การประยุกต์ใช้ Genetic Algorithm เพื่อลดค่าใช้จ่ายในการเดินทางไปตรวจสอบสาขาของธนาคาร

(The applied Genetic Algorithm for reduce bank's branch audit travelling cost)

โดย

เมธาวี มหาอัมพรพฤกษ์

รหัส 43067051

อาจารย์ที่ปรึกษา

ผศ.ดร. อาริต ธรรมโน

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2544

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การประยุกต์ใช้ Genetic Algorithm เพื่อลดค่าใช้จ่ายในการเดินทางไปตรวจสอบสาขาของธนาคาร
นักศึกษา	นางสาวเมธาวี มหาอัมพรพฤกษ์
อาจารย์ที่ปรึกษา	ผศ.ดร.อาริต ธรรมโน
ระดับการศึกษา	วิทยาศาสตร์มหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2544

บทคัดย่อ

เนื่องจากฝ่ายตรวจสอบของทุกธนาคาร ได้มอบหมายให้พนักงานภายในฝ่ายเดินทางไปยังสาขาต่าง ๆ ทั่วประเทศ เพื่อตรวจสอบการทำงานของสาขา ตรวจสอบเอกสารของลูกค้าและเอกสารของธนาคารว่าข้อมูลมีความถูกต้องหรือไม่ และในการเดินทางแต่ละครั้ง จะต้องมีค่าใช้จ่ายค่อนข้างสูง เนื่องจากบางครั้งต้องเดินทางไปยังสาขาที่อยู่ห่างไกลและต้องเดินทางไปหลายสาขาในแต่ละครั้ง จึงได้นำ Genetic Algorithm มาประยุกต์ใช้ เพื่อคำนวณหาเส้นทางในการเดินทางที่เสียค่าใช้จ่ายน้อยที่สุด และระยะทางสั้นที่สุดเพื่อให้เกิดประโยชน์แก่ธนาคารในการลดค่าใช้จ่ายในส่วนนี้

ดังนั้น การนำเอาเทคโนโลยีสารสนเทศมาประยุกต์ใช้ให้เหมาะสม จะช่วยให้ธนาคารสามารถที่จะได้รับข้อมูลที่ เป็นประโยชน์เพื่อช่วยในการวางแผนการเดินทางได้อย่างเหมาะสม รวมทั้งประหยัดเวลาและค่าใช้จ่ายในการเดินทาง

การประยุกต์ใช้ Genetic Algorithm เพื่อลดค่าใช้จ่ายในการเดินทางไปตรวจสอบสาขาของธนาคาร เป็นการพัฒนาระบบงานสำหรับให้ธนาคารได้ใช้ข้อมูลเพื่อวางแผนการเดินทางไปตรวจสอบที่สาขา

Title	The applied Genetic Algorithm for reduce bank's branch audit travelling cost
Student	Miss Maythavee Mahaampornpreuk
Advisor	Asst.Prof.Dr. Arit Thammano
Level of study	Master of Science in Information Technology
Major	Information Science
Academic year	2001

ABSTRACT

Audit Department in each bank send auditor go to their branches to audit branches' operation, customers' documents and bank's documents those correct or not. In each travelling has more expenses because auditors go to many branches and they have a long distance. Genetic Algorithm can be applied for calculate optimum path and minimum cost that reduce some bank's expenses. Information Technology can be applied for helping bank receive useful information to plan the travelling and reduce time and cost. The applied Genetic Algorithm for reduce bank's branch audit travelling cost is a system development for bank to use data and plan the travelling to audit branches.

กิตติกรรมประกาศ

ในการศึกษาโครงการพัฒนาระบบงานการประยุกต์ใช้ Genetic Algorithm เพื่อลดค่าใช้จ่ายในการเดินทางไปตรวจสอบสาขาของธนาคาร ผู้จัดทำขอขอบพระคุณ ผศ.ดร. อาริต ธรรมโน ที่ได้ให้คำปรึกษาและแนะนำแนวทางในการพัฒนาระบบจนเสร็จสมบูรณ์ และ โครงงานนี้ได้มีผู้เกี่ยวข้อง ที่สนับสนุนและให้ความช่วยเหลือหลายท่าน ดังนี้

- ขอกราบขอบพระคุณ พ่อแม่ และขอบคุณพี่จูนและลัทธ์ ที่คอยดูแลเป็นกำลังใจ และสนับสนุนในการทำงานเสมอมา
- ขอขอบคุณคำที่ช่วยหาข้อมูลและช่วยทำการ input ข้อมูลลงใน table
- ขอขอบคุณเพื่อน ๆ IS9 สมทบ ที่คอยช่วยเหลือ ให้คำแนะนำ ตลอดระยะเวลาที่ได้ศึกษาร่วมกัน
- ขอขอบคุณเพื่อนร่วมงานที่ช่วยทำงานแทนในยามที่ผู้จัดทำลางานเพื่อมาทำโปรเจ็ค
- ขอขอบคุณทุกคนที่ไม่ได้กล่าวถึงข้างต้นที่ให้กำลังใจและช่วยเหลือตลอดมา

เมธาวี มหาอัมพรพฤกษ์

14 กุมภาพันธ์ 2545

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมา

ปัจจุบันเทคโนโลยีสารสนเทศได้เข้ามามีบทบาทกับชีวิตประจำวันของเรามากขึ้น การนำเทคโนโลยีมาประยุกต์ใช้ให้เหมาะสมจึงเป็นสิ่งที่สำคัญ และการนำเทคโนโลยีสารสนเทศมาประยุกต์ใช้ในธุรกิจก่อให้เกิดประโยชน์มากมาย เพราะปัจจุบันธุรกิจในด้านต่าง ๆ มีอัตราการแข่งขันที่สูงขึ้น โดยเฉพาะธุรกิจธนาคาร ได้มีการปรับปรุงรูปลักษณ์ให้แตกต่างกันไป แต่ละธนาคารต่างก็ลงทุนเพื่อให้ดึงดูดความสนใจของลูกค้ามากที่สุด ทำให้ต้นทุนในการผลิตของธุรกิจประเภทนี้มีแนวโน้มที่จะสูงขึ้นเรื่อย ๆ ความซื่อสัตย์และจริงใจกับลูกค้าเป็นเหตุผลหนึ่งที่ทำให้ลูกค้าเลือกใช้บริการของธนาคาร ดังนั้นแต่ละธนาคารจึงต้องมีหน่วยงานตรวจสอบ เพื่อมาทำหน้าที่ตรวจสอบการทำงานของพนักงาน ทั้งในเรื่องการปฏิบัติงานที่ถูกต้องตามกระบวนการและความถูกต้องของเอกสาร รวมทั้งความถูกต้องของอุปกรณ์ที่ใช้ในการทำงานโดยเฉพาะโปรแกรมคอมพิวเตอร์ซึ่งทำหน้าที่หลักในการทำธุรกรรมของธนาคาร ดังนั้นพนักงานของฝ่ายตรวจสอบ จึงต้องเดินทางไปตรวจสอบดังกล่าวตามสาขาต่าง ๆ ทั่วประเทศเป็นประจำทุกปี และค่าใช้จ่ายในการเดินทางนั้นค่อนข้างสูง หากเรานำเทคโนโลยีสารสนเทศมาประยุกต์ใช้ จะช่วยให้ลดค่าใช้จ่ายในการเดินทางได้ ทำให้ธนาคารลดต้นทุนในการผลิตลง

1.2 วัตถุประสงค์ของการพัฒนาระบบงาน

การประยุกต์ใช้ Genetic Algorithm เพื่อลดค่าใช้จ่ายในการเดินทางไปตรวจสอบสาขาของธนาคาร ได้ถูกสร้างขึ้นมาเพื่อช่วยลดค่าใช้จ่ายของธนาคาร และอำนวยความสะดวกรวดเร็วในการได้มาซึ่งสารสนเทศที่ใช้ในการเดินทางไปตรวจสอบยังสาขาต่าง ๆ ทั่วประเทศ โดยจะสามารถลดระยะทางในการเดินทางในกรณีที่มีการเดินทางไปตรวจสอบในหลายสาขาพร้อม ๆ กัน โปรแกรมจะทำการคำนวณเส้นทางในรูปแบบต่าง ๆ เพื่อให้ได้มาซึ่งเส้นทางที่น้อยที่สุด เพื่อเป็นแนวทางให้ผู้ตรวจสอบเลือกเส้นทางและเป็นการประหยัดค่าใช้จ่าย

1.3 ขอบเขตของการพัฒนาระบบงาน

สำหรับขอบเขตในการพัฒนาของระบบงานนั้นโปรแกรมจะทำการคำนวณหาเส้นทางของการเดินทางไปยังจังหวัดต่าง ๆ ทั่วทั้งภาคกลางของประเทศไทย (หากต้องการใช้เพื่อเดินทางไปทั่ว

ประเทศสามารถเพิ่มข้อมูลใน Table ได้) ผู้ตรวจสอบสามารถเลือกว่าจะไปทั้งหมดกี่จังหวัดและใส่ข้อมูลว่าจะไปจังหวัดใดบ้าง โปรแกรมจะทำการใช้ Genetic Algorithm เพื่อคำนวณหาเส้นทางที่สั้นที่สุดพร้อมทั้งแสดงเส้นทางที่สั้นที่สุดที่จะไปยังจังหวัดเหล่านั้น

1.4 ประโยชน์คาดว่าจะได้รับจากโครงการ

ประโยชน์ที่คาดว่าจะได้รับในการวางแผนงานสำหรับองค์กร ช่วยในการหาเส้นทางในการเดินทาง มีดังนี้

1.4.1 เป็นข้อมูลให้ผู้ตรวจสอบเลือกเส้นทาง

1.4.2 ผู้ตรวจสอบได้ข้อมูลรวดเร็วและประหยัดเวลา

1.4.3 ประหยัดค่าใช้จ่ายในการเดินทาง

1.4.4 เพิ่มประสิทธิภาพในการทำงานขององค์กร

บทที่ 2
ความรู้พื้นฐานเกี่ยวกับจีเนติกอัลกอริทึม

Genetic Algorithm มีแนวคิดมาจากระบบการคัดเลือกสายพันธุ์ตามธรรมชาติ โดยกระบวนการสุ่มหาประชากรต้นกำเนิด (Initial Population) ของสิ่งมีชีวิต ในสิ่งมีชีวิตจะมีหลาย ๆ โครโมโซม และโครโมโซมก็จะถูกแบ่งเป็นส่วน ๆ เรียกว่า ยีน (Gene) หากยีนในโครโมโซมของสิ่งมีชีวิตชนิดหนึ่งไปรวมกับยีนในโครโมโซมของสิ่งมีชีวิตอีกชนิดหนึ่ง ก็จะเกิดเป็นพันธุ์ใหม่ถูกผสมออกมาและมีลักษณะบางประการคล้ายต้นกำเนิด (Parents) ซึ่งสายพันธุ์ที่ไม่ดี จะค่อย ๆ สูญพันธุ์ไป จนในที่สุดจะเหลือเพียงสายพันธุ์ที่ดีเท่านั้นที่สามารถอยู่รอดสืบต่อไปได้

John Holland นักคอมพิวเตอร์ เป็นผู้บุกเบิก Genetic Algorithm เมื่อปี ค.ศ. 1975 โดยได้รับแนวคิดมาจากรูปแบบทางชีววิทยา โดยเขาได้เชื่อมปัญหาของการที่ระบบนิเวศน์ปรับตัวให้เข้ากับสิ่งแวดล้อมเพื่อการอยู่รอดเทียบกับปัญหาของโปรแกรมคอมพิวเตอร์ ดังนั้นเขาจึงเริ่มศึกษาและสามารถแก้ปัญหาได้

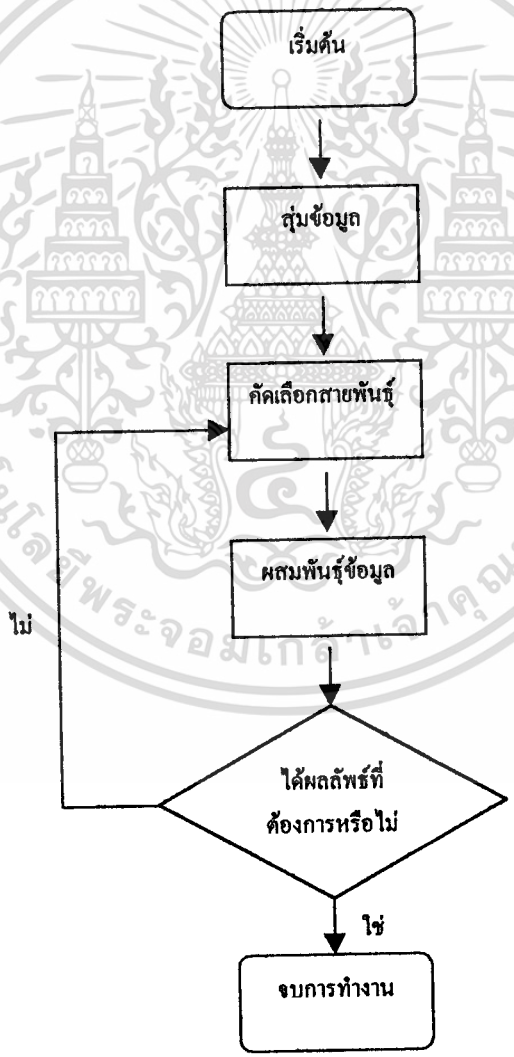
การสร้างคำตอบที่เป็นไปได้จากคำตอบของรุ่นก่อนหน้า ทำให้ไม่ต้องไปหาคำตอบทั้งหมดจากค่าที่เป็นไปได้ทั้งหมด การทดลองครั้งต่อ ๆ มา ทำให้การสุ่มเริ่มลดลง มันจะลดส่วนที่ไม่ดีของรุ่นก่อน ทำให้ค่อย ๆ ลดจำนวนของคำตอบลง จนได้คำตอบที่ดีที่สุด

ถ้าเราแก้ปัญหาใด ๆ แล้ว เราต้องการที่จะได้วิธีการแก้ปัญหาที่ดีชุดหนึ่ง การแก้ปัญหาที่จะสามารถเป็นไปได้ทั้งหมดเรียกว่า search space แต่ละจุดใน search space จะแสดงการแก้ปัญหาหนึ่ง ๆ ที่จะสามารถเป็นไปได้ แต่ละวิธีการที่เป็นไปได้จะถูกเป็นเป้าหมายโดยค่า fitness สำหรับปัญหานั้น ๆ การค้นหาวิธีการแก้ปัญหาจะคล้าย ๆ กับการค้นหาค่าสูงสุดหรือต่ำสุดใน search space ซึ่งค่าใน search space คือค่าของวิธีการที่เป็นไปได้ในชั่วระยะเวลาหนึ่ง แต่เราจะเลือกค่าของวิธีการที่เป็นไปได้มาจำนวนหนึ่งเพื่อหาวิธีการที่ต้องการ ส่วนที่เหลือจะใช้ในการค้นหาวิธีการในครั้งต่อ ๆ ไป

ปัญหาที่ต้องค้นหาวิธีแก้ปัญหาอย่างซับซ้อน เราไม่รู้ว่าจะหาวิธีการได้จากที่ไหนและจะเริ่มต้นที่ไหน ซึ่งมีหลายวิธีการที่จะหาการแก้ปัญหาที่เหมาะสม แต่อาจจะไม่ใช่วิธีการที่ดีที่สุด ได้แก่ hill climbing, tabu search, simulated annealing และ genetic algorithm วิธีการเหล่านี้มักจะได้วิธีแก้ปัญหาที่ดีพอสมควร เพราะเป็นไปได้ยากที่จะสามารถหาวิธีการที่ดีที่สุดซึ่งมีไม่บ่อยครั้ง

2.1 ขั้นตอนของ Genetic Algorithm

ขั้นตอนแรกของ Genetic Algorithm จะเริ่มจากการสุ่มข้อมูลขึ้นมาจำนวนหนึ่ง แล้วทำการวัดค่าความดีของข้อมูลแต่ละตัว คัดเลือกเอาเฉพาะข้อมูลที่มีค่าความดีเป็นที่น่าพอใจชุดหนึ่งเก็บไว้ ข้อมูลที่คัดเลือกไว้นั้นจะถูกนำมาทำการผสมพันธุ์กัน ได้เป็นข้อมูลชุดใหม่ ซึ่งเราจะนำข้อมูลชุดใหม่นี้ มาทำการคัดเลือกและผสมพันธุ์ต่อไป เมื่อกระบวนการนี้เกิดขึ้นซ้ำแล้วซ้ำอีกก็จะเหลือแต่ข้อมูลที่มีค่าความดีเป็นที่น่าพอใจ (ดังรูปที่ 2.1) ส่วนข้อมูลที่มีค่าความดีไม่เป็นที่น่าพอใจ เราก็จะไม่นำมาคัดเลือก ทำให้ข้อมูลดังกล่าวค่อย ๆ หายไป



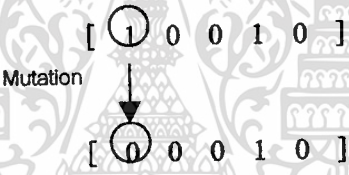
รูปที่ 2.1 แผนภาพแสดงการทำงานของ Genetic Algorithm

2.2 กระบวนการที่ถูกนำมาประยุกต์ใช้ร่วมกับเทคโนโลยีคอมพิวเตอร์

2.2.1 การเปลี่ยนรูปร่างของโครโมโซม (Mutation)

การเปลี่ยนรูปร่างเป็นตัวกระทำทางวิวัฒนาการ ซึ่งจะเลือกเปลี่ยนค่าของยีน 1 หรือมากกว่าในโครโมโซมที่เป็นสถานะเริ่มต้น ยีนค่าใหม่ที่ได้อาจจะทำให้การค้นหาคำตอบของ Genetic Algorithm เร็วขึ้น Mutation เป็นส่วนที่สำคัญสำหรับการค้นหา เนื่องจากมันจะทำให้ประชากรมีการเปลี่ยนแปลงตลอดเวลา ทำให้ประชากรไม่หยุดนิ่ง โอกาสที่จะพบคำตอบที่ดีที่สุดจึงมีมาก Mutation จะเกิดขึ้นระหว่างการวิวัฒนาการตามความน่าจะเป็นที่มีผู้กำหนดไว้ สามารถแบ่งประเภทของ Mutation ได้ดังนี้

1. **Flip Bit** เป็นวิธีการเปลี่ยนรูปร่างโครโมโซม โดยการสลับค่าของยีนที่ถูกเลือกมา เช่น จาก 1 เป็น 0 หรือจาก 0 เป็น 1 เพียงยีนเดียว วิธีการนี้สามารถใช้ได้กับเฉพาะ Binary Gene เท่านั้น (ดังรูปที่ 2.2)

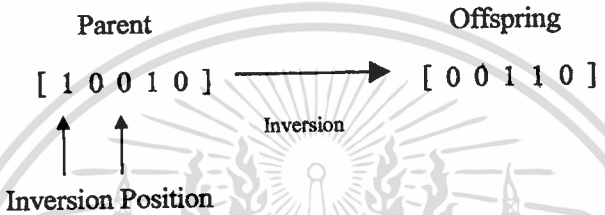


รูปที่ 2.2 แสดงตัวอย่างการเปลี่ยนรูปร่างของโครโมโซม ทำให้ได้สิ่งมีชีวิตที่มีคุณลักษณะใหม่ (ในที่นี้ใช้เลขฐานสองแสดงความแตกต่างของแต่ละหน่วยพันธุกรรม โดยเลขฐานสองหนึ่งชุดภายในวงเล็บแทนแต่ละโครโมโซม

2. **Boundary** เป็นวิธีการเปลี่ยนรูปร่างของโครโมโซมโดยการแทนที่ยีนด้วยค่าสูงสุดหรือต่ำสุดอย่างใดอย่างหนึ่งโดยการสุ่มของยีน วิธีการนี้ใช้ได้เฉพาะกับ Integer Genes และ Float Genes
3. **Non-Uniform** เป็นวิธีการเปลี่ยนรูปร่างของโครโมโซมที่จะเพิ่มความน่าจะเป็นของจำนวนการเปลี่ยนรูปร่างเข้าใกล้ 0 วิธีการเปลี่ยนรูปร่างนี้ จะทำให้ประชากรหยุดนิ่งหยุดการวิวัฒนาการ ทำให้ Genetic Algorithm ใช้หาคำตอบได้ดี วิธีการนี้ใช้ได้เฉพาะกับ Integer Genes และ Float Genes
4. **Uniform** เป็นวิธีการที่จะแทนค่าของยีนที่ถูกเลือกด้วยค่าสุ่มที่ไม่เปลี่ยนแปลง ซึ่งมีค่าระหว่างค่าสูงสุดและต่ำสุดที่ User ได้เลือกเอาไว้สำหรับยีนนั้น ๆ วิธีการนี้สามารถใช้ได้กับ Integer Genes และ Float Gene

5. **Gaussian** เป็นวิธีการที่เพิ่มค่าสุ่มกระจายแบบ Gaussian ให้กับยีนที่ถูกเลือก ค่ายีนใหม่ที่ได้จะถูกตัดออกเมื่อค่านั้นเกินขอบเขตที่ User ได้กำหนดไว้ วิธีการนี้สามารถใช้ได้กับ Integer Genes และ Float Genes

2.2.2 การสลับตำแหน่งของยีนภายในโครโมโซมเดียวกัน (Inversion) ให้เกิดคุณลักษณะใหม่ในรุ่นลูก (ดังรูปที่ 2.3)

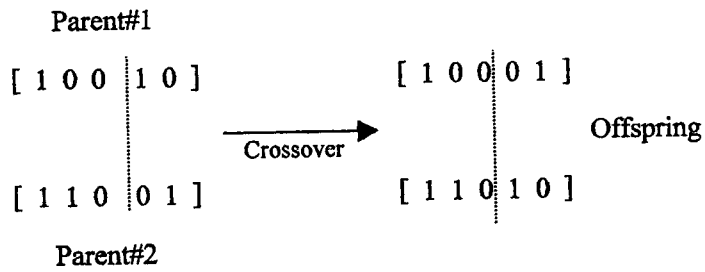


รูปที่ 2.3 แสดงตัวอย่างการสลับตำแหน่งของหน่วยถ่ายพันธุกรรมอย่างสุ่มภายในโครโมโซมของสิ่งมีชีวิตเพศเดียว ทำให้ได้สิ่งมีชีวิตรุ่นลูกที่มีคุณลักษณะใหม่

2.2.3 การสลับตำแหน่งระหว่างโครโมโซมของพ่อและแม่ (Crossover)

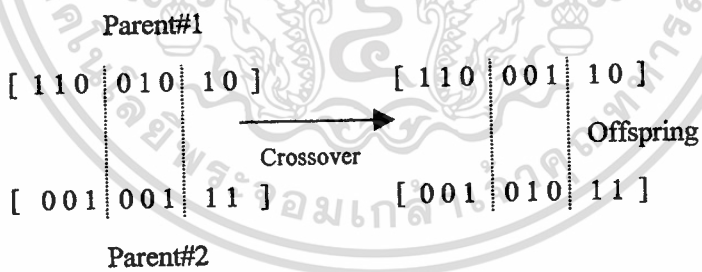
การสลับตำแหน่งเป็นตัวกระทำทางวิวัฒนาการ ซึ่งจะรวมโครโมโซมของพ่อส่วนหนึ่งกับของแม่อีกส่วนหนึ่งเพื่อให้เกิดเป็นโครโมโซมใหม่เป็นรุ่นลูกซึ่งอาจจะดีกว่ารุ่นพ่อแม่ถ้าได้รับส่วนที่ดีของพ่อแม่ไป ได้แบ่งประเภทของ crossover ไว้ดังนี้

1. **One Point** เป็นวิธีการสลับตำแหน่งระหว่างโครโมโซมของพ่อและแม่โดยการคัดเลือกแบบสุ่ม หลังจากนั้นก็จะสลับตำแหน่งโครโมโซมระหว่างพ่อและแม่ เพื่อจะได้รุ่นลูกที่เกิดขึ้นใหม่ 2 ตัว โดยจะคั่นตำแหน่งที่มีการสลับด้วยสัญลักษณ์ “ | ” (ดังรูปที่ 2.4)



รูปที่ 2.4 แสดงตัวอย่างการสลับตำแหน่งของหน่วยถ่ายพันธุกรรมอย่างสุ่ม ภายในโครโมโซมของพ่อและแม่ โดยลูก (Offspring) เกิดจากหน่วยถ่ายพันธุกรรมชุดแรกของพ่อ (Parent#1) และหน่วยถ่ายพันธุกรรมชุดหลังของแม่ (Parent#2) หรือ เกิดจากหน่วยถ่ายพันธุกรรมชุดแรกของแม่ (Parent#2) และหน่วยถ่ายพันธุกรรมชุดหลังของพ่อ (Parent#1)

2. **Two Point** เป็นวิธีการสลับตำแหน่งระหว่างโครโมโซมของพ่อและแม่โดยการคัดเลือกแบบสุ่ม หลังจากนั้นก็จะสลับตำแหน่งตรงกลางของโครโมโซมระหว่างพ่อและแม่ เพื่อจะได้รุ่นลูกที่เกิดขึ้นใหม่ 2 ตัว โดยจะคั่นตำแหน่งที่มีการสลับด้วยสัญลักษณ์ " | " (ดังรูปที่ 2.5)



รูปที่ 2.5 แสดงตัวอย่างการสลับตำแหน่งของหน่วยถ่ายพันธุกรรมอย่างสุ่ม ณ ตำแหน่งตรงกลางของโครโมโซมของพ่อและแม่ โดยลูก (Offspring) เกิดจากหน่วยถ่ายพันธุกรรมชุดแรกและชุดหลังของพ่อ (Parent#1) และหน่วยถ่ายพันธุกรรมชุดกลางของแม่ (Parent#2) หรือ เกิดจากหน่วยถ่ายพันธุกรรมชุดแรกและชุดหลังของแม่ (Parent#2) และหน่วยถ่ายพันธุกรรมชุดกลางของพ่อ (Parent#1)

3. **Uniform** เป็นวิธีการสลับตำแหน่งระหว่างโครโมโซมของพ่อและแม่โดยการผสมของยีน รุ่นลูกที่เกิดขึ้นจะมีลักษณะที่ยีนในโครโมโซมของพ่อจะกระจายอย่างไม่เป็นกลุ่ม

ไปยังโครโมโซมของแม่ หรือยีนในโครโมโซมของแม่จะกระจายอย่างไม่เป็นกลุ่มไปยังโครโมโซมของพ่อ ซึ่งจะเป็นอัตราส่วนที่ผสมกัน (mixing ratio) ถ้า มีค่า 0.5 แสดงว่ารุ่นลูกที่ได้เกิดจากยีนของพ่อและยีนของแม่อย่างละครึ่ง (ดังรูปที่ 2.6)

Parent#1 : 1 1 0 0 1 0 1 0

Parent#2 : 0 0 1 0 0 1 1 1

เมื่อ crossover แล้วจะได้

Offspring#1 : $1_1 0_2 1_2 0_1 0_2 0_1 1_1 1_2$

Offspring#2 : $0_2 1_1 0_1 0_2 1_1 1_2 1_2 0_1$

รูปที่ 2.6 แสดงตัวอย่างการสลับตำแหน่งของหน่วยถ่ายพันธุกรรมภายในโครโมโซมของพ่อและแม่ โดยลูก (Offspring) เกิดจากหน่วยถ่ายพันธุกรรมของพ่อ (Parent#1) และหน่วยถ่ายพันธุกรรมของแม่ (Parent#2) ซึ่งจะผสมกันโดยไม่เป็นกลุ่ม ตัวเลขที่ห้อยแทนการบอกว่าหน่วยถ่ายพันธุกรรมนี้มาจากพ่อหรือแม่

4. **Arithmetic** เป็นวิธีการสลับตำแหน่งระหว่างโครโมโซมของพ่อและแม่โดยการรวมกันเชิงเส้นของ vector ของโครโมโซมของพ่อและแม่ เพื่อจะได้ลูก 2 ตัว ตามสมการนี้ (ดังรูปที่ 2.7)

$$\text{Offspring\#1} = a * \text{Parent\#1} + (1 - a) * \text{Parent\#2}$$

$$\text{Offspring\#2} = (1\text{-}a) * \text{Parent\#1} + a * \text{Parent\#2}$$

เมื่อ a คือ random weighting factor (ต้องเลือกก่อนการทำ crossover)

ลองดูตัวอย่างของ Parents ที่แต่ละตัวมี 4 ยีน

Parent#1 : (0.3)(1.4)(0.2)(7.4)

Parent#2 : (0.5)(4.5)(0.1)(5.6)

ถ้า a = 0.7 จะได้ลูกดังนี้

Offspring#1 : (0.36)(2.33)(0.17)(6.86)

Offspring#2 : (0.44)(3.444)(0.13)(6.14)

รูปที่ 2.7 แสดงตัวอย่างการคำนวณหาค่าพันธุกรรมของลูกโดยใช้สมการ

5. **Heuristic** เป็นวิธีการสลับตำแหน่งระหว่างโครโมโซมของพ่อและแม่โดยใช้ค่าความอยู่รอดของโครโมโซมของพ่อและแม่ เพื่อกำหนดแนวทางในการค้นหา ลูกที่เกิดขึ้นจะเป็นดังสมการ (ดังรูปที่ 2.8)

Offspring#1 = BestParent + r * (BestParent - WorstParent)

Offspring#2 = BestParent

เมื่อ r คือเลขที่สุ่มที่มีค่าระหว่าง 0 ถึง 1

รูปที่ 2.8 แสดงตัวอย่างการคำนวณหาค่าพันธุกรรมของลูกโดยใช้สมการ

เป็นไปได้ว่าลูกตัวที่ 1 (Offspring#1) ที่เกิดขึ้นอาจจะได้ค่าที่ไม่เหมาะสม เมื่อค่า r ที่ได้มีค่าที่ไม่สามารถยอมรับได้คืออาจจะมากเกินไปหรือน้อยเกินไป จากเหตุผลนี้ Heuristic crossover จึงสามารถให้ User กำหนดตัวเลขมาจำนวนหนึ่ง เพื่อหาค่า r ที่สามารถทำให้เกิดเป็นโครโมโซมที่เหมาะสม แต่ถ้าตัวเลขจำนวนนั้นไม่สามารถทำให้เกิดโครโมโซมที่เหมาะสมแล้ว พ่อหรือแม่ที่มีค่าความอยู่รอดต่ำสุด (WorstParent) ก็จะกลายเป็นลูกตัวที่ 1

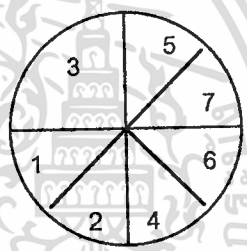
2.2.4 การคัดเลือก (Selection)

การคัดเลือก เป็นตัวดำเนินการในการวิวัฒนาการ ซึ่งจะเลือกโครโมโซมจากประชากรชุดปัจจุบัน เพื่อนำไปสร้างเป็นประชากรชุดใหม่ ก่อนที่จะเกิดเป็นประชากรชุดใหม่ จะต้องมีการคัดเลือกโครโมโซมเพื่อนำไปทำการ Crossover หรือ Mutation โครโมโซมของลูกหลานที่ได้ก็จะนำไปสร้างเป็นประชากรรุ่นใหม่ต่อไป การคัดเลือกมีหลายประเภท แต่จะยกตัวอย่างดังต่อไปนี้

1. **Roulette** เป็นตัวดำเนินการในการคัดเลือก ซึ่งโอกาสในการถูกเลือกขึ้นกับค่าความอยู่รอดที่เหมาะสม ได้รับแนวคิดมาจากการดำรงชีวิตให้อยู่รอดของสิ่งมีชีวิต จะสามารถใช้การเล่นเกม Roulette เป็นตัวอย่างของการคัดเลือก ดังนี้ Roulette Wheel เป็นอุปกรณ์ที่ใช้สำหรับเลือกตัวเลขมา 1 ตัว จากทั้งหมด n ตัว Roulette Wheel จะแบ่งออกเป็นส่วน ๆ ทั้งหมด n ส่วน และมีเลขกำกับ 1 ถึง n จะมีลูกบอล 1 ลูกวิ่งไปรอบ ๆ Roulette Wheel ถ้าลูกบอลไปหยุดที่ส่วนไหน ก็แสดงว่าตัวเลขในส่วนนั้นถูกเลือกขึ้นมาโดยวิธีการสุ่ม การกระจายความน่าจะเป็นของตัวเลข n ตัว มีโอกาสที่จะถูกเลือกเท่า ๆ กัน เพราะแต่ละส่วนมีลักษณะที่เหมือนกัน ในทางตรงกันข้าม ถ้าให้เลข i มี 2

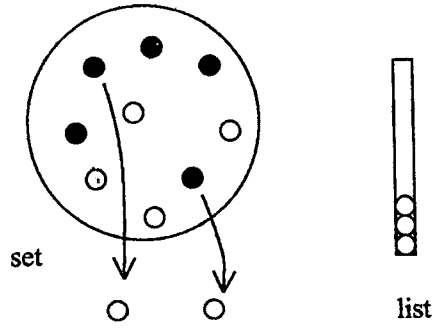
ส่วน ส่วนเลข j มี 1 ส่วน i จะมีโอกาสถูกเลือกเป็น 2 เท่าเมื่อเทียบกับ j กลยุทธ์ของ Roulette Wheel มีพื้นฐานมาจากการสังเกต มันจะจำลอง Roulette Wheel ที่มี 1 ส่วน แทนสิ่งมีชีวิต 1 สิ่งในประชากร ขนาดของแต่ส่วนเทียบได้กับค่าความอยู่รอดของสิ่งมีชีวิต ดังแสดงในรูปที่ 2.9 ซึ่งเป็นตัวอย่างของประชากรที่มีสิ่งมีชีวิต 7 ชนิด ลักษณะสำคัญของส่วนของ Roulette Wheel คือ มันจะเลือกโดยอาศัยหลักการความน่าจะเป็นมากกว่าการกำหนดไว้แต่แรก นั่นคือ เมื่อสิ่งมีชีวิตที่ 3 (ดังรูปที่ 2.9) มีค่าความอยู่รอดสูงสุดในประชากรทั้งหมด ก็ไม่ได้รับรองครั้งต่อไปมันจะถูกเลือกอีก ซึ่งเป็นที่แน่นอนว่าขึ้นกับค่าเฉลี่ยที่สิ่งมีชีวิตจะถูกเลือกด้วยอัตราความอยู่รอดที่เหมาะสม

Individual	Fitness
1	1.0
2	1.0
3	2.0
4	1.333
5	1.333
6	0.667
7	0.667

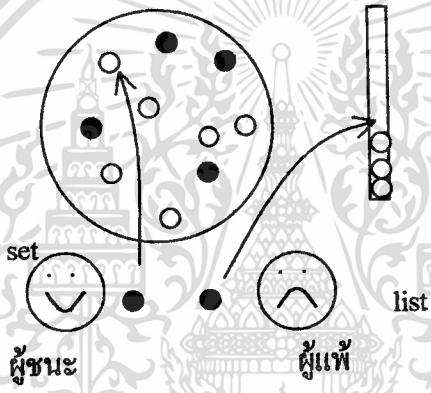


รูปที่ 2.9 แสดงกลยุทธ์การคัดเลือกของ Roulette Wheel

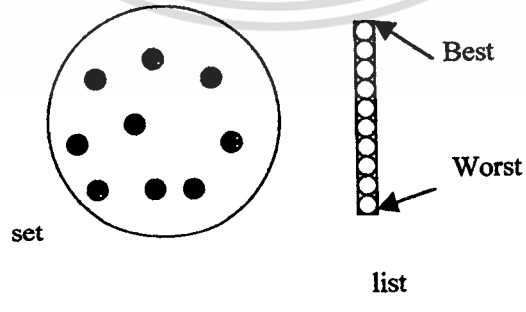
2. **Tournament** เทคนิคนี้แสดงดังรูปที่ 2.10
ขั้นแรก จะมีการสร้าง set ของ index ขึ้น กำหนดให้ index หนึ่งแทนสิ่งมีชีวิตหนึ่ง ในตอนแรก list จะว่างเปล่า หลังจากนั้นจะเริ่มแข่งขันกัน โดยการสุ่มคู่ต่อสู้มา 2 สิ่งแล้วเปรียบเทียบ สิ่งมีชีวิตที่ดีกว่า (อันที่ชนะ) จะถูกส่งกลับคืนเข้าไปใน set ส่วนสิ่งมีชีวิตอันที่ด้อยกว่า (อันที่แพ้) จะถูกส่งเข้าไปไว้ใน list เหตุการณ์นี้จะกระทำซ้ำไปเรื่อย ๆ จนกระทั่งใน set ไม่มีสิ่งมีชีวิตเหลืออยู่ จะไปอยู่ใน list ทั้งหมด ส่วนที่อยู่ล่างสุดของ list คือสิ่งมีชีวิตที่ด้อยที่สุด ส่วนที่อยู่บนสุดของ list คือสิ่งมีชีวิตที่ดีที่สุด
3. **Top Percent** เป็นตัวดำเนินการคัดเลือก ซึ่งจะคัดเลือกโครโมโซมอย่างสุ่มจาก N เปอร์เซนต์สูงสุดที่ดี ซึ่งค่า N นี้จะกำหนดโดย User



A. เลือกคู่ต่อสู้แล้วเปรียบเทียบกัน



B. อันที่แพ้เอาไปไว้ใน list ส่วนอันที่ชนะเก็บไว้ใน set



C. เลือกใช้คำตอบจากใน list

รูปที่ 2.10 แสดงกลยุทธ์การคัดเลือกแบบ Tournament

4. **Best** เป็นตัวดำเนินการคัดเลือกที่เลือกโครโมโซมที่ดีที่สุด (ดูจากค่าความอยู่รอด) ถ้ามีโครโมโซมที่มีค่าความอยู่รอดดีที่สุดตั้งแต่ 2 ตัวขึ้นไป ก็จะสุ่มมาเพียงตัวเดียว
5. **Random** เป็นตัวดำเนินการคัดเลือกโครโมโซมอย่างสุ่มจากประชากร

2.2.5 การสร้างสิ่งมีชีวิตที่มีคุณลักษณะใหม่ (Reproduction)

เป็นการเลียนแบบของสิ่งมีชีวิตจากรุ่นหนึ่งไปสู่อีกรุ่นหนึ่ง โดยไม่มีการเปลี่ยนแปลงใด ๆ ซึ่งจะมี 2 ตัวอย่างที่เกิดขึ้นตามธรรมชาติ คือ

1. **True Reproduction** คือสิ่งมีชีวิตชั้นต่ำที่สืบพันธุ์แบบไม่อาศัยเพศ ดังนั้นลูกที่เกิดมาจึงมีลักษณะทางพันธุกรรมเหมือนกับบรรพบุรุษ
2. **Simple Survival** เป็นลักษณะของสิ่งมีชีวิตที่มีอยู่ทั่วไป มันจะปรับตัวให้เข้ากับสิ่งแวดล้อม และสามารถดำรงชีวิตต่อไปได้เรื่อย ๆ จนกระทั่งรุ่นลูกรุ่นหลาน

ในส่วนของ Genetic Algorithm เป้าหมายของ Reproduction คือ ดำรงชีวิตให้อยู่รอดและรักษาลักษณะที่ดีไว้ เพื่อไม่ให้ลักษณะพันธุกรรมนี้หายไป แต่ในทางตรงกันข้าม จะมีการแพร่กระจายไปทั่วในอัตราที่สูงขึ้น จากนั้นลักษณะที่ดีก็จะดำรงอยู่ได้ อย่างไรก็ตาม Reproduction จะไม่มีลักษณะใหม่เกิดขึ้น มีแต่สิ่งมีชีวิตที่เกิดใหม่ลักษณะเหมือนเดิม ซึ่งจะทำให้ขอบเขตของคำตอบลดลง ทำให้เป็นประโยชน์มากในทางปฏิบัติ

2.2.6 การสิ้นสุด (Termination)

การสิ้นสุดเป็นตัวบอกว่าจะให้ Genetic Algorithm ดำเนินการค้นหาคำตอบหรือจะหยุด ซึ่งมีหลายเหตุการณ์ที่จะทำให้เกิดการสิ้นสุด ดังนี้

1. **Generation Number** เป็นวิธีการ Termination ที่ใช้สำหรับหยุดการวิวัฒนาการเมื่อได้ดำเนินการวิวัฒนาการไปจนถึงค่าสูงสุดที่ User ได้กำหนดไว้
2. **Evolution Time** เป็นวิธีการ Termination ที่ใช้สำหรับหยุดการวิวัฒนาการเมื่อเวลาที่ใช้ในการวิวัฒนาการที่ผ่านมามากกว่าเวลาของการวิวัฒนาการสูงสุดที่ User ได้กำหนดไว้ โดยปกติแล้ว การวิวัฒนาการจะไม่หยุดนิ่งจนกระทั่งการวิวัฒนาการของรุ่นปัจจุบันได้เสร็จสมบูรณ์ แต่พฤติกรรมนี้ได้เปลี่ยนไป ดังนั้น การวิวัฒนาการสามารถหยุดได้ในกรณีเช่นนี้
3. **Fitness Threshold** เป็นวิธีการ Termination ที่ใช้สำหรับหยุดการวิวัฒนาการเมื่อค่าความอยู่รอดที่ดีที่สุดของประชากรปัจจุบันน้อยกว่าที่ User ได้กำหนดค่า Fitness Threshold และเป้าหมายคือค่าต่ำสุดของความอยู่รอด หรือจะหยุดเมื่อค่าความอยู่รอดที่

ดีที่สุดของประชากรปัจจุบันมากกว่าที่ User ได้กำหนดค่า Fitness Threshold และเป้าหมายคือค่าสูงสุดของความอยู่รอด

4. **Fitness Convergence** เป็นวิธีการ Termination ที่ใช้สำหรับหยุดการวิวัฒนาการเมื่อค่าความอยู่รอด 2 แบบมาบรรจบกัน ซึ่งทั้งสองมีค่าที่ต่างกัน วิธีการนี้ได้เคยใช้ในการทำให้เกิดค่าความอยู่รอดที่ดีที่สุดโดยไม่มีอุปสรรคใด เมื่อค่าความอยู่รอดที่ดีที่สุดของ long filter แตกต่างจากค่าความอยู่รอดที่ดีที่สุดของ short filter เป็นเปอร์เซนต์ที่น้อยกว่าที่ User กำหนดไว้ ค่าความอยู่รอดก็จะมาบรรจบกัน การวิวัฒนาการก็จะหยุดลง
5. **Population Convergence** เป็นวิธีการ Termination ที่ใช้สำหรับหยุดการวิวัฒนาการเมื่อค่าของประชากรมาบรรจบกัน และจะบรรจบกันเมื่อค่าเฉลี่ยของค่าความอยู่รอดที่พาดผ่านประชากรปัจจุบันต่างจากค่าความอยู่รอดที่ดีที่สุดของประชากรเป็นเปอร์เซนต์น้อยกว่าที่ User กำหนดไว้
6. **Gene Convergence** เป็นวิธีการ Termination ที่ใช้สำหรับหยุดการวิวัฒนาการเมื่อเปอร์เซนต์ของยีนที่เกิดขึ้นเป็นโครโมโซมลู่เข้ามาบรรจบกัน ซึ่งจะบรรจบกันเมื่อค่าของยีนที่พาดผ่านโครโมโซมทั้งหมดในประชากรปัจจุบันต่างจากค่าสูงสุดของยีนที่พาดผ่านโครโมโซมเหล่านั้น เป็นเปอร์เซนต์น้อยกว่าที่ User กำหนดไว้

2.3 Chromosome

2.3.1 Chromosome Encoding

โดยปกติแล้ว chromosome จะแสดงวิธีแก้ปัญหาเป็นข้อมูลรายละเอียด แต่ในที่นี้เราจะใช้ตัวเลขฐาน 2 ในการแทนค่าของ chromosome ซึ่งจะแสดงดังรูปที่ 2.11

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

รูปที่ 2.11 แสดงตัวอย่างของ chromosome encoding โดยใช้เลขฐาน 2 แทนค่า

Chromosome แต่ละตัวจะแสดงเป็นเลขฐาน 2 แต่ละ bit จะแสดงลักษณะของวิธีแก้ปัญหาหรือตัวเลข ซึ่งอาจจะใช้เป็นเลขจำนวนเต็มหรือจำนวนจริงก็ได้

2.4. Parameter ของ Genetic Algorithm

ความน่าจะเป็นของ Crossover และ Mutation

ความน่าจะเป็นของ crossover กล่าวถึงโอกาสที่จะเกิดการ crossover มีความถี่มากน้อยเพียงใด ถ้าไม่มีการ crossover offspring จะทำการสำเนาจาก parent ถ้ามีการ crossover offspring จะเกิดจาก chromosome ของ parent ถ้าค่าความน่าจะเป็นเท่ากับ 100% offspring ทั้งหมดจะเกิดจากการ crossover แต่ถ้าค่าความน่าจะเป็นเท่ากับ 0% ประชากรที่เกิดขึ้นจะเหมือนกับประชากรรุ่นก่อน การ crossover ถูกสร้างขึ้นโดยที่คาดหมายไว้ว่าประชากรรุ่นใหม่จะได้ลักษณะที่ดีจากประชากรรุ่นก่อนหน้า และจะต้องดีกว่าประชากรรุ่นก่อนหน้าด้วย

ความน่าจะเป็นของ mutation กล่าวถึงโอกาสที่จะเกิดการ mutation มีความถี่มากน้อยเพียงใด ถ้าไม่มีการ mutation offspring จะทำการสำเนาจาก parent เช่นเดียวกับการ crossover ถ้าเกิดการ mutation ส่วนของ chromosome จะเปลี่ยนแปลงไป ถ้าค่าความน่าจะเป็นของการ mutation เท่ากับ 100% โครโมโซมทั้งหมดจะถูกเปลี่ยนแปลง ถ้าค่าความน่าจะเป็นเท่ากับ 0% จะไม่มีการเปลี่ยนแปลง

2.5 Encoding

การเข้ารหัสของ chromosome เป็นปัญหาอย่างหนึ่ง เมื่อเราเริ่มต้นที่จะแก้ปัญหาโดย genetic algorithm การจะเข้ารหัสโดยวิธีใดนั้น จะขึ้นกับประเภทของปัญหา ซึ่งในส่วนนี้จะแนะนำวิธีการเข้ารหัสในแบบต่าง ๆ ที่ใช้สำเร็จมาแล้ว

2.5.1 Binary Encoding เป็นแบบธรรมดามาก เพราะการทำงานของ genetic algorithm ใช้วิธีการนี้ ซึ่งจะประกอบไปด้วย bit 0 หรือ bit 1 ดังรูปที่ 2.12

Chromosome A 10110010110010101110010 1

Chromosome B 11111110000011000001111 1

รูปที่ 2.12 แสดงตัวอย่างของ chromosome ที่ใช้ Binary Encoding

ตัวอย่างของปัญหา ปัญหากระเป๋าเป้สะพายหลัง

ปัญหา มีของหลายสิ่งซึ่งมีมูลค่าต่าง ๆ กันและขนาดของกระเป๋าเป้สะพายหลังมีเนื้อที่เก็บของที่จำกัด ต้องเลือกของบางสิ่งให้ได้มูลค่ามากที่สุดใส่ในกระเป๋าเป้สะพายหลัง โดยที่ไม่ต้องขยายขนาดของกระเป๋า

Encoding แต่ละบิตจะกล่าวถึง สิ่งของต่าง ๆ ที่มีมูลค่าต่าง ๆ กันที่อยู่ในกระเป๋าเป้สะพายหลัง

2.5.2 Permutation Encoding

สามารถใช้ได้กับปัญหาแบบเป็นลำดับ เช่น ปัญหาการเดินทางของ salesman ไปตามเมืองต่าง ๆ ทุก chromosome จะแทนชื่อเมืองซึ่งแสดงอย่างเรียงลำดับ ดังรูปที่ 2.13

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

รูปที่ 2.13 แสดงตัวอย่าง chromosome ที่ใช้ permutation encoding

Permutation encoding เหมาะสำหรับปัญหาที่เป็นลำดับ เหตุการณ์ของปัญหานี้ หากใช้วิธีการ crossover หรือ mutation แล้ว จะทำให้ chromosome เปลี่ยนไป ซึ่งที่จริงแล้วตัวเลขเหล่านี้เป็นตัวเลขของลำดับเหตุการณ์

ตัวอย่างของปัญหา ปัญหาการเดินทางของ salesman ไปตามเมืองต่าง ๆ

ปัญหา มีเมืองหลาย ๆ เมืองซึ่งมีระยะห่างระหว่างกัน ซึ่ง salesman ต้องเดินทางไปทุกเมือง ให้หาเส้นทางที่สั้นที่สุดที่จะทำให้ salesman เดินทางได้ระยะทางสั้นที่สุด

Encoding chromosome จะแสดงลำดับของเมืองที่ salesman จะเดินทางไป

การนำ Crossover มาใช้ใน permutation encoding

การทำ crossover จุดเดียว ในส่วนแรก permutation จะสำเนาลักษณะมาจาก parent แรก หาก parent ที่ 2 ถูกตรวจดูแล้วว่าไม่มีตัวเลขซ้ำกับ offsping ก็จะถูกเพิ่มเป็นส่วนหลัง ดังรูปที่ 2.14

(1 2 3 4 5 6 7 8 9) + (4 5 3 6 8 9 7 2 1) = (1 2 3 4 5 6 8 9 7)

(4 5 3 6 8 9 7 2 1) + (1 2 3 4 5 6 7 8 9) = (4 5 3 6 8 1 2 7 9)

รูปที่ 2.14 แสดงการนำ crossover มาใช้ใน permutation encoding

การนำ Mutation มาใช้ใน permutation encoding

การทำ mutation ลำดับจะเปลี่ยนแปลง จะมีตัวเลข 2 ตัวถูกเลือกเพื่อเปลี่ยนตำแหน่งกัน ดังรูปที่ 2.15

(1 2 3 4 5 6 8 9 7) => (1 8 3 4 5 6 2 9 7)

รูปที่ 2.15 แสดงการนำ mutation มาใช้ใน permutation encoding

2.5.3 Value Encoding

การใช้ค่าของข้อมูลโดยตรงในการแก้ปัญหา ซึ่งเป็นค่าที่มีความซับซ้อน เช่น จำนวนจริง แต่หากใช้ Binary encoding จะทำให้การแก้ปัญหายากขึ้น ใน value encoding ทุก chromosome จะแสดงเป็นค่าซึ่งสัมพันธ์กับปัญหา เช่น ตัวเลข จำนวนจริง หรือตัวหนังสือ ดังรูปที่ 2.16

Chromosome A 1.2324 5.3243 0.4556 2.3293 2.4545

Chromosome B ABDJEIFJDHDIERJFDLDFLFEGT

Chromosome C (back), (back), (right), (forward), (left)

รูปที่ 2.16 แสดงตัวอย่าง chromosome ที่ใช้ value encoding

Value encoding ใช้ได้ดีกับปัญหาที่เฉพาะเจาะจงเป็นพิเศษ อีกทางหนึ่งคือ การ encoding ลักษณะนี้เหมาะสำหรับการพัฒนาวิธีการ crossover และ mutation ที่เฉพาะเจาะจงเป็นพิเศษกับปัญหา

ตัวอย่างของปัญหา การหา weight ของ neural network

ปัญหา สถาปัตยกรรมของ neural network มีหลายแบบ ให้หา weight ของ input ของ neuron เพื่อฝึกให้ network หา output ที่ต้องการ

Encoding ค่าของ chromosome แสดงค่า weight ของ input ที่เหมือนกัน

การนำ Crossover มาใช้ใน value encoding

กระทำเช่นเดียวกับ binary encoding

การนำ Mutation มาใช้ใน value encoding

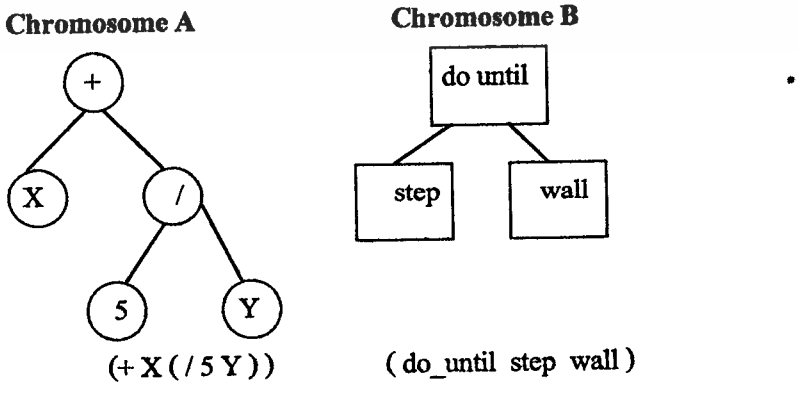
mutation เพิ่มตัวเลขที่มีค่าน้อย ๆ มา 1 จำนวน (ในกรณีที่เป็นเลขจำนวนจริง) เลือกค่าจาก chromosome เพื่อนำตัวเลขที่มีค่าน้อย ๆ นั้นมาบวกหรือลบ เช่น สมมติให้มีเลขที่มีค่าน้อย ๆ นั้น คือ 0.11 แล้วนำเลขจำนวนนี้ไปบวกกับค่าจาก chromosome ดังรูปที่ 2.17

(1.29 5.68 2.73 **4.11** 5.55) => (1.29 5.68 2.73 **4.22** 5.55)

รูปที่ 2.17 แสดงการนำ mutation มาใช้ใน value encoding

2.5.4 Tree Encoding

Tree Encoding ใช้ในการพัฒนาโปรแกรมสำหรับ genetic programming ใน tree encoding นั้น แต่ละ chromosome เป็น tree ของ object ต่าง ๆ เช่น ฟังก์ชัน หรือ คำสั่ง ในโปรแกรมภาษา ดังรูปที่ 2.18



รูปที่ 2.18 แสดงตัวอย่าง chromosome ที่ใช้ tree encoding

tree encoding ดีสำหรับการพัฒนาโปรแกรม โปรแกรมภาษา LISP ส่วนใหญ่จะใช้วิธีนี้ เพราะโปรแกรมภายในจะแสดงในรูปแบบนี้และสามารถกระจายได้โดยง่าย ดังนั้นวิธีการ crossover และ mutation สามารถใช้วิธีนี้ได้ง่ายเช่นเดียวกัน

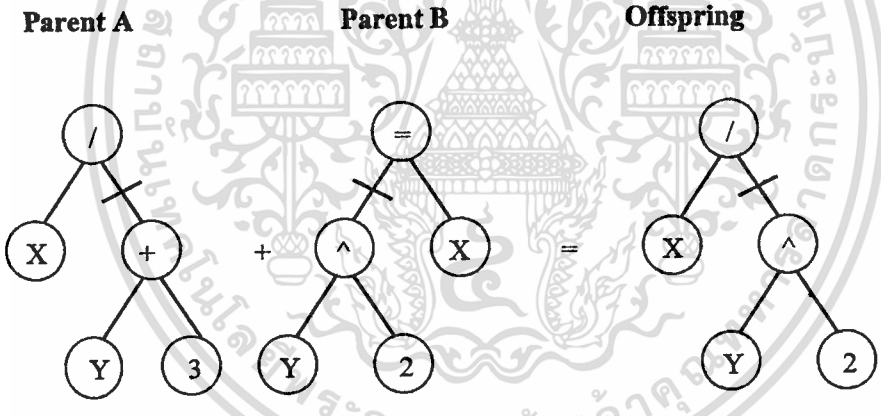
ตัวอย่างของปัญหา การหาฟังก์ชันจากค่าที่ให้มา

ปัญหา จะมีการกำหนด input และ output มาให้ แล้วให้หาฟังก์ชันซึ่งจะทำให้ได้ output ที่ดีที่สุดหรือใกล้เคียงสำหรับทุก input

Encoding chromosome เป็นฟังก์ชันที่แสดงใน tree

การนำ crossover มาใช้ใน tree encoding

สำหรับการ crossover จุดเดียว parent ทั้งสองจะต้องทำการแบ่งเป็น 2 ส่วน แล้วเปลี่ยนแปลง tree ส่วนที่แบ่งเพื่อให้เกิดเป็น offspring ใหม่ ดังรูปที่ 2.19



รูปที่ 2.19 แสดงการนำ crossover มาใช้ใน tree encoding

การนำ mutation มาใช้ใน tree encoding

Mutation ทำการเปลี่ยนค่า operator หรือ ตัวเลข ค่าของ node ที่ถูกเลือกจะเปลี่ยนแปลง

บทที่ 3

ตัวอย่างของการแก้ปัญหาโดยใช้ Genetic Algorithm

ตัวอย่างที่จะนำมาเสนอการประยุกต์ใช้ Genetic Algorithm นี้ มี 2 ตัวอย่าง ตัวอย่างแรกจะเป็นการประยุกต์ใช้ Genetic Algorithm เพื่อหาส่วนผสมระหว่างแป้งและน้ำตาลเพื่อทำขนมคุกกี้ให้มีคุณภาพมากที่สุด ส่วนตัวอย่างที่สอง เป็นการประยุกต์ใช้ Genetic Algorithm เพื่อหาค่าต่ำสุดของฟังก์ชัน

3.1 ตัวอย่างการหาส่วนผสมระหว่างแป้งและน้ำตาลเพื่อใช้ทำขนมคุกกี้ให้มีคุณภาพมากที่สุด

ตัวอย่างนี้ นายตุ้ยต้องการที่จะหาส่วนผสมระหว่างแป้งและน้ำตาลเพื่อใช้ทำขนมคุกกี้ให้มีคุณภาพมากที่สุด

SUGAR									
9	1	2	3	4	5	4	3	2	1
8	2	3	4	5	6	5	4	3	2
7	3	4	5	6	7	6	5	4	3
6	4	5	6	7	8	7	6	5	4
5	5	6	7	8	**9**	8	7	6	5
4	4	5	6	7	8	7	6	5	4
3	3	4	5	6	7	6	5	4	3
2	2	3	4	5	6	5	4	3	2
1	1	2	3	4	5	4	3	2	1
	1	2	3	4	5	6	7	8	9

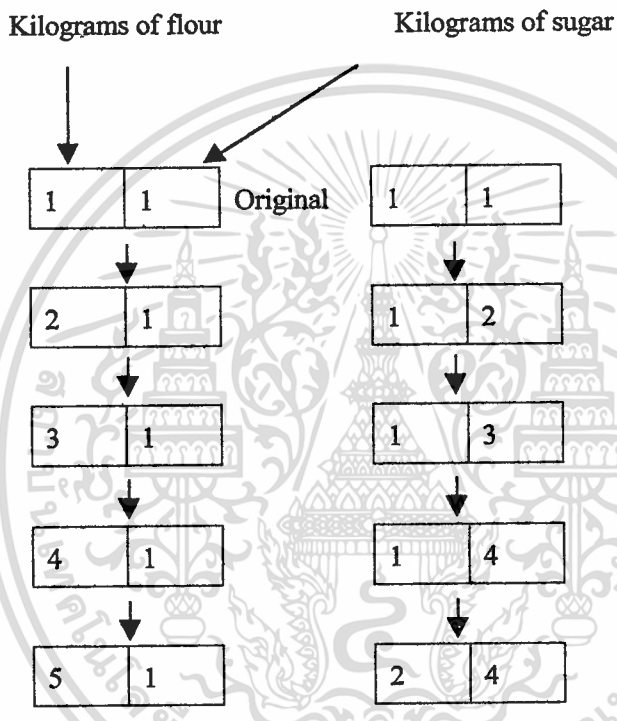
F L O U R

รูปที่ 3.1 แสดงปริมาณส่วนผสมของแป้งและน้ำตาลที่ต้องใช้เพื่อให้คุกกี้มีคุณภาพมากที่สุด

จากรูปที่ 3.1 แสดงให้เห็นส่วนผสมของแป้งและน้ำตาลที่จะทำให้คุกกี้มีคุณภาพเป็นค่าตัวเลข คุณภาพสูงสุดที่หาได้คือ 9 โดยเขาได้ลองผสมแป้งและน้ำตาล จาก 1 กิโลกรัม ถึง 9 กิโลกรัม

โดยเพิ่มขึ้นครั้งละ 1 กิโลกรัม เขาต้องทำการทดลองถึง 81 ครั้ง (9x9) จึงจะได้ค่าที่ต้องการ เขาจึงได้นำ Genetic Algorithm มาช่วยแก้ปัญหาเพื่อหาปริมาณของส่วนผสมที่น้อยที่สุดเพื่อให้ได้คุกกี้ที่มีคุณภาพมากที่สุด

สมมติว่าการทำขนมครั้งหนึ่งเปรียบได้กับสิ่งมีชีวิตสิ่งหนึ่ง จากรูปที่ 3.2 1 โครโมโซมจะประกอบด้วย 2 ยีน ยีนแต่ละอันจะมีค่า 1 ถึง 9 ยีนตัวแรกเป็นการกำหนดปริมาณแป้ง ส่วนยีนตัวที่สองเป็นการกำหนดปริมาณน้ำตาลที่ต้องใช้ในการทำคุกกี้



รูปที่ 3.2 แสดงให้เห็นการดำเนินการ Mutate อย่างต่อเนื่อง

จากรูปที่ 3.2 แสดงโครโมโซม 2 อัน ซึ่งแต่ละอันจะเปลี่ยนแปลงไปโดยการนำ 1 ไปบวกหรือลบค่ายีน และจะเปลี่ยนเพียงยีนเดียวเท่านั้น เราจะเรียกโครโมโซมต้นกำเนิดว่า โครโมโซม 1-1 มีค่าคุณภาพระดับ 1 เมื่อได้ผ่านกระบวนการ Mutation นั่นก็คือชุดแรกได้เป็นโครโมโซม 5-1 ส่วนชุดที่สองได้โครโมโซม 2-4 ทั้งสองแบบได้ค่าคุณภาพระดับ 5 เท่ากัน ก่อนที่นายตุ้ยจะลงมือทดลองเขาได้ตั้งเงื่อนไขไว้ดังนี้

1. เริ่มต้นที่โครโมโซม 1-1
2. ไม่มีโครโมโซมใดที่จะปรากฏมากกว่า 1 ครั้งในแต่ละ Generation
3. จะมีโครโมโซมอย่างมาก 4 อันเท่านั้นที่รอดชีวิตจาก Generation หนึ่งไป Generation ถัดไป

4. โครโมโซมที่รอดชีวิตจะต้องนำไปแข่งขันกับโครโมโซมที่จะถูกผลิตขึ้นมาใหม่ใน Generation ถัดไป
5. ยีน 1 อัน จะถูกเลือกและถูก mutate อย่างสุ่มเรียกว่า mutant ถ้า mutant ที่ได้มีค่าต่างจากโครโมโซมที่มีอยู่เดิม จะถูกเพิ่มเข้ามาใน Generation เพื่อมาแข่งขันให้มีชีวิตรอด
6. จะไม่มีการ Crossover
7. โครโมโซมที่มีค่าสูงสุด จะอยู่รอดไปยัง Generation ถัดไป
8. โครโมโซมที่เหลืออยู่ก็จะถูกคัดเลือกอย่างสุ่ม

เริ่มต้นที่ Generation 0 โดยเลือกโครโมโซม 1-1 ซึ่งมีคุณภาพระดับ 1

Generation 0:

โครโมโซม	คุณภาพ
1 1	1

เมื่อเกิดการ mutate จะเกิดโครโมโซม 1-2 เป็นประชากร ทำให้มีสมาชิก 2 อัน

Generation 1:

โครโมโซม	คุณภาพ
1 2	2
1 1	1

โครโมโซม 1-2 จะ mutate เป็นโครโมโซม 1-3 เป็นประชากร
โครโมโซม 1-1 จะ mutate เป็นโครโมโซม 1-2 ซึ่งมีอยู่แล้วในประชากร ใน Generation ถัดไปจึงเกิดประชากรใหม่เพียง 1 อัน

Generation 2:

โครโมโซม	คุณภาพ
1 3	3
1 2	2
1 1	1

โครโมโซม 1-3 จะ mutate เป็นโครโมโซม 1-4

โครโมโซม 1-2 จะ mutate เป็นโครโมโซม 2-2

โครโมโซม 1-1 จะ mutate เป็นโครโมโซม 2-1

จะทำให้เกิดเป็นประชากร 6 อัน

โครโมโซม	คุณภาพ
1 4	4
2 2	3
1 3	3
2 1	2
1 2	2
1 1	1

จากที่กล่าวไว้ในตอนต้นว่า ประชากรโครโมโซมต้องมีชีวิตรอดไป Generation ถัดไปไม่เกิน 4 อัน จึงทำการเลือกตาม standard fitness method ได้ดังนี้

Generation 3:

โครโมโซม	คุณภาพ
1 4	4
1 3	3
1 2	2
2 1	2

โครโมโซมทั้ง 4 อันนี้ จะทำการ mutate ได้เป็นโครโมโซมใหม่ดังนี้

โครโมโซม	คุณภาพ
2 4	5
2 3	4
3 1	3

จากโครโมโซมทั้ง 7 อัน เลือกมา 4 อันได้ดังนี้

Generation 4:

โครโมโซม	คุณภาพ
2 4	5
1 4	4
1 3	3
2 1	2

โครโมโซมทั้งหมดนี้จะทำการ mutate และโครโมโซมใหม่จะถูกคัดเลือกไปยัง Generation ถัดไป

Generation 5:

โครโมโซม	คุณภาพ
2 5	6
1 5	5
2 3	4
2 2	3

เช่นเดิมคือโครโมโซมทั้งหมดจะถูก mutate แต่ครั้งนี้ โครโมโซม 1-5 จะยังมีชีวิตรอดไปยัง Generation ถัดไป

Generation 6:

โครโมโซม	คุณภาพ
3 5	7
1 5	5
3 2	4
1 4	4

โครโมโซม 3-5 จะ mutate เป็นโครโมโซม 4-5

โครโมโซม 3-2 จะ mutate เป็นโครโมโซม 3-1

โครโมโซม 1-5 และโครโมโซม 1-4 จะ mutate เป็นอีกฝ่ายหนึ่ง โครโมโซม 4 อัน ต้องถูกคัดเลือกจากโครโมโซมทั้ง 6 อันนี้

โครโมโซม	คุณภาพ
4 5	8
3 5	7
1 5	5
3 2	4
1 4	4
3 1	3

Generation 7:

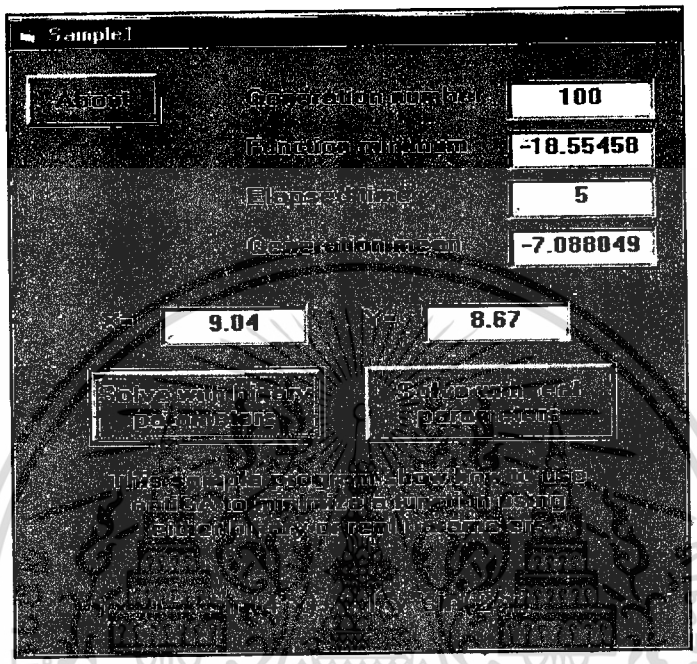
โครโมโซม	คุณภาพ
4 5	8
1 5	5
1 4	4
3 1	3

ตอนนี้ โครโมโซม 4-5 ได้ถูก mutate เป็นโครโมโซม 5-5 ซึ่ง Optimum แล้ว การทดลองสิ้นสุดลงแล้ว

Generation 8:

โครโมโซม	คุณภาพ
5 5	9
4 5	8
2 5	6
2 1	2

3.2 ตัวอย่างโปรแกรมการใช้ Genetic Algorithm เพื่อหาค่าต่ำสุดของฟังก์ชันโดยใช้ Binary Parameter หรือ Real Parameter



รูปที่ 3.3 แสดงหน้าจอโปรแกรมซึ่งใช้ Visual Basic เขียนโปรแกรม

โปรแกรมนี้ได้สร้าง Object ขึ้นมา ชื่อว่า optiGA ไว้ที่ Form ของโปรแกรมนี้ และที่ Form จะมีช่อง Label เพื่อแสดงรายละเอียดต่าง ๆ ของการ run โปรแกรมในแต่ละรอบ ดังนี้

Generation number : เป็นจำนวน Generation ที่ทำการ run

Function minimum : จะแสดงค่าต่ำสุดของฟังก์ชันในแต่ละ Generation ซึ่งท้ายที่สุดจะแสดงเฉพาะค่าต่ำสุดของฟังก์ชันในการ run ครบทุก Generation แล้ว

Elapsed time : จะแสดงเวลาที่ใช้ในการ run Genetic Algorithm เป็นหน่วยวินาที

Generation Mean : แสดงค่าเฉลี่ยของ Function ที่ run ในแต่ละ Generation

X, Y : จะแสดงค่าตัวแปร X และ Y ที่ทำให้ฟังก์ชันมีค่าต่ำสุด

ปุ่ม Solve with binary parameters : กดปุ่มนี้เมื่อต้องการแก้ปัญหาโดยใช้ binary parameters

ปุ่ม Solve with real parameters : กดปุ่มนี้เมื่อต้องการแก้ปัญหาโดยใช้ real parameter

จะอธิบายแต่ละโปรแกรมย่อยและฟังก์ชันที่ใช้ในโปรแกรมนี้ โดยใช้เครื่องหมาย " * " หน้าบรรทัดที่อธิบายคำสั่ง ซึ่งแสดงใต้คำสั่งของโปรแกรม (ตัวเอียง)

Sub ChangeOptiGAProperties()

* Sub Program นี้ ทำหน้าที่เปลี่ยนค่าของ Properties ใน optiGA Object ภายใต้คำสั่ง With … End With

```
With optiGA1
    .BinaryCrossoverType = ogaTwoPointsBCO
    .BinaryMutationType = ogaFlipBitBMU
    .Elitism = True
    .IntegerCrossoverType = ogaBlendingICO
    .IntegerMutationType = ogaRandomIMU
    .MaximumGenerationsWithNoChange = 10
    .MaximumRunTime = 3600
    .MutationProbability = 0.1
    .NumberOfGenerations = 100
    .ObjectiveFunctionType = ogaMinimum
    .PopulationSize = 100
    .RandomSeed = 0
    .RealCrossoverType = ogaBlendingRCO
    .RealMutationType = ogaRandomRMU
    .ReportEveryGeneration = 1
    .SelectionType = ogaTopMate
    .ShowErrorMessageBox = True
    .TerminationMode = ogaMaximumGenerations
    .CrossoverProbability = 0.95
End With
End Sub
```

Private Sub Command1_Click()

* เมื่อกดปุ่ม Solve with binary parameters แล้ว โปรแกรมจะเริ่มกระบวนการหาค่าต่ำสุดของฟังก์ชันโดย binary parameter

Dim B As Variant

* กำหนดตัวแปร B ให้เป็น Variant

ChangeOptiGAProperties

* เรียกใช้โปรแกรมย่อยนี้เพื่อเปลี่ยนค่า Properties ของ Object optiGA

B = Array(10, 10)

* กำหนดค่าตัวเลขให้ Gene ซึ่งก็คือ parameter แต่ละตัว โดยใช้คำสั่ง Array

With optiGA1

.ResetOptiGA

* ทำการ reset ให้ค่า Properties ของ Object optiGA เป็นค่าเริ่มต้น (Default)

.ReportEveryGeneration = 1

* กำหนดให้แสดงผลทุกการเปลี่ยนแปลง 1 Generation

i = .DefineBinaryGenes(2, B)

* เรียกใช้ method DefineBinaryGenes ซึ่ง 2 หมายถึงจำนวนของ Gene หรือ paramenter ที่กำหนดขึ้น ส่วน B เป็น array ที่มีตัวเลขแบบ Binary และอยู่ภายในแต่ละ Gene หรือ parameter

i = .RunOptiGA

* ทำการ run Genetic Process ซึ่งไปเรียกใช้ Sub optiGA1_FitnessFunction

End With

End Sub

Private Sub Command2_Click()

* เมื่อกดปุ่ม Solve with real parameters แล้ว โปรแกรมจะเริ่มกระบวนการหาค่าต่ำสุดของฟังก์ชันโดยใช้ real parameters

Dim vMin As Variant, vMax As Variant

* กำหนดค่าตัวแปร vMin และ vMax เป็น Variant

ChangeOptiGAProperties

* เรียกใช้โปรแกรมย่อยนี้เพื่อเปลี่ยนค่า Properties ของ Object optiGA

vMin = Array(0, 0)

* กำหนดค่าต่ำสุดของ parameter เป็น array

vMax = Array(10, 10)

* กำหนดค่าสูงสุดของ parameter เป็น array

With optiGA1

.ResetOptiGA

* ทำการ reset ให้ค่า Properties ของ Object optiGA เป็นค่าเริ่มต้น (Default)

.ReportEveryGeneration = 1

* กำหนดให้แสดงผลทุกการเปลี่ยนแปลง 1 Generation

i = .DefineRealGenes(2, vMin, vMax)

* เรียกใช้ method DefineRealGenes ซึ่ง 2 หมายถึงจำนวนของ Gene หรือ paramenter ที่กำหนดขึ้น ส่วน vMin และ vMax เป็นขอบเขตของตัวเลขแบบ real และอยู่ภายในแต่ละ Gene หรือ parameter

i = .RunOptiGA

* ทำการ run Genetic Process ซึ่งไปเรียกใช้ Sub optiGA1_FitnessFunction

.

End With

End Sub

Private Sub Command3_Click()

* เมื่อกดปุ่ม About แล้ว โปรแกรมจะแสดงรายละเอียดของชื่อโปรแกรมและที่อยู่บน web site ของโปรแกรม

optiGA1.ShowAboutBox

End Sub

Private Sub Form_Unload(Cancel As Integer)

* เมื่อสั่ง Close จะทำการออกจากโปรแกรม

End

End Sub

Private Sub optiGA1_FitnessFunction(BinaryGenes As Variant, RealGenes As Variant, IntegerGenes As Variant, GenerationNumber As Long, Fitness As Single)

* ฟังก์ชันนี้จะถูกเรียกใช้ทุกครั้งที่ algorithm ต้องการหาค่าของ fitness function เราต้องหาค่าของ fitness function ให้กับ chromosome ซึ่งแสดงในรูปแบบต่าง ๆ ดังนี้

1. BinaryGenes – แสดงในรูปทศนิยมของ binary parameters ใน chromosome ที่ต้องหาค่านั้น
2. RealGenes - แสดงในรูปทศนิยมของ real parameters ใน chromosome ที่ต้องหาค่านั้น
3. Fitness – คือค่า fitness ของ chromosome ซึ่งจะคืนค่าให้กับ optiGA

Dim X As Single, Y As Single

* กำหนดค่าตัวแปร X, Y เป็น Single

If Not IsNull(BinaryGenes) Then

*X = BinaryGenes(0) * 0.01*

*Y = BinaryGenes(1) * 0.01*

* ถ้า BinaryGenes ไม่เท่ากับ Null ให้เอา BinaryGenes ตัวแรกไปคูณ 0.01 เก็บค่าไว้ที่ตัวแปร X ถ้า BinaryGenes ไม่เท่ากับ Null ให้เอา BinaryGenes ตัวที่สองไปคูณ 0.01 เก็บค่าไว้ที่ตัวแปร Y

ElseIf Not IsNull(RealGenes) Then

X = RealGenes(0)

Y = RealGenes(1)

* ถ้า RealGenes ไม่เท่ากับ Null ให้เอา RealGenes ตัวแรกไป เก็บไว้ที่ตัวแปร X
ถ้า RealGenes ไม่เท่ากับ Null ให้เอา RealGenes ตัวที่สองไปเก็บไว้ที่ตัวแปร Y

End If

*Fitness = X * Sin(4 * X) + 1.1 * Y * Sin(2 * Y)*

* นำตัวแปร X, Y ที่เก็บค่าไว้มาทำการคำนวนแทนค่าในสมการเพื่อหาค่า Fitness

End Sub

Private Sub optiGA1_GenerationReport(BestFitness As Single, GenerationNumber As Long, BinaryGenes As Variant, RealGenes As Variant, IntegerGenes As Variant, ElapsedTime As Long, GenerationMeanFitness As Single)

* ฟังก์ชันนี้ใช้ในการแสดงสถานะของการ run Genetic ในแต่ละ Generation ซึ่งสามารถกำหนดได้ว่าจะให้แสดงสถานะทุกกี่ Generation ซึ่งในที่นี้ให้แสดงทุก Generation

Label7 = GenerationNumber

* แสดงสถานะของ GenerationNumber ซึ่งเป็น Generation ปัจจุบัน

Label8 = BestFitness

* แสดงสถานะของ BestFitness ซึ่งเป็นค่า Fitness ที่ดีที่สุดในประชากรปัจจุบัน

Label9 = ElapsedTime

* แสดงสถานะของ ElapsedTime ว่าใช้เวลาในการ run genetic ตั้งแต่เริ่มต้นมาเป็นเวลาเท่าไรแล้ว (หน่วยเป็นวินาที)

Label10 = GenerationMeanFitness

* แสดงสถานะของ GenerationMeanFitness ซึ่งเป็นค่าเฉลี่ยของ Fitness ในประชากรปัจจุบัน

If Not IsNull(BinaryGenes) Then

*Label11 = BinaryGenes(0) * 0.01*

*Label12 = BinaryGenes(1) * 0.01*

* ถ้า BinaryGenes ไม่เท่ากับ Null ให้เอา BinaryGenes ตัวแรกไปคูณ 0.01 แสดงค่าไว้ที่ตัวแปร X
ถ้า BinaryGenes ไม่เท่ากับ Null ให้เอา BinaryGenes ตัวที่สองไปคูณ 0.01 แสดงค่าไว้ที่ตัวแปร Y

ElseIf Not IsNull(RealGenes) Then

Label11 = RealGenes(0)

Label12 = RealGenes(1)

* ถ้า RealGenes ไม่เท่ากับ Null ให้เอา RealGenes ตัวแรกไปแสดงค่าไว้ที่ตัวแปร X
ถ้า RealGenes ไม่เท่ากับ Null ให้เอา RealGenes ตัวที่สองไปแสดงค่าไว้ที่ตัวแปร Y

End If

End Sub

3.3 การใช้ Genetic Algorithm เพื่อช่วยแก้ปัญหา Travelling Salesman Problem (TSP)

Travelling Salesman Problem (TSP) ที่จะกล่าวถึงนี้ เป็นการแสดงเมืองหลาย ๆ เมือง ซึ่งแต่ละเมืองจะมีระยะทางระหว่างเมือง travelling salesman ต้องการที่จะเดินทางไปยังทุก ๆ เมือง แต่ไม่ต้องการที่จะเสียเวลาในการเดินทางมาก ดังนั้นจึงต้องการหาลำดับของเมืองที่จะเดินทางโดยให้มีระยะทางน้อยที่สุด

การใช้ Genetic Algorithm (GA) เพื่อช่วยแก้ปัญหา Travelling Salesman Problem (TSP) มีเป้าหมายเพื่อให้ค้นหาเส้นทางการเดินทางที่ระยะทางสั้นที่สุดระหว่างเมืองต่าง ๆ จำนวนN เมือง ถ้าเราจะหาเส้นทางที่เป็นไปได้ทั้งหมดจะมี N! เส้นทาง ถ้าเราเดินทางผ่าน 30 เมือง จะมีเส้นทางที่เป็นไปได้ทั้งหมด 2.65×10^{32} เส้นทาง สมมติว่าใช้เวลาคำนวณ 1,000,000,000 เส้นทางต่อวินาที ดังนั้นจะใช้เวลาในการแก้ปัญหาถึง 8,000,000,000,000,000 ปี และถ้าเพิ่มจำนวนเมืองอีก 1 เมือง จะมีเส้นทางเพิ่มขึ้นเป็น 31! ซึ่งเป็นไปได้ยากในการแก้ปัญหา

Genetic Algorithm สามารถช่วยในการค้นหาคำตอบได้ในเวลาเพียงเล็กน้อย ถึงแม้ว่ามันอาจจะไม่ใช่เส้นทางที่ดีที่สุดก็ตาม แต่ก็สามารถหาเส้นทางได้เกือบดีที่สุดในเวลาเพียงไม่กี่นาที มี 2 ขั้นตอนในการแก้ปัญหา Travelling Salesman Problem โดยใช้ GA คือ ขั้นแรกสร้างกลุ่มของเส้นทางแบบสุ่มซึ่งเรียกว่า population และเก็บเส้นทางเหล่านี้เป็นลำดับ ขั้นที่ 2 เลือกเส้นทางที่สั้นจาก population มาเป็น parents จำนวน 2 เส้นทางและทำการ crossover จะได้ children ใหม่หรือเส้นทางใหม่ 2 เส้นทาง ซึ่งอาจได้เป็นเส้นทางที่สั้นกว่าใน population การ crossover นั้นต้องสุ่มหาจุดที่จะทำการ crossover หลังจากนั้นก็นำตัวเลขหลังจุดที่จะทำการ crossover มาสลับที่กัน

แนวความคิดของ Genetic Algorithm จำลองแบบมาจากการวิวัฒนาการโดยดูที่ค่าความอยู่รอด (Survival of the Fittest) ในการค้นหาคำตอบที่หลากหลายจาก population เส้นทางที่สั้นจะนำไปเป็น population แทนเส้นทางที่ยาวกว่าใน population

บางครั้ง GA ก็มีข้อจำกัดในการสร้างเส้นทาง ซึ่งแบ่งเป็น 2 ข้อคือ ข้อแรก การสร้าง population เริ่มต้นที่มีปริมาณมากทำให้ต้องใช้เวลามากในการหาเส้นทางที่จะได้คำตอบเหมือนเดิมทุกครั้ง ข้อสองคือการทำ mutation ซึ่งบางครั้งก็อาจจะทำให้ได้คำตอบดีกว่าการใช้ crossover

ความยากในการแก้ปัญหา TSP โดยใช้ GA คือการแทนค่าตัวเลขไปในคำตอบ ซึ่งลำดับการเดินทางอาจจะเรียงไม่ถูก จากรูปที่ 3.4 (ซ้าย) จะแสดงให้เห็นว่าวิธีการ crossover ใช้ไม่ได้ผล จากรูปได้ทำการ crossover ในตำแหน่งที่ 3 ตัวเลขใน parent 1 ก่อนจุดที่จะ crossover จะเป็นตัวเลขเดียวกับ Child 1 ดังนั้นหลังจุด crossover ในตำแหน่งที่ 3 จะเป็นตัวเลขที่มาจาก parent 2 และ Child 2 ก็ทำในลักษณะตรงกันข้าม

Parent 1	1 2 3 4 5
Parent 2	3 5 2 1 4
Child 1	1 2 3 1 4
Child 2	3 5 2 4 5

Parent 1	1 2 3 4 5
Parent 2	3 5 2 1 4
Child 1	1 2 3 5 4
Child 2	3 5 2 1 4

รูปที่ 3.4 แสดงลักษณะการ Crossover

จากรูปที่ 3.4 (ซ้าย) จะเห็นว่า Child 1 ซึ่งเกิดจากการ crossover ระหว่าง Parent 1 และ Parent 2 ที่ตำแหน่ง 3 ทำให้เกิดปัญหาว่า Child 1 ที่ได้จะแสดงเมือง 1 ถึง 2 ครั้ง ในขณะที่เมือง 5 ไม่ถูกแสดงเลย เราสามารถแก้ปัญหาดังกล่าวได้โดยการ นำตัวเลขจาก Parent 1 มาแสดงใน Child 1 โดยเลือกเฉพาะตัวเลขชุดก่อนตำแหน่งที่ crossover หลังจากนั้นนำตัวเลขจาก Parent 2 มาเทียบกับ Child 1 ทีละตัวตั้งแต่ตัวแรกถึงตัวสุดท้าย ถ้าเทียบแล้ว Child 1 ยังไม่มีตัวเลขที่นำมาเทียบ ก็ให้นำตัวเลขนั้นไปแสดงต่อท้าย Child 1 และ Child 2 ก็ทำคล้ายกับ Child 1 แต่เปลี่ยนเป็นนำตัวเลขใน Parent 2 ตำแหน่งก่อน crossover มาแสดงใน Child 2 ก่อน หลังจากนั้นนำตัวเลขใน Parent 1 มาเทียบกับ Child 2 ทีละตัว ถ้าเทียบแล้ว Child 2 ยังไม่มีตัวเลขที่นำมาเทียบ ก็ให้นำตัวเลขนั้นไปแสดงต่อท้าย Child 2 ดังรูปที่ 3.4 (ขวา)

ทุก ๆ ครั้ง ตัวเลขของเส้นทางจะถูกเลือกจากกลุ่มของเส้นทางซึ่งเรียกว่า Tournament set เส้นทางที่ดีที่สุด 2 เส้นทางแรกจะถูกนำมารวมกันเพื่อให้เกิดเป็นเส้นทางใหม่ 2 เส้นทางโดยการ crossover เส้นทางใหม่ที่ได้จะถูกนำไปแทนที่เส้นทางที่แย่ที่สุด (ระยะทางยาวที่สุด) 2 เส้นทาง

3.4 การประยุกต์ใช้ Travelling Salesman Problem (TSP)

การใช้ TSP ส่วนใหญ่ไม่ได้เพื่อต้องการนำมาประยุกต์ใช้โดยตรง แต่เพื่อนำสิ่งที่ได้จาก TSP มาศึกษาเกี่ยวกับวิธีการทั่ว ๆ ไปที่ถูกประยุกต์ใช้อย่างกว้างขวางในการทำ discrete optimization problem อย่างไรก็ตาม TSP ก็ไม่ได้ถูกนำมาใช้อย่างหลากหลาย แต่จริง ๆ แล้วการประยุกต์ใช้ TSP ตรง ๆ นำไปสู่การวิจัยและการทำงานในอนาคตได้

TSP เริ่มมาจากปัญหาการขนส่งหรือการส่งกำลังบำรุงในทหาร เช่น การหาเส้นทางของรถรับส่งนักเรียนที่จะต้องเดินทางไปรับนักเรียนยังที่ต่าง ๆ ซึ่งเป็นปัญหาที่สำคัญสมัยเริ่มแรกของ TSP

แม้ว่าการประยุกต์ใช้ TSP โดยส่วนใหญ่จะถูกใช้ในการขนส่ง แต่บางครั้งก็นำมาในรูปแบบอื่นได้ ได้แก่

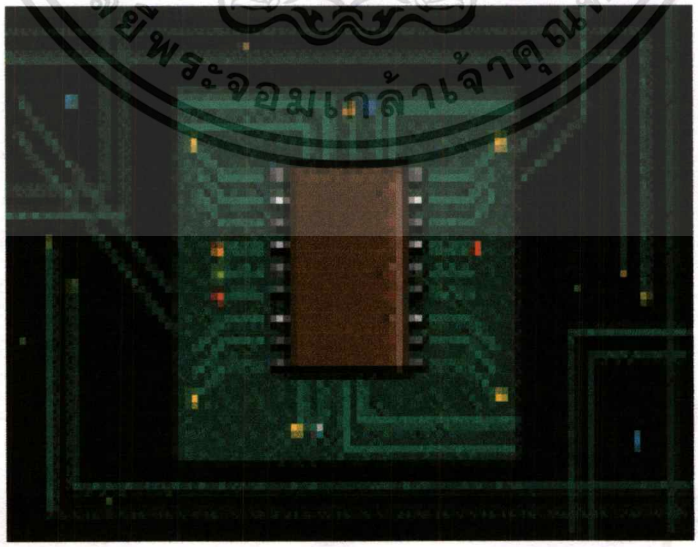
3.4.1 ตั้งโปรแกรมการทำงานของเครื่องจักรเพื่อเจาะรูแผงวงจรไฟฟ้า ซึ่งรูต่าง ๆ ก็เปรียบเสมือนเมืองต่าง ๆ และค่าใช้จ่ายก็เปรียบเสมือนเวลาที่ใช้ในการเลื่อนหัวเจาะจากรูหนึ่งไปอีกรูหนึ่ง ตราบใดที่เวลาในการเลื่อนหัวเจาะอยู่ในกระบวนการผลิต สามารถนำ TSP มาใช้เพื่อช่วยในการลดค่าใช้จ่าย

3.4.2 นักวิจัยแห่งสถาบันเพื่อสุขภาพแห่งชาติได้ใช้การแก้ปัญหาแบบ TSP เพื่อสร้างแผนที่ซึ่งใช้เป็นส่วนหนึ่งในงานของพวกเขาโดยจะใช้เป็นแผนที่ในการเดินทางไปยังท้องถิ่นต่าง ๆ โดยนำแผนที่ของหลาย ๆ ท้องถิ่นมารวมกันให้เป็นแผนที่ใหญ่อันเดียวแล้วใช้ TSP หาเส้นทางเพื่อที่จะเชื่อมต่อให้เป็นเส้นทางที่สั้นที่สุด ทำให้ช่วยลดค่าใช้จ่าย

3.4.3 ทีมงานวิศวกรแห่ง Hernandez Engineering ใน Houston และ Brigham Young University ได้ทำการทดลองโดยใช้ Chained Lin-Kerninghan เพื่อลดขั้นตอนในการสร้างภาพจำลองโดยใช้โปรแกรมการวัดลำแสงของดวงดาว เป้าหมายในการศึกษาเรื่องนี้คือเพื่อลดการใช้พลังงานทำภาพเคลื่อนไหวสำหรับดาวเทียมดู่หนึ่งที่ทำงานอยู่ (เปรียบเทียบว่าเมืองใน TSP คือวัตถุบนฟ้าที่ต้องการทำภาพ และค่าใช้จ่ายในการเดินทางคือพลังงานที่ถูกใช้ระหว่างที่ดาวเทียมถ่ายภาพจากดาวเทียมดวงแรกไปดวงถัดไป ดังรูปที่ 3.5

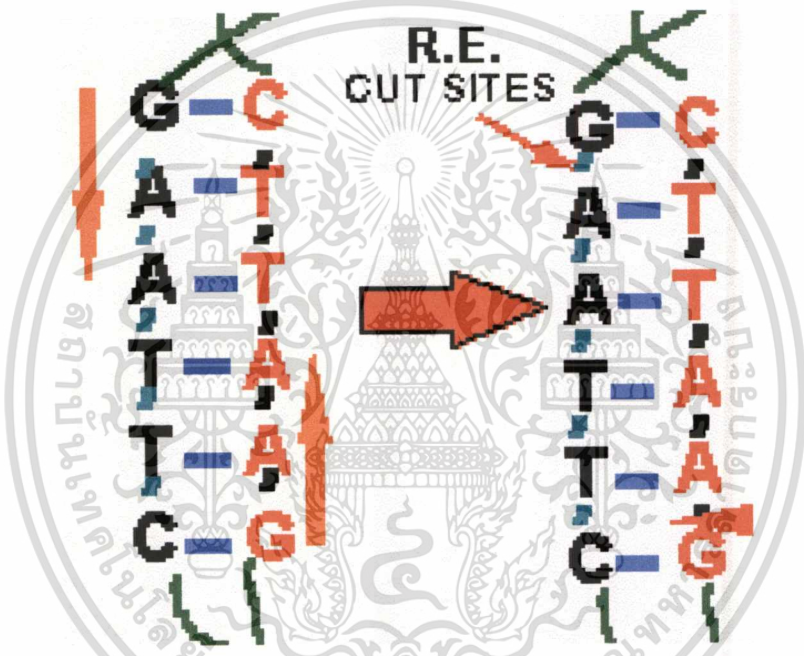
รูปที่ 3.5 แสดงการสร้างภาพจำลองโดยใช้ดาวเทียม

3.4.4 โรงงานผู้ผลิตเซมิคอนดักเตอร์ใช้ Chained Lin-Kerninghan ในการทดลองเพื่อลด Scan Chain ในแผงวงจรรวม ซึ่ง Scan Chain หมายถึงเส้นที่รวมเข้ากับ chip เพื่อทดสอบและมีประโยชน์ในการลดทั้งเวลาและพลังงาน ดังรูปที่ 3.6



รูปที่ 3.6 แสดง semi-conductor

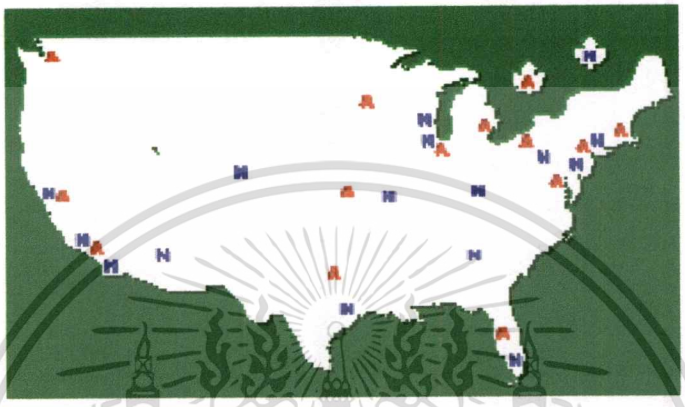
3.4.5 กลุ่มบริษัท AT&T ใช้ TSP เพื่อคำนวณลำดับของ DNA ในการทำโปรเจ็ควิจัยเกี่ยวกับพันธุวิศวกรรม โดยการนำเส้น DNA มารวมกัน ซึ่งแต่ละเส้นมีความยาว k และถูกเชื่อมต่อกับเส้นหลักภายนอก (แต่ละเส้นเป็นเส้นย่อย ๆ ของเส้นหลัก) มีเป้าหมายในการลดความยาวของเส้นหลัก เมืองใน TSP ก็เปรียบเสมือนเส้นหลักเหล่านั้นและค่าใช้จ่ายในการเดินทางคือค่า k ลบด้วยค่าสูงสุดของความแตกต่างระหว่างเส้นเหล่านั้น ดังรูปที่ 3.7



รูปที่ 3.7 แสดงการรวมกันของเส้น DNA

3.4.6 TSP ถูกใช้ในการแก้ปัญหาเส้นทางเดินรถ Whizzkids'96 จากการแสดงให้เห็นการแก้ปัญหาที่ชนะเลิศในปี 1996 ซึ่งปัญหาประกอบด้วยการค้นหาเส้นทางสำหรับเด็ก 4 คนเพื่อส่งหนังสือพิมพ์ให้ลูกค้า 120 ราย ซึ่งทีมประกอบด้วย David Applegate, William Cook, Sanjeeb Dash และ Andre Rohe

3.4.7 ผู้ชมเบสบอลต้องการที่จะหาเส้นทางเพื่อไปชมเบสบอล 30 นัดโดยใช้ TSP ช่วยในการค้นหาเส้นทาง ดังรูปที่ 3.8

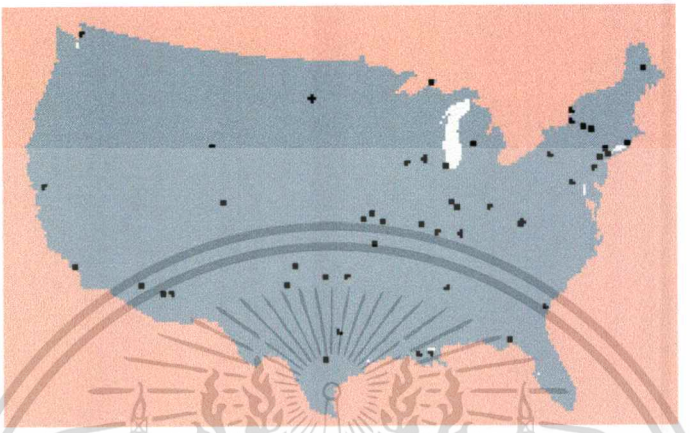


รูปที่ 3.8 แสดงแผนที่ของสวนสาธารณะ 30 แห่งที่มีการแข่งขันเบสบอล

3.4.8 การนำ TSP มาประยุกต์ใช้เพื่อช่วยในการจัดลำดับการเดินทางในการเก็บเหรียญที่โทรศัพท์สาธารณะตามเมืองต่าง ๆ ดังรูปที่ 3.9

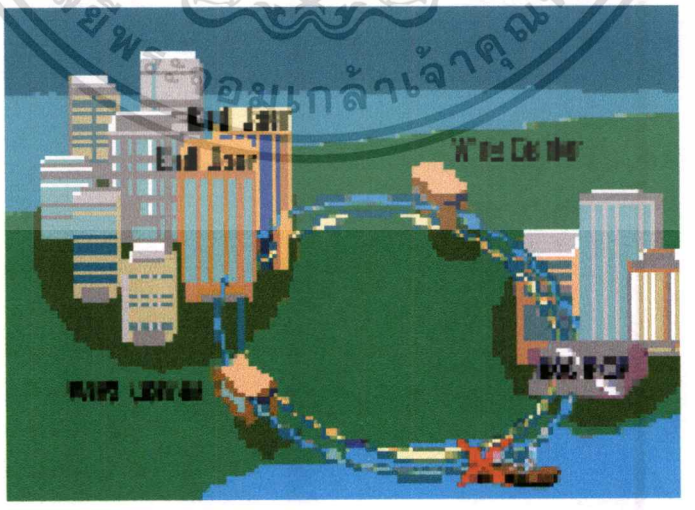
รูปที่ 3.9 แสดงโทรศัพท์สาธารณะที่หน่วยงานต้องเดินทางไปเก็บเหรียญตามพื้นที่ต่าง ๆ

3.4.9 การนำ TSP มาใช้เพื่อหาเส้นทางที่สั้นที่สุดที่จะไปเมืองต่าง ๆ โดยเลือกสนามบินให้เหมาะสม รูปภาพแสดงสนามบิน ณ ตำแหน่งต่าง ๆ แสดงดังรูปที่ 3.10



รูปที่ 3.10 แสดงแผนที่ตั้งสนามบิน

3.4.10 กำหนดการเดินทางขององค์กรที่ไม่แสวงหาผลกำไรถูกคำนวณโดยใช้ TSP การเดินทางซึ่งรวมถึงค่าเช่าเครื่องบินเพื่อเดินทางไปยังเมืองต่าง ๆ 48 รัฐในภาคพื้นทวีปรวมถึง Washington D.C. (เคยไปอลาสก้าและฮาวายมาแล้ว) ถ้าหากปัญหาที่เกิดขึ้นถูกแก้ไข



รูปที่ 3.11 แสดงการเดินทางไปยังเมืองต่าง ๆ

คล้ายกับปัญหาที่เกิดขึ้นในปี 1954 ซึ่งแก้ไขโดย Dantzig, Fulkerson และ Johnson แต่เมืองต่าง ๆ ในที่นี้บางเมืองก็เดินทางโดยรถยนต์จึงไม่สามารถนำมาคำนวณเพื่อเปรียบเทียบกับการเดินทางโดยเครื่องบินได้ ดังรูปที่ 3.11

3.4.11 Version ล่าสุดของ TSP เพื่อใช้ในการหาขั้นตอนการทำงานเป็นเครื่องมือในการออกแบบ fiber optical networks แห่งสถาบันวิจัย Bell Communication (ปัจจุบันคือ Telcordia) TSP มุ่งสู่ปัญหาในการเกิดเส้นทางของ sonet ring ซึ่งทำให้เกิดเครือข่ายเชื่อมโยงผ่านไปยังองค์กรต่าง ๆ ดังรูปที่ 3.12

รูปที่ 3.12 แสดงการเชื่อมต่อของ sonet ring

ผลจากการที่เคยใช้ TSP ช่วยในการติดตั้งสาย cable เพื่อส่งกำลังและอุปกรณ์อิเล็กทรอนิกส์รวมกับ fiber optic ที่เชื่อมต่อมายังบ้านของเรา

บทที่ 4
ขั้นตอนการสร้างระบบงาน

การใช้ทฤษฎีของ Genetic Algorithm เพื่อช่วยในการหาเส้นทางการเดินทางที่น้อยที่สุด (Shortest Path) สามารถช่วยให้ประหยัดเวลาในการเดินทางรวมถึงประหยัดค่าใช้จ่ายด้วย

การสร้างระบบงานเริ่มตั้งแต่การศึกษาทฤษฎีที่เกี่ยวข้องกับ Genetic Algorithm ตามที่ได้อธิบายไว้ในบทที่ 2 หลังจากนั้นจึงทำการวิเคราะห์และออกแบบระบบงาน เมื่อได้รูปแบบที่ต้องการแล้วจึงทำการเขียนโปรแกรมเพื่อให้ระบบสามารถแก้ปัญหาได้ตามทฤษฎีของ Genetic Algorithm หลังจากนั้นทำการทดสอบระบบและนำไปใช้งาน ต่อไปนี้จะได้อธิบายรายละเอียดของขั้นตอนการสร้างระบบโดยเริ่มที่การวิเคราะห์ปัญหา ส่วนทฤษฎีสามารถศึกษาได้จากบทที่ 2

4.1 การวิเคราะห์ปัญหา (Problem Definition)

กำหนดว่าสิ่งที่เราต้องการจะทราบคือการหาเส้นทางการเดินทางที่น้อยที่สุด (Shortest Path) ซึ่งจากการศึกษาทฤษฎีของ Genetic Algorithm สามารถนำมาช่วยในการแก้ปัญหาได้ เป็นการนำเทคโนโลยีสารสนเทศเข้ามาช่วยในการแก้ปัญหานี้

4.2 วิเคราะห์ความต้องการ (Requirement Analysis)

จากการศึกษาทฤษฎีของ Genetic Algorithm ทำให้สามารถทราบได้ว่าปัญหาในการหาเส้นทางการเดินทางที่น้อยที่สุด (Shortest Path) สามารถนำทฤษฎีนี้มาช่วยในการแก้ปัญหาได้ และมีความเป็นไปได้ที่จะได้ผลลัพธ์ที่ถูกต้องและรวดเร็วและผลลัพธ์ที่ได้สามารถนำมาประยุกต์ใช้กับธุรกิจของธนาคาร ทำให้ช่วยลดค่าใช้จ่ายลงได้

4.3 การออกแบบ (Design)

ระบบที่สร้างขึ้นได้พัฒนาบน Microsoft Access97 ซึ่งรองรับการพัฒนาระบบในรูปแบบของฐานข้อมูลและสามารถทำการเขียนโปรแกรม VBA ได้ด้วย ระบบที่สร้างขึ้นใช้ฐานข้อมูลเพียงเล็กน้อยคือใช้ในการเก็บชื่อจังหวัดและเส้นทางจากจังหวัดต้นทางไปจังหวัดปลายทาง จึงประกอบด้วยเพียง 2 Table และสามารถแสดง Table ออกมาในรูปแบบดังนี้

4.3.1 การออกแบบ Table

ประกอบด้วย 2 Table คือ

ตารางที่ 4.1 Data Dictionary ของระบบงาน (ข้อมูลรายชื่อจังหวัด)

ID	Item (English)	Item (Thai)	Unq	Type	Domain		Remark
					Lgth	Null	
1	**Province**	จังหวัด					
	ProvNo	รหัสจังหวัด	PK,FK	Integer		NN	
	ProvName	ชื่อจังหวัด		Text	50	NN	

ตารางที่ 4.2 ตัวอย่าง Table ของระบบงาน (ข้อมูลรายชื่อจังหวัด)

ProvNo	ProvName
1	กรุงเทพมหานคร
2	กำแพงเพชร
3	ชัยนาท
4	นครนายก
5	นครปฐม
6	นครสวรรค์
7	นนทบุรี
8	ปทุมธานี
9	พระนครศรีอยุธยา
10	พิจิตร
11	พิษณุโลก
12	เพชรบูรณ์
13	ลพบุรี
14	สมุทรปราการ

ตารางที่ 4.2 Data Dictionary ของระบบงาน (ข้อมูลระยะทางระหว่างจังหวัด)

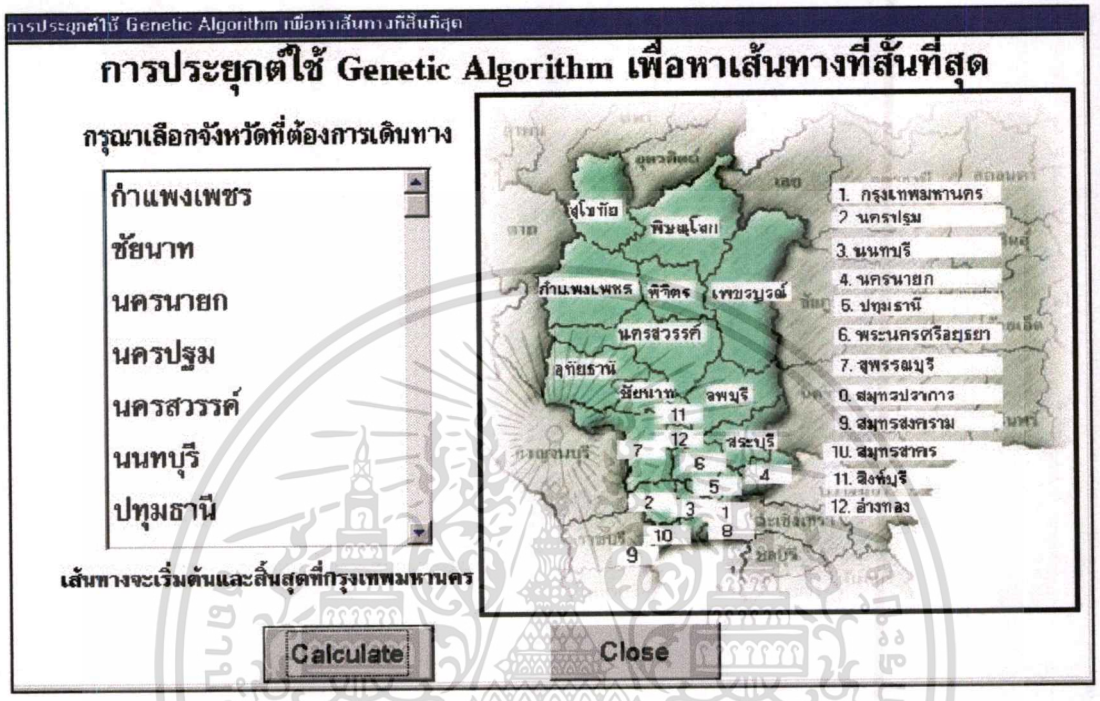
ID	Item (English)	Item (Thai)	Unq	Type	Domain		Remark
					Lgth	Null	
2	**Distance**	**ระยะทาง**					
	Prov1	รหัสจังหวัดต้นทาง	PK	Integer		NN	
	Prov2	รหัสจังหวัดปลายทาง	PK	Integer		NN	
	Distance	ระยะทาง		Integer		NN	

ตารางที่ 4.4 ตัวอย่าง Table ของระบบงาน (ข้อมูลระยะทางระหว่างจังหวัด)

Prov1	Prov2	Distance
1	3	194
2	3	181
1	4	107
2	4	465
3	4	301
4	5	163
1	5	56
3	5	205
2	5	414
3	6	64
2	6	117
5	6	296
4	6	347

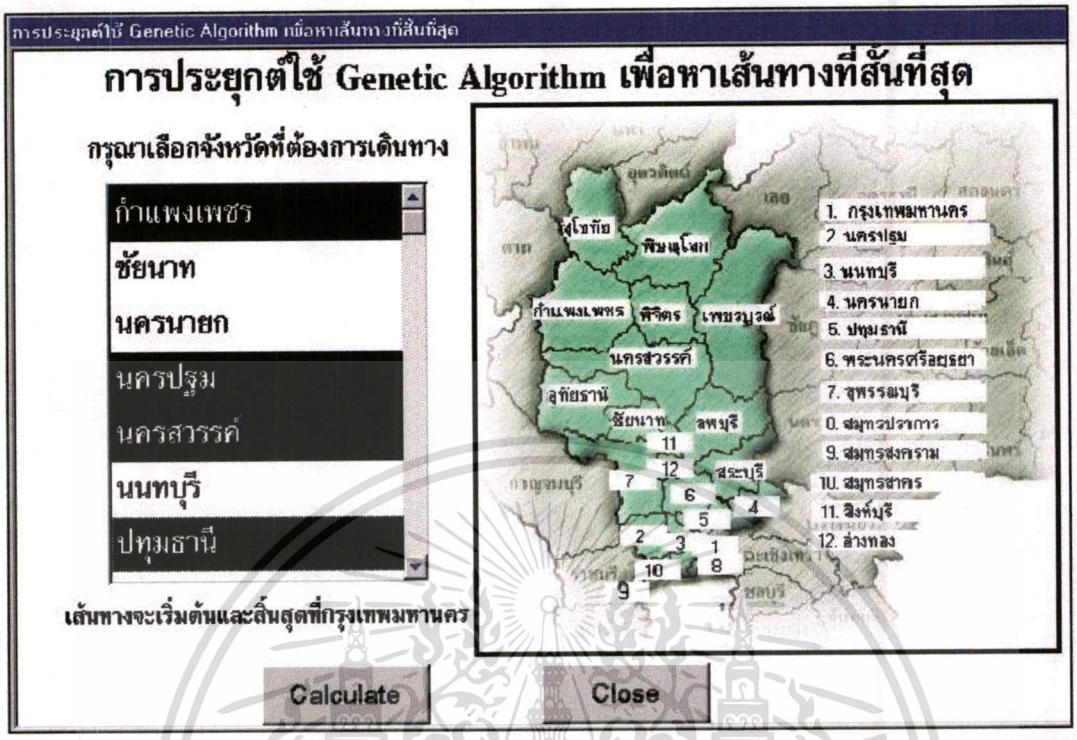
4.3.2 การออกแบบหน้าจอ

การออกแบบหน้าจอของโปรแกรมนี้แสดง 1 หน้าจอ เพื่อทำการ Input ข้อมูล ดังรูปที่ 4.1

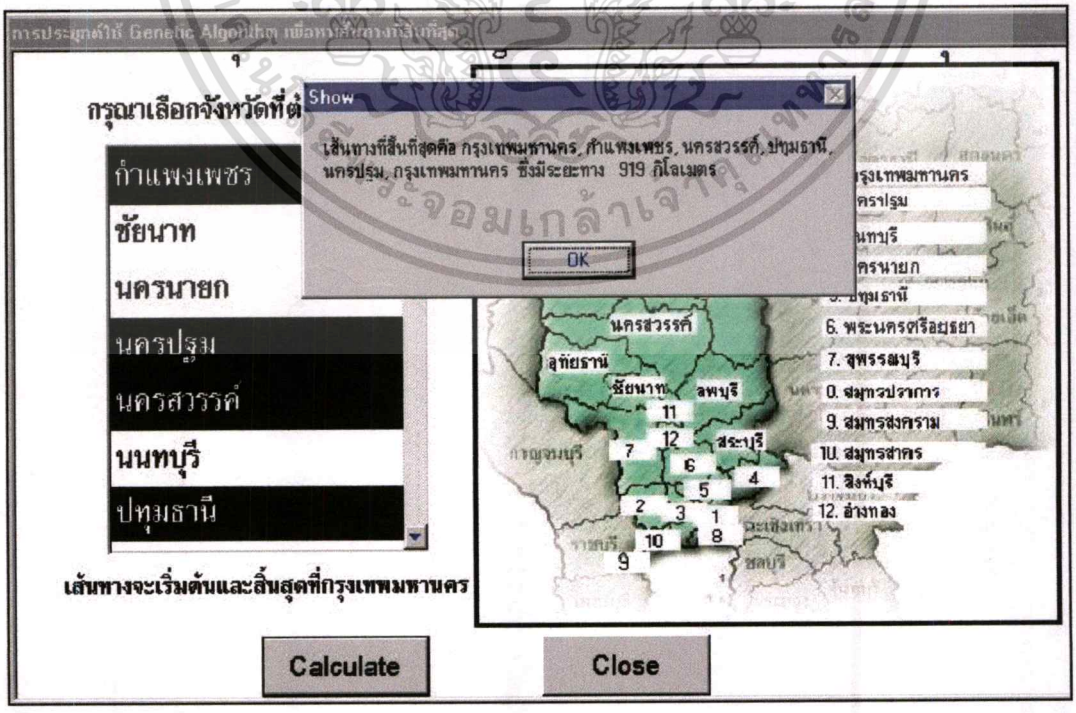


รูปที่ 4.1 แสดงหน้าจอเพื่อ Input ข้อมูล

จากหน้าจอ Input ข้อมูล จะแสดง List Box เพื่อให้ผู้ใช้สามารถเลือกชื่อจังหวัด โดยการใช้เม้าส์คลิกที่ชื่อจังหวัด หากต้องการเลือกจังหวัดที่ติดกันให้กดปุ่ม Shift แล้วคลิกชื่อจังหวัดที่ต้องการ หากต้องการเลือกชื่อจังหวัดที่ไม่อยู่ติดกันให้กดปุ่ม Ctrl แล้วคลิกชื่อจังหวัดที่ต้องการ ดังรูปที่ 4.2 เราสามารถดูตำแหน่งของจังหวัดต่าง ๆ ในประเทศได้จากแผนที่ในหน้าจอ เพื่อผู้ใช้จะได้ใช้ในการดูเส้นทาง ซึ่งในที่นี้จะแสดงเฉพาะข้อมูลของจังหวัดภาคกลาง เมื่อผู้ใช้เลือกจังหวัดเรียบร้อยแล้วให้กดปุ่ม “Calculate” เพื่อให้โปรแกรมทำการคำนวณเส้นทางที่สั้นที่สุดในการเดินทางไปยังจังหวัดที่ได้เลือกไว้ ผลจากการคำนวณแสดงดังรูปที่ 4.3

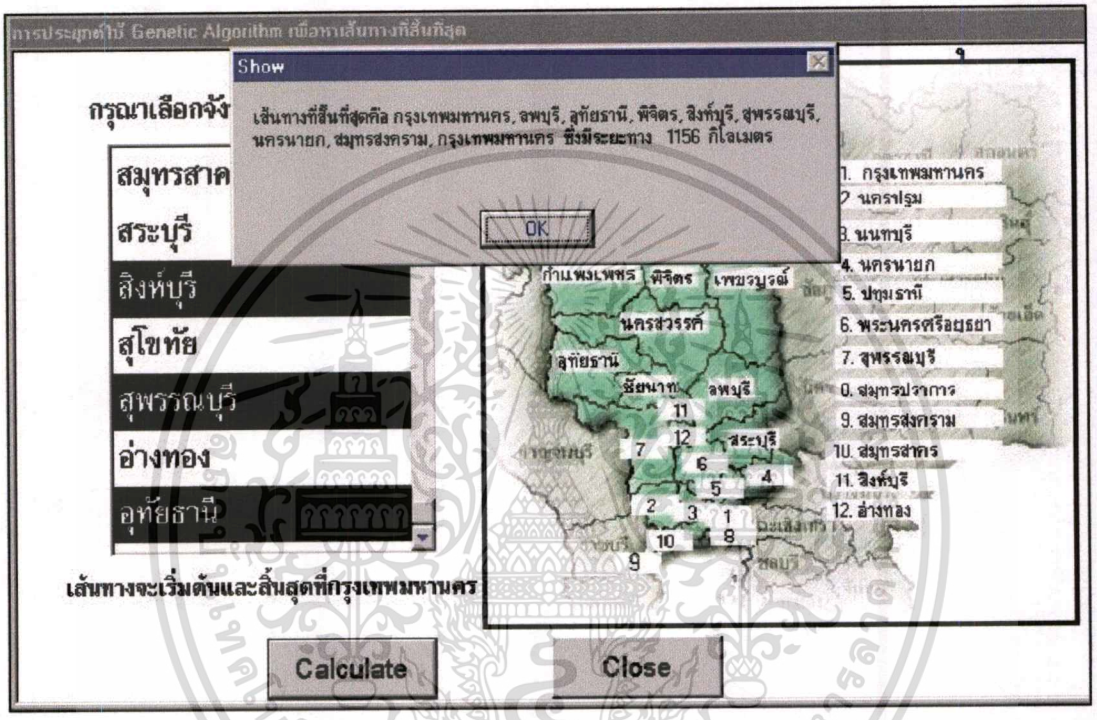


รูปที่ 4.2 แสดงตัวอย่างการเลือกจังหวัดที่จะเดินทาง



รูปที่ 4.3 หน้าจอแสดงผลการคำนวณจากโปรแกรม

จากหน้าจอได้เลือกไว้ 4 จังหวัด ได้แก่ กำแพงเพชร นครปฐม นครสวรรค์และปทุมธานี เมื่อกดปุ่ม “Calculate” แล้วโปรแกรมจะทำการคำนวณระยะทางได้ 919 กิโลเมตร โดยเริ่มต้นที่กรุงเทพมหานคร นครสวรรค์ กำแพงเพชร ปทุมธานี และไปสิ้นสุดที่กรุงเทพมหานคร ซึ่งเป็นระยะทางที่สั้นที่สุด เมื่อลองเลือกจำนวนจังหวัดให้มากขึ้น จะได้ผลลัพธ์ดังรูปที่ 4.4



รูปที่ 4.4 หน้าจอแสดงผลการคำนวณจากโปรแกรมเมื่อเลือกจำนวนจังหวัดมากขึ้น

จากหน้าจอได้เลือกไว้ 7 จังหวัด ได้แก่ นครนายก พิจิตร ลพบุรี สมุทรสงคราม สิงห์บุรี สุพรรณบุรี และอุทัยธานี เมื่อกดปุ่ม “Calculate” แล้วโปรแกรมจะทำการคำนวณระยะทางได้ 1,156 กิโลเมตร โดยเริ่มต้นที่กรุงเทพมหานคร ลพบุรี อุทัยธานี พิจิตร สิงห์บุรี สุพรรณบุรี นครนายก สมุทรสงคราม และไปสิ้นสุดที่กรุงเทพมหานคร ซึ่งเป็นระยะทางที่สั้นที่สุด เมื่อต้องการออกจากโปรแกรมให้กดปุ่ม “Close”

4.4 การเขียนโปรแกรม

โปรแกรมที่เขียนขึ้นมานี้พัฒนาบน Microsoft Access 97 ซึ่งมีขั้นตอนการทำงานของโปรแกรมดังนี้

4.4.1 เมื่อผู้ใช้เลือกจังหวัดที่ต้องการจะเดินทางจาก List Box เรียบร้อยแล้ว กดปุ่ม "Calculate" โปรแกรมจะเริ่มทำงานโดยการเก็บค่าจังหวัดที่ถูกเลือกเพื่อเดินทางใช้วิธีการวนลูปเพื่อเก็บค่าลำดับจังหวัดและนับว่ามีจังหวัดที่เลือกทั้งหมดกี่จังหวัด โดยเส้นทางจะต้องเริ่มต้นและสิ้นสุดที่กรุงเทพมหานครเสมอ โดยที่ i จะทำการอ่านค่าจาก List Box ทีละจังหวัด เมื่อมีจังหวัดใดถูกเลือก X จะทำการนับว่ามีการถูกเลือกกี่จังหวัด

4.4.2 ถ้ามีจังหวัดที่ถูกเลือกเพียงจังหวัดเดียวก็คำนวณระยะทางจากกรุงเทพมหานครถึงจังหวัดที่ถูกเลือกแล้วคูณด้วย 2 เพื่อเป็นระยะทางขากลับ ส่วนฟังก์ชันที่ใช้ในการอ่านค่าระยะทางมาจาก table คือ Dlookup

4.4.3 เมื่อมีจังหวัดถูกเลือก 2 จังหวัด โปรแกรมจะคิดระยะทางจากกรุงเทพมหานครไปยังจังหวัดที่ 1 และจังหวัดที่ 2 ตามลำดับ แล้วกลับมายังกรุงเทพมหานครอีกครั้ง แล้วนำระยะทางทั้งหมดมาบวกกัน

4.4.4 เมื่อมีจังหวัดถูกเลือก 3 จังหวัด โปรแกรมจะทำการสุ่มเลข 1 ถึง 3 เรียงกันไป 3 หลักและไม่ซ้ำกันในแต่ละหลักซึ่งเรียกว่าเป็น 1 โครโมโซม และต้องสร้างทั้งสิ้น 6 โครโมโซม และแต่ละโครโมโซมก็ต้องไม่ซ้ำกันด้วย เมื่อได้ครบทั้ง 6 โครโมโซมแล้วก็มาตรวจสอบว่าระยะทางของโครโมโซมใดมีเส้นทางสั้นที่สุด ก็เลือกเส้นทางนั้น แสดงดังรูปที่ 4.5

เลือกจังหวัด A, B, C รวม 3 จังหวัด จะได้เส้นทางต่าง ๆ ดังนี้

1. A B C ระยะทาง 50 กิโลเมตร
2. A C B " 61 "
3. B A C " 59 "
4. B C A " 54 "
5. C A B " 48 "
6. C B A " 53 "

รูปที่ 4.5 ตัวอย่างแสดงการหาเส้นทางจากการเลือก 3 จังหวัด

จากตัวอย่างจะเห็นว่าโครโมโซมที่ 5 มีระยะทางสั้นที่สุดคือ 48 กิโลเมตร จึงเลือกโครโมโซมนี้เป็นคำตอบ

4.4.5 เมื่อมีจังหวัดถูกเลือกมากกว่า 3 จังหวัด โปรแกรมจะทำงานดังเช่นข้อ 4.4.4 แต่จะทำการสุ่มขึ้นมาเป็น 10 โครโมโซม แสดงดังรูปที่ 4.6

เลือกจังหวัด A, B, C, D จะสุ่มมาสร้าง 10 เส้นทาง

1. A B C D ระยะทาง 87 กิโลเมตร
2. A B D C " 96 "
3. B D C A " 78 "
4. B A D C " 85 "
5. B C D A " 90 "
6. C D A B " 100 "
7. C B A D " 99 "
8. C A B D " 96 "
9. D A B C " 89 "
10. D B A C " 81 "

รูปที่ 4.6 แสดงการสุ่มเลือกเส้นทางจากการเลือกมากกว่า 3 จังหวัด

หลังจากนั้นจะเพิ่มกระบวนการ CrossOver ขึ้นมาเพื่อให้เกิดการวิวัฒนาการไปในทางที่ดีขึ้น โครโมโซมที่จะถูกใช้ในการ CrossOver จะถูกสุ่มมาจากโครโมโซมทั้ง 10 ที่ได้สร้างไว้แล้ว ส่วนหลักการในการสุ่มคือโครโมโซมใดแทนเส้นทางที่มีระยะทางน้อยจะมีสิทธิ์ในการถูกสุ่มมากกว่าโครโมโซมที่แทนเส้นทางที่มีระยะทางมาก โดยใช้หลักการของ Roulette Wheel ในการสุ่ม คือ แบ่งวงกลมเป็นส่วน ๆ ทั้งวงแทน 100 เปอร์เซ็นต์ โครโมโซมใดที่ระยะทางน้อย จะมีสิทธิ์ถูกเลือกมาก ดังนั้นจะมีพื้นที่มาก ดังรูปที่ 4.7

ใช้ Roulette ในการสุ่มเลือกเส้นทางมา Crossover

1. A B C D ระยะทาง 87 กิโลเมตร
3. B D C A " 78 "
4. B A D C " 85 "
9. D A B C " 89 "

รูปที่ 4.7 แสดงตัวอย่างของโครโมโซมที่ถูกสุ่มมา 4 ตัว

เมื่อสุ่มได้มา 2 โครโมโซมแล้ว ให้นำโครโมโซมทั้งสองมาทำการ CrossOver กัน โดยดูหลักการ CrossOver ได้จากบทที่ 2 ผลลัพธ์ที่ได้จากการ CrossOver จะเกิดเป็นโครโมโซมใหม่ซึ่งอาจจะแทนเส้นทางที่มีระยะทางมากกว่าหรือน้อยกว่า 10 โครโมโซมแรกก็ได้จำนวน 2 โครโมโซม และทำการ CrossOver 2 ชุด ดังนั้นจะเกิดเป็นประชากรใหม่ 4 โครโมโซม ดังรูปที่ 4.8

หลังจาก Crossover แบบ One Point จะได้

ชุดที่ 1

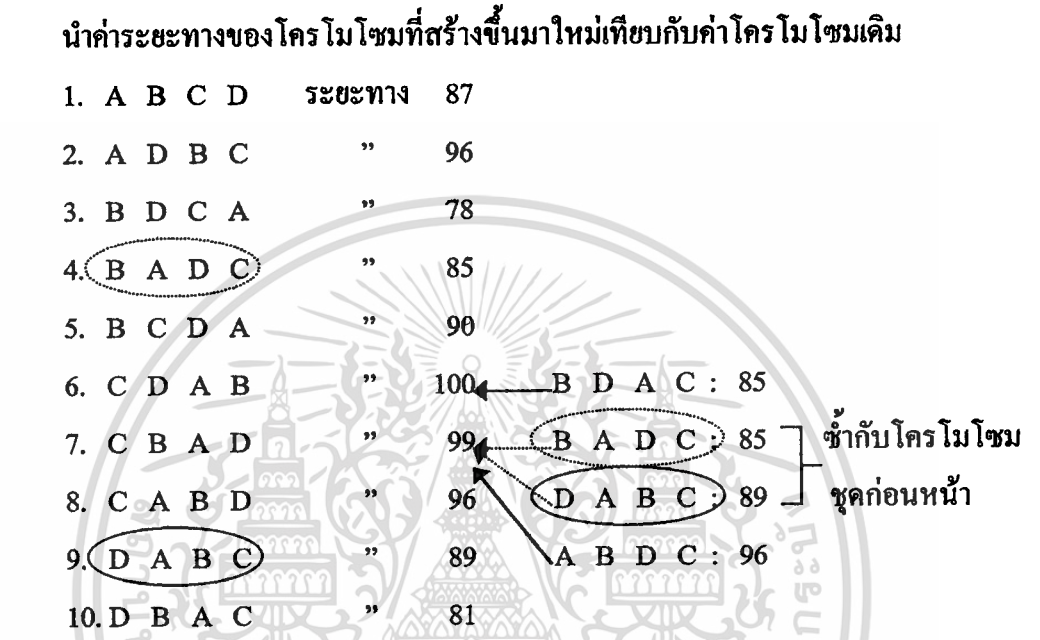
1. A B | C D → A B D C = 96 กิโลเมตร
3. B D | C A *Crossover* B D A C = 85 กิโลเมตร

ชุดที่ 2

4. B A | D C → B A D C = 85 กิโลเมตร
9. D A | B C *Crossover* D A B C = 89 กิโลเมตร

รูปที่ 4.8 แสดงการ Crossover ของโครโมโซมที่ถูกเลือก

จากนั้นนำโครโมโซม 10 โครโมโซมแรกมาเปรียบเทียบกับโครโมโซมใหม่ที่เกิดขึ้นอีก 4 โครโมโซม หากค่าที่น้อยที่สุดของโครโมโซมชุด 4 อันหลังน้อยกว่าค่าที่มากที่สุดของโครโมโซมชุด 10 อันแรก ให้แทนค่าน้อยนั้นลงไปแทนค่าของโครโมโซมที่มากที่สุดในชุด 10 อันแรกดังรูปที่ 4.9



รูปที่ 4.9 นำโครโมโซมชุดใหม่ที่สร้างขึ้นมาเทียบกับโครโมโซมชุดเดิม

จากรูปที่ 4.9 จะเห็นว่าโครโมโซมชุดใหม่ที่สร้างขึ้น เมื่อนำมาเทียบกับโครโมโซมชุดเดิมแล้ว หากพบว่าเป็นโครโมโซมที่ซ้ำกับของเดิม จะไม่นำมาแทนที่ แต่จะนำโครโมโซมที่ดีตัวถัดมาทำการเปรียบเทียบต่อไป หลังจากที่ไม่พบว่ามีค่าที่น้อยที่สุดของโครโมโซมชุดใหม่น้อยกว่าค่าที่มากที่สุดของโครโมโซมชุดเดิม โปรแกรมจะหยุดการเปรียบเทียบและแทนค่า แล้วนำโครโมโซมที่ได้เปรียบเทียบเรียบร้อยแล้ว นำไปเป็นโครโมโซมต้นกำเนิดในรอบถัดไป ดังรูปที่ 4.10 ซึ่งโครโมโซมชุดที่ได้นี้ จะถูกนำค่าระยะทางของโครโมโซมที่น้อยที่สุดไปเก็บไว้ในตัวแปรไว้ก่อน แล้วทำขั้นตอน CrossOver ซ้ำ หากระยะทางที่ได้ครั้งใหม่น้อยกว่าค่าระยะทางน้อยที่สุดที่เก็บไว้ในตัวแปร ให้แทนค่าระยะทางน้อยที่สุดครั้งใหม่ทับไปในตัวแปรเลย ทำเช่นนี้ซ้ำไปเรื่อย ๆ จนกระทั่งค่าของตัวแปรเท่าเดิมหรือลดลงติดต่อกันเป็นจำนวน 500 รอบ ให้นำค่าที่เก็บไว้ในตัวแปรมาแสดงผลโดยใช้ msgbox ในการแสดงผล

เส้นทางที่จะนำไปใช้สุ่มในรอบถัดไป

1. A B C D	ระยะทาง	87	กิโลเมตร
2. A D B C	”	96	”
3. B D C A	”	78	”
4. B A D C	”	85	”
5. B C D A	”	90	”
6. B D A C	”	85	”
7. A B D C	”	96	”
8. C A B D	”	96	”
9. D A B C	”	89	”
10. D B A C	”	81	”

รูปที่ 4.10 แสดงเส้นทางที่จะใช้สุ่มในรอบถัดไป

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการศึกษา

การสร้างโปรแกรมเพื่อใช้ในการคำนวณหาเส้นทางและระยะทางที่สั้นที่สุดโดยอาศัยหลักการของ Genetic Algorithm นั้น เป็นเครื่องมือที่ช่วยคำนวณให้สามารถแก้ปัญหาได้อย่างรวดเร็ว และคำตอบที่ได้เป็นที่น่าพอใจ เป็นผลให้สามารถนำคำตอบที่ได้ไปใช้ในการกำหนดเส้นทางเพื่อเดินทางไปตรวจสาขาต่างจังหวัด

ในการศึกษานี้ ได้ทำการสร้างโปรแกรมซึ่งช่วยในการคำนวณเส้นทางโดยอาศัยหลักการของ Genetic Algorithm เพื่อให้ได้คำตอบในเวลาอันรวดเร็ว ซึ่งโปรแกรมที่สร้างขึ้นมีการทำงานที่ไม่ซับซ้อน ใช้งานง่าย ประกอบด้วยขั้นตอนหลัก ๆ ดังนี้

- การ Input ข้อมูล โดยเลือกจังหวัดต่าง ๆ ที่อยู่ใน List Box
- การคำนวณหาเส้นทางและระยะทางที่สั้นที่สุด
- การแสดงผลลัพธ์ทางหน้าจอต่อผู้ใช้

ผลจากการสร้างโปรแกรม นอกจากจะนำไปใช้ในธุรกิจของธนาคารในการคำนวณหาเส้นทางเพื่อเดินทางไปตรวจสาขายังต่างจังหวัดซึ่งก่อให้เกิดประโยชน์ในการลดค่าใช้จ่ายในการเดินทางแล้ว ยังสามารถนำโปรแกรมดังกล่าวไปใช้กับงานอื่น ๆ ได้โดยการเพิ่มหรือเปลี่ยนค่าข้อมูลใน Table เช่น จากเดิมเป็นข้อมูลของจังหวัดต่าง ๆ ในภาคกลาง อาจจะเพิ่มเป็นจังหวัดทั่วทั้งประเทศ หรืออาจจะเปลี่ยนเป็นข้อมูลของประเทศต่าง ๆ ทั่วโลกก็ได้ ซึ่งก็คือนำไปใช้ในธุรกิจท่องเที่ยว หรือประยุกต์ใช้กับงานด้านอื่น ๆ ได้ในลักษณะเดียวกันนี้

5.2 ข้อเสนอแนะ

เนื่องจากระยะเวลาในการสร้างและพัฒนาโปรแกรมมีค่อนข้างจำกัด การทดสอบโปรแกรมอาจจะยังไม่ทั่วถึง แต่จากการที่ได้ทดสอบยังไม่พบข้อผิดพลาด หากภายหลังได้มีการนำโปรแกรมดังกล่าวไปประยุกต์ใช้กับระบบงานอื่นอาจจะพบข้อผิดพลาดที่ผู้จัดทำไม่เคยพบมาก่อน แต่ผู้จัดทำมีความมั่นใจว่าโปรแกรมสามารถทำงานได้อย่างมีประสิทธิภาพ สิ่งที่ผู้จัดทำคิดว่าต้องแก้ไขเพิ่มเติมหากมีผู้สนใจจะพัฒนาโปรแกรมต่อไปคือ

- แผนที่ของภาคกลางที่ได้แสดงในหน้าจอของโปรแกรมควรจะสามารถแสดงให้เห็นเส้นเชื่อมต่อของจังหวัดต่าง ๆ ที่เป็นผลลัพธ์จากการคำนวณ

- ข้อมูลที่นำมาใช้อาจจะยังไม่มากพอที่จะแสดงให้เห็นประสิทธิภาพในการคำนวณโดยใช้หลักการของ Genetic Algorithm ได้ เนื่องจากการใช้หลักการดังกล่าวเหมาะสมกับข้อมูลจำนวนมาก ๆ หากข้อมูลมีจำนวนน้อยสามารถคำนวณเองได้อย่างคร่าว ๆ โดยที่ไม่ต้องเสียเวลาในการสร้างโปรแกรมขึ้นมาคำนวณ

5.3 ประโยชน์ที่ได้รับจากโครงการ

ประโยชน์ที่ได้รับในการวางแผนงานสำหรับองค์กร ช่วยในการหาเส้นทางในการเดินทาง มีดังนี้

5.3.1 เป็นข้อมูลให้ผู้ตรวจสอบเลือกเส้นทาง

5.3.2 ผู้ตรวจสอบได้ข้อมูลรวดเร็วและประหยัดเวลา

5.3.3 ประหยัดค่าใช้จ่ายในการเดินทาง

5.3.4 เพิ่มประสิทธิภาพในการทำงานขององค์กร

บรรณานุกรม

บัณฑิต เลขะนุกิจ. 1998. **Microcomputer Magazine.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น

เทพฤทธิ์ บัณฑิตวัฒนวงศ์. 2000 **Internet Magazine.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

Applegate, D. et.al. 2001. **Solving Travelling Salesman Problems.** [Online]. Available : http://www.math.princeton.edu/tsp.

Falkenauer,E. 1998. **GENETIC ALGORITHMS AND GROUPING PROBLEMS.** John Wiley & Sons Ltd.

MarekObitko. 1998. **Introduction to Genetic Algorithms.** [Online]. Available : http://www.cs.felk.cvut.cz/~xobitko/ga/index.html.

NeuroDimension. 2002. **Genetic Algorithm.** [Online]. Available : http://www.nd.com/products/genetic.htm.

Peterson and Ivars. 1990. **Islands of Truth: a mathematical mystery cruise.** New York : W. H Freeman and Co.

Ramsey, M. 2000. **Genetic Algorithm Optimizer.** [Online]. Available : http://www.ai.bpa.arizona.edu/~mramsey/ga.html.

ประวัติผู้เขียน

ชื่อผู้เขียน	น.ส.เมธาวี มหาอัมพรพฤกษ์
วันเดือนปีเกิด	23 พฤษภาคม พ.ศ. 2517
สถานที่เกิด	นครราชสีมา
วุฒิการศึกษาระดับปริญญาตรี	วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์
สถานที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ มหาวิทยาลัยขอนแก่น
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2538
อาชีพปัจจุบัน	พนักงานธนาคาร ตำแหน่งพนักงานตรวจสอบ ฝ่ายตรวจสอบสารสนเทศ ธนาคารทหารไทย จำกัด (มหาชน)