บทที่ 6

งานวิจัยที่เกี่ยวข้อง

บทนี้จะนำเสนองานวิจัยที่เกี่ยวข้องกับงานวิจัยครั้งนี้ ซึ่งเป็นการวิจัยที่เกี่ยวข้องกับการใช้วิธีการตามขั้นตอนวิธีพันธุการและการโปรแกรมพันธุการในหลายๆวิธีที่จะแก้ปัญหาต่างๆ โดยนำเสนอแยกตามหัวข้อต่างๆของงานวิจัย ซึ่งแต่ละงานวิจัยแสดงให้เห็นถึงความแตกต่างกันของวิธีการเหล่านี้ที่ทำให้หุ่นยนต์เรียนรู้ว่ามีวิธีการเรียนรู้ได้อย่างไรและเรียนรู้อะไรบ้าง และสรุปท้ายบท

6.1 การวิวัฒนาการที่ละขั้น(The staged evolution)

Lewis, Fagg และ Solidum (1992) ได้อธิบายถึงการวิวัฒนาการทีละขั้นของเครื่องควบคุมการทำงานของมอเตอร์ที่ใช้กับหุ่นยนต์เดินได้ ซึ่งประกอบด้วยโครงสร้างของระบบเครือข่ายหน่วยประสาท โดยอาศัยขบวนการขั้นตอนวิธีพันธุการเพื่อค้นหาค่าตัวเลขน้ำหนัก (weights)ที่เหมาะสมกับการควบคุมการเคลื่อนที่ของหุ่นยนต์ โดยเสนอแนวคิดในการแบ่งชบวนการวิวัฒนาการออกเป็น 2 ช่วง โดยการวิวัฒนาการช่วงแรกจะเป็นการวิวัฒนาการในส่วนของการควบคุมการเคลื่อนที่ของขาหุ่นยนต์เพียงอย่างเดียว เพื่อให้สามารถควบคุมการเคลื่อนที่ของขาหุ่นยนต์ได้ จากนั้นจะเป็นการวิวัฒนาการของการควบคุมการเคลื่อนที่ของหุ่นยนต์ทั้งหมดที่ประกอบด้วยขาหุ่นยนต์ 4 ขาให้สามารถทำงานประสานกันได้ โดยอาศัยค่าตัวเลขน้ำหนักที่เชื่อมต่อกันทุกๆขาเข้าด้วยกัน ซึ่งได้ผลเป็นอย่างดี และได้แสดงความคิดเห็นว่าขั้นตอนวิธีพันธุการมีประสิทธิภาพเพียงพอสำหรับนำไปใช้ในระบบที่ซับซ้อนภายใต้สภาพแวดล้อมใดๆ โดยไม่จำเป็นต้องทำการออกแบบล่วงหน้าถึงผลกระทบของสภาพแวดล้อมที่มีต้องระบบนั้นๆไว้ก่อน

6.2 แผนการเคลื่อนที่ของแขนกลหุ่นยนต์

Chan และ Zalzala (1993) ได้นำเสนอการใช้ขั้นตอนวิธีพันธุการที่ใช้สายอักขระที่มีความยาวไม่คงที่ เพื่อหาแผนการเคลื่อนที่ของแขนกลหุ่นยนต์ที่จำลองขึ้น ในการขยับปลายแขนกลไปยังเป้าหมายที่กำหนด โดยต้องใช้เวลาน้อยที่สุดด้วย การวัดค่าความเหมาะจะวัดจากเวลาที่ใช้ในการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดเป้าหมาย การทดลองกำหนดให้มีประชากร 100 สายอักขระในขบวนการวิวัฒนาการถึง 5000 รุ่น ผลการทดลองพบว่าในรุ่นที่ 4717 ของการวิวัฒนาการสามารถหาแผนการเคลื่อนที่ของแขนกลหุ่นยนต์ที่ดีที่สุด โดยใช้เวลาน้อยกว่าการใช้วิธี-การค้นหาแบบศึกษาสำนึก(the heuristic search)

งานวิจัยครั้งนี้พิจารณาเฉพาะประสิทธิภาพของเวลาที่ใช้ในการปฏิบัติการ ซึ่งการออกแบบไม่ได้พิจารณาความเป็นไปได้ในโลกจริง

Khoogar และ Parker (1991) นำเสนอวิธีการใหม่ของการหาแผนการเคลื่อนที่เพื่อการหลบหลีกสิ่งกีดขวาง โดยทดลองกับแขนกลหุ่นยนต์ที่มี 3 ข้อต่อ โดยอาศัยวิธีการขั้นตอน-วิธีพันธุการ เพื่อค้นหาลำดับการเคลื่อนที่ของแขนกล ซึ่งแทนแต่ละพารามิเตอร์ในระบบด้วยรหัสโดยใช้สายอักขระที่มีความยาว 30 ตัวอักษรในการแทนแผนการเคลื่อนที่ โดยวัดค่าความเหมาะจากจำนวนครั้งของการเคลื่อนที่ที่ใช้ในการเคลื่อนที่ไปยังเป้าหมาย

การทดลองครั้งนี้เป็นการจำลองการทำงานบนเครื่องคอมพิวเตอร์ โดยกำหนดจุดเริ่มต้น, จุดเป้าหมาย, และจุดปลายแขน การเคลื่อนที่ของแขนกลขึ้นกับการหมุนของแต่ละข้อ-ต่อ ซึ่งขึ้นกับค่า ตัวเลขฐานสาม(Ternary number) คือ {1, 0, -1} โดย 1 หมายถึงหมุนเพิ่มในทางบวก, 0 หมายถึง ไม่มีการเคลื่อนที่ และ -1 หมายถึงหมุนเพิ่มไปในทางลบ ดังนั้น ในการเคลื่อน-ที่แต่ละครั้งของแขนกล N ข้อต่อ จะต้องใช้ N หลักของตัวเลขฐานสาม และในแต่ละ M ครั้งของการเคลื่อนที่ที่ประสบความสำเร็จ จะต้องใช้สายอักขระที่ยาว N * M หลักของตัวเลขฐานสาม

สองสิ่งที่ถูกนำมาพิจารณาในการกำหนดค่าความเหมาะของปัญหานี้คือ ต้องนำปลายแขนไปยังจุดเป้าหมาย และการเคลื่อนที่ต้องไม่มีการชนกับสิ่งกีดขวาง ส่วนคุณสมบัติอื่นที่จะถูกนำมาพิจารณาเป็นอันดับต่อมาคือ พยายามให้ใช้การเคลื่อนที่ให้น้อยที่สุด และให้มีการชนน้อยที่สุด

ค่าความเหมาะได้ถูกออกแบบตามเงื่อนไขข้างต้นดังนี้

(1) เข้าใกล้จุดเป้าหมายมากที่สุดโดยพิจารณาเป็นค่าความผิดพลาดคือ

$Error_1$ = ระยะห่างจากจุดเป้าหมาย

(2) หลบหลีกสิ่งกีดขวาง

$Error_2$ = ค่าคงที่ , ถ้าเกิดการชน

$Error_2$ = 0 , ถ้าไม่ชน

(3) เคลื่อนที่น้อยที่สุด

$Error_3 = W * N$

โดย W เป็นค่าคงที่แทนตัวเลขน้ำหนัก

และ N แทนจำนวนครั้งของการเคลื่อนที่

(4) ค่าความผิดพลาดรวม

$Error_{total} = Error_1 + Error_2 + Error_3$

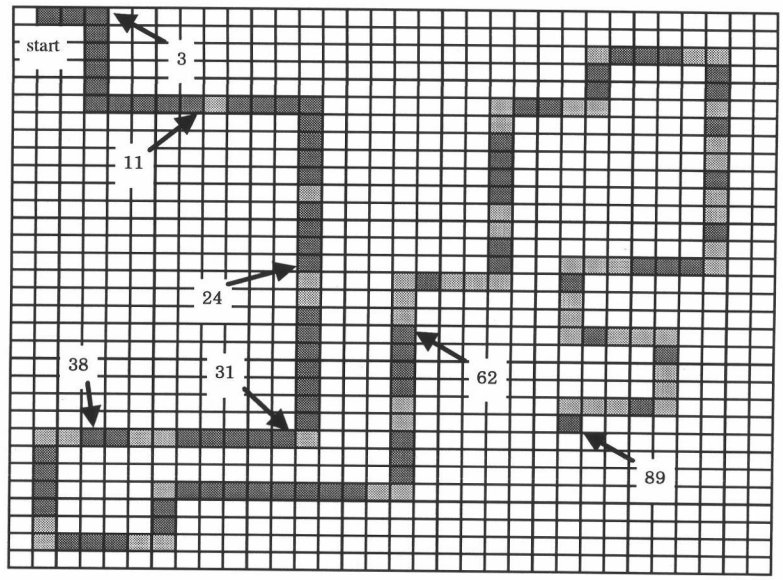
โดยจะพิจารณาค่าความผิดพลาดรวมน้อยที่สุด

เนื่องจากข้อจำกัดของคอมพิวเตอร์ในการกำหนดจำนวนครั้งของการเคลื่อนที่ที่มากที่สุด การเคลื่อนที่จะถูกกำหนดให้เคลื่อนที่ครั้งละ 10 ครั้ง นั้นคือ 30 ตัวอักษรของสาย-อักขระ โดยในกรณีที่การเคลื่อนที่ไม่พอจะเพิ่มเป็นอีกหนึ่งชุดโดยเริ่มจากจุดนั้น เพื่อให้สามารถเข้าถึงจุดเป้าหมายได้ ถ้าตรวจสอบพบว่าหุ่นยนต์ไม่สามารถเดินได้ดีขึ้นหรือติดอยู่ในสภาพเดิมๆ เป็นเวลานาน จะยกเลิกการเคลื่อนที่นั้น

จากการทดลองพบว่าวิธีการที่ใช้สามารถทำงานได้ดี แต่การทดลองดังกล่าวเป็นเพียงการใช้สมการฟอร์เวิร์ดไคนีเมติก(forward kinematic equations) เพื่อใช้หาการเคลื่อนที่โดยใช้ตัวแปรของข้อต่อต่างๆเพื่อที่จะหลีกเลี่ยงการใช้อินเวอร์สไคนีเมติก(inverse kinematic) การทดลองครั้งนี้ใช้เวลาในการดำเนินการบนเครื่องคอมพิวเตอร์ IBM-AT เพียง 2 นาทีเท่านั้นเอง ซึ่งเวลาที่ใช้ขึ้นกับขนาดของประชากรและจำนวนรุ่นของการวิวัฒนาการ และการทดลองครั้งนี้ไม่ได้ทดลองในโลกจริง

6.3 <u>สมการอินเวอร์สไคนีเมติก</u>

Khoogar, Parker และ Goldberg (1989) แสดงการใช้ขั้นตอนวิธีพันธุการในการหาค่าอินเวอร์สไคนีเมติกสำหรับแขนกลหุ่นยนต์ที่ง่ายขึ้น โดยสมการอินเวอร์สไคนีเมติก (inverse kinematic equations) เป็นการหาตำแหน่งของแต่ละข้อต่อ และการจัดวางรูปร่างของแขนกลหุ่น-ยนต์ โดยให้ค่าตำแหน่งของปลายแขน ซึ่งตำแหน่งของแต่ละข้อต่อ และการจัดวางรูปร่างที่ได้มีได้มากมายหลายค่าที่แตกต่างกันไปสำหรับค่าตำแหน่งของปลายแขนค่าหนึ่ง งานวิจัยนี้ได้เสนอแนวทางใหม่ที่ใช้ขั้นตอนวิธีพันธุการ เพื่อหาค่าของสมการอินเวอร์สไคนีเมติก ที่ไม่จำเป็นต้องใช้การ



รูปที่ 6.1 ภาพแสดงการเดินของมดประดิษฐ์บนพื้นที่ 32 x 32 ที่มีก้อนหิน 89 ก้อน

คำนวณ Jacobian Matrix ซึ่งเพิ่มขึ้นอย่างมากตามจำนวนข้อต่อที่เพิ่มขึ้น สมการอินเวอร์สไคนีเมติกต้องการเพียงสมการฟอร์เวิร์ดไคนีเมติกที่ทำการคำนวณได้ง่ายกว่า ไม่ต้องการข้อจำกัดใดๆ ของแต่ละข้อต่อ ข้อจำกัดต่างๆของการหมุนแขนกลก็เกิดขึ้นโดยตรงกับตัวหุ่นยนต์เอง ขั้นตอนวิธีพันธุการจะทำการแทนค่าข้อต่อต่างๆในรูปของตัวอักขระ ซึ่งสามารถนำไปแทนเป็นโปรแกรมคอมพิวเตอร์เพื่อใช้ควบคุมระบบของหุ่นยนต์ได้อีกด้วย

6.4 <u>ตัวอย่างปัญหาที่ใช้การโปรแกรมพันธุการ</u>

Koza (1990) ได้เสนอแนวคิดในการพัฒนาการโปรแกรมพันธุการที่ใช้ขบวนการคัดเลือกและการสืบพันธุ์ของประชากรในโปรแกรมคอมพิวเตอร์เพื่อนำมาใช้ในการแก้ปัญหาในงานต่างๆในระบบปัญญาประดิษฐ์ โดยการแสดงตัวอย่างปัญหาขึ้นมา 3 ตัวอย่างด้วยกัน ตัวอย่างแรกเป็นการจำลองการเดินทางของมดประดิษฐ์ที่ชอบเดินทางไปมาตามเส้นทางที่ไม่แน่นอน ให้เดินผ่านก้อนหินให้ครบทุกก้อนภายในจำนวนการเดินที่จำกัด ถัดมาเป็นการหาการวางแผนยุทธวิธีที่เหมาะสมในการเล่นเป็นผู้ไล่ล่า(Pursuer) หรือ การเล่นเป็นผู้หลบหลีก(Evader) ในการเล่นเกมความแตกต่างกันระหว่างผู้ไล่ล่ากับผู้หลบหลีก(a differential pursuer-evader game) สุดท้ายเป็นการหายุทธวิธีที่เหมาะสมสำหรับผู้เล่นเกมง่ายๆที่ใช้ผู้เล่น 2 คน โดยใช้โครงสร้างต้นไม้

6.4.1 <u>การเดินของมดประดิษฐ์</u> เป็นการจำลองการเดินทางของมดประดิษฐ์ที่ชอบเดินทางไปมาตามเส้นทางที่ไม่แน่นอน ให้เดินผ่านก้อนหินให้ครบทุกก้อนจำนวน 89 ก้อน ดังตัวอย่างในรูปที่ 6.1 ภายในจำนวนการเดินที่จำกัดเพียง 400 ครั้ง

ปัญหานี้เคยใช้ขั้นตอนวิธีพันธุการในการแก้ปัญหาสำเร็จมาแล้ว โดยใช้สายอักขระที่ยาว 453 บิต(64 สายอักขระย่อยที่มีความยาว 7 บิตซึ่งแทนการเปลี่ยนสถานะ และบวกด้วย 5 บิตที่แทนสถานะเริ่มต้น) ในจำนวนประชากร 65536 สายอักขระ ในรุ่นที่ 200 ใช้เวลาถึง 10 ชั่วโมงบนเครื่อง Connection Machine ภายในจำนวนการเดินที่จำกัดเพียง 200 ครั้ง

สำหรับปัญหานี้กำหนดให้

Function set = {If-sensor, PROGN}

Terminal set = {ADVANCE, TURN-RIGHT, TURN-LEFT}

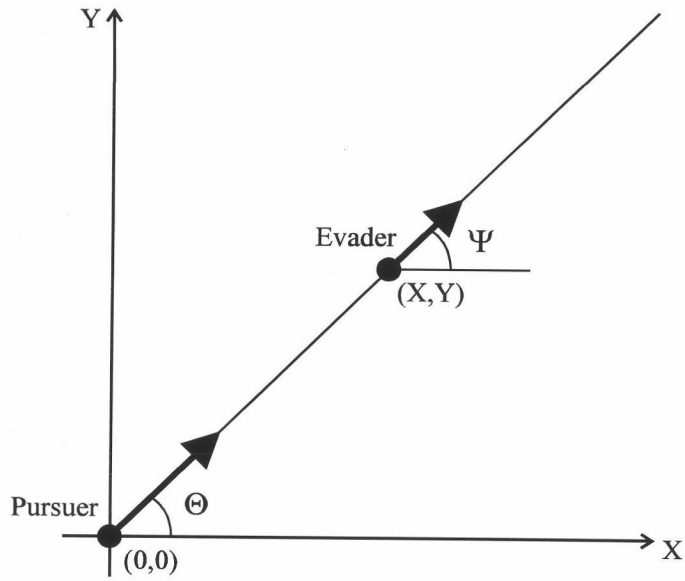
มีรูปแบบการใช้คำสั่งดังต่อไปนี้ If-sensor(arg1,arg2) หมายถึง จะทำการตรวจสอบว่าเครื่องตรวจวัดของมดประดิษฐ์พบก้อนหินข้างหน้าหรือไม่ ถ้าตรวจพบให้ทำตามอาร์กิวเมนต์แรก และถ้าตรวจไม่พบให้ทำตามอาร์กิวเมนต์ที่สอง, PROGN(arg1,arg2,...) หมายถึงเป็นคำสั่งในภาษาลิสป์ ซึ่งจะทำตามคำสั่งของอาร์กิวเมนต์ของมันตามลำดับ , ADVANCE หมายถึง เดินตรงไปข้างหน้าหนึ่งขั้น , TURN-RIGHT หมายถึง หันไปทางขวา 90 องศา , TURN-LEFT หมายถึง หันไปทางซ้าย 90 องศา

ค่าความเหมาะวัดได้จากจำนวนก้อนหินที่เดินผ่าน ภายในการเดินที่จำกัดเพียง 400 ครั้ง ยังกำหนดให้มีการใช้การกำเนิดใหม่ 10% และการผสมพันธุ์ 90% ของประชากรทั้งหมด โดยกำหนดให้มีการผสมพันธุ์ที่ฟังก์ชัน 90 % และการผสมพันธุ์ที่เทอร์มินอล 10 % ในปัญหานี้ไม่ใช้การกลายพันธุ์

โปรแกรมที่สังเคราะห์ได้ดังนี้คือ

```
(IF-SENSOR  (ADVANCE)
    (PROGN   (TURN-RIGHT)
          (IF-SENSOR  (ADVANCE)  (TURN-LEFT))
          (PROGN   (TURN-LEFT)
                (IF-SENSOR  (ADVANCE)  (TURN-RIGHT))
                (ADVANCE))))
```

การทำงานของโปรแกรมคือ มดจะเดินไปข้างหน้าถ้าตรวจพบว่ามีก้อนหินอยู่ข้างหน้า ถ้าไม่พบมดจะหันไปทางขวา แล้วจะเดินไปข้างหน้าถ้าตรวจพบว่ามีก้อนหินอยู่ข้าง-หน้า หรือถ้าไม่พบจะหันไปทางซ้าย(กลับไปวางตัวแบบเดิม) จากนั้นจะหันไปทางซ้าย และจะเดินไปข้างหน้าถ้าตรวจพบว่ามีก้อนหินอยู่ข้างหน้า หรือถ้าไม่พบจะหันไปทางขวา(กลับไปวางตัวแบบเดิม) จากนั้นจะเดินตรงไปข้างหน้า



รูปที่ 6.2 ภาพแสดงลักษณะการอ้างอิงตำแหน่งของผู้ไล่ล่าและผู้หลบหลีก

โปรแกรมที่ได้ ใช้เวลา 6 ชั่วโมงบนเครื่อง Texas Instruments Explorer II+ ได้โปรแกรมที่ประสบผลสำเร็จในรุ่นที่ 7 ของการวิวัฒนาการในประชากร 1000 ตัว และพบว่าความเป็นไปได้ของความสำเร็จของการวิวัฒนาการมีค่าเป็น 43% ภายใน 51 รุ่น,, 67% ภายใน 51 รุ่น,ในประชากร 2000 ตัว และ 81% ภายใน 51 รุ่น,ในประชากร 4000 ตัว

6.4.2 <u>เกมความแตกต่างกันระหว่างผู้ไล่ล่ากับผู้หลบหลีก</u> จุดประสงค์ของปัญหานี้คือ ค้นหายุทธวิธีที่เหมาะสมสำหรับผู้เล่น 1 คนที่เป็นผู้ไล่ล่า เพื่อเคลื่อนที่ไล่ตามผู้หลบหลีกให้ทัน

วิธีการเล่นของเกมนี้ผู้เล่นคนหนึ่งซึ่งเป็นผู้ไล่ล่าจะพยายามที่จะตามจับผู้-หลบหลีกซึ่งเคลื่อนที่ช้ากว่า การเคลื่อนที่จะใช้การเลือกทิศทาง(มุม)ในแนวที่จะไป ดังแสดงในรูปที่ 6.2 โดยที่ผู้ไล่ล่าจะเคลื่อนที่ด้วยความเร็วเท่ากับ 1.0 ซึ่งมากกว่าการเคลื่อนที่ของผู้หลบหลีกที่เคลื่อนที่ด้วยความเร็วเท่ากับ 0.67

สำหรับปัญหานี้กำหนดให้

*Function set = {+, -, *, %, EXP}*

Terminal set = {X, Y, R}

มีรูปแบบการใช้คำสั่งดังต่อไปนี้

ฟังก์ชัน + (arg1,arg2) หมายถึง ทำการบวก arg1 ด้วย arg2,ซึ่งฟังก์ชัน - (ลบ), * (คูณ) และ % (หาร) มีลักษณะเช่นเดียวกัน ส่วนฟังก์ชัน EXP(arg1) หมายถึง e^{arg1} ส่วน X กับ Y เป็นตัวแปรสถานะแทนตำแหน่งของผู้หลบหลีกหรือผู้ไล่ล่า ดังรูปที่ 18 และ R แทนค่าจำนวนจริงคงที่สุ่มระหว่าง -1 ถึง 1 เพื่อใช้ในการแก้ปัญหา

ค่าความเหมาะจะวัดจากเวลาที่ใช้ โดยที่ผู้ไล่ล่าจะต้องใช้เวลาให้น้อยที่สุด ในขณะที่ผู้หลบหลีกจะต้องใช้เวลาให้มากที่สุด สำหรับปัญหานี้กำหนดให้ใช้เวลาได้มากที่สุดเพียง 100 ขั้นเวลา

ยุทธวิธีที่ดีที่สุดของการเคลื่อนที่ของผู้ไล่ล่าคือ เดินตรงไปยังทิศทางที่ผู้-หลบหลีกอยู่ให้มากที่สุด ถ้าเดินไปในทิศทางที่เบี่ยงเบนไปจะทำให้ต้องเสียเวลามากขึ้นอีก ในทางเดียวกันยุทธวิธีที่ดีที่สุดของการเคลื่อนที่ของผู้หลบหลีกคือ เดินตรงไปยังทิศทางที่เบี่ยงเบนไปให้มากที่สุด ถ้าเดินไปในทิศทางที่ตรงกันข้ามกับที่ผู้ไล่ล่าเดินจะทำให้เวลาที่ใช้น้อยลงกว่าเดิม

ในประชากรทั้งหมด 500 ยุทธวิธี สามารถสังเคราะห์ยุทธวิธีสำหรับผู้ไล่ล่าที่แก้ปัญหานี้ได้ในทุกสภาพแวดล้อมที่แตกต่างกัน 10 แบบ ในรุ่นที่ 17 ดังนี้คือ

```
(% ( - (% (% (+ (*  2.0  Y)  - 0.066)  - 0.365)
                (% Y  - 0.124))
          (+ (EXP  X)  Y  - 0.579))
```

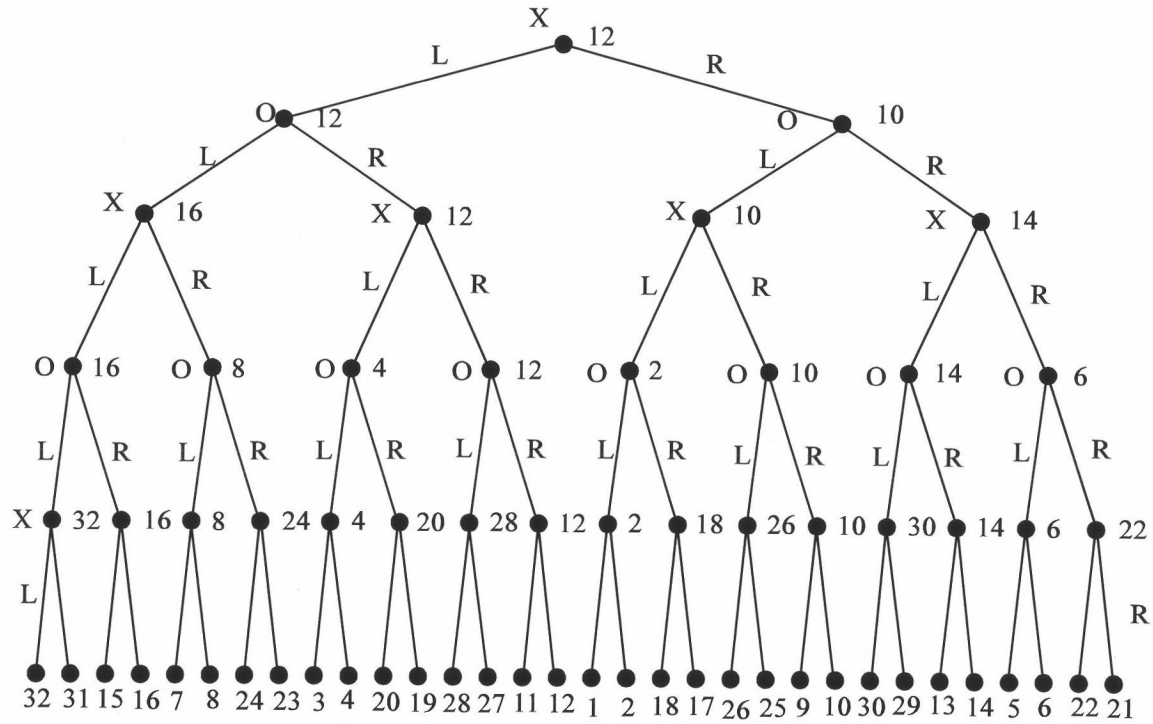
ซึ่งสามารถเขียนแทนด้วยฟังก์ชันดังนี้

$$\frac{\frac{2y-0.066}{-0.365} \quad \frac{y}{+0.124}}{e^x+y-0.579}$$

และจากการทดลองพบว่าค่าความเป็นไปได้ของความสำเร็จของผู้ไล่ล่ามีค่าเท่ากับ 55% หลังจากวิวัฒนาการถึงรุ่นที่ 51 แล้ว

6.4.3 <u>ยุทธวิธีสำหรับการเล่นเกม</u> เกมที่มีผู้เล่น 2 คนแต่ละคนผลัดกันเล่นโดยในการเล่นแต่ละครั้งผู้เล่นจะเลือกเพียงแค่ว่าจะเลือกด้านซ้ายหรือด้านขวา ผลตอบแทนที่ได้จากการเลือกแต่ละครั้งตามโครงสร้างต้นไม้ดังรูปที่ 6.3 โดยที่ผู้เล่นคนแรก(แทนด้วย X)ได้เลือกเดินสามครั้งและผู้เล่นคนที่สอง(แทนด้วย O)ได้เลือกเดินสองครั้ง และหลังจากเดินเสร็จแล้วผู้เล่นคนแรกจะได้ค่าตอบแทนตามที่ได้เลือกเดินซึ่งผู้เล่นคนที่สองจะเป็นผู้เสียให้

การเลือกเดินของแต่ละผู้เล่นจะขึ้นกับข้อมูลจากการเคลื่อนที่ผ่านมา ข้อมูลต่างๆของการเคลื่อนที่จะถูกเก็บอยู่ใน 5 ตัวแปร ดังต่อไปนี้ XM1 (การเดินของ X ครั้งที่ 1),



รูปที่ 6.3 ต้นไม้แสดงผลตอบแทนของการเลือกเดินแต่ละครั้ง

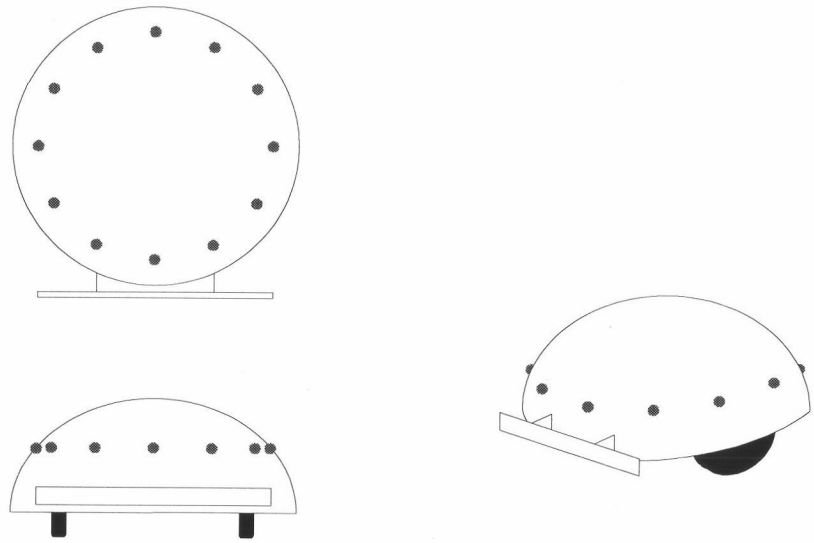
OM1(การเดินของ O ครั้งที่ 1), XM2(การเดินของ X ครั้งที่ 2), OM2(การเดินของ O ครั้งที่ 2) และ XM3(การเดินของ X ครั้งที่ 3) โดยจะเก็บค่าที่เป็นไปได้เพียง 3 ค่าเท่านั้นคือ L(ซ้าย), R(ขวา) และ U (ไม่ได้กำหนด) ตัวแปรทั้งหมดจะถูกกำหนดให้มีค่าเป็น U ก่อนตอนเริ่มเล่นเกม และจะกำหนดค่าให้กับแต่ละตัวแปรเมื่อได้รับการเลือกทางเดินแล้ว

ยุทธวิธีที่ผู้เล่นแต่ละคนจะต้องใช้ได้กับทุกๆสถานะการของการเล่นไม่ว่าจะเป็นการเล่นครั้งแรก หรือการเล่นครั้งที่สอง, สาม, สี่ และ ห้า และค่าความเหมาะของแต่ละยุทธวิธีจะวัดจากค่าตอบรวมที่ได้จากการเล่นในทุกรูปแบบการเดินของผู้เล่นอีกคนหนึ่งของเกม นั้นคือยุทธวิธี สำหรับผู้เล่นคนแรกจะทดสอบกับการเดิน 4 รูปแบบของผู้เล่นคนที่สองสำหรับการเลือกเดินครั้งที่ 2 และครั้งที่ 4 ในทำนองเดียวกันยุทธวิธี สำหรับผู้เล่นคนที่สองจะทดสอบกับการเดิน 16 รูปแบบของผู้เล่นคนแรกสำหรับการเลือกเดินครั้งที่ 1 ครั้งที่ 3 และครั้งที่ 5

สำหรับปัญหานี้กำหนดให้

Function set = {CXM1, COM1, CXM2, COM2}

Terminal set = {L, R}



รูปที่ 6.4 ภาพหุ่นยนต์จำลองที่ถูกใช้ในการทดลองหุ่นยนต์ผลักกล่อง

มีรูปแบบการใช้คำสั่งดังต่อไปนี้ CXM1(arg1,arg2,arg3) หมายถึง จะตรวจค่าของตัวแปร XM1 ถ้ามีค่าเป็น U จะทำตาม arg1 ถ้ามีค่าเป็น L จะทำตาม arg2 และถ้ามีค่าเป็น R จะทำตาม arg3 ซึ่งฟังก์ชัน COM1, CXM2 และ COM2 มีลักษณะเช่นเดียวกัน ส่วน L คือเลือกเดินทางซ้าย และ R คือเลือกเดินทางขวา

ยุทธวิธีที่ดีที่สุดสำหรับผู้เล่นคนแรก ได้จากรุ่นที่ 6 ของการวิวัฒนาการได้ค่าตอบแทนเท่ากับ 88 ยุทธวิธีที่ได้คือ

(COM2 (COM1 (COM1 L (CXM2 R (COM2 L L L) (CXM1 L R L))
(CXM1 L L R)) L R) L (COM1 L R R))

สามารถเขียนเป็นอย่างง่ายคือ

(COM2 (COM1 L L R) L R)

ยุทธวิธีที่ดีที่สุดสำหรับผู้เล่นคนที่สอง ได้จากรุ่นที่ 9 ของการวิวัฒนาการได้ค่าตอบแทนเท่ากับ 52 ยุทธวิธีที่ได้คือ

(CXM2 (CXM1 L (COM1 R L L) L) (COM1 R L (CXM2 L L R))
(COM1 L R (CXM2 R (COM1 L L R) (COM1 R L R))))

สามารถเขียนเป็นอย่างง่ายคือ

(CXM2 (CXM1 # R L) L R)

สัญลักษณ์ # แทนเหตุการณ์ที่ไม่มีทางเกิดขึ้น

6.5 การทดลองหุ่นยนต์ผลักกล่อง

Koza และ Rice (1992) ได้แสดงให้เห็นวิธีการนำเอาการโปรแกรมพันธุการไปใช้เพื่อสังเคราะห์โปรแกรมที่สามารถใช้ควบคุมหุ่นยนต์ที่จำลองขึ้นให้ทำงานที่กำหนดให้ได้อย่างเหมาะสม โดยทำการเปรียบเทียบกับการใช้วิธี การเรียนรู้แบบเสริมกำลัง(Reinforcement Learning) ที่มีลักษณะการสร้างโปรแกรมตามสถาปัตยกรรมแบบเพิ่มพูน ซึ่งต้องมีการคำนวณในทุกขั้นตอน จากการทดลองแสดงให้เห็นว่าการโปรแกรมพันธุการสามารถใช้ได้ผลใกล้เคียงสำหรับการเข้าถึงเป้าหมายที่กำหนดได้ โดยไม่ต้องใช้การโปรแกรมที่มีลักษณะเฉพาะเจาะจงตายตัว และยังแสดงให้เห็นว่ากระบวนการทำงานของการโปรแกรมพันธุการจะถูกบังคับโดยค่าความเหมาะซึ่งจะคล้อยตามธรรมชาติของการทำงานของโปรแกรม และโครงสร้างการเรียนรู้ของตัวมันเอง

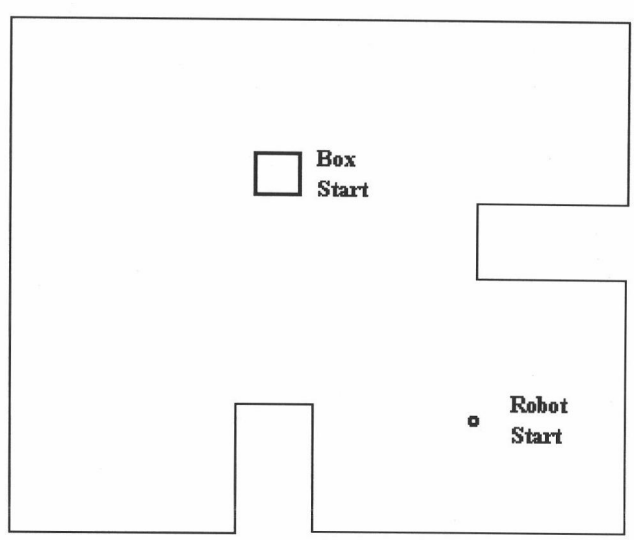
การทดลองใช้หุ่นยนต์จำลอง มีลักษณะดังรูปที่ 6.4 ซึ่งมีเครื่องตรวจจับการชนและการติดขัด(Bump & Stuck detector) และ มีเครื่องตรวจวัดโซน่า(Sonar Sensor) 12 ตัวรอบๆ ตัวหุ่นซึ่งจะวางไว้ทุกๆ 30 องศาเทียบกับด้านหน้าของหุ่นยนต์ โดยจะให้ค่าเป็นระยะห่างจากสิ่ง-กีดขวาง(ทั้งกำแพงและกล่อง)ในทิศทางนั้นๆ

เป้าหมายของหุ่นยนต์ที่ถูกกำหนดในการทดลองนี้คือ ค้นหากล่องซึ่งวางอยู่กลางห้องที่มีรูปร่างที่ได้กำหนดไว้ (ดังรูปที่ 6.5) และผลักกล่องให้ชิดผนังด้านใดด้านหนึ่ง

การทดลองกำหนดให้

Function set = {IFSTK, IFBMP, IFLTE, PROGN2, MF, TL, TR}

Terminal set = {S00, S01, S02 , S03 , ... , S11}



รูปที่ 6.5 ภาพแสดงรูปร่างของห้อง, ตำแหน่งเริ่มต้นของกล่อง และตำแหน่งเริ่มต้นของหุ่นยนต์

มีรูปแบบการใช้คำสั่งดังต่อไปนี้ IFSTK(arg1,arg2) หมายถึง ถ้าหุ่นยนต์เกิดอาการติดขัดจากการเดินคือสั่งให้เดินแล้วไม่สามารถเดินไปได้ตามคำสั่ง จะปฏิบัติการตาม arg1 และถ้าไม่ติดขัดให้ปฏิบัติการตาม arg2 , IFBMP(arg1,arg2) หมายถึง ถ้าหุ่นยนต์ชนกับสิ่งกีดขวางที่เกิดจากการเดิน จะปฏิบัติการตาม arg1 และถ้าไม่มีการชนกับสิ่งกีดขวางให้ปฏิบัติการตาม arg2 , IFLTE(arg1,arg2,arg3,arg4) หมายถึง จะนำค่าที่ได้จากการปฏิบัติการตาม arg1 และ arg2 มาเปรียบเทียบกัน ถ้า arg1 น้อยกว่าหรือเท่ากับ arg2 จะปฏิบัติการตาม arg3 และ ถ้า arg1 มากว่า arg2 ให้ปฏิบัติการตาม arg4 , PROGN2(arg1,arg2) หมายถึง ให้ปฏิบัติการตาม arg1 และปฏิบัติการตาม arg2 โดยส่งค่ากลับตาม การปฏิบัติการของ arg2 , MF() หมายถึง ให้หุ่นยนต์เดินหน้า 1 ขั้น โดยส่งค่ากลับเป็น 0 , TL() หมายถึง ให้หุ่นยนต์หมุนไปทางซ้าย 30 องศา โดยส่งค่ากลับเป็น 0 , TR() หมายถึง ให้หุ่นยนต์หมุนไปทางขวา 30 องศา โดยส่งค่ากลับเป็น 0 และ S00,S01,S02,...,S11 นั้นแทนคำสั่งให้เครื่องตรวจวัดโซน่าตัวที่ 1 ถึง 12 ส่งค่ากลับเป็นระยะห่างจากสิ่งกีดขวาง(ทั้งกำแพงและกล่อง)ในทิศทางตามหมายเลขเทียบหัวของหุ่นยนต์

ค่าความเหมาะวัดจากผลรวมของระยะห่างของกล่องจากกำแพงทั้งสี่ด้านวัดจากส่วนที่ใกล้กำแพงที่สุดของกล่องเมื่อสิ้นสุดการทำงาน ถ้ากล่องถูกผลักชิดกับผนังแล้วทำให้สิ้นสุดการทำงานค่าความเหมาะจะมีค่าเป็นศูนย์

การทดลองใช้ขนาดของประชากรหุ่นยนต์ 2000 ตัว โดยกำหนดจำนวนรุ่นที่มากที่สุดในการวิวัฒนาการเพียง 51 รุ่น สามารถสังเคราะห์โปรแกรมที่ทำงานได้ตามเป้าหมายในรุ่นที่ 20

6.6 <u>สรุปท้ายบท</u>

ในบทนี้ได้แสดงตัวอย่างของงานวิจัยที่ใช้วิธีการของขั้นตอนวิธีพันธุการหรือการโปรแกรมพันธุการเพื่อแก้ปัญหาของงานต่างๆโดยเฉพาะงานของการเรียนรู้ของหุ่นยนต์ในหลายๆ ลักษณะของงาน ซึ่งได้ผลเป็นที่น่าพอใจ ดังต่อไปนี้

พบว่าสามารถใช้การวิวัฒนาการที่ละขั้นในการการค้นหาค่าตัวเลขน้ำหนักที่เหมาะสมกับโครงสร้างของระบบเครือข่ายหน่วยประสาทเพื่อการควบคุมการทำงานของมอเตอร์ที่ใช้กับหุ่นยนต์เดินได้ทั้งการทำงานส่วนเดียวและการทำงานร่วมกัน

สามารถใช้ขั้นตอนวิธีพันธุการเพื่อค้นหาแผนการเคลื่อนที่ที่เหมาะสมสำหรับแขนหุ่นยนต์โดยอาศัยการจำลองการทำงานบนเครื่องคอมพิวเตอร์ ที่สามารถทำงานได้ดีและสามารถทำได้ง่ายกว่าการหาอินเวอร์สไคนีเมติก

สามารถหาสมการอินเวอร์สไคนีเมติกที่เหมาะสมสำหรับแขนหุ่นยนต์ที่สามารถใช้งานได้ง่ายและมีประสิทธิภาพ โดยใช้ขั้นตอนวิธีพันธุการ

พบว่าการโปรแกรมพันธุการเป็นวิธีการที่สามารถนำมาประยุกต์ใช้แก้ปัญหาต่างๆได้มากมาย โดยที่แสดงให้เห็นการแก้ปัญหาที่กำหนดในลักษณะที่แตกต่างกัน 3 ลักษณะได้อย่างมีประสิทธิภาพ

และ โดยเฉพาะงานของการเรียนรู้ของหุ่นยนต์ ได้แสดงให้เห็นว่า การโปรแกรมพันธุการถูกนำมาใช้ในการสังเคราะห์โปรแกรมเหมาะสมที่สามารถใช้ควบคุมหุ่นยนต์ที่จำลองขึ้นบนเครื่องคอมพิวเตอร์ ให้สามารถทำงานที่มีความซับซ้อนได้โดยไม่ต้องใช้วิธีการที่ยุ่งยากเกินไปและได้ผลเป็นที่น่าพอใจ