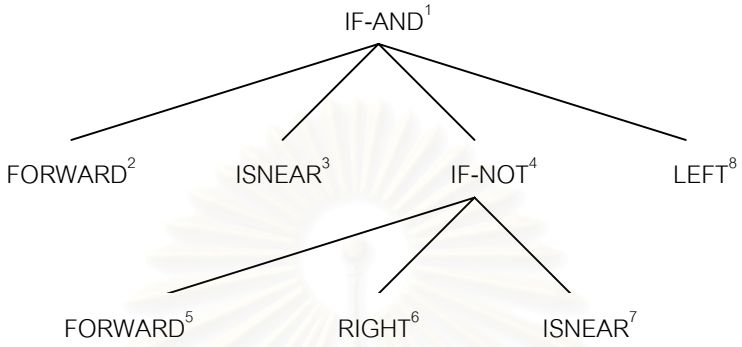
รูปที่ 4.5 ลักษณะต้นไม้ของฟังก์ชัน IF-NOT

สำหรับเทอร์มินอลที่ใช้สร้างโปรแกรมควบคุมหุ่นยนต์นี้ เป็นคำสั่งพื้นฐานการทำงาน ซึ่งสอดคล้องกับลักษณะทางกายภาพของหุ่นยนต์ที่ได้อธิบายไว้ในหัวข้อที่ 4.1 ชุดเทอร์มินอลพื้นฐานนี้มี 4 คำสั่ง ประกอบไปด้วย

- FORWARD เป็นการเคลื่อนที่ไปข้างหน้า 1.0 หน่วยในทิศทางเดียวกับหัวหุ่นยนต์ ค่าส่งคืนมีค่าเป็นจริงเมื่อสามารถเดินหน้าได้ และมีค่าเป็นเท็จเมื่อมีสิ่งกีดขวางอยู่เบื้องหน้าไม่สามารถเคลื่อนที่ได้
- LEFT เป็นการหมุนตัวหุ่นยนต์ไป 22.5 องศา ในทิศทางทวนเข็มนาฬิกา ค่าส่งคืนมีค่าเป็นจริง
- RIGHT เป็นการหมุนตัวหุ่นยนต์ไป 22.5 องศา ในทิศทางตามเข็มนาฬิกา ค่าส่งคืนมีค่าเป็นจริง

- ISNEAR เป็นเทอร์มินอลที่รับค่าจากเซนเซอร์เป้าหมาย เพื่อตรวจสอบว่าเข้าใกล้เป้าหมายหรือไม่ ถ้าเคลื่อนเข้าหาเป้าหมายมากขึ้นค่าส่งคืนมีค่าเป็นจริง แต่ถ้าเคลื่อนออกห่างเป้าหมาย ค่าส่งคืนมีค่าเป็นเท็จ

ฟังก์ชัน และเทอร์มินอลเหล่านี้สามารถสร้างเป็นโปรแกรมหุ่นยนต์ได้ดังรูปที่ 4.6



รูปที่ 4.6 ตัวอย่างโปรแกรมหุ่นยนต์

จากโปรแกรมหุ่นยนต์ในรูปที่ 4.6 เมื่อนำไปประมวลผลสามารถอธิบายเป็นขั้นตอนได้ดังนี้ เทอร์มินอล FORWARD[2] และ ISNEAR[3] จะถูกประมวลประมวลผลก่อน จากนั้นถ้าค่าส่งคืนของทั้งสองเทอร์มินอลนี้เป็นจริงฟังก์ชัน IF-NOT[4] จะได้รับการประมวลผล แต่ถ้าค่าส่งคืนของทั้งสองฟังก์ชันตัวใดตัวหนึ่งเป็นเท็จเทอร์มินอล LEFT[8] จะได้รับการประมวลผลแทน ในกรณีที่ฟังก์ชัน IF-NOT[4] ได้รับการประมวลผลเทอร์มินอล FORWARD[5] จะได้รับการประมวลด้วย และถ้าค่าส่งคืนที่ได้จาก FORWARD[5] เป็นจริงเทอร์มินอล ISNEAR[7] จะได้รับการประมวลผลแต่ถ้าเป็นเท็จ เทอร์มินอล RIGHT[6] จะได้รับการประมวลผลแทน สำหรับค่าส่งคืนของ IF-AND[1] จะเป็นค่าส่งคืนจากฟังก์ชัน IF-NOT[4] หรือเทอร์มินอล LEFT[8] ขึ้นอยู่กับการประมวลผล จากการประมวลผลนี้จะเห็นได้ว่า เมื่อมีฟังก์ชันไปเป็นอาร์กิวเมนต์ของอีกฟังก์ชันหนึ่งแล้ว การประมวลผลจะวนซ้ำ (Recursive) เข้าไปในต้นไม้ย่อย (Subtree) ที่เกิดจากฟังก์ชันนั้น

4.2.2 การสร้างประชากรผลเฉลยเริ่มต้น

หลังจากได้ฟังก์ชัน และเทอร์มินอลสำหรับสร้างโปรแกรมหุ่นยนต์แล้ว กำหนดการเชิงพันธุกรรมจะทำการสุ่มฟังก์ชัน และเทอร์มินอลด้วยความน่าจะเป็นเท่าๆกัน มาสร้างโปรแกรมหุ่นยนต์เพื่อเป็นประชากรผลเฉลยเริ่มต้นจำนวน 1,500 โปรแกรม โปรแกรมหุ่นยนต์ที่สร้างเหล่านี้มีความสูงของต้นไม้ไม่เกิน 4 หรือโดยเฉลี่ยโปรแกรมหุ่นยนต์จะมีจำนวนฟังก์ชัน และเทอร์มินอลรวม 180 คำสั่ง

4.2.3 การวัดค่าความเหมาะของผลเฉลย

เมื่อสร้างกลุ่มประชากรผลเฉลยเริ่มต้นแล้ว ขั้นต่อมาโปรแกรมหุ่นยนต์แต่ละตัวในกลุ่มประชากรผลเฉลยจะถูกประมวลผลเพื่อวัดค่าความเหมาะในการแก้ปัญหา ค่าความเหมาะที่ได้นี้จะนำไปใช้เป็นเกณฑ์ในการคัดเลือกโปรแกรม เพื่อดำเนินการทางพันธุกรรมต่อไป

ในการวัดค่าความเหมาะโปรแกรมหุ่นยนต์แต่ละตัวจะถูกประมวลผล เริ่มจากราก (Root) ลงไปยังโหนดล่างสุดของต้นไม้ (Leaf) และวนกลับมาประมวลผลจากรากลงไปยังโหนดล่างสุดของต้นไม้อย่างนี้ซ้ำหลายครั้ง จนกระทั่งพบเงื่อนไขที่ทำให้สิ้นสุดการประมวลผลต่อไปนี้

- ตัวหุ่นยนต์ถึงเป้าหมาย
- จำนวนฟังก์ชัน และเทอร์มินอลที่ถูกปฏิบัติการครบตามที่กำหนด ในเงื่อนไขนี้จะมีการทดลองย่อยในบทต่อไปเพื่อหาจำนวนที่เหมาะสม

หลังจากสิ้นสุดการประมวลผลโปรแกรมแต่ละตัวแล้ว โปรแกรมนั้นจะได้ค่าคุณสมบัติต่างๆ ซึ่งนำไปใช้คำนวณค่าความเหมาะของตัวโปรแกรม

จุดประสงค์ของปัญหาหุ่นยนต์นี้ คือ การเคลื่อนหุ่นยนต์ไปยังจุดเป้าหมาย ดังนั้นค่าความเหมาะจะต้องบอกได้ว่า โปรแกรมที่สามารถควบคุมหุ่นยนต์ให้เคลื่อนที่เข้าใกล้เป้าหมายได้มากเท่าไหร่ยิ่งเป็นโปรแกรมที่ดี และการทดลองนี้ยังต้องการโปรแกรมหุ่นยนต์ที่มีการทำงานน้อยที่สุด เพื่อให้โปรแกรมควบคุมหุ่นยนต์ถึงเป้าหมายได้เร็ว เพราะฉะนั้นค่าความเหมาะที่ได้จะต้องบอกถึงจำนวนฟังก์ชัน และเทอร์มินอลที่ใช้ประมวลผลด้วย จากความต้องการเหล่านี้ผู้วิจัยได้สร้างฟังก์ชันความเหมาะดังสมการ

$$\textit{Fitness function} = (10{,}000 \times \textit{Distance}) + \textit{Term} \qquad (1)$$

โดยที่ Distance หมายถึง ระยะขจัดระหว่างหุ่นยนต์กับเป้าหมาย หลังจากเสร็จสิ้นการประมวลผล

Term หมายถึง จำนวนรวมของฟังก์ชัน และเทอร์มินอลที่ถูกปฏิบัติการหลังจากสิ้นสุดการประมวลผล

10,000 เป็นค่าคงที่เพื่อให้น้ำหนักกับตัวแปร Distance ซึ่งเป็นปัจจัยหลักของปัญหาหุ่นยนต์นี้

หมายเหตุ ค่าที่ได้จากฟังก์ชันความเหมาะนี้ ค่ายิ่งน้อยโปรแกรมยิ่งมีประสิทธิภาพในการแก้ปัญหา

โปรแกรมทุกตัวในกลุ่มประชากรจะถูกประมวลผลแล้วหาค่าความเหมาะ จากนั้นจะเข้าสู่ขั้นตอนวิธีทางพันธุกรรมเพื่อสร้างกลุ่มประชากรใหม่ โดยมีความคาดหวังว่าโปรแกรมในกลุ่มประชากรใหม่จะมีประสิทธิภาพในการแก้ปัญหาดีขึ้น

4.2.4 การสร้างประชากรผลเฉลยใหม่

ขั้นตอนนี้โปรแกรมหุ่นยนต์ในกลุ่มประชากรผลเฉลยจะถูกเรียงลำดับตามค่าความเหมาะจากน้อยไปมาก โปรแกรมที่มีประสิทธิภาพในการแก้ปัญหามากที่สุดจะถูกจัดให้อยู่ในลำดับแรก ส่วนโปรแกรมที่มีประสิทธิภาพในการแก้ปัญหาน้อยที่สุดจะถูกจัดให้อยู่อันดับสุดท้าย จากนั้นจะดำเนินการทางพันธุกรรมเพื่อสร้างกลุ่มประชากรใหม่ โดยมีวิธีทางพันธุกรรมดังนี้

4.2.4.1 การสืบพันธุ์

การสืบพันธุ์จะคัดลอกเอาโปรแกรมหุ่นยนต์ 150 โปรแกรมแรกที่ได้เรียงลำดับไว้ของกลุ่มประชากรเดิมมาเป็นกลุ่มประชากรใหม่ โดยไม่มีการเปลี่ยนแปลงตัวโปรแกรม การสืบพันธุ์นี้คิดเป็น 10% ของกลุ่มประชากร

4.2.4.2 การไขว้เปลี่ยน

การไขว้เปลี่ยนจะนำโปรแกรมหุ่นยนต์ 150 โปรแกรมที่ได้จากการสืบพันธุ์ มาเป็นกลุ่มต้นแบบ จากนั้นจะสุ่มเลือกโปรแกรมหุ่นยนต์ต้นแบบ (Parents) ไม่ซ้ำกันสองโปรแกรมมาทำการไขว้เปลี่ยน ซึ่งจะได้โปรแกรมใหม่สองโปรแกรม (Children หรือ Offspring) มาเป็นกลุ่มประชากรใหม่ การไขว้เปลี่ยนจะวนทำจนได้โปรแกรมใหม่จำนวน 1,350 โปรแกรมคิดเป็น 90% ของกลุ่มประชากร โดยการไขว้เปลี่ยนแต่ละครั้งโปรแกรมต้นแบบอาจซ้ำกันได้

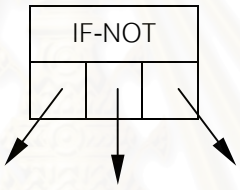
เมื่อได้กลุ่มประชากรรุ่นใหม่ครบแล้ว กลุ่มประชากรรุ่นนี้จะถูกนำไปวัดค่าความเหมาะของผลเฉลยอีก กำหนดการเชิงพันธุกรรมจะวนทำในขั้นตอนการวัดค่าความเหมาะผลเฉลย และการสร้างประชากรผลเฉลยใหม่ จนกระทั่งครบตามจำนวนรุ่นที่ต้องการ โดยจำนวนรุ่นเป็นอีกพารามิเตอร์หนึ่งที่จะทำการทดลองย่อยในบทต่อไปเพื่อหาจำนวนที่เหมาะสม

4.2.5 การหาคำตอบ

เมื่อกำหนดการเชิงพันธุกรรมวิวัฒนาการกลุ่มประชากรครบตามจำนวนรุ่นที่ต้องการแล้ว กลุ่มประชากรในรุ่นสุดท้ายจะเป็นกลุ่มคำตอบสำหรับปัญหานั้น โดยโปรแกรมที่ดีที่สุด และสามารถแก้ปัญหาได้จะเป็นโปรแกรมคำตอบ

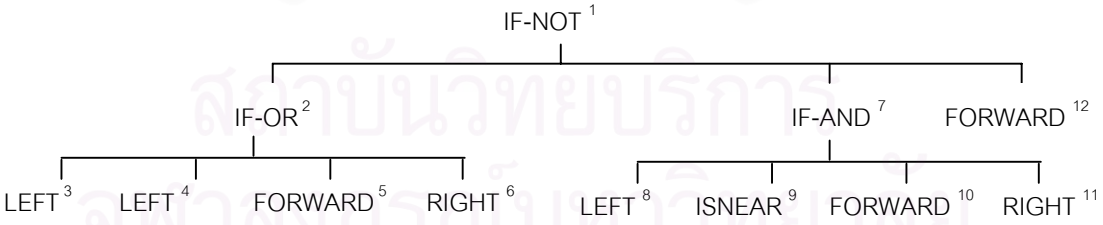
4.2.6 โครงสร้างข้อมูลในกำหนดการเชิงพันธุกรรม

โดยทั่วไปโครงสร้างข้อมูลของโปรแกรมคอมพิวเตอร์ในกำหนดการเชิงพันธุกรรมจะเป็นแบบต้นไม้ (Tree) กล่าวคือ จะมีโหนดที่เก็บค่าฟังก์ชัน หรือเทอร์มินอล และเก็บตัวเชื่อม (Link) เพื่อชี้ไปยังตำแหน่งของโหนดต่อไปไว้ด้วยกัน ตัวโหนดนี้หลายๆตัวจะประกอบรวมกันเป็นต้นไม้ รูปที่ 4.7 แสดงโครงสร้างข้อมูลของฟังก์ชัน IF-NOT



รูปที่ 4.7 โครงสร้างข้อมูลของโหนด IF-NOT

จากโครงสร้างข้อมูลนี้เมื่อนำโปรแกรมคอมพิวเตอร์ในรูปที่ 4.8 ไปเก็บในหน่วยความจำ (Memory) จะเป็นดังรูปที่ 4.9



รูปที่ 4.8 ตัวอย่างโปรแกรมคอมพิวเตอร์

หน่วยความจำ				
IF-NOT [1]	-> IF-OR [2]	->IF-AND [7]	-> FORWARD [12]	IF-OR [2]
->LEFT [3]	->LEFT [4]	->FORWARD [5]	->RIGHT [6]	LEFT [3]
LEFT [4]	FORWARD [5]	RIGHT [6]	IF-AND [7]	->LEFT [8]
->ISNEAR [9]	->FORWARD [10]	->RIGHT [11]	LEFT [8]	ISNEAR [9]
FORWARD [10]	RIGHT [11]	FORWARD [12]		

หมายเหตุ เครื่องหมาย "->" หมายถึง ตำแหน่งหน่วยความจำ

รูปที่ 4.9 แสดงภาพจำลองของหน่วยความจำที่ใช้เก็บโปรแกรมคอมพิวเตอร์

ถ้าหนึ่งช่องในหน่วยความจำของรูปที่ 4.9 ใช้เนื้อที่ขนาด 4 ไบต์ เพื่อเก็บฟังก์ชันเทอร์มินอล และตำแหน่งหน่วยความจำ โปรแกรมคอมพิวเตอร์ตัวอย่างจะใช้เนื้อที่ทั้งหมด 92 ไบต์ จะเห็นได้ว่าโครงสร้างข้อมูลแบบนี้ต้องใช้หน่วยความจำคอมพิวเตอร์จำนวนมาก จึงไม่เหมาะกับการแก้ปัญหาที่ต้องการกลุ่มประชากรขนาดใหญ่ เช่น การทดลองนี้ ดังนั้นผู้วิจัยได้นำโครงสร้างข้อมูลที่เสนอโดย Rodkaew, Y., and Chongstitvatana, P (2000) มาใช้

โครงสร้างข้อมูลแบบใหม่จะเก็บโปรแกรมคอมพิวเตอร์ให้อยู่ในรูปของเวกเตอร์ (Vector) โดยเรียงลำดับฟังก์ชัน และเทอร์มินอลของโปรแกรมแบบพรีออเดอร์ (Preorder) และไม่เก็บตำแหน่งเชื่อมโยงไปยังฟังก์ชัน หรือเทอร์มินอลอื่น โครงสร้างข้อมูลแบบเวกเตอร์แสดงได้ดังรูปที่ 4.10

หน่วยความจำ				
IF-NOT [1]	IF-OR [2]	LEFT [3]	LEFT [4]	FORWARD [5]
RIGHT [6]	IF-AND [7]	LEFT [8]	ISNEAR [9]	FORWARD [10]
RIGHT [11]	FORWARD [12]			

รูปที่ 4.10 โปรแกรมคอมพิวเตอร์ที่เก็บในโครงสร้างข้อมูลแบบเวกเตอร์

ในรูปที่ 4.10 จะเห็นได้ว่าใช้หน่วยความจำเพียง 12 หน่วยเท่านั้น และเนื่องจากโครงสร้างข้อมูลแบบนี้ไม่ต้องเก็บตำแหน่งเชื่อมโยงทำให้แต่ละช่องในหน่วยความจำใช้เนื้อที่เพียง 1 ไบต์ก็สามารถแทนค่าฟังก์ชัน และเทอร์มินอลได้ทั้งหมด 256 แบบ ซึ่งเพียงพอสำหรับการแก้ปัญหา ดังนั้นโปรแกรมตัวอย่างจะใช้หน่วยความจำเพียง 12 ไบต์ ส่วนความเร็วในการประมวลผลของโครงสร้างข้อมูลทั้งสองแบบนี้ก็ไม่แตกต่างกัน

จากการที่โครงสร้างข้อมูลแบบเวกเตอร์ใช้หน่วยความจำคอมพิวเตอร์น้อยกว่าแบบต้นไม้ 7.5 เท่า และมีความเร็วในการประมวลผลไม่แตกต่างกันทำให้โครงสร้างข้อมูลแบบเวกเตอร์เหมาะสมสำหรับงานวิจัยนี้

ขั้นตอนการทำงานของกำหนดการเชิงพันธุกรรมที่ได้กล่าวมาทั้งหมดเป็นวิธีพื้นฐานที่ไม่มีวิธีเพิ่มความทนทานให้กับคำตอบ รายละเอียดต่างๆ และพารามิเตอร์ของกำหนดการเชิงพันธุกรรมที่ใช้สามารถสรุปได้ดังตารางที่ 4.1

ตารางที่ 4.1 รายละเอียดของกำหนดการเชิงพันธุกรรมที่ใช้ในการทดลอง

จำนวนประชากร	1,500 โปรแกรม
จำนวนรุ่น	ทำการทดลองย่อยในบทต่อไป เพื่อหาค่าที่เหมาะสม
ฟังก์ชัน	IF-AND, IF-OR, IF-NOT
เทอร์มินอล	FORWARD, LEFT,RIGTH,ISNEAR
ความสูงเริ่มต้นของโปรแกรม (ต้นไม้)	4
เกณฑ์สิ้นสุดการประมวลผลโปรแกรม เพื่อวัดค่าความเหมาะ	- หุ่นยนต์ถึงเป้าหมาย - จำนวนฟังก์ชัน และเทอร์มินอลครบตามจำนวนที่ต้องการ ซึ่งจะทำการทดลองย่อยในบทต่อไป เพื่อหาค่าที่เหมาะสม
ฟังก์ชันความเหมาะ	*F = (10,000xDistance)+Term* Distance = ระยะขจัดระหว่างหุ่นยนต์กับเป้าหมาย Term = จำนวนฟังก์ชัน และเทอร์มินอลที่ถูกปฏิบัติการ ค่าที่ได้จากฟังก์ชันความเหมาะ ค่ายิ่งน้อยยิ่งมีประสิทธิภาพในการแก้ปัญหา
อัตราการสืบพันธุ์	10% หรือ 150 โปแกรม
อัตราการไขว้เปลี่ยน	90% หรือ 1,350 โปรแกรม
อัตราการกลาย	0%
วิธีการสืบพันธุ์	คัดเลือกเอา 150 โปรแกรมแรกตามค่าความเหมาะไปไว้ในกลุ่มประชากรรุ่นใหม่โดยไม่มีการเปลี่ยนแปลงตัวโปรแกรม

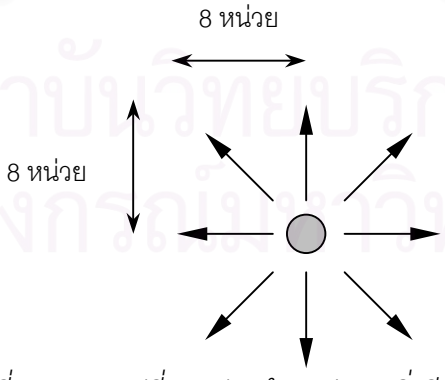
ตารางที่ 4.1 รายละเอียดของกำหนดการเชิงพันธุกรรมที่ใช้ในการทดลอง(ต่อ)

วิธีการไขว้เปลี่ยน	สุ่มเลือกสองโปรแกรมจาก 150 โปรแกรมที่ได้จากวิธีสืบพันธุ์ จากนั้นทำการไขว้เปลี่ยนโดยสองโปรแกรมใหม่ที่เป็นผลลัพธ์จะอยู่ในกลุ่มประชากรรุ่นใหม่ การไขว้เปลี่ยนจะวนทำจนได้โปรแกรมครบ 1,350 โปรแกรม
โครงสร้างข้อมูล	แบบเวกเตอร์

4.3 <u>การทดสอบความทนทานของคำตอบ</u>

ความทนทานในงานวิจัยนี้ หมายถึง การที่โปรแกรมคำตอบสามารถควบคุมหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายได้ในสภาพแวดล้อมที่ไม่ได้เรียนรู้มา เพื่อทดสอบความทนทานดังกล่าวของโปรแกรมคำตอบผู้วิจัยใช้วิธีทดสอบดังนี้

ผู้วิจัยได้สร้างสนามขึ้นมาใหม่โดยใส่สัญญาณรบกวนเข้าไปในสนามต้นแบบ เพื่อให้เกิดสนามใหม่ที่แตกต่างจากสนามเดิม สนามใหม่ที่ได้จะมีตำแหน่งของสิ่งกีดขวางแตกต่างไปจากสนามต้นแบบ กล่าวคือ สิ่งกีดขวางของสนามต้นแบบจะถูกสุ่มมาจำนวนหนึ่ง เพื่อเปลี่ยนแปลงตำแหน่งให้เลื่อนไปในแนวแกนนอน แนวแกนตั้ง หรือทั้งสองแนวเป็นระยะทาง 8 หน่วยจากตำแหน่งเดิม หรือประมาณ 40 % ของขนาดสิ่งกีดขวาง ซึ่งการเลื่อนตามแนวต่างๆ เกิดจากการสุ่มเช่นกัน การเปลี่ยนแปลงตำแหน่งสิ่งกีดขวางแสดงได้ดังรูปที่ 4.11



รูปที่ 4.11 การเปลี่ยนแปลงตำแหน่งของสิ่งกีดขวางทั้ง 8 ทิศ

สนามใหม่ที่ใช้ทดสอบความทนทานนี้มีจำนวน 10,000 สนาม แบ่งเป็น 10 กลุ่มๆ ละ 1,000 สนามตามเปอร์เซ็นต์สัญญาณรบกวนตั้งแต่ 10% 20% จนถึง100% ซึ่งเปอร์เซ็นต์สัญญาณรบกวนนี้หมายถึง จำนวนของสิ่งกีดขวางที่ถูกสุ่มให้เคลื่อนที่จากตำแหน่งเดิมเทียบกับจำนวนสิ่งกีดขวางทั้งหมด

การทดสอบความทนทานของการทดลอง จะนำโปรแกรมคำตอบที่ดีที่สุดที่แก้ปัญหาได้ในการทดลองมาทดสอบกับสนามทดสอบที่สร้างขึ้นใหม่แยกตามเปอร์เซ็นต์สัญญาณรบกวนของสนาม ซึ่งสามารถคำนวณได้ด้วยสมการนี้

$$\text{ความทนทาน} = \frac{\text{จำนวนสนามที่โปรแกรมควบคุมหุ่นยนต์ถึงเป้าหมาย}}{\text{จำนวนสนามทั้งหมด}} \times 100\% \qquad (2)$$

4.4 **สรุปท้ายบท**

บทนี้ได้อธิบายถึงเป้าหมายของปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย รายละเอียดของกำหนดการเชิงพันธุกรรมในการแก้ปัญหา ซึ่งเป็นวิธีพื้นฐานที่ยังไม่มีการเพิ่มความทนทานให้กับคำตอบ และอธิบายถึงวิธีวัดความทนทานของโปรแกรมคำตอบ รายละเอียดเหล่านี้จะเป็นพื้นฐานในการทดลองเพื่อเพิ่มความทนทานให้กับคำตอบในบทต่อไป

บทที่ 5

การทดลอง

เนื้อหาของบทนี้อธิบายวิธีการทดลองเพื่อเพิ่มความทนทานให้กับโปรแกรมคำตอบโดยใช้ การรังควาน การสุ่ม และการรังควานร่วมกับการสุ่ม จากนั้นอธิบายการเลือกค่าพารามิเตอร์ต่างๆที่ใช้ในการทดลอง พร้อมทั้งเสนอผลการทดสอบความความทนทานของโปรแกรมคำตอบที่ได้จากวิธีเพิ่มความทนทานทั้งสามวิธีเปรียบเทียบกัน และสรุปท้ายบท

5.1 วิธีเพิ่มความทนทาน

ในส่วนนี้อธิบายรายละเอียดการทำงานของวิธีเพิ่มความทนทานโดยแยกเป็นหัวข้อ การเพิ่มความทนทานโดยใช้การรังควาน การเพิ่มความทนทานโดยใช้การสุ่ม และเสนอวิธีเพิ่มความทนทานโดยใช้การรังควานร่วมกับการสุ่ม

5.1.1 การเพิ่มความทนทานโดยใช้การรังควาน

การใช้การรังควานเพื่อเพิ่มความทนทานให้กับคำตอบ เป็นวิธีที่นำมาจากงานวิจัยของ รุ่งโรจน์ นพสุวรรณชัย (2541) วิธีนี้เป็นการปรับปรุงกระบวนการวิวัฒนาการของกำหนดการเชิงพันธุกรรม จากเดิมที่เรียนรู้กับสภาพแวดล้อมเดียว ซึ่งมีลักษณะคงที่ (Static Environment) มาเป็นเรียนรู้กับสภาพแวดล้อมหลายๆ แบบที่แตกต่างกัน (Dynamic Environment) ผลที่ได้จากการปรับปรุงกระบวนการวิวัฒนาการส่งผลให้โปรแกรมคำตอบมีความยืดหยุ่นสูงสามารถปรับตัวเข้ากับสภาพแวดล้อมใหม่ที่ไม่ได้เรียนรู้มาได้ดี ทำให้โปรแกรมคำตอบมีความทนทาน

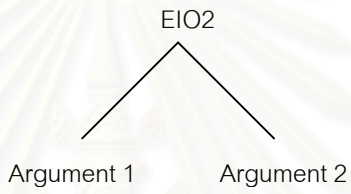
การปรับปรุงกระบวนการวิวัฒนาการเป็นการปรับปรุงขั้นตอนการวัดค่าความเหมาะของผลเฉลยที่ได้อธิบายไว้ในหัวข้อที่ 4.2.3 โดยการปรับปรุงนี้โปรแกรมทุกตัวในกลุ่มประชากรจะถูกประมวลผลกับสภาพแวดล้อมหลายๆแบบ ค่าความเหมาะที่ได้ของโปรแกรมแต่ละตัวจะเป็นค่าเฉลี่ยที่ได้จากทุกสภาพแวดล้อม สำหรับโปรแกรมคำตอบที่ได้รับการปรับปรุงกระบวนการวิวัฒนาการจะสามารถควบคุมหุ่นยนต์ไปถึงเป้าหมายได้ในทุกสภาพแวดล้อมที่เรียนรู้

สำหรับสภาพแวดล้อมเรียนรู้สร้างโดยใส่สัญญาณรบกวนเข้าไปในสภาพแวดล้อมต้นแบบ เพื่อให้ได้สภาพแวดล้อมใหม่ที่สิ่งกีดขวางเคลื่อนย้ายออกไปจากตำแหน่งเดิม เหมือนกับการสร้างสภาพแวดล้อมเพื่อใช้ทดสอบความทนทานในหัวข้อที่ 4.3

การเพิ่มความทนทานของโปรแกรมคำตอบด้วยวิธีนี้มีปัจจัยที่มีผลต่อความทนทานอยู่สองอย่าง คือ จำนวนสภาพแวดล้อม และเปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อม ซึ่งการหาค่าที่เหมาะสมของทั้งสองปัจจัยจะกล่าวในหัวข้อที่ 5.2.2

5.1.2 การเพิ่มความทนทานโดยใช้การสุ่ม

วิธีเพิ่มความทนทานโดยใช้การสุ่มนำมาจากงานวิจัยของ มาเรีย ประทีปทองคำ (2541) โดยมีแนวคิดว่าถ้าโปรแกรมคำตอบมีความหลากหลายของเส้นทางเดินมากขึ้นแล้วโปรแกรมคำตอบจะมีความทนทานเพิ่มขึ้น วิธีเพิ่มความทนทานนี้เป็นการปรับปรุงชุดฟังก์ชันที่ใช้ในการทดลอง โดยเพิ่มฟังก์ชันพิเศษ EIO2 ซึ่งมีลักษณะต้นไม้ดังรูปที่ 5.1



รูปที่ 5.1 ลักษณะต้นไม้ของฟังก์ชัน EIO2

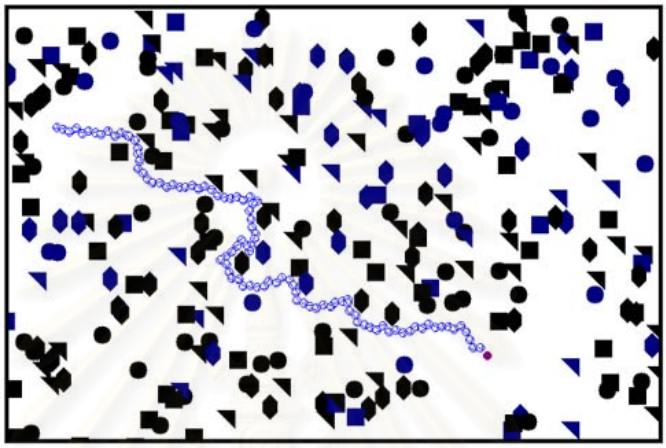
EIO2 เป็นฟังก์ชันที่มีอาร์กิวเมนต์สองตัว การประมวลผลของฟังก์ชันขึ้นอยู่กับการสุ่มตัวเลขศูนย์ หรือหนึ่งด้วยความน่าจะเป็นเท่าๆ กัน ถ้าเลขที่ได้จากการสุ่มมีค่าเป็น 0 อาร์กิวเมนต์ตัวแรกจะได้รับการประมวลผล แต่ถ้าเลขที่ได้จากการสุ่มมีค่าเป็น 1 อาร์กิวเมนต์ตัวที่สองจะได้รับการประมวลผลแทน ส่วนค่าส่งคืนของฟังก์ชัน EIO2 จะเป็นค่าส่งคืนที่ได้จากอาร์กิวเมนต์ที่หนึ่ง หรือสองขึ้นอยู่กับการประมวลผล

ฟังก์ชัน EIO2 แตกต่างจากฟังก์ชันพื้นฐาน IF-AND IF-OR และ IF-NOT เนื่องจากการประมวลผลขึ้นอยู่กับการสุ่มทำให้ลำดับการทำงานไม่แน่นอนในการประมวลผลแต่ละครั้ง ขณะที่ฟังก์ชันพื้นฐานทั้งสามฟังก์ชัน มีลำดับการประมวลผลที่แน่นอนในการประมวลผลแต่ละครั้ง ผลที่ได้จากฟังก์ชัน EIO2 ทำให้เกิดการเคลื่อนที่ของหุ่นยนต์ที่แตกต่างกันส่งผลให้มีความหลากหลายของเส้นทางในการเดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมาย

การทำงานของกำหนดการเชิงพันธุกรรมด้วยวิธีเพิ่มความทนทานนี้มีความแตกต่างจากกระบวนการทำงานแบบธรรมดา โดยการประมวลผลเพื่อหาค่าความเหมาะของโปรแกรมที่มีฟังก์ชัน EIO2 นั้นเมื่อประมวลผลสองครั้งมีโอกาสที่เส้นทางเดิน และค่าความเหมาะจะไม่เหมือนกัน ดังนั้นโปรแกรมหนึ่งโปรแกรมจึงมีการประมวลผลซ้ำในสภาพแวดล้อมเดิม 6 ครั้งด้วยเงื่อนไขเริ่มต้นเดิม เพื่อให้ได้ค่าที่แน่นอน ส่วนค่าความเหมาะของโปรแกรมที่เพิ่มความทนทานด้วยวิธีนี้จะมีค่าเท่ากับค่าเฉลี่ยของการประมวลผลทั้ง 6 ครั้ง

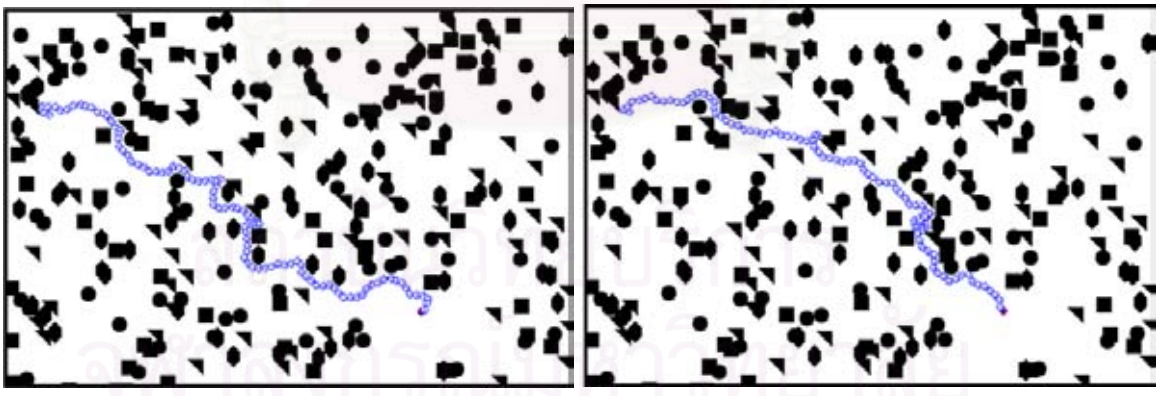
5.1.3 การเพิ่มความทนทานโดยใช้การรังควานร่วมกับการสุ่ม

การเพิ่มความทนทานให้กับโปรแกรมคำตอบของสองวิธีข้างต้นกล่าวได้ว่าเป็นการใส่ความไม่แน่นอนเข้าไปในกำหนดการเชิงพันธุกรรมเหมือนกัน แต่โปรแกรมคำตอบที่ได้กลับมีลักษณะแตกต่างกันกล่าวคือ โปรแกรมคำตอบที่ได้จากวิธีการรังควานจะมีรูปแบบทางเดินไปยังเป้าหมายที่แน่นอน รูปที่ 5.2 เป็นเส้นทางเดินของหุ่นยนต์ที่ได้จากวิธีการรังควาน



รูปที่ 5.2 เส้นทางเดินของหุ่นยนต์ที่ได้จากวิธีการรังควาน

ส่วนโปรแกรมคำตอบที่ได้จากวิธีการสุ่มจะมีเส้นทางเดินไปยังเป้าหมายได้หลายเส้นทางไม่แน่นอนดังรูปที่ 5.3



(ก) *(ข)*

รูปที่ 5.3 แสดงเส้นทางเดินที่ได้จากโปรแกรมเดียวกัน ซึ่งเพิ่มความทนทานด้วยวิธีการสุ่ม

จะเห็นได้ว่าโปรแกรมหุ่นยนต์ที่ได้จากการเพิ่มความทนทานทั้งสองวิธีมีข้อดีที่แตกต่างกัน ดังนั้นงานวิจัยนี้จึงรวมวิธีการรังควาน และการสุ่มเข้าด้วยกันเพื่อสร้างโปรแกรมหุ่นยนต์ให้มีความทนทานเพิ่มขึ้น

วิธีใหม่นี้จะใส่ฟังก์ชันพิเศษ EIO2 เข้าไปในชุดฟังก์ชัน เพื่อเป็นส่วนประกอบของโปรแกรมหุ่นยนต์ และปรับปรุงกระบวนการวิวัฒนาการของกำหนดการเชิงพันธุกรรมให้เรียนรู้กับสภาพแวดล้อมหลายๆแบบ ซึ่งผลจากการปรับปรุงความทนทานนี้ทำให้ในขั้นตอนการวัดค่าความเหมาะของผลเฉลยเปลี่ยนไป โดยโปรแกรมทุกตัวของกลุ่มประชากรจะประมวลผลกับสภาพแวดล้อมทุกสภาพแวดล้อม และประมวลผลซ้ำสภาพแวดล้อมละ 6 ครั้ง ดังนั้นค่าความเหมาะที่ได้ของโปรแกรมแต่ละโปรแกรมจะเป็นค่าเฉลี่ยของค่าความเหมาะในทุกการประมวลผล

5.2 <u>พารามิเตอร์สำหรับการทดลองร่วม</u>

หัวข้อนี้อธิบายการเลือกค่าพารามิเตอร์ที่ใช้ร่วมกันในการทดลอง เพื่อให้ได้ผลการทดลองที่มีประสิทธิภาพสามารถเปรียบเทียบความทนทานของโปรแกรมคำตอบทั้งสามวิธีได้อย่างเป็นกลาง หัวข้อนี้แบ่งย่อยออกเป็นพารามิเตอร์พื้นฐานของกำหนดการเชิงพันธุกรรม และพารามิเตอร์สำหรับการเรียนรู้ที่ใช้สภาพแวดล้อมหลายๆแบบ

5.2.1 **พารามิเตอร์พื้นฐานของกำหนดการเชิงพันธุกรรม**

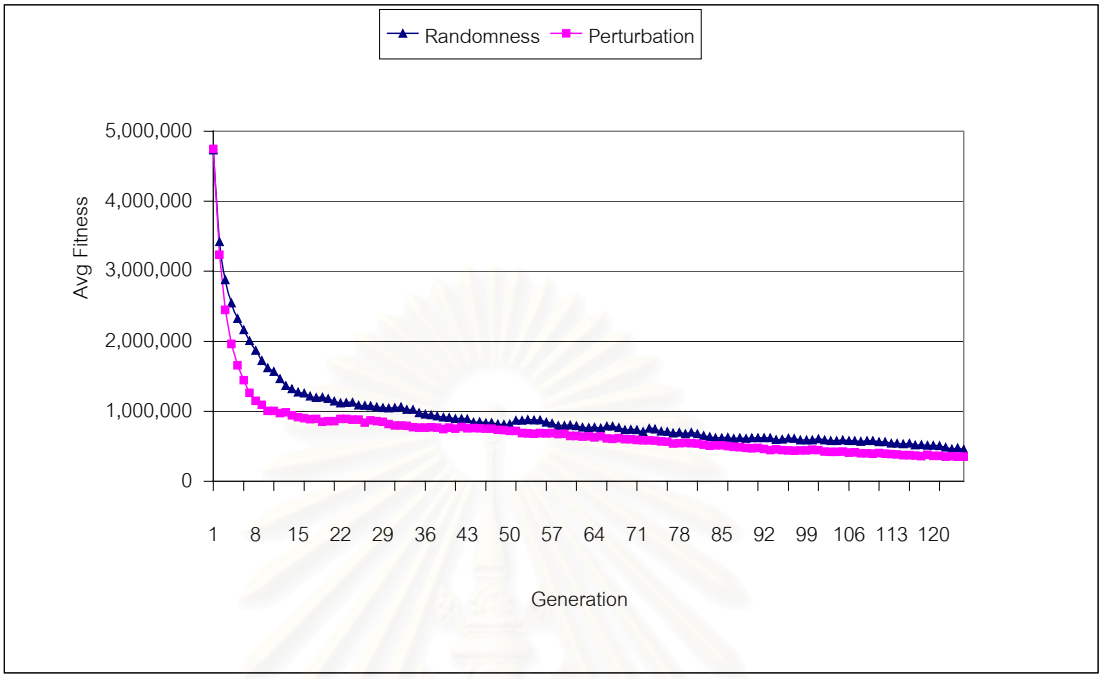
การทดลองนี้นำเอาวิธีเพิ่มความทนทานในงานวิจัยของ รุ่งโรจน์ นพสุวรรณชัย (2541) และ มาเรีย ประทีปทองคำ (2541) มารวมกัน เนื่องจากงานวิจัยทั้งสองต่างตั้งค่าพารามิเตอร์ของกำหนดการเชิงพันธุกรรมให้เหมาะสมกับวิธีของตัวเอง ทำให้ค่าพารามิเตอร์ของทั้งสองงานแตกต่างกัน งานวิจัยนี้จึงต้องหาค่าพารามิเตอร์ที่เหมาะสม เพื่อใช้กับกำหนดการเชิงพันธุกรรมทั้งแบบธรรมดา และแบบที่เพิ่มความทนทาน

จากที่ได้พิจารณาค่าพารามิเตอร์ในงานวิจัยทั้งสองปรากฏว่า จำนวนรุ่นในการทดลองและเงื่อนไขสิ้นสุดการประมวลผลโปรแกรมมีความแตกต่างกัน ซึ่งพารามิเตอร์ทั้งคู่นี้มีผลสำคัญต่อการทดลอง ผู้วิจัยจึงทำการทดลองย่อยเพื่อหาค่าที่เหมาะสมสำหรับพารามิเตอร์ทั้งสอง

5.2.1.1 **การหาจำนวนรุ่นในการทดลอง**

จุดประสงค์การทดลองย่อยนี้เพื่อปรับหาจำนวนรุ่นที่มากที่สุดในการทดลอง ซึ่งเป็นเงื่อนไขที่ทำให้กำหนดการเชิงพันธุกรรมจบการทำงาน การทดลองจะเปรียบเทียบค่าเฉลี่ยความเหมาะของกลุ่มประชากรที่ได้จากวิธีเพิ่มความทนทานโดยการรังควาน กับวิธีเพิ่มความทนทานโดยการสุ่ม ตั้งแต่รุ่นที่ 1 จนถึงรุ่นที่ 125 และเพื่อความแน่นอนกราฟหนึ่งเส้นจะทำการทดลองซ้ำ 20 ครั้ง

สำหรับการเพิ่มความทนทานโดยการรังควาน จะใช้จำนวนสภาพแวดล้อมในการเรียนรู้ 15 สภาพแวดล้อม ที่เปอร์เซ็นต์สัญญาณรบกวน 10% ผลการทดลองเป็นดังรูปที่ 5.4



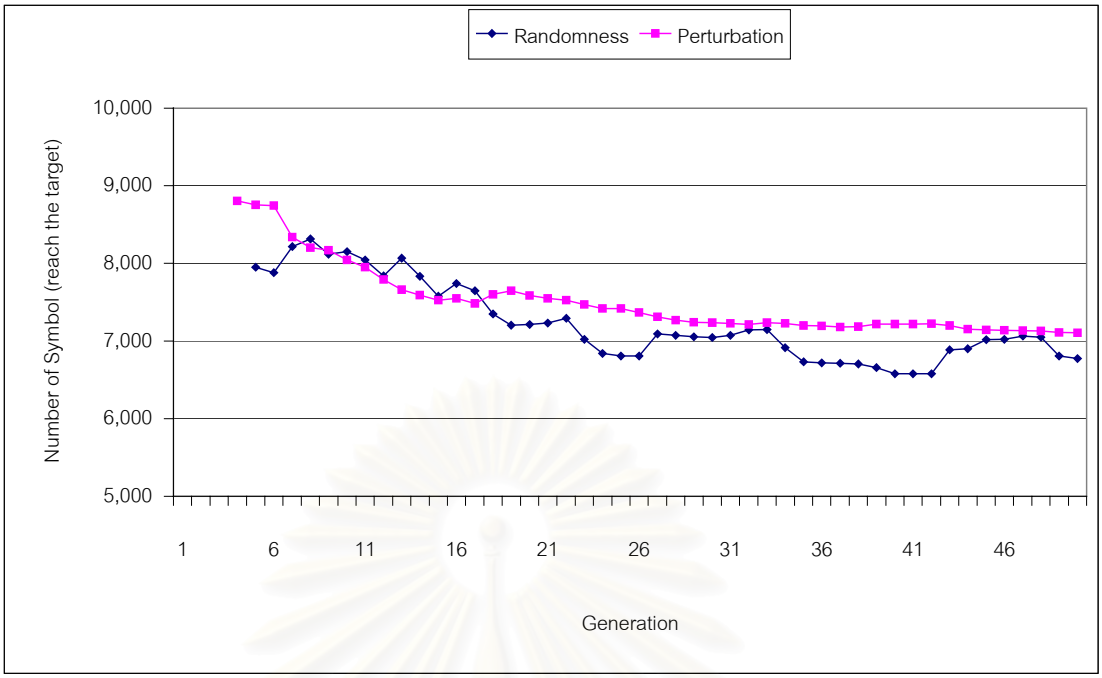
รูปที่ 5.4 กราฟแสดงค่าเฉลี่ยความเหมาะของวิธีเพิ่มความทนทานโดยการรังควาน และการสุ่ม

จากผลการทดลองแสดงให้เห็นว่าเส้นกราฟของทั้งสองวิธีลู่เข้าหากันอย่างเห็นได้ชัด ตั้งแต่รุ่นที่ 60 เป็นต้นไป และค่าเฉลี่ยความเหมาะดีขึ้นเรื่อยๆ เมื่อจำนวนรุ่นสูงขึ้น แต่ในช่วงท้ายค่าความเหมาะลดลงเพียงเล็กน้อย ดังนั้นการทดลองทั้งหมดในงานวิจัยกำหนดใช้จำนวนรุ่นที่ 100 เนื่องจากที่จำนวนรุ่น 100 ให้ค่าความเหมาะที่ดี และหลังจากรุ่นที่ 100 ไปแล้วค่าความเหมาะแตกต่างเพียงเล็กน้อย

5.2.1.2 การหาเงื่อนไขสิ้นสุดการประมวลผลโปรแกรม

เงื่อนไขสิ้นสุดการประมวลผลโปรแกรมในงานวิจัยนี้ใช้จำนวนฟังก์ชัน และเทอร์มินอลที่ถูกประมวลผลเป็นเกณฑ์ในการตัดสิน ดังนั้นเป้าหมายของการทดลองย่อยนี้ คือ การหาจำนวนฟังก์ชันและเทอร์มินอลดังกล่าว

การทดลองจะหาค่าเฉลี่ยจำนวนฟังก์ชัน และเทอร์มินอลที่ถูกประมวลผลทั้งวิธีการรังควาน และวิธีการสุ่ม โดยนับเฉพาะโปรแกรมที่ควบคุมหุ่นยนต์ไปถึงเป้าหมายเท่านั้น สำหรับจำนวนสภาพแวดล้อม และเปอร์เซ็นต์สัญญาณรบกวนของวิธีการรังควาน เลือกใช้ที่ 15 สภาพแวดล้อม และ 15% ตามลำดับ และแต่ละวิธีเพิ่มความทนทานทำการทดลองซ้ำ 10 ครั้ง



รูปที่ 5.5 กราฟแสดงค่าเฉลี่ยจำนวนฟังก์ชัน และเทอร์มินอล ของวิธีเพิ่มความทนทานทั้งสองวิธี

จากรูปที่ 5.5 จะเห็นได้ว่าในรุ่นแรกๆยังไม่มีโปรแกรมที่สามารถควบคุมหุ่นยนต์ไปถึงเป้าหมายได้สำเร็จ แต่หลังจากรุ่นที่ 4 และ 5 แล้วจะพบโปรแกรมที่สามารถควบคุมหุ่นยนต์ไปถึงเป้าหมายได้ โดยจำนวนฟังก์ชัน และเทอร์มินอลที่ถูกประมวลผลจะลดลงเมื่อจำนวนรุ่นมากขึ้น

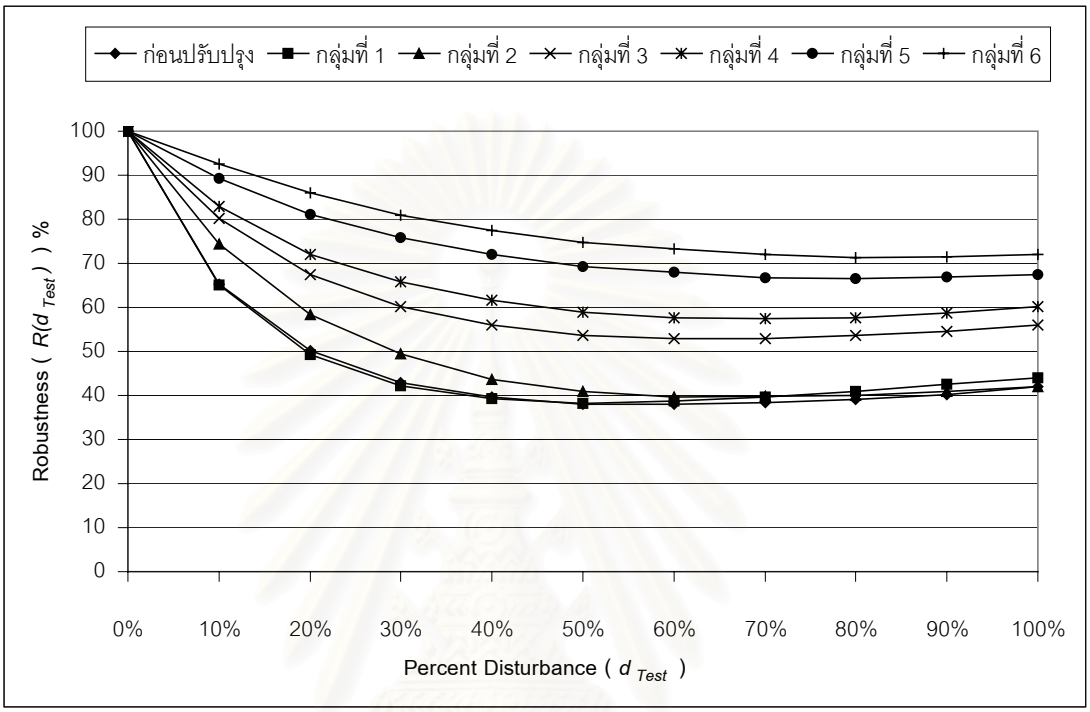
ผลการทดลองทำให้ผู้วิจัยกำหนดจำนวนฟังก์ชัน และเทอร์มินอลที่ถูกประมวลผลที่ 10,000 สำหรับทุกการทดลอง เนื่องจากต้องการให้ทุกการทดลองสามารถหาโปรแกรมคำตอบได้แน่นอน

5.2.2 พารามิเตอร์สำหรับการเรียนรู้ที่ใช้สภาพแวดล้อมหลาย ๆแบบ

จากงานวิจัยของรุ่งโรจน์ นพสุวรรณชัย (2541) พบว่าการปรับปรุงกระบวนการวิวัฒนาการโดยเรียนรู้จากสภาพแวดล้อมหลายๆแบบนั้น จำนวนสภาพแวดล้อมในการเรียนรู้ และเปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อม มีผลกับความทนทานของโปรแกรมคำตอบ ดังนั้นจะต้องมีการกำหนดจำนวนสภาพแวดล้อม และเปอร์เซ็นต์สัญญาณรบกวนที่เหมาะสม เพื่อให้โปรแกรมคำตอบที่ได้มีประสิทธิภาพ ซึ่งพารามิเตอร์เหล่านี้จะนำไปใช้กับวิธีเพิ่มความทนทานโดยการรังควาน และวิธีเพิ่มความทนทานโดยการรังควานร่วมกับการสุ่ม

5.2.2.1 การหาจำนวนสภาพแวดล้อมในการเรียนรู้

ในงานวิจัยของ รุ่งโรจน์ นพสุวรรณชัย (2541) ได้มีการทดลองเพื่อทดสอบความทนทานของโปรแกรมคำตอบ โดยตั้งสมมุติฐานว่าจำนวนสภาพแวดล้อมเรียนรู้มีผลต่อความทนทานหรือไม่ ผลการทดลองเป็นดังรูปที่ 5.6



รูปที่ 5.6 กราฟแสดงความทนทานของคำตอบ ในการทดลองเรียนรู้กับสภาพแวดล้อมกลุ่มต่างๆ จากงานวิจัยของ รุ่งโรจน์ นพสุวรรณชัย (2541)

จากรูปที่ 5.6 แกนตั้ง หมายถึง เปอร์เซ็นต์ความทนทาน แกนนอน หมายถึง เปอร์เซ็นต์สัญญาณรบกวนของสนามทดสอบ และกราฟทั้ง 6 เส้น หมายถึง

- ก่อนการปรับปรุง เรียนรู้จากสภาพแวดล้อมต้นแบบ 1 สภาพแวดล้อม
- กลุ่มที่ 1 เรียนรู้จากสภาพแวดล้อม 5 สภาพแวดล้อม
- กลุ่มที่ 2 เรียนรู้จากสภาพแวดล้อม 10 สภาพแวดล้อม
- กลุ่มที่ 3 เรียนรู้จากสภาพแวดล้อม 15 สภาพแวดล้อม
- กลุ่มที่ 4 เรียนรู้จากสภาพแวดล้อม 20 สภาพแวดล้อม
- กลุ่มที่ 5 เรียนรู้จากสภาพแวดล้อม 30 สภาพแวดล้อม
- กลุ่มที่ 6 เรียนรู้จากสภาพแวดล้อม 50 สภาพแวดล้อม

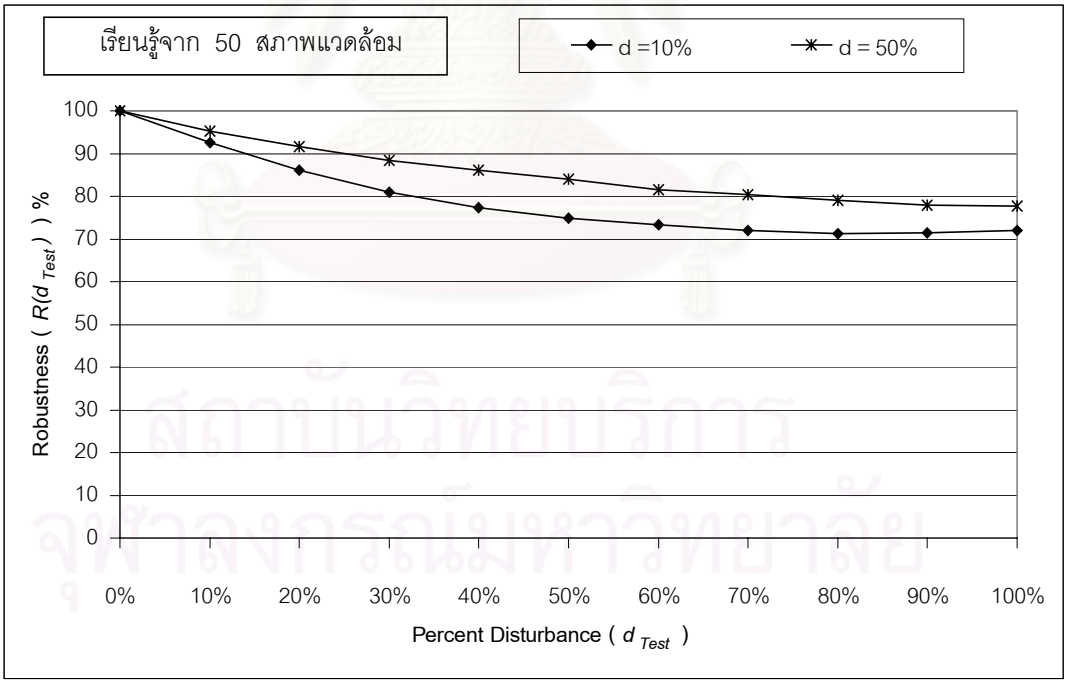
หมายเหตุ กลุ่มที่ 1-6 จะมีสภาพแวดล้อมต้นแบบรวมอยู่ด้วย 1 สภาพแวดล้อม และทุกกลุ่มมีเปอร์เซ็นต์สัญญาณรบกวนสภาพแวดล้อมที่ 10%

ผลการทดลองในรูปที่ 5.6 แสดงให้เห็นว่ายิ่งเพิ่มสภาพแวดล้อมเรียนรู้โปรแกรมคำตอบยิ่งมีความทนทาน โดยกลุ่มที่ 6 มีความทนทานมากที่สุด แต่จะสังเกตได้ว่าการเพิ่มจำนวนสภาพแวดล้อมจะปรับเพิ่มขึ้นจาก 5 สภาพแวดล้อมในกลุ่มที่ 2 3 4 เป็น 10 สภาพแวดล้อมในกลุ่มที่ 5 และ 20 สภาพแวดล้อมในกลุ่มที่ 6 เพื่อให้ได้ค่าความทนทานเพิ่มขึ้น ชี้ให้เห็นว่าความทนทานมีจุดอิ่มตัว เมื่อไปถึงจุดหนึ่งการเพิ่มสภาพแวดล้อมในการเรียนรู้ย่อมไม่มีผลต่อความทนทาน

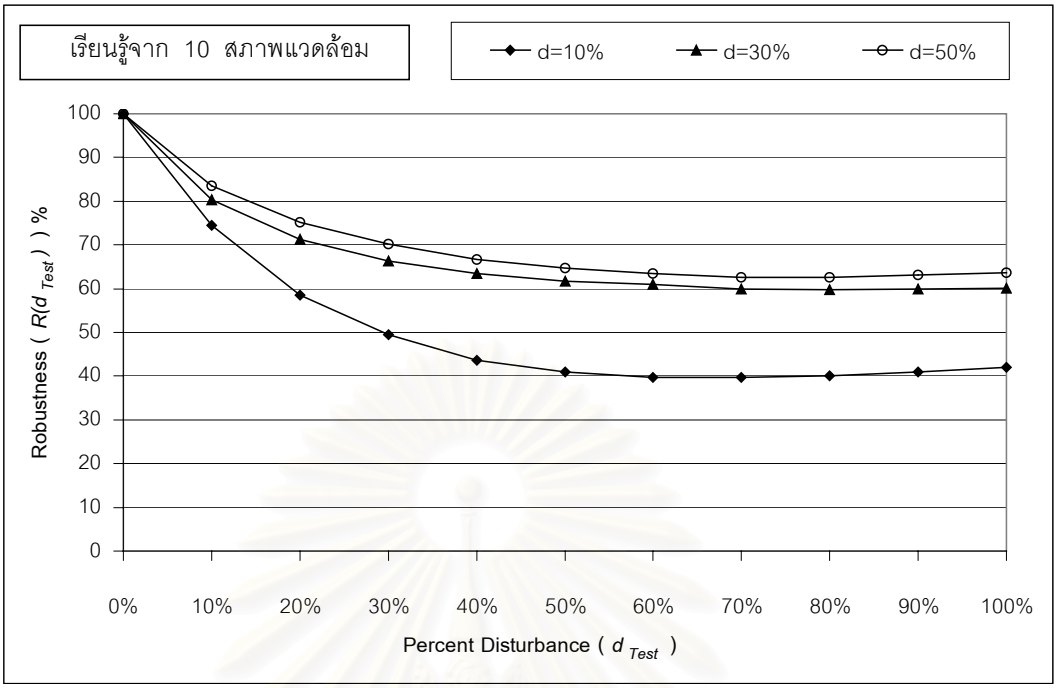
เนื่องจากการเรียนรู้จากสภาพแวดล้อม 50 สภาพแวดล้อมนั้นส่งผลต่อความทนทานเพียงเล็กน้อยเมื่อเทียบกับ 30 สภาพแวดล้อม แต่ต้องประมวลผลเพิ่มขึ้นถึง 20 สภาพแวดล้อม ซึ่งมีผลทำให้การประมวลผลล่าช้า ดังนั้นเพื่อประสิทธิภาพที่ดีทั้งในแง่ของความทนทาน และความเร็วในการประมวลผล การทดลองนี้จึงกำหนดจำนวนสภาพแวดล้อมในการเรียนรู้ที่ 30 สภาพแวดล้อม

5.2.2.2 การหาเปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อม

สำหรับการกำหนดค่าเปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อมที่ใช้เรียนรู้ พบว่าในงานวิจัยของ รุ่งโรจน์ นพสุวรรณชัย (2541) ได้มีการทดลองเพื่อทดสอบความทนทานของโปรแกรมคำตอบ โดยการเรียนกับกลุ่มสภาพแวดล้อมที่มีเปอร์เซ็นต์สัญญาณรบกวนแตกต่างกัน ผลการทดลองเป็นดังรูปที่ 5.7 และ 5.8



รูปที่ 5.7 กราฟแสดงความทนทานของคำตอบ ในการเรียนรู้กับสภาพแวดล้อม 50 สภาพแวดล้อมที่เปอร์เซ็นต์สัญญาณรบกวน 10% และ50%



รูปที่ 5.8 กราฟแสดงความทนทานของคำตอบ ในการเรียนรู้กับสภาพแวดล้อม 10 สภาพแวดล้อมที่เปอร์เซ็นต์สัญญาณรบกวน 10% 30 % และ50%

ในรูปที่ 5.7 และ 5.8 แกนตั้ง คือ ค่าความทนทานเป็นเปอร์เซ็นต์ แกนนอน คือ เปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อมที่ใช้ทดสอบ ค่า d=10% 30% และ50% คือ เปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อมที่ใช้เรียนรู้ รูปที่ 5.7 และ 5.8 แตกต่างกันที่จำนวนสภาพแวดล้อมที่ใช้เรียนรู้ โดยในรูปที่ 5.7 ใช้ 50 สภาพแวดล้อม ส่วนรูปที่ 5.8 ใช้ 10 สภาพแวดล้อม

ผลการทดลองจากทั้งสองรูปแสดงให้เห็นว่า เปอร์เซ็นต์สัญญาณรบกวนมีผลต่อความทนทาน และจำนวนสภาพแวดล้อมที่ใช้เรียนรู้ก็มีผลกระทบต่อเปอร์เซ็นต์สัญญาณรบกวนที่ทำให้เกิดความทนทานด้วย โดยในรูปที่ 5.7 จะเห็นได้ว่าการเรียนรู้จาก 50 สภาพแวดล้อมนั้น ทำให้ค่าความทนทานที่ได้จากการเรียนรู้กับสภาพแวดล้อมที่เปอร์เซ็นต์สัญญาณรบกวน 10% และ 50% แตกต่างกันไม่มากนัก ในขณะที่รูปที่ 5.8 เรียนรู้จาก 10 สภาพแวดล้อมค่าความทนทานที่ได้จากเปอร์เซ็นต์สัญญาณรบกวน 10% และ 50% มีความแตกต่างกันมาก ส่วนที่เปอร์เซ็นต์สัญญาณรบกวน 30% จะมีค่าความทนทานใกล้เคียงกับเปอร์เซ็นต์สัญญาณรบกวน 50%

จากการวิเคราะห์กราฟทั้งสองรูป ผู้วิจัยได้เลือกเปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อมเรียนรู้ที่ 30% เนื่องจากค่าความทนทานที่ได้จากเปอร์เซ็นต์สัญญาณรบกวนที่ 30% และ50% แตกต่างกันเพียงเล็กน้อย อีกทั้งการเลือกจำนวนสภาพแวดล้อมเรียนรู้ที่ 30 สภาพแวดล้อมจากหัวข้อ 5.2.2.1 มีผลทำให้ค่าความทนทานที่ได้จากเปอร์เซ็นต์สัญญาณรบกวนที่ 30% และ 50% ไม่แตกต่างกัน

5.3 ผลการทดลอง

จากที่ได้อธิบายวิธีเพิ่มความทนทานแต่ละวิธี และกำหนดค่าพารามิเตอร์สำหรับการทดลองร่วมกันแล้ว หัวข้อนี้เสนอผลการทดสอบความทนทานของโปรแกรมคำตอบที่ได้จากวิธีเพิ่มความทนทานทั้งสามวิธีเทียบกับวิธีธรรมดาที่ไม่ได้เพิ่มความทนทาน

สำหรับวิธีเพิ่มความทนทานทั้งสามวิธีนั้น สามารถสรุปวิธีการได้ดังนี้

- การรังควาน เรียนรู้กับสภาพแวดล้อม 30 สภาพแวดล้อมที่เปอร์เซ็นต์สัญญาณรบกวน 30%
- การสุ่ม เพิ่มฟังก์ชันพิเศษ EIO2
- การรังควานร่วมกับการสุ่ม เรียนรู้กับสภาพแวดล้อม 30 สภาพแวดล้อมที่เปอร์เซ็นต์สัญญาณรบกวน 30% และเพิ่มฟังก์ชันพิเศษ EIO2

ตารางที่ 5.1 แสดงผลการวัดความทนทาน ซึ่งการทดลองแต่ละวิธีทำซ้ำ 20 ครั้ง ค่าความทนทานที่ได้จะแยกตามเปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อมทดสอบ และค่าจากตารางนี้สามารถแสดงเป็นกราฟได้ดังรูปที่ 5.9

ตารางที่ 5.1 แสดงค่าความทนทานของโปรแกรมคำตอบที่ได้แต่ละวิธี

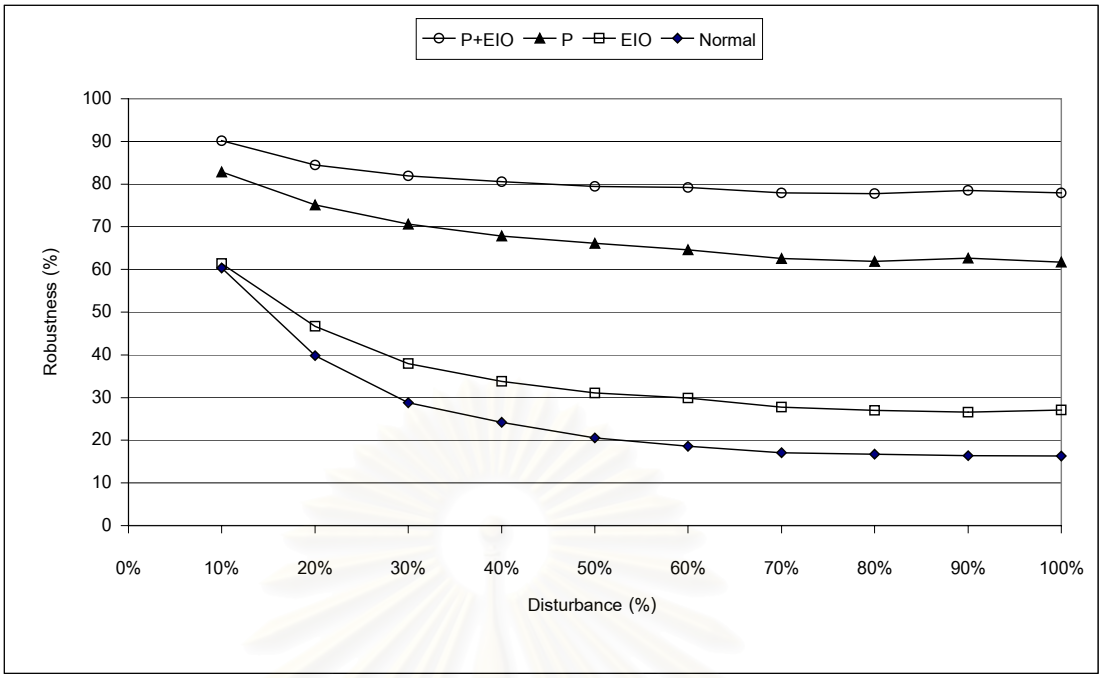
เปอร์เซ็นต์สัญญาณรบกวน	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Normal	60.32	39.81	28.80	24.19	20.58	18.56	17.05	16.75	16.35	16.29
EIO	61.39	46.69	37.99	33.80	31.06	29.85	27.75	26.99	26.58	27.06
P	82.88	75.10	70.61	67.83	66.16	64.57	62.60	61.92	62.65	61.73
P+EIO	90.16	84.46	81.95	80.57	79.46	79.22	77.91	77.77	78.50	77.93

หมายเหตุ Normal หมายถึง วิธีธรรมดา

EIO หมายถึง วิธีการสุ่ม

P หมายถึง วิธีการรังควาน

P+EIO หมายถึง วิธีการรังควานร่วมกับการสุ่ม



รูปที่ 5.9 กราฟแสดงความทนทานของโปรแกรมคำตอบ

ผลการทดลองแสดงให้เห็นว่าวิธีการรังควานร่วมกับการสุ่ม ทำให้โปรแกรมคำตอบมีความทนทานสูงสุด เมื่อเทียบกับวิธีการรังควาน หรือการสุ่มเพียงอย่างใดอย่างหนึ่ง โดยเมื่อนำวิธีทั้งสามเทียบกับวิธีธรรมดาแล้ว วิธีการสุ่มมีความทนทานกว่าวิธีธรรมดาประมาณ 10% ส่วนวิธีการรังควานมีความทนทานมากกว่าวิธีธรรมดาประมาณ 40% และวิธีการรังควานร่วมกับการสุ่มมีความทนทานสูงกว่าวิธีธรรมดาถึง 60% นอกจากนี้ยังเห็นได้ว่าเมื่อเปอร์เซ็นต์สัญญาณรบกวนสภาพแวดล้อมเพิ่มสูงขึ้นค่าความทนทานจะลดลง และค่าความทนทานของแต่ละวิธีจะยิ่งแตกต่างกัน

5.4 **สรุปท้ายบท**

บทนี้ได้อธิบายรายละเอียดวิธีเพิ่มความทนทานโดยใช้การรังควาน และการสุ่ม จากนั้นได้เสนอการรวมวิธีการรังควานกับการสุ่มเข้าด้วยกัน ซึ่งจากการทดสอบโปรแกรมคำตอบแต่ละวิธี แสดงให้เห็นว่า วิธีการรังควานร่วมกับการสุ่มที่เสนอทำให้โปรแกรมคำตอบที่ได้มีความทนทานมากกว่าการใช้รังควาน หรือการสุ่มเพียงอย่างเดียว โดยในบทต่อไปจะเป็นการหาสาเหตุที่ทำให้โปรแกรมคำตอบมีความทนทานเพิ่มขึ้น

บทที่ 6

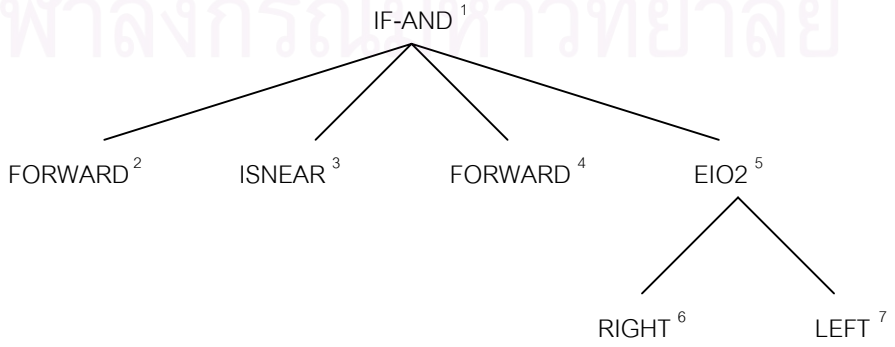
การวิเคราะห์ความทนทาน

ความทนทานเป็นคุณสมบัติสำคัญของโปรแกรมคำตอบที่ได้จากกำหนดการเชิงพันธุกรรม เพื่อค้นหาสาเหตุที่ทำให้โปรแกรมคำตอบเกิดความทนทาน บทนี้ได้รวบรวมข้อมูลที่ได้จากการทดลองในบทที่ 5 มาทำการวิเคราะห์โดยใช้ “เส้นทางการปฏิบัติงาน” และ "ประสบการณ์" ซึ่งเป็นแนวคิดจากงานวิจัยของ รุ่งโรจน์ นพสุวรรณชัย (2541) เป็นปัจจัยในการอธิบายความทนทาน

6.1 <u>เส้นทางปฏิบัติงาน</u>

งานวิจัยนี้ได้เปลี่ยนโครงสร้างข้อมูลสำหรับโปรแกรมควบคุมหุ่นยนต์ จากเดิมโครงสร้างแบบต้นไม้มาเป็นโครงสร้างแบบเวกเตอร์ โดยโครงสร้างแบบเวกเตอร์นี้เมื่อนำไปประมวลผล การประมวลผลจะเหมือนกับโครงสร้างแบบต้นไม้ กล่าวคือ โครงสร้างแบบเวกเตอร์จะจำลองตัวเองเหมือนโครงสร้างแบบต้นไม้ และเริ่มการประมวลผลตั้งแต่รากลงไปยังโหนดล่างสุดของต้นไม้ จากนั้นจะวนกลับขึ้นมาประมวลผลที่รากลงไปยังโหนดล่างสุดของต้นไม้อีก ขั้นตอนการประมวลผลนี้จะวนซ้ำจนกระทั่งพบเงื่อนไขที่ทำให้จบการประมวลผล ซึ่งในงานวิจัยนี้ คือ การประมวลผลฟังก์ชัน และเทอร์มินอลครบ 10,000 คำสั่ง หรือหุ่นยนต์เคลื่อนที่ถึงเป้าหมาย

ในการประมวลผลจากรากมายังโหนดล่างสุดของต้นไม้แต่ละครั้ง จะได้ข้อมูลที่เป็นลำดับพรีออเดอร์ (Preorder) ของฟังก์ชัน และเทอร์มินอลที่ถูกปฏิบัติงาน หรือเรียกอีกอย่างว่าเส้นทางปฏิบัติงาน (Trace) โดยโปรแกรมหนึ่งโปรแกรม เมื่อนำไปประมวลผลจะมีเส้นทางปฏิบัติงานได้หลายเส้นทางขึ้นอยู่กับว่าขณะนั้นสถานการณ์หุ่นยนต์ในสภาพแวดล้อมที่ประมวลผลเป็นอย่างไร และเมื่อโปรแกรมสิ้นสุดการประมวลผลจะได้เส้นทางปฏิบัติงานหลายๆเส้นทางซึ่งอาจซ้ำกันได้ เพื่อความเข้าใจเกี่ยวกับเส้นทางปฏิบัติงานมากขึ้น ผู้วิจัยขอยกตัวอย่างโปรแกรมดังรูปที่ 6.1



รูปที่ 6.1 ตัวอย่างโปรแกรมหุ่นยนต์

โปรแกรมในรูปที่ 6.1 นี้เมื่อนำไปประมวลผลจะมีเส้นทางปฏิบัติงานได้หลากหลายเส้นทาง ยกตัวอย่างเช่น ในกรณีที่หุ่นยนต์อยู่หน้าสิ่งกีดขวาง การปฏิบัติงานเทอร์มินอล FORWARD[2] จะส่งค่าคืนเป็นเท็จ เนื่องจากไม่สามารถเดินหน้าได้ ส่วนเทอร์มินอล ISNEAR[3] เมื่อเดินหน้าไม่ได้ก็ไม่เข้าใกล้เป้าหมายขึ้นจึงส่งค่าคืนเป็นเท็จ ดังนั้นเมื่อฟังก์ชัน IF-AND[1] รับค่าส่งคืนจากทั้งสองเทอร์มินอลมาดำเนินการด้วยตัวตรรกะ AND ฟังก์ชัน EIO[5] ก็จะได้รับการปฏิบัติงาน ส่วนเทอร์มินอล RIGHT[6] หรือ LEFT[7] ตัวใดจะถูกปฏิบัติงานก็ขึ้นกับค่าที่สุ่มได้ในขณะนั้นจากการประมวลผลนี้ จะเห็นได้ว่ามีเส้นทางปฏิบัติงานอยู่สองเส้นทาง คือ

ก) { IF-AND[1], FORWARD[2], ISNEAR[3], EIO[5], RIGHT[6] }

ข) { IF-AND[1], FORWARD[2], ISNEAR[3], EIO[5], LEFT[7] }

สำหรับกรณีที่ไม่มีสิ่งกีดขวางอยู่หน้าหุ่นยนต์การปฏิบัติงานของเทอร์มินอล FORWARD[2] จะได้ค่าส่งคืนเป็นจริง และถ้าการเดินหน้าครั้งนี้เข้าใกล้เป้าหมายมากขึ้นการปฏิบัติงานของเทอร์มินอล ISNEAR[3] ก็จะได้ค่าส่งคืนเป็นจริง เมื่อนำค่าส่งคืนที่ได้จากทั้งสองเทอร์มินอลมาดำเนินการด้วยตัวตรรกะ AND แล้วฟังก์ชัน FORWARD[4] จะได้รับการปฏิบัติงาน จากการประมวลผลจะได้เส้นทางปฏิบัติงานใหม่อีกหนึ่งเส้นทางคือ

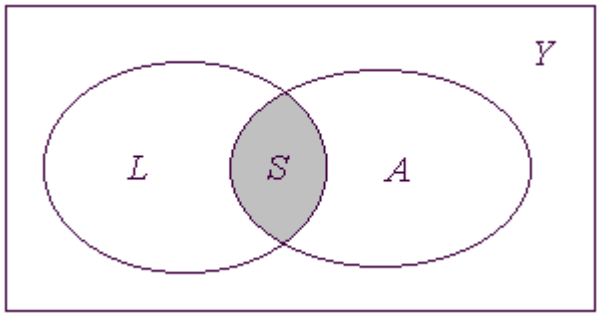
ค) { IF-AND[1], FORWARD[2], ISNEAR[3], FORWARD[4] }

สำหรับกรณีที่ปฏิบัติงานเทอร์มินอล FORWARD[2] แล้วเดินออกห่างเป้าหมาย ค่าส่งคืนจากเทอร์มินอล ISNEAR[3] ก็จะเป็นเท็จทำให้เส้นทางปฏิบัติงานที่ได้เป็น ก) หรือ ข) เหมือนกรณีข้างบน

จะเห็นได้ว่าโปรแกรมนี้มีเส้นทางปฏิบัติงานที่เป็นไปได้ทั้งหมดสามเส้นทาง การประมวลผลจากรากมายังโหนดล่างสุดของต้นไม้แต่ละครั้งจะได้เส้นทางปฏิบัติงานที่แตกต่างกัน หรือเหมือนกัน ขึ้นอยู่กับสถานการณ์ในขณะประมวลผล ดังนั้นโปรแกรมหนึ่งโปรแกรมเมื่อประมวลผลเสร็จเรียบร้อยอาจได้เส้นทางปฏิบัติงานที่เป็นไปได้ทั้งหมดของโปรแกรม หรืออาจได้เส้นทางปฏิบัติงานเพียงส่วนหนึ่ง ซึ่งเส้นทางเหล่านี้จะนำไปวิเคราะห์ความทนทานต่อไป

6.2 <u>วิธีวิเคราะห์ความทนทาน</u>

เพื่อหาสาเหตุที่ทำให้โปรแกรมเกิดความทนทานการวิเคราะห์นี้ใช้แนวคิดที่ว่า ถ้าโปรแกรมคำตอบสามารถนำ “ประสบการณ์“ กลับมาใช้ได้มากเท่าไรโปรแกรมคำตอบนั้นยิ่งมีความทนทาน โดยการวัดประสบการณ์ที่นำกลับมาใช้นี้สามารถวาดเป็นแผนภาพเวนน์-ออยเลอร์ได้ดังรูปที่ 6.2



รูปที่ 6.2 แสดงประสบการณ์ของโปรแกรมคำตอบ

รูปที่ 6.2 นี้ เซต Y หมายถึง เส้นทางปฏิบัติงานทั้งหมดที่เป็นไปได้ของโปรแกรมคำตอบ เซต L หมายถึง เส้นทางปฏิบัติงานของโปรแกรมคำตอบในขณะเรียนรู้ หรือเรียกว่า “ประสบการณ์” เซต A หมายถึง เส้นทางปฏิบัติงานของโปรแกรมคำตอบในขณะทดสอบ และส่วนที่ซ้อนทับกัน (Intersect) S เป็นส่วนที่บอกถึงการนำประสบการณ์ที่ได้เรียนรู้มาใช้เมื่อประมวลผลกับสภาพแวดล้อมทดสอบ ซึ่งยิ่งมีส่วนนี้มากเท่าไรโปรแกรมคำตอบยิ่งมีความทนทาน

สำหรับวิธีวัดค่า S ทำโดยนำโปรแกรมคำตอบที่ได้ไปประมวลผลกับสภาพแวดล้อมที่เรียนรู้มาจากกำหนดการเชิงพันธุกรรม เพื่อเก็บเส้นทางปฏิบัติงาน (เซต L) โดยเส้นทางปฏิบัติงานที่ซ้ำจะไม่เก็บอีก ขั้นตอนนี้แบ่งออกเป็น 4 วิธีตามวิธีเพิ่มความทนทาน

- วิธีธรรมดา ประมวลผลครั้งเดียวกับสภาพแวดล้อมที่เรียนรู้ 1 สภาพแวดล้อม
- วิธีการสุ่ม ประมวลผลซ้ำ 6 ครั้งกับสภาพแวดล้อมที่เรียนรู้ 1 สภาพแวดล้อม
- วิธีการรังควาน ประมวลผลกับสภาพแวดล้อมที่เรียนรู้ 30 สภาพแวดล้อม
- วิธีการรังควานร่วมกับการสุ่ม ประมวลผลซ้ำ 6 ครั้งกับสภาพแวดล้อมที่เรียนรู้ 30 สภาพแวดล้อม

จากนั้นนำโปรแกรมคำตอบเดิมไปประมวลผลกับสภาพแวดล้อมทดสอบ เพื่อเก็บเส้นทางปฏิบัติงาน (เซต A) หาเส้นทางปฏิบัติงานที่เหมือนกัน และคำนวณด้วยสมการ

$$S = \frac{|L \cap A|}{|A|} \times 100\ \% \quad \text{.....(1)}$$

สมการนี้เป็นการวัดค่า S กับสภาพแวดล้อมทดสอบเพียงสภาพแวดล้อมเดียว แต่เนื่องจากสภาพแวดล้อมทดสอบถูกแบ่งออกเป็นกลุ่มตามเปอร์เซ็นต์สัญญาณรบกวน ดังนั้นค่า S จะวัดค่าตามกลุ่มเปอร์เซ็นต์สัญญาณรบกวนด้วยสมการ

$$S = \frac{\sum_{i}^{N} \frac{|L \cap Ai|}{|Ai|}}{N} \times 100\% \quad \text{.....(2)}$$

โดยที่ N หมายถึง จำนวนสภาพแวดล้อมทดสอบ

ค่า S ที่ได้ยังแบ่งออกเป็น 3 ประเภทตามจำนวนสภาพแวดล้อมทดสอบ

- *Sall* เป็นการคำนวณกับภาพแวดล้อมทดสอบทั้งหมดในกลุ่มเปอร์เซ็นต์สัญญาณรบกวน โดย N มีค่าเท่ากับ 1,000
- *Ssucc* เป็นการคำนวณกับสภาพแวดล้อมทดสอบที่หุ่นยนต์ถึงเป้าหมายเท่านั้น โดย N มีค่าเท่ากับจำนวนสภาพแวดล้อมทดสอบที่หุ่นยนต์ถึงเป้าหมาย
- *Sfail* เป็นการคำนวณกับสภาพแวดล้อมทดสอบที่หุ่นยนต์ไม่ถึงเป้าหมายเท่านั้น โดย N มีค่าเท่ากับจำนวนสภาพแวดล้อมทดสอบที่หุ่นยนต์ไม่ถึงเป้าหมาย

6.3 ผลการวิเคราะห์ความทนทาน

ผลการวิเคราะห์ความทนทานของโปรแกรมคำตอบที่ได้จากวิธีเพิ่มความทนทานแต่ละวิธีได้ผลดังตารางที่ 6.1 โดยค่า *Sall* *Ssucc* และ *Sfail* ในตารางจะเป็นค่าเฉลี่ยที่ได้จากการประมวลผลโปรแกรมคำตอบ 20 โปรแกรมในแต่ละวิธี และเนื่องจากการวิเคราะห์ความทนทานนี้สนใจค่า *Sall* เป็นพิเศษจึงนำค่า *Sall* มาสร้างเป็นแผนภูมิแท่งเพื่อสังเกตได้ง่ายดังรูปที่ 6.3

สำหรับวิธีเพิ่มความทนทานแต่ละวิธีจะใช้อักษรย่อแทนดังนี้

Normal หมายถึง วิธีธรรมดา

EIO หมายถึง วิธีการสุ่ม

P หมายถึง วิธีการรังควาน

P+EIO หมายถึง วิธีการรังควานร่วมกับการสุ่ม

ตารางที่ 6.1 (ก) และ (ข) แสดงผลการวิเคราะห์ความทนทานของโปรแกรมคำตอบแต่ละวิธี แยกตามเปอร์เซ็นต์สัญญาณรบกวน

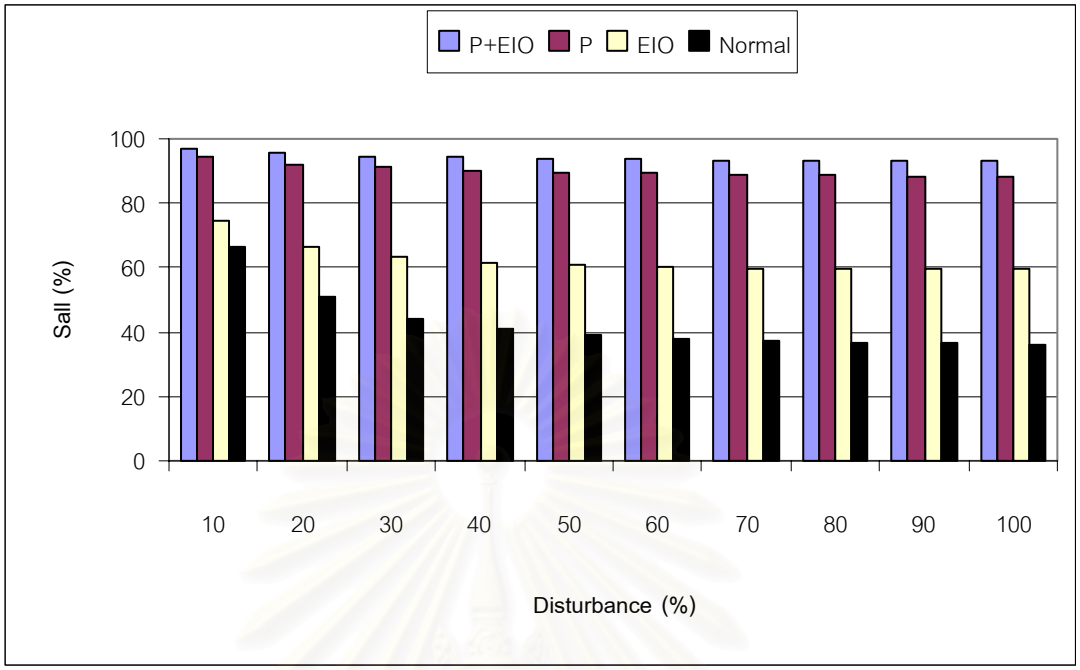
	Dtest = 10 %			Dtest = 20 %			Dtest = 30 %			Dtest = 40 %			Dtest = 50 %		
	Sall	*Ssucc*	*Sfail*	*Sall*	*Ssucc*	*Sfail*	*Sall*	*Ssucc*	*Sfail*	*Sall*	*Ssucc*	*Sfail*	*Sall*	*Ssucc*	*Sfail*
Normal	66.52	84.16	40.05	51.20	71.23	38.95	43.93	60.35	37.76	40.73	53.14	37.04	39.02	49.01	36.50
EIO	74.54	83.49	61.66	66.72	75.52	60.50	63.20	69.97	59.82	61.72	67.09	59.31	60.80	65.24	58.94
P	94.49	96.18	85.01	92.15	94.35	84.61	91.00	93.39	84.52	90.20	92.68	84.26	89.67	92.19	84.26
P+EIO	96.91	97.95	87.22	95.38	96.83	87.19	94.65	96.18	87.42	94.21	95.77	87.50	93.88	95.45	87.56

(ก)

	Dtest = 60 %			Dtest = 70 %			Dtest = 80 %			Dtest = 90 %			Dtest = 100 %		
	Sall	*Ssucc*	*Sfail*	*Sall*	*Ssucc*	*Sfail*	*Sall*	*Ssucc*	*Sfail*	*Sall*	*Ssucc*	*Sfail*	*Sall*	*Ssucc*	*Sfail*
Normal	37.85	45.60	35.96	37.15	44.10	35.56	36.78	42.73	35.33	36.47	42.17	35.10	36.05	41.20	34.73
EIO	60.31	64.30	58.65	59.93	63.89	58.44	59.67	63.09	58.30	59.94	63.53	58.03	59.32	63.14	57.96
P	89.30	91.82	84.17	88.86	91.50	83.99	88.55	91.29	83.64	88.44	91.04	83.60	88.27	90.86	83.64
P+EIO	93.67	95.21	87.58	93.42	94.96	87.57	93.29	94.87	87.45	93.32	94.82	87.52	93.16	94.64	87.53

(ข)

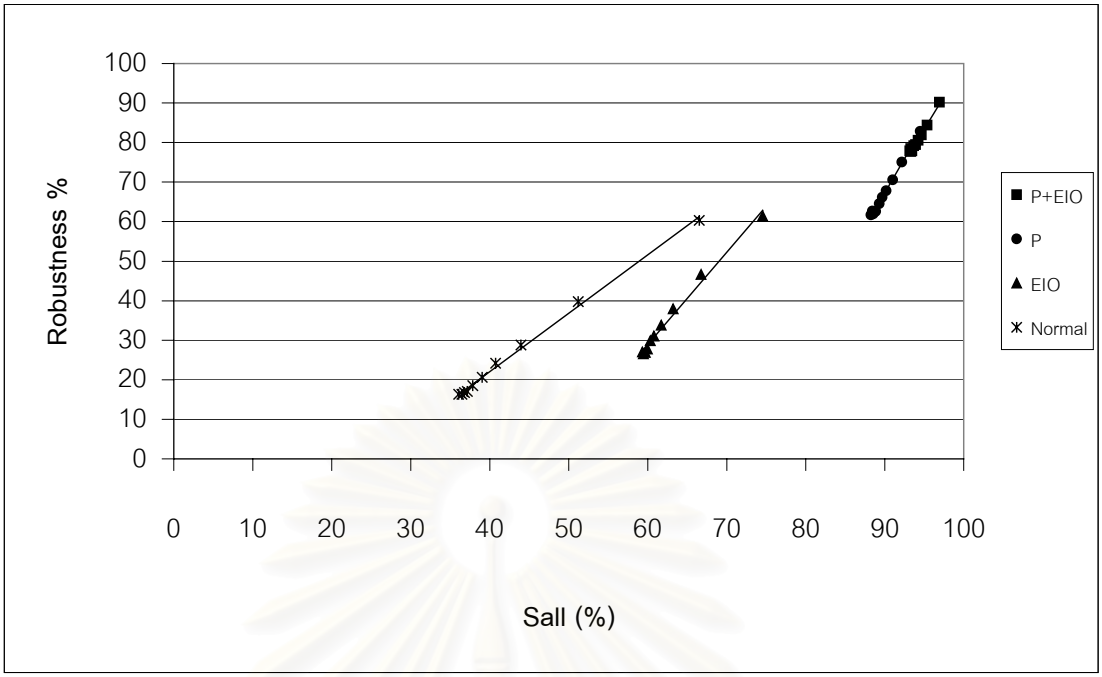
หมายเหตุ Dtest หมายถึง เปอร์เซ็นต์สัญญาณรบกวนของสภาพแวดล้อมทดสอบ



รูปที่ 6.3 แสดงค่า Sall ของแต่ละวิธีเพิ่มความทนทาน แยกตามเปอร์เซ็นต์สัญญาณรบกวน

เมื่อพิจารณาค่า *Sall Ssucc* และ *Sfail* ในตารางที่ 6.1 และรูปที่ 6.3 จะเห็นได้ว่าวิธีเพิ่มความทนทานโดยการรังควานร่วมกับการสุ่มมีค่า *Sall* สูงสุด ตามด้วยวิธีการรังควาน วิธีการสุ่ม และวิธีธรรมดา ซึ่งค่า *Sall* นี้เรียงลำดับสอดคล้องกับค่าความทนทานของโปรแกรมคำตอบที่ได้จากการทดลองในตารางที่ 5.1 นอกจากนี้ยังสังเกตได้ว่าค่า *Ssucc* ยังสูงกว่าค่า *Sfail* ทุกกลุ่มเปอร์เซ็นต์สัญญาณรบกวนในทุกวิธีเพิ่มความทนทาน แสดงให้เห็นว่าสภาพแวดล้อมที่ประสบความสำเร็จมีการนำประสบการณ์กลับมาใช้สูงกว่าสภาพแวดล้อมที่ไม่ประสบความสำเร็จ

ผลการวิเคราะห์ความทนทานแสดงให้เห็นว่า ประสบการณ์ที่นำกลับมาใช้มีความสัมพันธ์กับความทนทาน เนื่องจากโปรแกรมคำตอบที่นำประสบการณ์กลับมาใช้สูงจะมีความทนทานสูงตามไปด้วย เพื่อหาความสัมพันธ์ดังกล่าวผู้วิจัยได้สร้างกราฟความสัมพันธ์ระหว่างค่าความทนทานของโปรแกรมคำตอบในตารางที่ 5.1 กับค่า *Sall* เป็นดังรูปที่ 6.4 นอกจากนี้ผู้วิจัยยังได้คำนวณค่าสัมประสิทธิ์สหสัมพันธ์ (Correlation coefficient) ระหว่างค่า *Sall* กับค่าความทนทานเพื่อหาระดับความสัมพันธ์ของค่าทั้งสอง ผลเป็นดังตารางที่ 6.2



*รูปที่ 6.4 กราฟแสดงความสัมพันธ์ระหว่าง **Sall** กับค่าความทนทาน*

ตารางที่ 6.2 แสดงค่าสัมประสิทธิ์สหสัมพันธ์ของแต่ละวิธีเพิ่มความทนทาน

วิธีเพิ่มความทนทาน	ค่าสัมประสิทธิ์สหสัมพันธ์
Normal	0.9985
EIO	0.9941
P	0.9980
P+EIO	0.9954

จะเห็นได้ว่ากราฟในรูปที่ 6.4 นี้ เมื่อปรับเส้นแนวโน้มของชุดข้อมูลแล้วพบว่าความสัมพันธ์ระหว่าง ***Sall*** กับค่าความทนทานมีแนวโน้มเป็นเส้นตรง ส่วนค่าสัมประสิทธิ์สหสัมพันธ์ของแต่ละวิธีในตารางที่ 6.2 ก็มีค่าเข้าใกล้ 1 ชี้ให้เห็นความสัมพันธ์แบบแปรผันตามที่ดี

ผลที่ได้จากกราฟที่ 6.4 และตารางที่ 6.2 นี้สรุปได้ว่า ***Sall*** หรือการนำประสบการณ์กลับมาใช้เป็นปัจจัยหนึ่งที่ทำให้เกิดความทนทาน โดยโปรแกรมคำตอบจากวิธีการรังควานร่วมกับการสุ่ม ที่มีความทนทานสูงสุดนั้น ก็เนื่องมาจากสามารถดึงประสบการณ์กลับมาใช้ได้ดีที่สุด

6.4 **สรุปท้ายบท**

บทนี้ได้วิเคราะห์ความทนทานของโปรแกรมคำตอบโดยใช้เส้นทางปฏิบัติงาน และประสบการณ์เป็นปัจจัยในการอธิบายความทนทาน ซึ่งผลการวิเคราะห์พบว่าการดึงประสบการณ์กลับมาใช้เป็นปัจจัยหนึ่งที่ทำให้เกิดความทนทาน โดยโปรแกรมคำตอบที่ดึงประสบการณ์กลับมาใช้ได้ดีจะมีความทนทานสูง

บทที่ 7

สรุปผลการวิจัยและข้อเสนอแนะ

บทนี้เป็นการสรุปเนื้อหาทั้งหมดของงานวิจัยที่ได้ทำมา และเสนอแนะข้อคิดเห็นต่างๆที่ได้จากการทดลอง

7.1 <u>สรุปผลการวิจัย</u>

งานวิจัยนี้เป็นการทดลองเพื่อเพิ่มความทนทานให้กับโปรแกรมคำตอบ โดยใช้ปัญหาหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายเป็นปัญหาทดลอง และนิยามความทนทานของโปรแกรมคำตอบหมายถึง การที่โปรแกรมคำตอบสามารถควบคุมหุ่นยนต์เดินหลบหลีกสิ่งกีดขวางไปหาเป้าหมายได้ในสภาพแวดล้อมที่ไม่ได้เรียนรู้มา

ผลจากงานวิจัยนี้แสดงให้เห็นว่าการปรับปรุงกระบวนการวิวัฒนาการ โดยเรียนรู้จากสภาพแวดล้อมที่แตกต่างกันหลายๆแบบหรือการรังควาน และการเพิ่มฟังก์ชันพิเศษที่มีลำดับการทำงานไม่แน่นอนหรือการสุ่ม สามารถรวมกันเพื่อเพิ่มความทนทานให้กับโปรแกรมคำตอบได้ เนื่องจากโปรแกรมคำตอบที่สร้างจากวิธีการรังควาน และการสุ่มมีข้อดีที่แตกต่างกัน

โดยขั้นตอนการทดลองเริ่มจากศึกษาข้อมูล และทำการทดลองย่อยเพื่อหาค่าพารามิเตอร์ของกำหนดการเชิงพันธุกรรมที่เหมาะสม จากนั้นให้กำหนดการเชิงพันธุกรรมสร้างโปรแกรมคำตอบ 4 แบบ ได้แก่ โปรแกรมคำตอบแบบการรังควานร่วมกับการสุ่ม แบบการรังควาน แบบการสุ่ม และแบบธรรมดา เพื่อทดสอบความทนทานกับสภาพแวดล้อมที่ไม่ได้เรียนรู้มา ซึ่งผลการทดสอบพบว่าโปรแกรมคำตอบที่ได้จากวิธีการรังควานร่วมกับการสุ่มนั้นมีความทนทานสูงสุด คือ โดยเฉลี่ยแล้วมีความทนทานมากกว่าโปรแกรมคำตอบแบบธรรมดาถึง 60% และเมื่อเทียบกับวิธีการรังควาน หรือวิธีการสุ่มเพียงอย่างใดอย่างหนึ่งก็มีความทนทานสูงกว่าประมาณ 20% และ 50% ตามลำดับ

งานวิจัยนี้ยังได้วิเคราะห์ถึงสาเหตุที่ทำให้โปรแกรมคำตอบที่ได้จากวิธีการรังควานร่วมกับการสุ่มมีความทนทานเพิ่มขึ้น โดยใช้เส้นทางปฏิบัติการ และประสบการณ์เป็นแนวคิดในการอธิบาย ซึ่งผลที่เกิดขึ้นพบว่าโปรแกรมคำตอบที่ได้จากวิธีการรังความร่วมกับการสุ่มสามารถดึงประสบการณ์กลับมาใช้ได้ดีที่สุดคิดเป็นอัตราส่วนถึง 90% ส่วนโปรแกรมคำตอบที่ได้จากวิธีอื่นก็มีการดึงประสบการณ์กลับมาใช้ลดหลั่นกันไปตามค่าความทนทานที่ได้ และเมื่อหาความสัมพันธ์ระหว่างความทนทานกับการนำประสบการณ์กลับมาใช้พบว่าค่าทั้งสองมีความสัมพันธ์แบบแปรผันตาม ซึ่งทำให้สรุปได้ว่าโปรแกรมคำตอบที่สามารถนำประสบการณ์กลับมาใช้ได้ดีจะมีความทนทานสูง

7.2 ข้อเสนอแนะ

1. สาเหตุที่ทำให้เกิดปัญหาความทนทานสืบเนื่องจากการประมวลผลกำหนดการเชิงพันธุกรรม เพื่อสร้างโปรแกรมคำตอบนั้นประมวลผลบนสภาพแวดล้อมจำลองที่สร้างจากคอมพิวเตอร์ไม่ได้ประมวลผลบนสภาพแวดล้อมจริง เพราะมีข้อจำกัดหลายอย่าง เช่น ข้อจำกัดด้านเวลา ซึ่งผลที่เกิดขึ้นทำให้โปรแกรมคำตอบมีปัญหาเรื่องความทนทานเมื่อนำไปใช้กับสภาพแวดล้อมจริง งานวิจัยนี้ได้เสนอวิธีเพิ่มความทนทาน โดยทดสอบโปรแกรมคำตอบบนสภาพแวดล้อมจำลองเท่านั้นไม่ได้ทดสอบบนสภาพแวดล้อมจริงจึงน่าจะมีการทดลอง โดยนำโปรแกรมคำตอบที่ได้ไปทดสอบความทนทานกับสภาพแวดล้อมจริง เพื่อศึกษาว่าความทนทานในสภาพแวดล้อมจริงสอดคล้องกับสภาพแวดล้อมจำลองหรือไม่

2. การนำประสบการณ์กลับมาใช้น่าจะเป็นหนึ่งในหลายๆปัจจัยที่ทำให้โปรแกรมคำตอบมีความทนทานจึงควรมีการศึกษาปัจจัยอื่นที่มีผล เพื่อให้ได้องค์ประกอบรวมที่แท้จริง ซึ่งทำให้เกิดความทนทาน แนวทางหนึ่งที่น่าสนใจ คือ การศึกษาพฤติกรรม และโครงสร้างภายในของโปรแกรมคำตอบ เช่น พฤติกรรมแบบใดทำให้เกิดความทนทาน หรือโปรแกรมคำตอบควรจะมีชุดฟังก์ชัน และเทอร์มินอลเรียงตัวกันอย่างไร เพื่อให้มีความทนทาน

3. โปรแกรมคำตอบที่ได้จากวิธีการรังควานร่วมกับการสุ่มที่เสนอในงานวิจัยนี้ เป็นวิธีที่นำประสบการณ์กลับมาใช้ได้ดี แต่วิธีนี้ใช้เวลาประมวลผลเพื่อหาโปรแกรมคำตอบนานมาก จึงควรมีการค้นคว้าหาเทคนิคพิเศษ หรือวิธีเพิ่มความทนทานใหม่ที่สามารถดึงประสบการณ์กลับมาใช้ได้ดี และใช้เวลาประมวลผลเพื่อหาโปรแกรมคำตอบน้อย

4. ควรมีการศึกษาเพิ่มเติมโดยนำวิธีเพิ่มความทนทานโดยการรังควานร่วมกับการสุ่ม ไปประยุกต์ใช้กับปัญหาอื่น เพื่อทดสอบว่าวิธีนี้มีความเป็นทั่วไปสามารถเพิ่มความทนทานให้กับโปรแกรมคำตอบในปัญหาอื่นๆได้

รายการอ้างอิง

ภาษาไทย

ธีระ โตสุโขวงศ์ และ ทัศสิน บัวชื่น. “วิธีปรับปรุงความทนทานในกระบวนการเรียนรู้ของหุ่นยนต์ด้วยการโปรแกรมพันธุการ”. *โครงงานวิศวกรรมคอมพิวเตอร์* ปริญญาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย, 2540.

มาเรีย ประทีปทองคำ. “การเพิ่มความทนทานของวิธีการเรียนรู้แบบกำหนดการเชิงพันธุกรรมโดยการปรับพารามิเตอร์สำหรับปัญหาการนำร่องหุ่นยนต์”. *วิทยานิพนธ์ปริญญามหาบัณฑิต* ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2541.

รุ่งโรจน์ นพสุวรรณชัย. “การปรับปรุงกระบวนการวิวัฒนาการในวิธีกำหนดการเชิงพันธุกรรมเพื่อสร้างคำตอบที่มีความทนทานสำหรับปัญหาการนำร่องหุ่นยนต์”. *วิทยานิพนธ์ปริญญามหาบัณฑิต* ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2541.

ภาษาอังกฤษ

Ito, T., Iba, H., and Kimura, M. “Robustness of robot programs generated by genetic programming,” *In Genetic Programming: Proceedings of the First Annual Conference*, pp.321-326. MIT Press, 1996.

Koza, J. R. *Genetic Programming: On the programming of computers by means of natural selection*. MA, USA: MIT Press, 1992.

McNutt, G. “Using Co-Evolution to Produce Robust Robot Control,” *Proc. Conf. Decision and Control 36th*. 1997.

Mitchell, M. *An Introduction of genetic algorithms*. MA, USA: MIT Press, 1996.

Reynolds, C. W. “Evolution of obstacle avoidance behavior using noise to promote robust solution,” In K. E. Kinear, Jr. (Ed.), *Advance in Genetic Programming*, Chapter 10, pp.221-241. MIT Press, 1994.

Reynolds, C. W. “Evolution of corridor following behavior in a noisy world”. *In Simulation of Adaptive Behavior*. (n.p.), 1994.

Rodkaew, Y., and Chongstitvatana, P. "How to reduce the memory requirement in genetic programming," *Proc. of 4th Annual National Symposium on Computational Science and Engineering*. Bangkok, 2000.

ภาคผนวก

ภาคผนวก ก

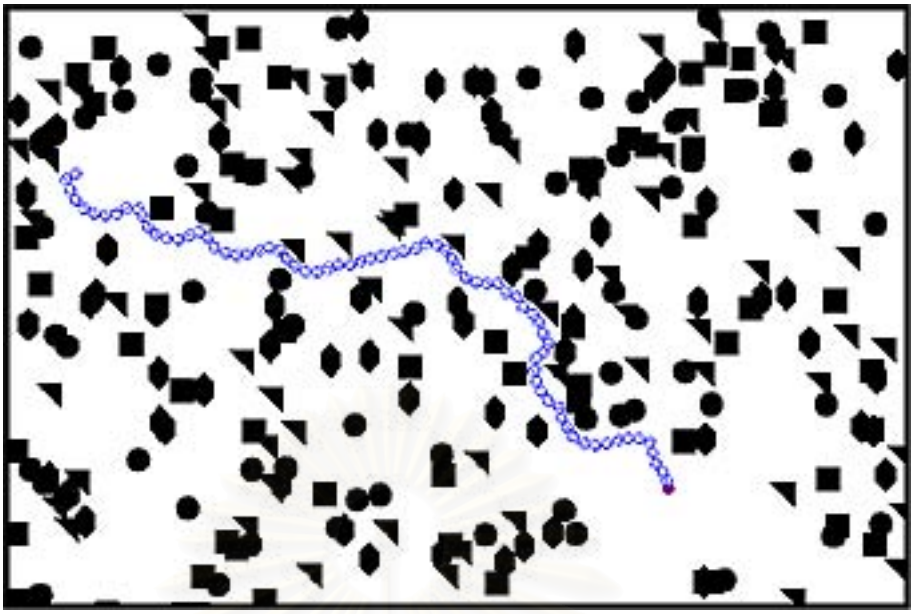
ตัวอย่างโปรแกรมหุ่นยนต์

ภาคผนวกนี้แสดงตัวอย่างโปรแกรมหุ่นยนต์ การเคลื่อนที่ของหุ่นยนต์ไปหาเป้าหมายในสภาพแวดล้อมจำลอง โดยโปรแกรมหุ่นยนต์ที่แสดงนี้คัดเลือกมาเพียง 1 โปรแกรมจากทั้งหมด 20 โปรแกรมตามแต่ละวิธีเพิ่มความทนทาน

<u>โปรแกรมหุ่นยนต์ที่ได้จากวิธีธรรมดา</u>

(IF-AND (IF-AND (IF-NOT (IF-AND right forward left forward) (IF-AND right left right right) (IF-AND right right isnear isnear)) (IF-NOT (IF-NOT right left isnear) (IF-AND right isnear (IF-AND right forward left forward) isnear) (IF-NOT forward right forward)) (IF-AND (IF-OR left forward forward left) (IF-NOT forward right forward) (IF-OR isnear left right right) (IF-OR left isnear isnear forward)) (IF-NOT (IF-NOT (IF-AND forward right (IF-AND right forward left forward) isnear) left right) (IF-NOT right left left) (IF-NOT right right left))) (IF-OR (IF-NOT isnear left (IF-AND right forward left forward)) (IF-AND (IF-AND right right right isnear) (IF-OR right left isnear right) (IF-OR isnear left right right) (IF-AND isnear forward forward forward)) (IF-AND (IF-AND left isnear isnear left) (IF-AND left left forward isnear) (IF-NOT isnear isnear forward) (IF-AND right forward left left)) isnear) (IF-OR (IF-NOT (IF-AND forward left isnear isnear) (IF-OR forward forward left right) (IF-AND isnear right left (IF-NOT right right forward))) (IF-NOT (IF-AND isnear isnear right right) (IF-OR right left isnear left) (IF-NOT forward isnear left)) (IF-NOT (IF-OR left forward isnear right) (IF-NOT forward forward right) (IF-NOT isnear right left)) (IF-NOT (IF-OR (IF-NOT isnear isnear (IF-NOT forward right forward)) isnear forward left) (IF-AND (IF-OR forward forward right right) (IF-NOT (IF-OR left forward isnear right) (IF-NOT forward forward right) (IF-NOT isnear right left)) forward forward) (IF-NOT isnear forward (IF-AND (IF-NOT (IF-AND right forward left forward) (IF-AND right left right right) (IF-AND right right isnear isnear)) (IF-NOT (IF-NOT (IF-AND isnear isnear right right) (IF-OR right left isnear left) (IF-NOT forward isnear left)) (IF-AND right isnear isnear isnear) (IF-NOT forward right forward)) (IF-AND (IF-OR left forward forward left) (IF-NOT right right forward) (IF-AND isnear isnear forward forward) (IF-OR left isnear isnear forward)) (IF-NOT (IF-NOT isnear left right) (IF-NOT right left left) (IF-NOT right right left)))))) (IF-NOT (IF-OR (IF-AND forward right isnear isnear) (IF-AND isnear isnear left right) left (IF-AND forward isnear left isnear)) (IF-AND (IF-NOT forward isnear isnear) (IF-OR isnear forward right forward) (IF-AND left isnear forward right) (IF-NOT forward right forward)) (IF-NOT (IF-NOT isnear right forward) (IF-NOT right left forward) (IF-OR isnear right isnear isnear))))

รูปที่ ก.1 แสดงตัวอย่างโปรแกรมหุ่นยนต์ที่ได้จากวิธีธรรมดา

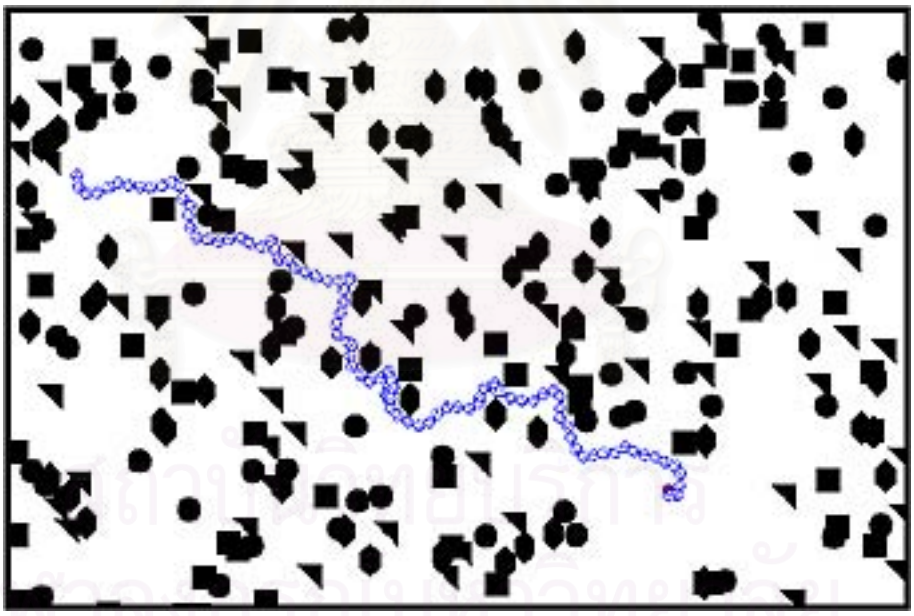


รูปที่ ก.2 แสดงการเคลื่อนที่ของหุ่นยนต์ซึ่งควบคุมโดยโปรแกรมที่ได้จากวิธีธรรมดา

โปรแกรมหุ่นยนต์ที่ได้จากวิธีการสุ่ม

(IF-NOT (EIO2 (IF-NOT forward right (IF-NOT right isnear isnear)) (IF-AND (IF-OR forward right isnear isnear) (IF-OR isnear right forward forward) (IF-OR left forward forward forward) (IF-AND (IF-NOT right forward right) right forward right))) right (EIO2 (IF-AND (IF-OR forward (IF-NOT forward left left) left isnear) (IF-AND left right left right) (IF-OR isnear isnear (EIO2 (IF-NOT right (IF-OR forward right isnear isnear) (IF-NOT right isnear isnear)) (IF-AND (IF-OR forward right isnear isnear) (IF-OR isnear right forward forward) (IF-OR left forward forward forward) (IF-AND (IF-NOT right forward forward) right forward (IF-AND forward (IF-OR isnear isnear forward left) left (IF-OR left right right forward))))) left) (EIO2 (IF-NOT forward isnear (IF-NOT right isnear isnear)) (IF-AND (IF-OR forward right isnear isnear) (IF-OR isnear right forward forward) forward (IF-AND (IF-NOT right forward forward) right forward right)))) (IF-AND (IF-NOT forward left left) (IF-AND isnear isnear forward left) left (IF-OR left right (IF-OR left right right left) left))))

รูปที่ ก.3 แสดงตัวอย่างโปรแกรมที่ได้จากวิธีการสุ่ม

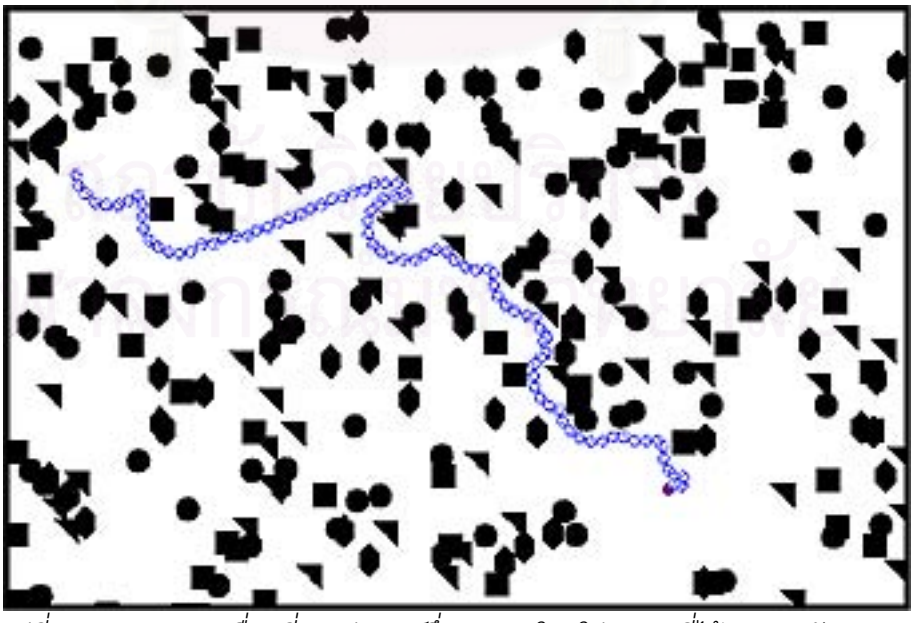


รูปที่ ก.4 แสดงการเคลื่อนที่ของหุ่นยนต์ซึ่งควบคุมโดยโปรแกรมที่ได้จากวิธีการสุ่ม

โปรแกรมหุ่นยนต์ที่ได้จากวิธีการรังควาน

(IF-NOT (IF-OR (IF-NOT (IF-NOT forward isnear forward) (IF-NOT (IF-OR (IF-AND (IF-OR forward left right forward) (IF-OR forward left (IF-AND forward isnear forward right) forward) left (IF-AND isnear left forward left)) (IF-OR right left forward left) left isnear) forward isnear) (IF-OR left forward left (IF-AND right isnear forward right))) right (IF-AND (IF-NOT isnear isnear right) (IF-AND forward isnear forward right) (IF-OR left isnear isnear forward) (IF-AND forward (IF-OR forward left right forward) isnear left)) (IF-NOT left forward (IF-NOT (IF-OR (IF-AND (IF-OR forward left forward forward) (IF-OR forward left (IF-NOT forward isnear forward) forward) left left) (IF-OR right left forward left) (IF-AND isnear isnear left isnear) left) forward isnear))) (IF-NOT (IF-NOT (IF-NOT isnear isnear forward) (IF-NOT left isnear left) forward) forward (IF-AND (IF-OR left isnear right right) (IF-OR isnear left isnear right) (IF-AND right right right isnear) (IF-NOT forward forward left))) (IF-OR (IF-OR isnear (IF-OR forward left isnear isnear) (IF-OR isnear left isnear right) (IF-OR left left forward forward)) (IF-AND (IF-AND right forward left isnear) (IF-NOT right (IF-AND (IF-NOT isnear isnear (IF-OR right (IF-OR right (IF-AND right left forward left) forward left) (IF-AND isnear forward left forward) (IF-OR isnear forward left forward))) (IF-AND (IF-AND forward forward (IF-OR isnear left right forward) left) forward right right) (IF-OR left isnear isnear forward) (IF-AND left forward isnear left)) isnear) (IF-AND left right forward forward) (IF-NOT isnear right left)) (IF-NOT (IF-AND right isnear forward right) (IF-OR right isnear right (IF-OR isnear (IF-OR forward forward isnear isnear) (IF-OR right left isnear right) (IF-OR left isnear forward left))) (IF-AND forward forward forward isnear)) (IF-NOT (IF-AND right left forward left) (IF-AND forward isnear left right) (IF-AND left forward forward left))))

รูปที่ ก.5 แสดงตัวอย่างโปรแกรมหุ่นยนต์ ที่ได้จากวิธีการรังควาน

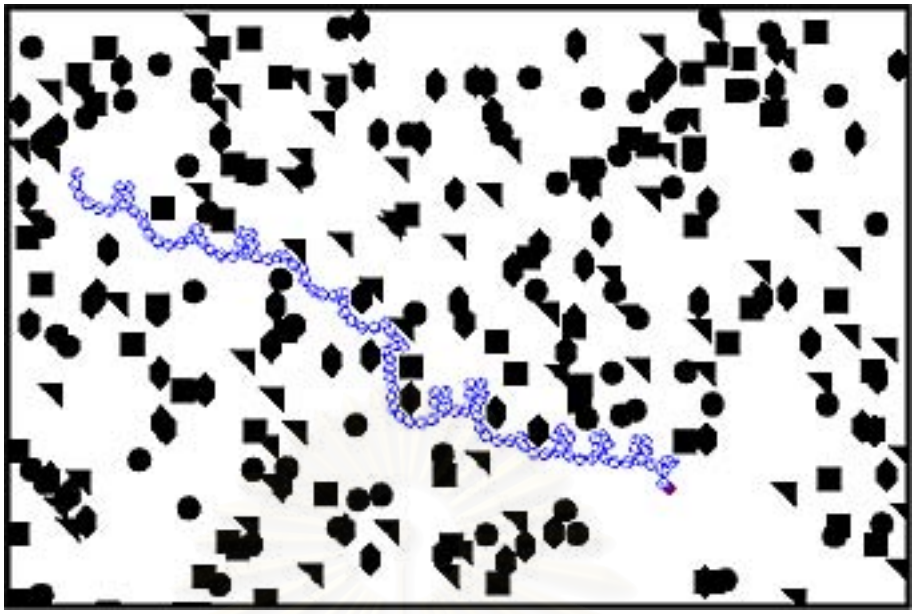


รูปที่ ก.6 แสดงการเคลื่อนที่ของหุ่นยนต์ซึ่งควบคุมโดยโปรแกรมที่ได้จากการรังควาน

โปรแกรมหุ่นยนต์ที่ได้จากวิธีการรังควานร่วมกับการสุ่ม

(IF-NOT (IF-NOT (IF-NOT isnear left forward) (IF-AND (IF-OR (IF-OR left right isnear left) left left left) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (EIO2 left isnear)) (IF-OR right forward left left)) (EIO2 (IF-AND (IF-AND forward left right left) (IF-OR isnear left isnear forward) (IF-OR right right forward right) (IF-OR forward forward forward left)) (EIO2 (EIO2 (EIO2 forward isnear) (EIO2 forward isnear)) (IF-OR left isnear right isnear))) (IF-OR (IF-OR (IF-OR left forward right isnear) (IF-AND forward forward forward right) (IF-NOT forward (IF-AND right isnear right (IF-AND (IF-OR isnear right isnear isnear) (IF-AND forward right forward isnear) (IF-AND left forward forward isnear) (IF-OR right isnear right right))) left) (IF-AND right forward right left)) (IF-NOT (IF-AND right right forward right) (IF-AND isnear left (IF-NOT (IF-NOT isnear left forward) (IF-AND (IF-OR left forward left forward) (IF-NOT right (IF-OR right forward left left) isnear) (IF-AND right forward (IF-AND (IF-OR (IF-OR left right isnear left) left (IF-AND forward forward forward isnear) (EIO2 (EIO2 right right) (IF-OR left forward right left))) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (EIO2 (IF-AND (IF-OR left isnear left forward) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (IF-NOT (IF-NOT isnear left forward) (IF-AND (IF-OR (IF-OR left right isnear left) left left (EIO2 (EIO2 right right) (IF-OR left (IF-OR right forward left left) right left))) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (EIO2 right isnear)) (IF-OR right forward left left))) isnear)) isnear) (IF-NOT (IF-NOT isnear left forward) (IF-AND (IF-OR (IF-OR left right isnear left) left left (EIO2 (EIO2 right left) (IF-OR left (IF-OR right forward left left) right left))) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (EIO2 right isnear)) (IF-OR right forward left left))) (IF-OR (IF-NOT (IF-NOT isnear (IF-OR isnear right isnear isnear) forward) (IF-AND (IF-OR (IF-OR left right isnear left) left (IF-AND forward forward forward isnear) (EIO2 (EIO2 right right) (IF-OR left forward right left))) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (EIO2 (IF-AND (IF-OR left isnear left forward) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (IF-NOT (IF-NOT isnear left (IF-AND (IF-OR (IF-OR left right isnear left) left left left) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (EIO2 left isnear))) (IF-AND (IF-OR (IF-OR left right isnear left) left left (EIO2 (EIO2 right right) (IF-OR left (IF-OR right forward left left) right left))) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) (EIO2 right isnear)) (IF-OR right forward left left))) isnear)) (IF-OR right forward left left)) forward left (IF-OR (IF-NOT (IF-NOT isnear left forward) (IF-AND (IF-OR (IF-OR left right isnear left) left (IF-AND forward forward forward isnear) (EIO2 (EIO2 right right) (IF-OR left forward right left))) (IF-NOT right isnear isnear) (IF-AND right forward right isnear) left) (IF-OR right forward left left)) left left left))) right) (IF-AND forward left isnear forward)) (IF-OR (IF-NOT left left isnear) isnear (IF-OR right left forward left) (IF-NOT left forward isnear)) (IF-AND (EIO2 isnear isnear) (IF-OR isnear left forward right) (IF-AND isnear isnear left isnear) (IF-NOT forward isnear forward))))

รูปที่ ก.7 แสดงตัวอย่างโปรแกรมหุ่นยนต์ ที่ได้จากวิธีการรังควานร่วมกับการสุ่ม



รูปที่ ก.8 แสดงการเคลื่อนที่ของหุ่นยนต์ซึ่งควบคุมโดยโปรแกรมที่ได้จากการรังควานร่วมกับการสุ่ม

ประวัติผู้เขียน

นายธนัท สุขกาญจนนท์ เกิดวันที่ 29 พฤศจิกายน พ.ศ. 2519 ที่กรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิทยาศาสตร์บัณฑิต สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์ ในปีการศึกษา 2540 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2541