ขั้นตอนวิธีเชิงพันธุกรรมแบบบีบอัด

Compressed Genetic Algorithm

วรเศรษฐ สุวรรณิก
ภาควิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์
Email: worasait.s@ku.ac.th

นริศ กุณาศล
ภาควิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์
Email: g4764228@ku.ac.th

ประภาส จงสถิตย์วัฒนา
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
Email: prabhas@chula.ac.th

บทคัดย่อ

ขั้นตอนวิธีเชิงพันธุกรรม (GA) เป็นวิธีการค้นหาที่เลียนแบบการวิวัฒนาการของสิ่งมีชีวิต GA ค้นหาคำตอบโดยใช้ประชากรของโครโมโซมที่เป็นเลขฐานสอง เมื่อโครโมโซมมีจำนวนบิตมากขึ้น การค้นหาอาจจะต้องใช้เวลามากขึ้น ทั้งนี้เพราะปริภูมิการค้นหามีขนาดใหญ่ขึ้นเป็นทวีคูณ บทความนี้เสนอการพัฒนาประสิทธิภาพของ GA โดยใช้การลดขนาดของโครโมโซมด้วยการบีบอัดข้อมูล เรียกวิธีการนี้ว่าขั้นตอนวิธีเชิงพันธุกรรมแบบบีบอัด (Compressed GA - CGA) การวิวัฒนาการใน CGA จะทำกับโครโมโซมที่ถูกบีบอัดไว้ ดังนั้นเมื่อนำ CGA และ GA มาแก้ปัญหาเดียวกัน โครโมโซมใน CGA จะมีจำนวนบิตน้อยกว่าโครโมโซมใน GA ผลการทดลองกับปัญหา OneMax ขนาด 128 บิต พบว่า CGA สามารถทำงานได้เร็วกว่า GA ถึง 805 เท่า และสำหรับปัญหาการโปรแกรมแขนหุ่นยนต์ CGA ทำงานได้เร็วกว่า GA ถึง 4.5 เท่า

คำสำคัญ ขั้นตอนวิธีเชิงพันธุกรรม, การบีบอัด, ปัญหา OneMax, การโปรแกรมแขนหุ่นยนต์

1. บทนำ

ขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithms - GA) เป็นวิธีการค้นหาที่เลียนแบบการวิวัฒนาการของสิ่งมีชีวิตในธรรมชาติ [1] GA จะค้นหาโดยใช้กลุ่มหรือประชากรของคำตอบ การค้นหาดำเนินแบบสุ่มโดยมีค่าความเหมาะสมของคำตอบเป็นตัวกำหนดทิศทาง คำตอบแต่ละตัวจะถูกเข้ารหัส (encode) อยู่ในรูปของโครโมโซม (chromosome) ที่เป็นเลขฐานสอง โครโมโซมแต่ละตัวในกลุ่มประชากรเดียวกันจะมีจำนวนบิตเท่ากัน

GA มักจะถูกใช้ในการหาคำตอบของปัญหาที่มีปริภูมิการค้นหาที่ใหญ่เกินกว่าจะหาโดยการค้นทุกรูปแบบ (exhaustive search) ขนาดของปริภูมิการค้นหาเป็นปัจจัยหนึ่งที่กำหนดความยากง่ายของปัญหา [2] ปัญหาประเภทเดียวกันที่มีคำตอบที่ต้องใช้จำนวนบิตมากกว่าจะใช้เวลาในการค้นหามากกว่า ทั้งนี้เพราะการค้นหาจะต้องทำในปริภูมิการค้นหา (search space) ที่มีขนาดใหญ่กว่า

จำนวนบิตที่เพิ่มขึ้นของโครโมโซมนั้นทำให้ปริภูมิการค้นหามีขนาดใหญ่ขึ้น ยกตัวอย่างปัญหาที่สามารถเข้ารหัสด้วยเลข 10 บิต จะมีคำตอบที่เป็นไปได้ทั้งหมด 2^{10} หรือ 1,024 แบบ แต่ถ้าปัญหาที่ต้องเข้ารหัสด้วยเลข 30 บิต นั้นจะมีคำตอบที่เป็นไปได้ทั้งหมด 2^{30} หรือ 1,073,741,824 แบบ จำนวนบิตที่เพิ่มขึ้นเพียง 20 บิตทำให้ปริภูมิการค้นหาใหญ่ขึ้นกว่าเดิมถึง 1,048,576 เท่า

บทความนี้เสนอวิธีที่ช่วยให้ขั้นตอนวิธีเชิงพันธุกรรมแก้ปัญหาได้เร็วขึ้นโดยใช้การบีบอัด (compress) โครโมโซมให้มีจำนวนบิตน้อยลง การวิวัฒนาการจะกระทำกับโครโมโซมที่ถูกบีบอัด ในขณะที่ค่าความเหมาะสมของโครโมโซมจะวัดกับโครโมโซมที่ถูกคลาย (extract) เรียกขั้นตอนวิธีดังกล่าวว่าขั้นตอนวิธีเชิงพันธุกรรมแบบบีบอัด (Compressed Genetic Algorithm - CGA)

2. ขั้นตอนวิธีเชิงพันธุกรรม

ขั้นตอนวิธีเชิงพันธุกรรมสามารถเขียนในรูปชุดคำสั่งเทียมได้ดังนี้

1. สร้างประชากรรุ่นแรกโดยการสุ่ม
2. วัดค่าความเหมาะสมของโครโมโซมแต่ละตัวในประชากรที่เพิ่งถูกสร้าง
3. สร้างประชากรรุ่นใหม่โดยการเลือกโครโมโซมที่ดีมาทำการไขว้เปลี่ยน (crossover), กลายพันธุ์ (mutation) หรือ สืบพันธุ์ (reproduction)
4. ทำซ้ำขั้นตอนที่ 2 และ 3 จนกระทั่งพบคำตอบที่ต้องการหรือถึงรุ่นการวิวัฒนาการที่กำหนด

2.1 ประชากร

ขั้นตอนวิธีเชิงพันธุกรรมค้นหาโดยใช้ประชากรของโครโมโซม โครโมโซมจะเป็นเลขฐานสองที่มีขนาดคงที่ดังแสดงในรูปที่ 1 การสร้างประชากรรุ่นแรกทำโดยการสุ่มเลขฐานสองขึ้นมา ในขณะที่การสร้างประชากรรุ่นถัดไปในขั้นตอนที่ 3 จะมีประชากรเพียงบางส่วนเท่านั้นที่จะถูกคัดเลือกเพื่อให้กำเนิดประชากรในรุ่นถัดไป โดยขั้นตอนวิธีเชิงพันธุกรรมจะคัดเลือกโครโมโซมที่มีค่าความเหมาะสมสูง

0	1	1	1	0	1	0	1

รูปที่ 1. โครโมโซม

2.2 การวัดค่าความเหมาะสม

เนื่องจากโครโมโซมเป็นตัวเลขฐานสองที่ใช้นำเสนอคำตอบของปัญหาต่างๆกัน เราจึงต้องนำโครโมโซมมาถอดรหัส (decode) เสียก่อน การถอดรหัสคือการแปลงเลขฐานสองให้อยู่ในรูปแบบที่เราสามารถนำไปใช้งานอย่างอื่นต่อได้ เช่น อาจจะแปลเลขฐานสองที่ต่อเรียงกันให้เป็นเลขจำนวนเต็ม เลขทศนิยม หรือโปรแกรมหุ่นยนต์เป็นต้น

2.3 การคัดเลือก

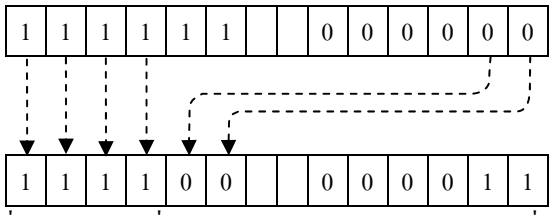
การคัดเลือกสามารถทำได้หลายวิธี แต่ละวิธีมีแนวความคิดคล้ายๆกันคือเลือกโครโมโซมที่มีค่าความเหมาะสมสูงเพื่อสร้างเป็นโครโมโซมของประชากรในรุ่นถัดไป บทความนี้ใช้การเลือกโดยการประลอง (tournament selection) การคัดเลือกวิธีนี้จะสุ่มประชากรขึ้นมาจำนวนหนึ่ง เช่น อาจจะสุ่มมา 4 ตัว และเลือกโครโมโซมที่มีค่าความเหมาะสมสูงที่สุดในกลุ่มโครโมโซมที่ถูกสุ่มขึ้นมา

2.4 ตัวดำเนินการทางพันธุกรรม

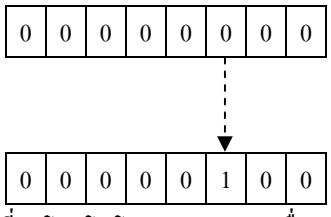
การสร้างประชากรในรุ่นถัดไปทำได้โดยใช้ตัวดำเนินการทางพันธุกรรม (genetic operator) กับโครโมโซมที่ถูกเลือก ตัวดำเนินการทางพันธุกรรมได้แก่

- **การไขว้เปลี่ยน** เป็นการนำเอาโครโมโซม 2 ตัว มาผสมกันเพื่อให้ได้โครโมโซมใหม่ โครโมโซมตัวใหม่อาจเกิดจากการนำเอาส่วนแรกของโครโมโซมในรุ่นก่อนตัวหนึ่งมาต่อกับส่วนหลังของโครโมโซมรุ่นก่อนอีกตัวหนึ่ง การไขว้เปลี่ยนแสดงในรูปที่ 2
- **การกลายพันธุ์** เป็นการสร้างโครโมโซมตัวใหม่โดยนำโครโมโซมจากรุ่นก่อนมาปรับเปลี่ยนส่วนใดส่วนหนึ่งอย่างสุ่ม การกลายพันธุ์แสดงในรูปที่ 3

- **การสืบพันธุ์** เป็นการสร้างโครโมโซมตัวใหม่โดยนำโครโมโซมจากรุ่นก่อนไปสร้างเป็นโครโมโซมตัวใหม่โดยไม่มีการเปลี่ยนแปลง



รูปที่ 2. การไขว้เปลี่ยนโครโมโซม 111111 และ 000000 ที่ตำแหน่งที่ 4 คือโครโมโซม 111100 และ 000011



รูปที่ 3 โครโมโซม 00000000 เมื่อกลายพันธุ์ที่ตำแหน่ง 6 จะได้เป็นโครโมโซม 00000100

3. ขั้นตอนวิธีเชิงพันธุกรรมแบบบีบอัด

ขั้นตอนวิธีเชิงพันธุกรรมแบบบีบอัด (Compressed Genetic Algorithm หรือ CGA) มีลักษณะการทำงานคล้ายกับขั้นตอนวิธีเชิงพันธุกรรม (GA) สิ่งที่แตกต่างก็คือโครโมโซมของ CGA จะถูกบีบอัดอยู่ ก่อนที่จะวัดค่าความเหมาะสม เราต้องนำโครโมโซมมาคลายออก

3.1 การบีบอัด

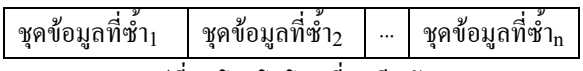
การบีบอัดข้อมูลคือการลดขนาดข้อมูลเพื่อให้ใช้พื้นที่ในการเก็บน้อยลง หรือประหยัดเวลาที่ใช้ส่งข้อมูล อัลกอริทึมที่ใช้บีบอัดข้อมูลมีมากมาย และให้ผลการบีบอัดที่มีรูปแบบแตกต่างกัน ใน CGA การบีบอัดทำโดยสังเกตการซ้ำกันของช่วงข้อมูล โครโมโซมใน CGA เกิดจากการต่อเรียงกันของชุดข้อมูลที่ซ้ำกันดังแสดงในรูปที่ 4 รูปแบบของชุดข้อมูลที่ซ้ำประกอบด้วย 3 ส่วนดังแสดงในรูปที่ 5 ความยาวของช่วงที่ซ้ำและจำนวนช่วงที่ซ้ำใช้พื้นที่ในการเก็บอย่างละ 4 บิต ส่วนจำนวนบิตของช่วงที่ซ้ำนั้นขึ้นอยู่กับค่าของความยาวของช่วงที่ซ้ำ

ยกตัวอย่าง เลข 100000010000001000000 (จำนวน 21 บิต) เกิดจากการซ้ำกัน 3 ครั้งของเลข 1000000 ข้อมูลชุดที่ซ้ำกันดังกล่าวจะถูกเก็บโดยใช้พื้นที่แค่ 15 บิตดังแสดงในรูปที่ 6

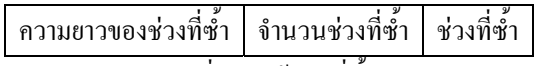
3.2 การคลาย

เนื่องจากโครโมโซมใน CGA นั้นถูกบีบอัดอยู่ ก่อนที่จะวัดค่าความเหมาะสมจึงต้องมีการคลายข้อมูลออกมาก่อน การคลายข้อมูลสามารถทำได้อย่างรวดเร็ว โดยจะคลายข้อมูลทีละชุด

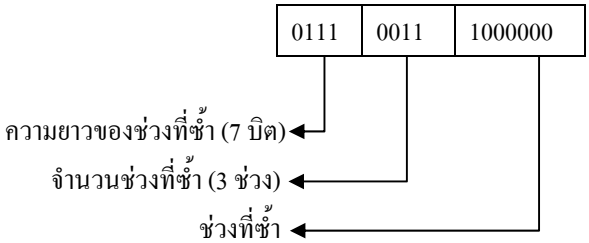
ยกตัวอย่าง ดังแสดงในรูปที่ 7 โครโมโซม 010000101000001101111100 นั้นประกอบด้วยชุดของข้อมูลที่ซ้ำกัน 2 ชุด คือ 010000101000 และ 00110111100 รูปที่ 8 แสดงส่วนต่างๆของชุดข้อมูลที่ซ้ำชุดแรก เมื่อนำมาคลายออกจะได้ 10001000 รูปที่ 9 แสดงส่วนต่างๆของชุดข้อมูลที่ซ้ำชุดที่สอง เมื่อนำมาคลายออกจะได้ 100100100100100100100 และเมื่อนำผลการคลายของข้อมูลทั้งสองชุดมาเขียนต่อกัน จะได้ 1000100010010010010010010100 ซึ่งเราจะนำเลขนี้ไปวัดค่าความเหมาะสมต่อไป



รูปที่ 4. โครโมโซมที่ถูกบีบอัด



รูปที่ 5. ชุดข้อมูลที่ซ้ำ



รูปที่ 6. ตัวอย่างชุดข้อมูลที่ซ้ำ สำหรับชุดข้อมูลนี้เมื่อคลายแล้วจะได้ผลคือ 100000010000001000000

010000101000	00110111100

รูปที่ 7. ตัวอย่างโครโมโซมของ CGA ที่ประกอบด้วยชุดข้อมูลที่ซ้ำ 2 ชุด

0100	0010	1000

รูปที่ 8. ชุดข้อมูลที่ซ้ำชุดแรก ช่วงที่ซ้ำคือ 1000 ซึ่งมีความยาว 4 หรือ 0100_2 บิต และมีการซ้ำกัน 2 หรือ 0010_2 ครั้ง

0011	0111	100

รูปที่ 9. ชุดข้อมูลที่ซ้ำชุดที่สอง ช่วงที่ซ้ำคือ 100 ซึ่งมีความยาว 3 หรือ 0011_2 บิต และมีการซ้ำกัน 7 หรือ 0111_2 ครั้ง

3.3 การสร้างประชากรรุ่นแรก

โครโมโซมในรุ่นแรกของ CGA ถูกสร้างขึ้นโดยการสุ่ม รูปแบบการบีบอัดที่ใช้ใน CGA ทำให้โครโมโซมสามารถนำมาคลายได้ ไม่ว่าจะถูกสุ่มขึ้นมาเป็นแบบใดก็ตาม ซึ่งต่างจากการสร้างโครโมโซมที่บีบอัดโดยใช้อัลกอริทึมบางอย่างที่ให้ผลการบีบอัดที่มีรูปแบบที่เคร่งครัด เช่น มีการใช้พจนานุกรม, มีการใช้รหัสเติมหน้า (prefix code) หรือมีส่วนของผลรวมตรวจสอบ (checksum) [3]

3.4 การวัดค่าความเหมาะสม

โครโมโซมใน CGA มีจำนวนบิตน้อยกว่าโครโมโซมใน GA ที่ใช้แก้ปัญหาเดียวกันเพราะว่าโครโมโซมใน CGA ถูกบีบอัดไว้ ดังนั้นก่อนที่จะวัดค่าความเหมาะสมเราต้องนำโครโมโซมมาคลายออกก่อน บางครั้งผลการคลายอาจจะได้จำนวนบิตน้อยกว่าโครโมโซมที่ใช้แก้ปัญหาด้วย GA ซึ่งในกรณีนั้นจะเติม 0 เข้าไป สำหรับกรณีที่คลายแล้วได้จำนวนบิตมากกว่าโครโมโซมที่ใช้แก้ปัญหาด้วย GA เราจะตัดส่วนที่เกินทิ้งไป

3.5 การคัดเลือกและการใช้ตัวดำเนินการทางพันธุกรรม

CGA สามารถใช้วิธีการคัดเลือกและตัวดำเนินการทางพันธุกรรมแบบเดียวกับ GA ตัวดำเนินการทางพันธุกรรมจะกระทำกับโครโมโซมที่ถูกบีบอัดอยู่

4. การทดลอง

เราเปรียบเทียบประสิทธิภาพของ CGA กับ GA ในการแก้ปัญหา OneMax และปัญหาแขนหุ่นยนต์ เนื่องจากการทำงานของ CGA และ GA เป็นแบบสุ่ม เพื่อความน่าเชื่อถือของผลการทดลอง แต่ละปัญหาจะทดลองซ้ำ 12 ครั้ง และตัดผลการทดลองที่ดีที่สุดและแย่ที่สุดทิ้งไป และรายงานผลด้วยค่าเฉลี่ยของผลการทดลองที่เหลือ 10 ครั้ง

4.1 ปัญหา OneMax

งานวิจัยหลายชิ้นในด้าน GA นิยมทดลองกับปัญหา OneMax [4] เพราะปัญหา OneMax เป็นปัญหาที่เข้าใจและวิเคราะห์ได้ง่าย ปัญหา OneMax คือ ปัญหาที่ว่าด้วยการนับจำนวนเลข 1 ในโครโมโซมโดยค่าความเหมาะสมที่ได้ก็คือ จำนวนเลข 1 ทั้งหมดที่พบในโครโมโซมนั้น ดังนั้นโครโมโซมที่ดีที่สุดคือโครโมโซมที่ทุกบิตมีค่าเป็นหนึ่ง

การทดลองนี้ใช้ GA แก้ปัญหา OneMax ขนาด 128 บิต และใช้ CGA แก้ปัญหา OneMax ขนาด 128, 256 และ 512 บิต สาเหตุที่ไม่ใช้ GA แก้ปัญหา OneMax ขนาด 256 และ 512 บิต เนื่องจากผลการทดลองเบื้องต้นแสดงให้เห็นว่า GA ใช้เวลาในการค้นหาคำตอบนานมาก

พารามิเตอร์ที่ใช้ในการแก้ปัญหา OneMax ของ GA และ CGA แสดงในตารางที่ 1 และตารางที่ 2 ตามลำดับ โครโมโซมที่บีบอัดของ CGA จะมีจำนวนบิตที่น้อยกว่าขนาดของปัญหา 4 เท่า

พารามิเตอร์	ค่า
จำนวนประชากร	1,000 ตัว
วิธีการคัดเลือก	แบบประลอง โครโมโซมที่ประลองกันมี 4 ตัว
อัตราการไขว้เปลี่ยน	80 %
วิธีการไขว้เปลี่ยน	แบบจุดเดียว (single point)
อัตราการกลายพันธุ์	5 %
เก็บโครโมโซมที่ดีสุดไว้	10 ตัว

ตารางที่ 1. พารามิเตอร์ที่ใช้ในการแก้ปัญหา OneMax ขนาด 128 บิตโดยใช้ GA

พารามิเตอร์	ค่า
จำนวนบิตของโครโมโซม สำหรับปัญหา OneMax ขนาด 128, 256 และ 512 บิต ตามลำดับ	32, 64, และ 128 บิต
จำนวนประชากร	1,000 ตัว
วิธีการคัดเลือก	แบบประลอง โครโมโซมที่ประลองกันมี 4 ตัว
อัตราการไขว้เปลี่ยน	80 %
วิธีการไขว้เปลี่ยน	แบบจุดเดียว (single point)
อัตราการกลายพันธุ์	5 %
เก็บโครโมโซมที่ดีสุดไว้	10 ตัว

ตารางที่ 2. พารามิเตอร์ที่ใช้ในการแก้ปัญหา OneMax ขนาด 128, 256 และ 512 บิตโดยใช้ CGA

พารามิเตอร์	ค่า
จำนวนประชากร	1,000 ตัว
วิธีการคัดเลือก	แบบประลอง โครโมโซมที่ประลองกันมี 4 ตัว
อัตราการไขว้เปลี่ยน	80 %
วิธีการไขว้เปลี่ยน	แบบจุดเดียว (single point)
อัตราการกลายพันธุ์	5 %
เก็บโครโมโซมที่ดีสุดไว้	10 ตัว

ตารางที่ 3. พารามิเตอร์ที่ใช้ในการแก้ปัญหาแขนหุ่นยนต์โดยใช้ GA และ CGA

4.2 ปัญหาแขนหุ่นยนต์

ขั้นตอนวิธีเชิงพันธุกรรมสามารถนำไปใช้แก้ปัญหาหุ่นยนต์ได้ [5] รูปที่ 10 และรูปที่ 11 แสดงแขนหุ่นยนต์จำลองในสิ่งแวดล้อมการทำงาน แขนหุ่นยนต์จำลองมีข้อต่อทั้งหมด 8 ข้อ งานของแขนหุ่นยนต์คือขยับให้ปลายแขนไปหาเป้าหมาย โดยในสิ่งแวดล้อมการทำงานจะมีสิ่งกีดขวางวางไว้สามเส้น แขนหุ่นยนต์ไม่สามารถเคลื่อนที่ผ่านสิ่งกีดขวางได้

โครโมโซมของ GA จะเข้ารหัสคำสั่งในการขยับแขนหุ่นยนต์ไว้ โครโมโซมประกอบด้วยคำสั่งเรียงต่อกันดังแสดงในรูปที่ 12 โดยแต่ละคำสั่งประกอบด้วยหมายเลขข้อต่อที่จะขยับและทิศทางดังแสดงในรูป 13 แต่ละคำสั่งมีความยาว 4 บิต โดย 3 บิตแรกใช้เก็บหมายเลขของข้อต่อตั้งแต่ 000 ถึง 111 และบิตสุดท้ายใช้เก็บทิศทาง ว่าต้องการหมุนข้อต่อทวนเข็มนาฬิกาหรือว่าตามเข็มนาฬิกา การหมุนหนึ่งครั้งเป็นมุม 15 องศา

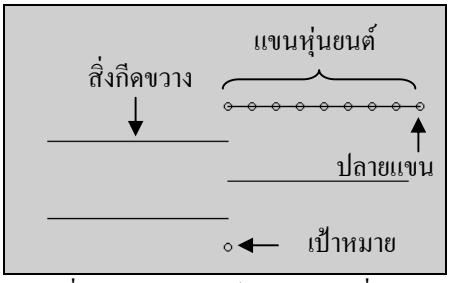
เนื่องจากโครโมโซมมี 600 คำสั่ง และแต่ละคำสั่งยาว 4 บิต โครโมโซมที่ GA ใช้จึงมีความยาว 2,400 บิต สำหรับโครโมโซมที่บีบอัดของ CGA จะถูกกำหนดให้มีความยาว

เพียง 600 บิต เพราะโครโมโซมใน CGA เก็บคำสั่งที่ถูกบีบอัดไว้

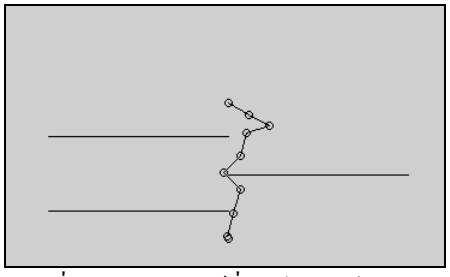
การวัดความเหมาะสมของโครโมโซมทำได้โดยการทำงานตามคำสั่งในโครโมโซมจนครบ 600 คำสั่ง หรือจนกว่าปลายแขนจะถึงเป้าหมาย หลังจากนั้นจึงวัดระยะทางจากปลายแขนไปหาเป้าหมาย ถ้าระยะห่างมากจะได้ค่าความเหมาะสมน้อย แต่ถ้าระยะห่างน้อยจะได้ค่าความเหมาะสมมาก ค่าความเหมาะสม f ถูกนิยามไว้ดังนี้

$$f = \begin{cases} 1000 & ; \text{ถ้าระยะจากปลายแขนถึงเป้าหมาย} < 5 \\ 1000 - \text{ระยะจากปลายแขนถึงเป้าหมาย} & ; \text{อื่นๆ} \end{cases}$$

ตารางที่ 3 แสดงพารามิเตอร์ที่ใช้ในการแก้ปัญหาแขนหุ่นยนต์โดยใช้ GA และ CGA



รูปที่ 10. แขนหุ่นยนต์ในตำแหน่งเริ่มต้น



รูปที่11. แขนหุ่นยนต์ที่ขยับไปถึงเป้าหมาย

คำสั่ง$_1$	คำสั่ง$_2$	คำสั่ง$_3$	...	คำสั่ง$_{600}$

รูปที่ 12. โครโมโซมของโปรแกรมแขนหุ่นยนต์ประกอบด้วยคำสั่งต่อเรียงกัน

หมายเลขข้อต่อที่จะขยับ	ทิศทาง

รูปที่ 13. แต่ละคำสั่งในของโปรแกรมแขนหุ่นยนต์ประกอบด้วย 2 ส่วนคือหมายเลขข้อต่อที่จะขยับ (3 บิต) และทิศทางการขยับ (1 บิต)

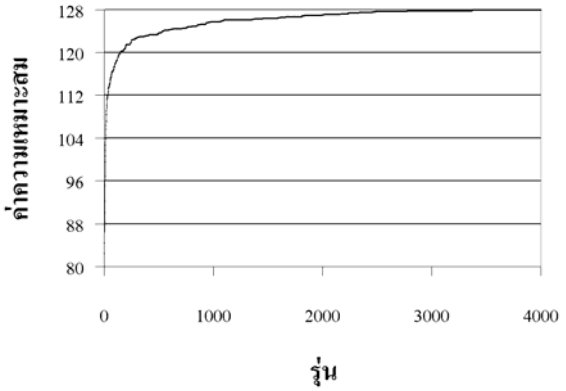
5. ผลการทดลอง

5.1 ปัญหา OneMax

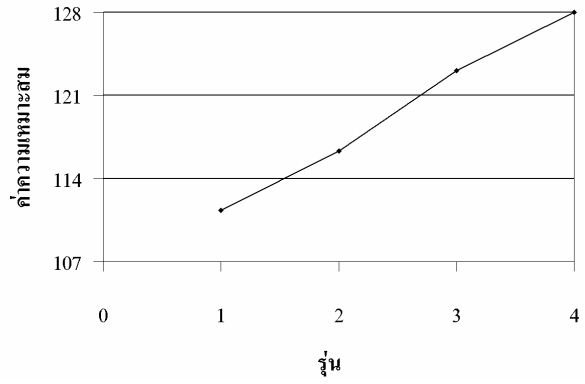
เราเปรียบเทียบประสิทธิภาพของ GA และ CGA ในการแก้ปัญหา OneMax ขนาด 128 บิต รูปที่ 14 และ 15 แสดงค่าความเหมาะสมเฉลี่ยของโครโมโซมที่ดีที่สุดของแต่ละรุ่นของการวิวัฒนาการด้วย GA และ CGA ตามลำดับ จากรูปจะเห็นว่า CGA สามารถแก้ปัญหาได้โดยใช้จำนวนรุ่นน้อยกว่า GA ซึ่งเมื่อนำจำนวนรุ่นที่ GA และ CGA ใช้แก้ปัญหามาหาค่าเฉลี่ย จะพบว่า CGA ใช้จำนวนรุ่นในการหาคำตอบน้อยกว่าของ GA ถึง 805 เท่า โดยที่ GA ใช้ 2,416 รุ่น และ CGA ใช้ 3 รุ่น ดังแสดงในตารางที่ 4

เพื่อที่จะทดสอบความสามารถของ CGA ในการแก้ปัญหาที่ยากขึ้น เราจึงทดลองใช้ CGA แก้ปัญหา OneMax ขนาด 256 และ 512 บิต ผลการวิวัฒนาการแสดงในรูปที่ 16 และ 17 ตามลำดับ ปัญหา OneMax ที่มีขนาดใหญ่ขึ้นต้องใช้เวลาแก้มากขึ้น จำนวนบิตของปัญหา OneMax กับจำนวนครั้งการวัดค่าความเหมาะสมของโครโมโซมจนกว่าจะพบคำตอบ มีความสัมพันธ์เป็นแบบชี้กำลังดังแสดงในรูปที่ 18

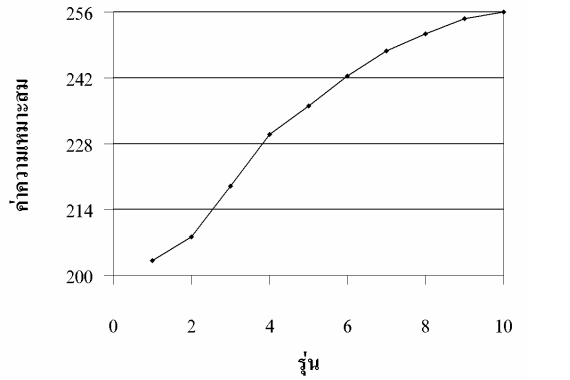
โครโมโซมของ CGA มีขนาดเล็กกว่า GA โครโมโซมที่ถูกบีบอัดมีขนาดเท่ากับ 25% ของโครโมโซมที่ใช้ใน GA และเมื่อนำโครโมโซมที่เป็นคำตอบจาก CGA มาคลายแล้วพบว่าจำนวนบิตที่ใช้จริงนั้นน้อยกว่าที่จองไว้ จำนวนบิตที่เป็นคำตอบจาก CGA แสดงในตารางที่ 5 คำตอบของปัญหา OneMax ที่ CGA ค้นพบ ใช้พื้นที่ประมาณ 15% ของ GA



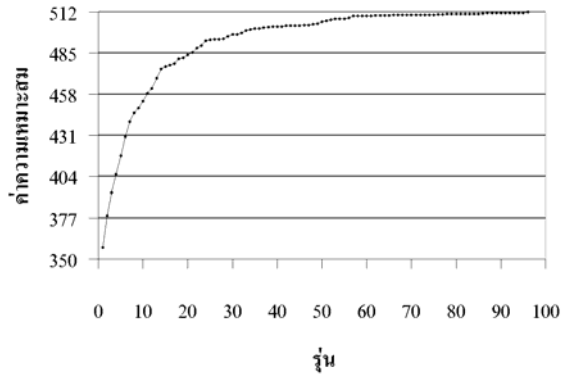
รูปที่14. ค่าความเหมาะสมเฉลี่ยของโครโมโซมที่ดีที่สุดเมื่อใช้ GA แก้ปัญหา OneMax ขนาด 128 บิต



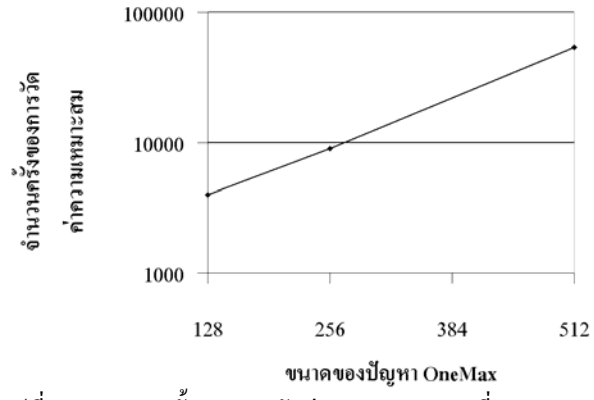
รูปที่ 15. ค่าความเหมาะสมของโครโมโซมที่ดีที่สุดเมื่อใช้ CGA แก้ปัญหา OneMax ขนาด 128 บิต



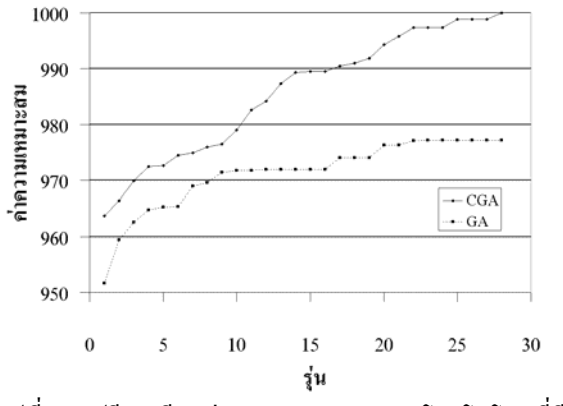
รูปที่ 16. ค่าความเหมาะสมของโครโมโซมที่ดีที่สุดเมื่อใช้ CGA แก้ปัญหา OneMax ขนาด 256 บิต



รูปที่ 17. ค่าความเหมาะสมของโครโมโซมที่ดีที่สุดเมื่อใช้ CGA แก้ปัญหา OneMax ขนาด 512 บิต



รูปที่ 18. จำนวนครั้งของการวัดค่าความเหมาะสมที่ CGA ใช้ในการแก้ปัญหา OneMax ที่ขนาดต่างๆ



รูปที่ 19. เปรียบเทียบค่าความเหมาะสมของโครโมโซมที่ดีที่สุดเมื่อใช้ CGA และ GA แก้ปัญหาแขนหุ่นยนต์

1000010010111001000010101010100001011000101101101100000110101000001001111011110100100000001000001101111110011100011100010101001101110000011010110010...

รูปที่ 20. ส่วนของคำตอบของโปรแกรมแขนหุ่นยนต์ที่ได้จาก GA

ขนาดของปัญหา OneMax	ขั้นตอนวิธี	จำนวนรุ่น
128	GA	2,416
	CGA	3
256	CGA	9
512	CGA	54

ตารางที่ 4. จำนวนรุ่นที่ใช้แก้ปัญหา OneMax

ขนาดของปัญหา OneMax	จำนวนบิตของคำตอบที่บีบอัด	ขนาดลดลง (%)
128	19	85.16
256	39	84.77
512	76	85.16

ตารางที่ 5. จำนวนบิตที่ CGA ใช้เพื่อแก้ปัญหา OneMax

5.2 ปัญหาแขนหุ่นยนต์

เราเปรียบเทียบการวิวัฒนาการโปรแกรมแขนหุ่นยนต์โดยใช้ GA และ CGA การวิวัฒนาการจะสิ้นสุดลงเมื่อพบกับโปรแกรมที่สามารถขยับปลายแขนของหุ่นยนต์ไปหาเป้าหมายได้ รูปที่ 19 แสดงค่าความเหมาะสมของโครโมโซมที่ดีที่สุดเมื่อการวิวัฒนาการด้วย GA และ CGA ได้ดำเนินไป ค่าความเหมาะสมของโครโมโซมที่ดีที่สุดใน CGA ดีกว่าของ GA ตั้งแต่รุ่นแรกถึงแม้ว่าประชากรของทั้งสองวิธีถูกสร้างขึ้นโดยการสุ่มเหมือนกัน CGA พบคำตอบเร็วกว่า GA โดยจำนวนรุ่นเฉลี่ยที่ GA ใช้ในการหาคำตอบคือ 59.6 รุ่น ในขณะที่ CGA ใช้ 13.3 รุ่น หรือ CGA สามารถแก้ปัญหาแขนหุ่นยนต์ได้เร็วกว่า GA ถึง 4.5 เท่า

6. สรุป

งานวิจัยนี้เปรียบเทียบผลการทำงานของขั้นตอนวิธีเชิงพันธุกรรม (GA) กับขั้นตอนวิธีเชิงพันธุกรรมแบบบีบอัด (CGA) โดยทดลองกับปัญหา OneMax และปัญหาแขนหุ่นยนต์ ผลการทดลองกับปัญหา OneMax ขนาด 128 บิต CGA ทำงานเร็วกว่า GA ถึง 805 เท่า และเมื่อทดลองกับปัญหาแขนหุ่นยนต์ CGA จะทำงานเร็วกว่า GA ถึง 4.5 เท่า

สาเหตุที่ CGA ทำงานได้ดีในปัญหา OneMax มากกว่าปัญหาแขนหุ่นยนต์นั้นน่าจะเพราะปัญหา OneMax มีคำตอบที่มีความเป็นรูปแบบ (regularity) มากกว่าปัญหาแขนหุ่นยนต์ คำตอบของปัญหา OneMax เป็นโครโมโซมที่ทุกบิตเป็นเลข 1 ในขณะที่คำตอบของปัญหาแขนหุ่นยนต์นั้นมีความเป็นรูปแบบน้อยกว่าดังแสดงในรูปที่ 20

นอกจาก CGA จะทำงานได้เร็วกว่าในปัญหาทั้งสองแล้ว CGA ยังใช้งานหน่วยความจำน้อยกว่า เพราะโครโมโซมที่ใช้มีขนาดน้อยกว่า ในการทดลองกับทั้งสองปัญหา โครโมโซมของ CGA ถูกจองไว้ด้วยขนาดเพียง 1 ใน 4 ของโครโมโซมของ GA

ถึงแม้ CGA จะมีประสิทธิภาพดีกว่า GA ในปัญหาที่เราได้นำมาทดลอง แต่สำหรับบางปัญหา CGA อาจจะทำงานได้ช้ากว่าหรืออาจจะหาคำตอบไม่พบ ทั้งนี้เพราะว่าโครโมโซมของ CGA ไม่สามารถคลายออกมาแล้วเป็นโครโมโซมของ GA ได้ทุกแบบ ยกตัวอย่างเช่น โครโมโซมขนาด 40 บิต สามารถแทนทำตอบ 1,099,511,627,776 คำตอบ แต่ถ้าต้องการบีบอัดให้เหลือแค่

10 บิต จะนำเสนอได้ 1,024 คำตอบ ดังนั้นจะมีคำตอบมากมายในเซ็ตของคำตอบขนาด 40 บิต ที่คำตอบที่ถูกบีบอัดอยู่ด้วยขนาด 10 บิตไม่สามารถจับคู่ได้

7. เอกสารอ้างอิง

[1] Goldberg, D.E., The Design of Innovation, Kluwer Academic Publishers, USA, 2002.

[2] Michalewicz, Z., Fogel, D.B., How to Solve It: Modern Heuristics, 2nd edition, Springer, Germany, 2004.

[3] Wayner, P., Compression Algorithms for Real Programmers, Academic Press, 2000.

[4] Thierens D., "Scalability Problems of Simple Genetic Algorithms", Evolutionary Computation, 7(4), 331-352, 1999.

[5] Jassadapakorn, C., Chongstitvatana, P., "Reactive Planning with Evolutionary Computation", Proceedings of the National Computer Science and Engineering Conference, 357-361, 2002.