

รายงาน

เรื่อง Genetic Algorithm

จัดทำโดย

นางสาวอลิสสา แฉล้มในเมือง รหัสนักศึกษา 521733022003-4

นายกมลภพ กวดสันเทียะ รหัสนักศึกษา 521733022004-2

นายสงคราม พริยะอนุพนธ์ รหัสนักศึกษา 521733022016-6

เสนอ

อาจารย์ธรรมกร ครองไตรภาพ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ปัญญาประดิษฐ์ (Artificial Intelligence)

สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน

ภาคเรียนที่ 2 ปีการศึกษาที่ 2555

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ปัญญาประดิษฐ์ (Artificial Intelligence) ทางคณะผู้จัดทำได้ค้นคว้าหาข้อมูลต่างๆจากหนังสือของห้องสมุด และ ค้นคว้าหาจากอินเทอร์เน็ต เกี่ยวกับ Genetic Algorithm ขั้นตอนทางพันธุศาสตร์นำมาประยุกต์ใช้โดยมีเนื้อหา และ องค์ประกอบของ Genetic Algorithm ได้แก่ Chromosome Encoding, Initial population, Fitness Function, Genetic Operator, Parameter มีขั้นตอนกระบวนการทำงานของ Genetic Algorithm ในการค้นหาคำตอบที่ถูกต้องของปัญหา และ สรุปข้อมูล Genetic Algorithm ไว้ในรายงานฉบับนี้ ซึ่งจะเป็นประโยชน์กับผู้สนใจเกี่ยวกับ Genetic Algorithm หากผิดพลาดประการใดทางคณะผู้จัดทำขออภัยไว้ ณ ที่นี้ด้วย

โจทย์

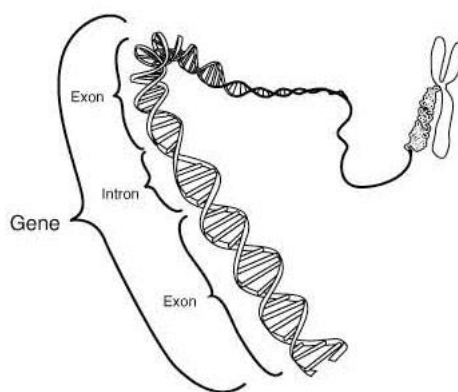
จงเขียนโปรแกรม ขั้นตอนวิธีเชิงพันธุกรรมหาค่าสูงสุดของฟังก์ชัน $y = x^2$ โดยกำหนดให้ x มีค่าอยู่ระหว่าง 0 ถึง 31?

กำหนดลักษณะการทำงานของขั้นตอนวิธีเชิงพันธุกรรม โดยกำหนด

- ให้สร้างโครโมโซมอยู่ในรูปแบบไบนารีโครโมโซมขนาด 5 บิต โดยโครโมโซมของปัญหาจะมีค่าอยู่ระหว่าง 00000 ถึง 11111 ซึ่งเมื่อถอดรหัสและจะมีค่าอยู่ระหว่าง 0 ถึง 31
- ฟังก์ชันเป้าหมายเป็น $y = x^2$
- ฟังก์ชันความเหมาะสมเป็น $Y = x^2$
- ขนาดของประชากร (Population Size: N) เป็น 10
- สามารถกำหนดอัตราการไขว้เปลี่ยน (Crossover Rate: P_C) ได้
- สามารถกำหนดอัตราการเกิดการกลายพันธุ์ (Mutation Rate: P_M) ได้
- และสามารถกำหนดให้สร้างประชากรโครโมโซมสูงสุดกี่รุ่นก็ได้ (Generations)

ประวัติ Genetic Algorithm

Evolutionary strategies ค้นพบโดย I.Rechenber ในประเทศ เยอรมันปี 1960 เป็นจุดเริ่มต้นการพัฒนา Genetic Algorithm ใน สหรัฐอเมริกา 1975 Genetic Algorithm ถูกค้นพบโดย John Holland คือกระบวนการเรียนแบบวิวัฒนาการของสิ่งมีชีวิตที่เกิดขึ้นในธรรมชาติโดยเกี่ยวข้องกับยีน ต่อมา ในปี 1992 John Koza ได้ใช้ Genetic Algorithm พัฒนา Evolve Programs หรือ เรียกว่า Genetic Programming ทั้ง Genetic Algorithm, Genetic Programming อยู่บนพื้นฐานการพัฒนาการของสิ่งมีชีวิตตามหลักของธรรมชาติที่ว่าด้วย ทฤษฎีการคัดเลือกตามธรรมชาติ และทฤษฎีการสืบทอดลักษณะทางพันธุกรรมของยีน



จินเนติก อัลกอริทึม การปรับเปลี่ยนสารพันธุกรรมของยีน หมายถึงวิธีการแก้ปัญหาที่ใช้แนวทางเดียวกับวิธีการที่สิ่งมีชีวิตปรับตัวเองให้เข้ากับสภาพแวดล้อม หรือวิวัฒนาการ วิธีการที่ถูกโปรแกรมให้ทำการเปลี่ยนแปลง หรือปรับปรุงส่วนประกอบ ของระบบโดยการสร้างขึ้นมาใหม่ การดัดแปลงและการคัดสรรวิธธรรมชาติใช้ ให้เกิดสิ่งใหม่ๆมาใช้งาน ประโยชน์ของการแก้ปัญหา ได้แก่ การออกแบบสินค้า และตรวจสอบระบบการทำงานต่างๆ เป็นต้น

Genetic Algorithm เป็นการรวมกันระหว่างคำว่า Genetic และ Algorithm ซึ่งแต่ละคำมีความหมายดังนี้

Genetic คือ วิชาพันธุศาสตร์ที่ว่าด้วยการศึกษากระบวนการถ่ายทอดลักษณะทางพันธุกรรมจากบรรพบุรุษสู่รุ่นลูกหลานโดย โครโมโซมจะเป็นตัวแทนในการถ่ายทอด โครโมโซมคือสายรหัสของ DNA ประกอบไปด้วยยีนซึ่งแต่ละยีนจะมีการบ่งบอกลักษณะพิเศษเอาไว้ เช่น สีตา สีผม ซึ่งโครโมโซมเหล่านี้มีบทบาทอย่างมากในกระบวนการถ่ายทอดลักษณะทางพันธุกรรมจากรุ่นหนึ่งไปยังรุ่นหนึ่ง ซึ่งจะมีปรากฏการณ์ธรรมชาติเกิดขึ้นในระหว่างการถ่ายทอดลักษณะทางพันธุกรรม ทำให้เกิดการเปลี่ยนแปลงในแต่ละรุ่น ซึ่งจะมีกระบวนการที่เกิดขึ้นเหล่านี้ ทำให้เกิดวิวัฒนาการ ซึ่งจะเกิดปรากฏการณ์ Crossover การข้ามสายพันธุ์ และ mutation การกลายพันธุ์เป็นกระบวนการเปลี่ยนแปลงที่สำคัญที่เรียกว่า วิวัฒนาการ Evolution

Evolution วิวัฒนาการ ในสิ่งมีชีวิต คือ กระบวนการเปลี่ยนแปลงหรือกลายไปสู่สถานะที่ดีขึ้นหรือเจริญขึ้นเป็นการเปลี่ยนแปลงในทางชีววิทยาจากสิ่งที่ย่ำๆ ไปสู่สิ่งที่ยิ่งขึ้นมากขึ้น หรือ ซับซ้อนน้อยลง เพื่อการดำรงอยู่ตามความเหมาะสมของสิ่งมีชีวิตชนิดนั้น การเปลี่ยนแปลงนี้จะต้องเปลี่ยนในลักษณะค่อยเป็นค่อยไป และต้องใช้เวลานาน

ในการถ่ายทอดลักษณะทางพันธุกรรมจะมีกระบวนการที่ทำให้เกิดการเปลี่ยนแปลง ที่เรียกว่า วิวัฒนาการ นั่นคือ กระบวนการ Selection, Crossover และ Mutation โดยคำนวณหาค่าความเหมาะสม Fitness Function ที่สอดคล้องกับวัตถุประสงค์ของปัญหา Objective Function กำหนดให้กับโครโมโซมแต่ละตัว เพื่อนำไปสู่กระบวนการคัดเลือก แต่บางปัญหาไม่สามารถที่จะคำนวณหาเพื่อให้ได้ค่าความเหมาะสม ที่ตรงกับความจริงที่โครโมโซมนั้นควรจะได้รับ และสิ่งนี้จึงทำให้เกิดปัญหาขึ้น ดังนั้นในการกำหนดค่าความเหมาะสมของโครโมโซมแต่ละตัว จึงเป็นสิ่งที่สำคัญ และจะต้องถูกประมาณค่าที่ดี เพราะถ้าหากค่าความเหมาะสมสูงกว่าความเป็นจริงที่โครโมโซมนั้นควรได้รับ อาจทำให้เกิดความเสียหายในการสืบค้นข้อมูลในแต่ละครั้ง และครั้งต่อไปได้ซึ่งในที่นี้หมายความว่า สมการหรือฟังก์ชันที่ใช้ในการหาค่าความเหมาะสมให้กับโครโมโซมแต่ละตัวไม่สามารถกำหนดได้แน่นอน และหาค่าได้ไม่ดีเท่ากับค่าความจริงที่คำนวณได้จากมนุษย์หรือผู้ใช้ เช่น ทางด้านศิลปะ เพลงสถาปัตยกรรม และการออกแบบ เป็นต้น ผู้ใช้สามารถให้ค่าความจริงว่าสิ่งที่ผู้ใช้เห็น หรือได้ยินนั้นอยู่ในระดับใด

Interactive Genetic Algorithm จะช่วยแก้ปัญหาในส่วนที่เกิดขึ้น และคำนึงถึงในส่วนของความรู้สึกรับของมนุษย์หรือนำค่าความจริงที่เกิดขึ้นจากความรู้สึกรับของมนุษย์ นำมาใช้ในการติดต่อสื่อสารกันระหว่างมนุษย์กับคอมพิวเตอร์ Emotion Human – Computer Interfaces นั้นหมายความว่า ค่าความเหมาะสมที่โครโมโซมแต่ละตัวควรได้รับนั้น ต้องเป็นค่าที่เกิดขึ้นจากความรู้สึกของผู้ใช้เองว่ารูปร่างลักษณะใดที่ผู้ใช้ต้องการ และนำค่าที่ได้นำมาใช้ในการสืบค้น หรือดึงข้อมูลรูปร่างที่สื่อถึงอารมณ์ตามความต้องการของผู้ใช้ เพื่อได้ผลลัพธ์ที่ผู้ใช้พอใจมากที่สุด

Algorithm คือ วิธีคิดหรือชุดคำสั่งที่มีการเรียงลำดับขั้นตอนด้วยกระบวนการทางคณิตศาสตร์ เพื่อแก้ปัญหา และปัญหามีหลายแบบจึงทำให้มีวิธีการแก้ปัญหาที่เหมาะสมกับแต่ละแบบ

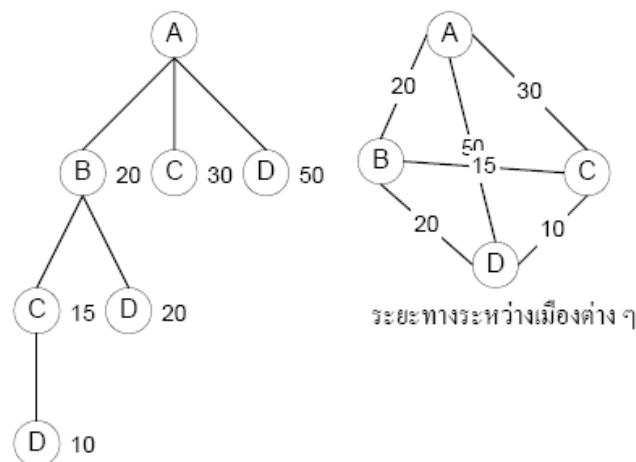
Genetic Algorithm สองคำนี้ได้มารวมกันจึงเป็นวิธีการค้นหาและแก้ปัญหาเพื่อได้สิ่งที่เหมาะสมที่สุด Optimum points โดยพัฒนาและจำลองมาจากกระบวนการทางพันธุกรรม คือ ทฤษฎีการอยู่รอดของสิ่งมีชีวิต Genetic Algorithm เป็นทางเลือกที่ใช้แก้ปัญหาที่ดีเพราะวิธีการใช้แก้ปัญหาเป็นวิธีการแบบสุ่มและช่วยแก้ปัญหาที่มีขนาดใหญ่และซับซ้อนได้เนื่องจากมีคุณสมบัติการเรียนรู้แบบการถ่ายทอดทางพันธุกรรมตามธรรมชาติ ซึ่งจะนำค่าที่เหมาะสมที่สุดจากประชากรรุ่นก่อนมาใช้ในการพิจารณาในการหาคำตอบของประชากรรุ่นถัดมา ซึ่งจะนำถ่ายทอดลักษณะทางพันธุกรรมจากบรรพบุรุษไปสู่รุ่นลูกหลานโดยใช้ค่า Fitness Function ที่สอดคล้องกับ Objective

Function ในการพิจารณาคำตอบโดยมีการพิจารณาว่า โครโมโซมใดควรที่จะนำมาสืบสายพันธุ์ หรือไม่ควรนำมาสืบสาย จะทำให้สามารถหาคำตอบที่มีค่าสูงสุดหรือต่ำสุดที่สมบูรณ์ได้

Algorithm พื้นฐานที่ใช้อย่างแพร่หลายในปัจจุบัน ได้แก่

1. Greedy Algorithms หมายถึง เป็นอัลกอริทึมที่จะหาคำตอบโดยการเลือกทางออกที่ดีที่สุดที่พบได้ในขณะนั้นเพื่อให้ได้คำตอบที่ดีที่สุด แต่ในบางครั้ง Greedy Algorithms อาจจะไม่สามารถหาคำตอบของปัญหาที่ดีที่สุดได้เสมอไป ขั้นตอนหา Greedy Algorithms มีดังนี้

1. เลือกโหนดเริ่มต้นมาหนึ่งโหนด
2. โหนดที่เลือกมานี้เป็นสถานะปัจจุบัน
3. ให้ทำตามขบวนการข้างล่างนี้จนกว่าจะไม่สามารถสร้างโหนดลูกได้อีก
 - 3.1 สร้างสถานะใหม่ที่เป็นโหนดลูกที่เป็นไปได้ทั้งหมดจากสถานะปัจจุบัน
 - 3.2 จากสถานะใหม่ที่สร้างขึ้นมาทั้งหมด ให้เลือกสถานะ หรือ โหนดลูก ที่ดีที่สุดออกมาเพียงโหนดเดียว
4. กลับไปที่ขั้นตอนที่ 2 อีกครั้ง



จากรูปที่9 การแก้ปัญหาเริ่มจาก การเลือก a เป็นเมืองเริ่มแรก จากนั้นทำการสร้างโหนดลูก B C และ D หาระยะทางระหว่าง A ถึงเมืองที่เหล่านี้นี้ได้ 20 30 และ 50 ตามลำดับเลือก B เป็นเมืองที่จะเดินทางต่อมาจากนั้นสร้างโหนดลูกของ B ได้ C และ D และได้ระยะทางเท่ากับ 15 และ 20 ตามลำดับ เลือก C เป็นเมืองที่จะเดินทางต่อไป จากนั้นสร้างโหนดลูกให้ C ได้ D มีค่าเท่ากับ 10 เลือกเดินทางที่ D เป็นเมืองสุดท้ายก่อนกลับไป A รวมระยะทางเท่ากับ $20+15+10+50=90$ แต่ในบางครั้ง Greedy Method อาจจะไม่สามารถหาคำตอบของปัญหาที่ดีที่สุดได้เสมอไป

2. Dynamic Programming หมายถึง วิธีการหลีกเลี่ยงการคำนวณหาคำตอบซ้ำๆ โดยการแก้ปัญหาย่อยๆ

ในบางครั้งเราไม่สามารถแบ่งปัญหาออกเป็นปัญหาย่อยๆได้ ถ้าเราพยายามแบ่งปัญหานั้นๆ ออกเป็นปัญหาย่อยที่เล็กที่สุด ขั้นตอนของเราอาจจะใช้เวลาทำงานเป็นแบบ Exponential ได้แต่เวลาที่เรากลับไปปัญหาต่างๆ มักจะพบว่าเราต้องแก้ปัญหาย่อยๆ ที่เหมือนกันและซ้ำไปซ้ำมาเพื่อหลีกเลี่ยงการคำนวณหาคำตอบซ้ำๆ ซากๆ Dynamic Programming จึงแก้ปัญหาย่อยๆ เหล่านี้เพียงครั้งเดียวจากนั้นก็เก็บผลลัพธ์ไว้ ถ้าหากพบว่าต้องแก้ปัญหานี้อีกเราก็สามารถนำคำตอบจากคำตอบจากคำตอบที่เคยคำนวณเก็บไว้มาใช้ได้เลย โดยไม่ต้องประมวลผลใหม่จะช่วยให้ประหยัดเวลาในการทำงานได้มาก

3. Iterative method หมายถึง วิธีการทำซ้ำเพื่อใช้ในการหาคำตอบของระบบสมการเชิงเส้นที่มีขนาดใหญ่ อย่างมีประสิทธิภาพได้คำตอบที่เที่ยงตรง และมีค่าผิดพลาดน้อย

4. Divide and conquer หมายถึง การแตกปัญหาเป็นปัญหาย่อย แล้วหาคำตอบ จากนั้นรวมคำตอบของปัญหาเป็นอัลกอริทึมที่จะมีการนำปัญหาหลักที่ได้มาทำการแยกออกเป็นปัญหาย่อยๆ แล้วนำคำตอบที่ได้จากปัญหาย่อยต่างๆ มารวมกันเข้าด้วยกัน โดยอัลกอริทึมนี้เราสามารถหาคำตอบของปัญหาได้ง่ายขึ้น จากการรวมคำตอบของปัญหาหลักนั่นเอง

อย่างเช่น การแก้ปัญหาคาดค่าอย่างง่าย มีอยู่ที่จะต้องส่งคูปองไปจำนวนเท่าไร เพื่อให้ได้ผลกำไรที่คุ้มค่าในการส่งคูปองไปพร้อมกับจดหมายคูปองในครั้งแรก จะเห็นว่ามีปัญหาง่ายๆ ให้แก้ได้ง่ายๆ ก็เพียงส่งคูปองให้มากที่สุดเท่าที่จะทำได้ ดังนั้นลูกค้าที่จะรับและใช้คูปองอย่างแท้จริงจะต้องมีประสิทธิภาพมากที่สุดเท่าที่จะเป็นไปได้ ปัญหาจะสร้างความยุ่งยากที่ละน้อยอย่างไรรู้ตาม เพราะว่ามีปัจจัยอื่น ๆ อีกที่จะทำให้ข้อเสนอในจดหมายทำกำไร จำนวนของคูปองที่มีจดหมายที่มีน้ำหนักมาก และมีมูลค่าสูง ด้วยเหตุนี้ผลกำไรจึงลดลง คูปองบางอันที่ไม่ถูกใช้ ผลรวมของรายได้ก็จะหายไปด้วย ถ้ามีคูปองมากเกินไปในจดหมายที่ส่ง ลูกค้าก็จะมามากเกินไป และไม่สามารถใช้คูปองได้ ปัญหานี้ สามารถถอดรหัสได้ใน Genetic Algorithms อย่างง่ายซึ่งส่วนประกอบแต่ละอันของระบบองค์กร มีขึ้นเดียวที่แสดงถึงการคาดเดาที่ดีที่สุด ขององค์กรที่จะหาจำนวนคูปอง โปรแกรมคอมพิวเตอร์นี้จะง่ายเหมือนตัวเลขหนึ่งตัวที่สะท้อนถึงคูปองหลายๆ ใบที่จะใส่ไปกับจดหมาย Genetic Algorithm สามารถดำเนินการร่วมกับการสร้างประสิทธิภาพด้วยการสร้างประชากรของยีนเดียวจากองค์กรด้วยวิธีการสุ่มและผ่านการเลียนแบบการวิวัฒนาการ การปรับปรุงยีน ทั้งคนที่ไร้ประสิทธิภาพที่สุดออกและทำการเลียนแบบและแก้ไขคนที่ดีที่สุด เมื่อเวลาผ่านไปก็จะได้จำนวนของคูปองที่ให้ผลกำไรที่แน่นอน

5. Case Study หมายถึง เรื่องราวหรือเหตุการณ์ที่เกิดขึ้นจริง ซึ่งได้มีการรวบรวมมาเสนอให้ทราบข้อเท็จจริงพร้อมทั้งข้อมูลต่างๆ ที่เกี่ยวข้องเพื่อจะได้ศึกษาอภิปราย แลกเปลี่ยนความคิดเห็น

และวิเคราะห์เรื่องที่เกิดขึ้น แล้วสรุปแนวทางการตัดสินใจ หรือวิธีแก้ปัญหาที่เห็นว่าดีที่สุด
เหมาะสมที่สุด และอำนวยความสะดวกมากกว่าแนวทางหรือวิธีแก้ปัญหาอื่นๆ

Genetic Algorithm มีองค์ประกอบที่สำคัญ 5 ส่วนด้วยกัน ดังนี้

1. **Chromosome Encoding** หรือรูปแบบโครโมโซมที่ใช้ในการนำเสนอทางเลือกที่สามารถจะเป็นไปได้ของแต่ละปัญหา
2. **Initial Population** คือ ประชากรต้นกำเนิดที่จะนำไปในกระบวนการถ่ายทอดลักษณะทางพันธุกรรม
3. **Fitness Function** ฟังก์ชันสำหรับประเมินค่าความเหมาะสม เพื่อให้คะแนนแต่ละทางเลือกของคำตอบ
4. **Genetic Operator** ซึ่งใช้ในการปรับเปลี่ยนองค์ประกอบของข้อมูลตลอดกระบวนการ ได้แก่ Selection, Crossover และ Mutation
5. **Parameter** ที่สำคัญสำหรับ Genetic Algorithm เช่น ขนาดของประชากร (Population size) ความน่าจะเป็นของการ crossover (Probability crossover) ความน่าจะเป็นของการ mutation (Probability mutation) และจำนวนรุ่น เป็นต้น

องค์ประกอบหลักๆของ Genetic Algorithm มีรายละเอียดดังนี้

1. **Chromosome Encoding** คือ ขั้นตอนสำหรับแปลงทางเลือกสำหรับการแก้ปัญหาที่เป็นไปได้ให้อยู่ในรูปแบบของ Chromosome ในการแปลงวิธีการสำหรับแก้ปัญหาที่เป็นไปได้ให้อยู่ในรูปแบบของ Chromosome นั้นสามารถที่จะทำได้ในหลายรูปแบบซึ่งแล้วแต่ความเหมาะสมของแต่ละปัญหา
- รูปแบบโครโมโซมที่ได้จากปัญหา ในการถอดรหัสนั้นจะขึ้นอยู่กับปัญหา และในปัจจุบันปัญหามีมากมาย จึงทำให้รูปแบบของโครโมโซมมีความแตกต่างกันออกไปตามปัญหานั้นๆ
- เช่น

-Binary ทุกตำแหน่งของยีนบนโครโมโซมจะมีค่าเป็นบิต 0 หรือ 1 เช่น โครโมโซม A :
00110010001110

Chromosome A	101100101100101011100101
Chromosome B	111111100000110000011111

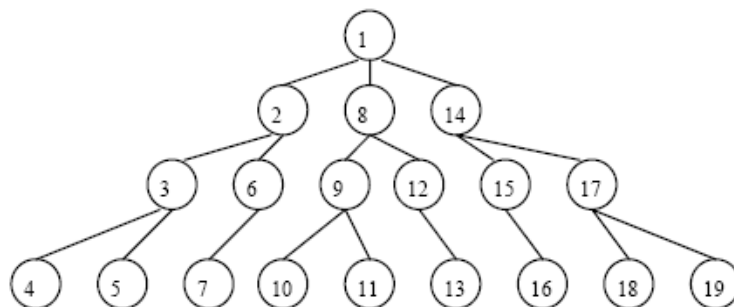
-Value Encoding หรือ Direct ทุกตำแหน่งของยีนบนโครโมโซมจะมีค่าบางค่า ที่สามารถเชื่อมโยงไปยังปัญหาได้ เช่น ตัวอักษร จำนวนจริง คำสั่ง หรืออื่นๆ รูปแบบโครโมโซมแบบนี้สามารถใช้ได้กับปัญหาที่ค่อนข้างซับซ้อนค่าได้

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLEFGTYTGABA
Chromosome C	(back), (back), (right), (forward), (left)

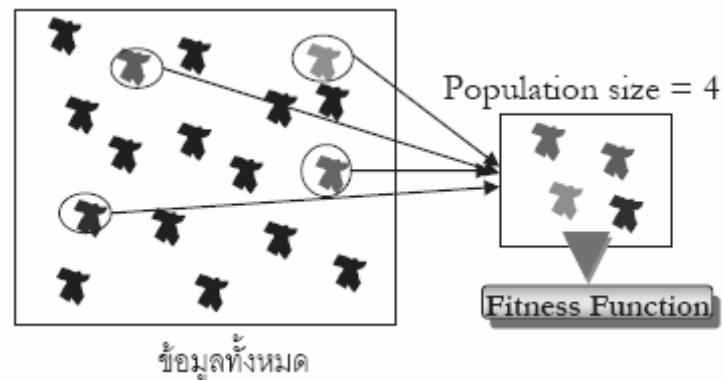
-Permutation Encoding เป็นการกระทำคั้งแรกก่อนที่จะเข้ากระบวนการของ Genetic Algorithm ประชากรที่เกิดจากการสุ่ม Random เพื่อนำประชากรเข้าไปในกระบวนการ ในการสุ่ม จะต้องสุ่มให้ได้จำนวนเท่ากับขนาดของรุ่นที่ได้กำหนดไว้โดยที่ยังไม่มีการสนใจค่าความเหมาะสมของแต่ละโครโมโซม B: 9 5 2 1 4 6 7 8 3

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

- Tree Encoding ทุกตำแหน่งของยีนจะเป็น node ของต้นไม้



2. Initial population คือ ลักษณะที่เป็นต้นแบบหรือต้นกำเนิดที่จะนำเข้าไปในกระบวนการถ่ายทอดลักษณะทางพันธุกรรม โดยการสุ่มเลือกเรื่องสร้างประชากรต้นแบบขึ้นมาเพื่อใช้เป็นจุดเริ่มต้นของขั้นตอนการวิวัฒนาการขั้นตอนนี้จะเป็นขั้นตอนแรกที่เกิดขึ้นก่อนที่จะเริ่มเข้ากระบวนการของ จินเนติก อัลกอริทึม โดยประชากรกลุ่มแรก หรือประชากรต้นกำเนิด จะเกิดจากการสุ่มเลือกขึ้นมาจาก กลุ่มของประชากรทั้งหมดที่มีอยู่ โดยในการสุ่มเลือกจะทำการสุ่มตามจำนวนของประชากรที่ได้กำหนดไว้เป็น Parameter ของ Algorithm



จากรูป แสดงการสุ่มหาประชากรเริ่มต้นจำนวน 4 โครโมโซม

3. Fitness Function คือ เป็นวิธีการสำหรับประเมินค่าความเหมาะสม เพื่อให้คะแนนแต่ละทางเลือกของคำตอบต่างๆอย่างเหมาะสม โครโมโซมทุกตัวจะมีค่าความเหมาะสมของตัวเองเพื่อใช้สำหรับพิจารณาว่า โครโมโซมตัวนั้น เหมาะสมหรือไม่ ที่จะนำมาใช้ในการสืบทอดพันธุกรรม สำหรับสร้างโครโมโซมรุ่นใหม่ โดยวิธีการสำหรับคิดค่าความเหมาะสมนั้น จะใช้สมการที่สอดคล้องกับแต่ละปัญหา เช่น กำหนดให้ค่าความเหมาะสม = จำนวนของบิต 1 ทั้งหมดในโครโมโซม A: 100011100 ดังนั้นโครโมโซม A มีค่าความเหมาะสมเท่ากับ 4

4. Genetic Operator ซึ่งเป็นวิธีการปรับเปลี่ยนองค์ประกอบของข้อมูลทุกขั้นตอนได้แก่ Selection, Crossover และ Mutation

-Selection เพื่อให้เกิดการอยู่รอดของสิ่งมีชีวิตนั้น โดยคัดเลือกมาเป็นโครโมโซมพ่อและโครโมโซมแม่ หรือที่เรียกว่า Parent ในการสืบสายพันธุ์ ทำให้เกิดปัญหาว่าจะทำอย่างไรให้เกิดจากการคัดเลือกโครโมโซมที่น่าพอใจเพื่อที่จะเกิดการอยู่รอดของสิ่งมีชีวิตตามทฤษฎีของ Charles Darwin จึงทำให้เกิดรูปแบบมากมายในการเลือกโครโมโซมที่น่าพอใจที่สุดเพื่อนำไปสืบสายพันธุ์ทำให้เกิดรูปแบบการคัดเลือกมากมายเพื่อให้เกิดผลลัพธ์ที่น่าพอใจที่สุด เช่น การคัดเลือกแบบ Roulette Wheel การคัดเลือกแบบ Ranking การคัดเลือกแบบ Tournament การคัดเลือกแบบ Elitist การคัดเลือกแบบ Steady-state และอื่นๆมากมายหลายวิธีเพื่อให้ได้มาซึ่งวิธีการคัดเลือกโครโมโซมที่ดี

-Crossover เป็นกระบวนการที่สำคัญ Genetic Algorithm เมื่อเกิดการ Crossover ขึ้นในทางพันธุศาสตร์จะทำให้เกิดการเปลี่ยนแปลงของสิ่งมีชีวิตที่หลากหลาย ซึ่งการ Crossover จะต้องอาศัยวิวัฒนาการเป็นเวลานาน จึงสามารถเลือกเอาคำตอบที่เหมาะสมกับความต้องการได้มากที่สุดและขั้นตอนในการ Crossover คือ นำ 2 โครโมโซม Parent มาผสมกันเพื่อให้ได้โครโมโซมใหม่ขึ้นมา จากนั้นใช้วิธีการที่ง่ายที่สุด คือ สุ่มตำแหน่ง Crossover และทำการคัดลอกทุกอย่างที่อยู่หน้าตำแหน่ง Crossover ของพ่อและคัดลอกทุกอย่างหลังตำแหน่ง Crossover ของแม่รวมกันจะได้ลูกตัวที่ 1 ออกมา จากนั้นทำการคัดลอกทุกอย่างที่อยู่หน้าตำแหน่ง Crossover ของแม่ และคัดลอกหลังตำแหน่ง Crossover ของพ่อรวมกันจะได้ลูกตัวที่ 2 ออกมาดังรูป

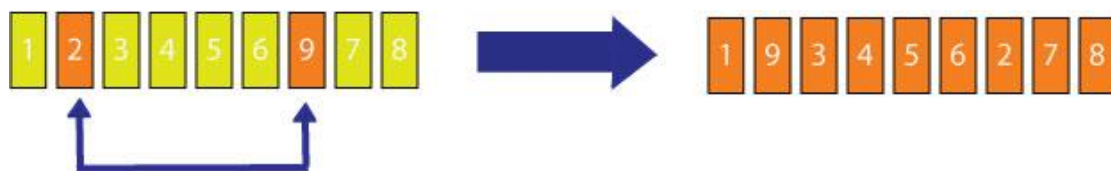


รูปแสดงการCrossover

เราสามารถเลือกตำแหน่งอื่นๆ ในการ Crossover ได้ขึ้นอยู่กับ การเข้ารหัสChromosome

ลักษณะของการ Crossover จะสร้างตามลักษณะของปัญหาซึ่งสามารถปรับปรุงได้ตามประสิทธิภาพของ Genetic Algorithm ได้

-Mutation เกิดขึ้นหลังจากการ Crossover เสร็จสิ้น จะทำการสุ่มประชากรเปลี่ยนแปลงผลที่ได้จากการ Crossover หมายความว่า รุ่นลูกที่เกิดจากผสมจากรุ่นพ่อแม่แล้วจึงนำรุ่นลูกมาดำเนินการ Mutation ต่อไป ซึ่งการ Mutation ทางพันธุศาสตร์จะทำให้ได้ลักษณะใหม่ๆ เกิดขึ้น ขั้นตอนในการ Mutation เมื่อได้ตำแหน่ง Mutation แล้วเปลี่ยน แปลงค่า ณ ตำแหน่งที่สุ่มนั้น ในตัวอย่างต่อไปนี้ จะดำเนินการ Mutation กับรูปแบบโครโมโซมแบบ Binary เราจะสุ่มเลือก bit เพียงเล็กน้อย จะมีการเปลี่ยนแปลงค่าจาก 1 เป็น 0 หรือจาก 0 เป็น 1 ดังรูป ภายใต้เงื่อนไขของการ Mutation โครโมโซมรูปแบบ Permutation Encoding



Offspring 1 :1101111000011110

Offspring 2 :1101100100110110

Mutated Offspring 1 :1100111000011110

Mutated Offspring 2 :1101101100110110

จากรูป แสดงการ Mutation กับโครโมโซมรูปแบบ Permutation Encoding

Fitness Function การวัดค่าความเหมาะสม หรือ ฟังก์ชันวัดค่าความเหมาะสม คือ ฟังก์ชันที่ใช้ในการประมาณว่าเส้นทางแต่ละเส้นทางเลือก นั้นมีความเหมาะสมหรือไม่ โครโมโซมทุกตัว

จะต้องมีค่าซึ่งบ่งบอกถึงความเหมาะสมที่จะพิจารณาว่าสมควรนำไปสืบสายพันธุ์หรือไม่สมควร ดังนั้นจึงต้องมีการให้ค่าความเหมาะสมกับแต่ละโครโมโซมเพื่อนำค่าความเหมาะสมไปพิจารณา โดยใช้สมการหาค่าความเหมาะสมที่สอดคล้องกับปัญหา สรุปได้ว่า ค่าความเหมาะสม คือตัวที่ใช้ ประเมินว่าแต่ละเส้นทางเลือก Solution นั้นมีความเหมาะสม หรือสามารถใช้แก้ปัญหาได้ดีเพียงใด ตัวอย่างของฟังก์ชันหาค่าความเหมาะสม เช่นค่าความเหมาะสม = จำนวนของ bit 1 ทั้งหมดใน โครโมโซม Fitness Function มี 2 รูปแบบคือ Binary Term Vectors และ Weighted Term Vectors

5.Parameter เป็นวิธีการที่ใช้ในการสร้างจำนวนโครโมโซมรุ่นถัดไป ถ้ากำหนดให้จำนวน โครโมโซมในแต่ละรุ่นมากจะทำให้ Genetic Algorithm ประมวลผลได้ช้าลง เช่น ขนาดของ ประชากร Population size ความน่าจะเป็นของ Crossover หรือ Probability Crossover ส่วนใหญ่มี ค่าอยู่ที่ 60% - 95% ความน่าจะเป็นของ Mutation หรือ Probability Mutation ส่วนใหญ่มีค่าอยู่ที่ 0%-1% และจำนวนโครโมโซมที่ใช้ในการสร้างรุ่นถัดไป ถ้ากำหนดให้จำนวนโครโมโซมในแต่ละ รุ่นมากจะทำให้ Genetic Algorithm ประมวลผลได้ช้าลง

Genetic Algorithm

จินเนติก อัลกอริทึม เป็นวิธีการทำให้มีประสิทธิภาพ Optimization ที่มีแนวคิดมาจากการวิวัฒนาการของสิ่งมีชีวิต โดยการนำค่าของตัวแปรต่างๆ ที่มีผลต่อค่าใช้จ่ายมาแปลงให้อยู่ในรูปแบบของโครโมโซม ซึ่งอาจจะแสดงในรูปแบบของ bit string หรือ real value string ก็ได้

ตัวอย่างเช่น กำหนดให้ตัวแปรที่มีผลต่อค่าใช้จ่ายมี 3 ตัวแปร คือ x, y, z โดยทั้งสามสามารถแปลงเป็นเลขจำนวนเต็ม Integer ในช่วง 0-127 ในกรณีที่ $x = 16, y = 77, z = 15$ จะสามารถแปลงเป็นโครโมโซม ในรูปแบบ bit string ได้โดย

$x = 16$ แปลงเป็น bit sting ได้ 0010000

$y = 77$ แปลงเป็น bit sting ได้ 1001101

$z = 15$ แปลงเป็น bit sting ได้ 0001111

นำ Binary จาก x, y , และ z มาต่อกันจะได้โครโมโซมดังนี้

Chromosome = $[x, y, z] = [001000010011010001111]$

ทั้งนี้ในการแปลงค่าจากปัจจัยที่นำมาคิดเป็น Binary string นั้นสามารถกำหนดวิธีการแปลงค่าได้เอง ไม่จำเป็นต้องแปลงตามค่าตัวเลขทางคณิตศาสตร์ numerical หรืออาจจะแปลงเป็นโครโมโซม ในรูปแบบ real value string จะได้ Chromosome = $[x, y, z] = [16|77|15]$

ในการดำเนินการของจินเนติก อัลกอริทึม จะเป็นการดำเนินการโดยมีโครโมโซมอยู่จำนวนหนึ่งซึ่งเรียกโครโมโซมกลุ่มนี้ว่า ประชากร

กระบวนการสืบค้นข้อมูล

แบ่งออกเป็น 2 ขั้นตอนคือ Genetic Algorithm และกระบวนการเปรียบเทียบรูปภาพ ดังนี้

1 .Genetic algorithm

-Selection เป็นอัลกอริทึมสำหรับเลือกรูปภาพที่มีค่าความเหมาะสมสูงที่สุด และตามจำนวนของ Generation ที่ได้มาจากผู้ใช้

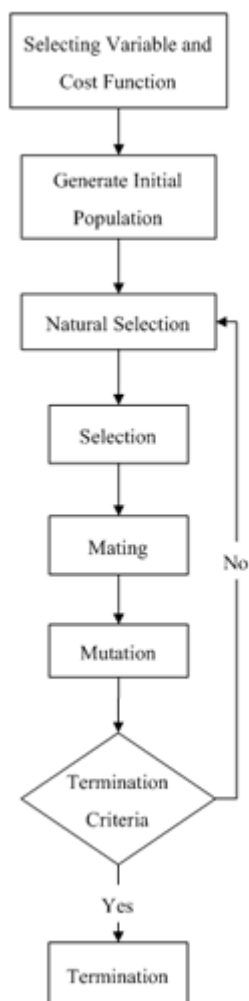
-Crossover เป็นอัลกอริทึม สำหรับเปลี่ยนแปลงข้อมูลระหว่าง 2 โครโมโซม ภายใต้เงื่อนไขของอัตราความน่าจะเป็นของการ Crossover และรูปแบบของการ Crossover ได้

ขั้นตอนการทำงาน Genetic Algorithm

ขั้นตอนการทำงานค้นหาคำตอบของ Genetic Algorithm อย่างง่าย มีดังนี้

- เริ่มทำการค้นหา ปัญหาที่เกิดขึ้น
- ถ้ายังไม่พบคำตอบ แต่ครบจำนวนรอบที่ได้กำหนดไว้ ก็จะหยุดทำการค้นหา
- ทำการค้นหาจนพบเป้าหมายหรือคำตอบที่ต้องการ ก็จะหยุดทำการค้นหา
- พบว่าคำตอบที่ได้เริ่มลู่เข้าสู่คำตอบที่เป็นคำตอบที่ดีที่สุด เช่น คำตอบที่ได้จากประชากรแต่ละรุ่น ไม่มีการเปลี่ยนแปลงหรือคงที่เป็นจำนวนที่ติดต่อกัน ถึงร้อยละ 95

จินเนติกอัลกอริทึม Genetic Algorithm มีขั้นตอนการทำงาน ดังรูป



แสดงขั้นตอนการทำงานของจินเนติก อัลกอริทึม

รายละเอียดของการทำงานในแต่ละขั้นมีดังต่อไปนี้

1. การกำหนดค่าตัวแปร

สมการค่าใช้จ่ายที่ใช้ในจินเนติก อัลกอริทึม Selection Variable and Cost Function กำหนดว่า ในโจทย์ที่ต้องการ Optimize นั้นมีปัจจัยอะไรที่มีผลต่อค่าใช้จ่ายและทำการสร้างฟังก์ชัน สำหรับคำนวณค่าค่าใช้จ่ายขึ้นมาเพื่อใช้ในขั้นต่อไป

2. สร้างประชากรต้นกำเนิด Generate Initial Population

ทำการสร้างประชากรชุดแรก เท่ากับจำนวนประชากรสูงสุดที่กำหนดไว้ ซึ่งอาจสร้างขึ้นมาโดยการสุ่ม หรือกำหนดขึ้นเอง

3. การคัดเลือกทางธรรมชาติ Natural Selection

เป็นการคัดเลือกโครโมโซม ที่มีค่าใช้จ่ายมากที่สุดออกตามอัตราส่วนที่กำหนดไว้ ทำให้เหลือโครโมโซม อยู่จำนวนหนึ่งสำหรับการเลือกคู่ Mating

4. การเลือกสรร Selection

ทำการจับคู่โครโมโซมที่เหลือเพื่อทำการเลือกคู่ Mating โดยใช้วิธีการจับคู่ที่กำหนดขึ้น ซึ่งวิธีการเลือกคู่โครโมโซมขึ้นมา ทำการเลือกคู่ mating มีหลายวิธี ดังต่อไปนี้

- จับคู่โครโมโซม ที่อยู่ติดกันจากบนลงล่าง
- จับคู่โดยการสุ่ม โดยความน่าจะเป็นที่โครโมโซม แต่ละตัวจะถูกสุ่มขึ้นมานั้นมีเท่ากัน
- จับคู่โดยการสุ่มแบบถ่วงน้ำหนัก วิธีนี้ความน่าจะเป็นที่โครโมโซม แต่ละตัวจะถูกสุ่มขึ้นมานั้นไม่เท่ากัน โดยวิธีการถ่วงน้ำหนักมี 2 วิธี คือ

1. ถ่วงน้ำหนักโดยดูจากอันดับ วิธีนี้จะคิดความน่าจะเป็นที่โครโมโซม แต่ละตัวจะถูกสุ่ม ขึ้นมาตามลำดับที่เรียงจากโครโมโซมที่มี Cost น้อยที่สุด ไปยังโครโมโซมที่มีค่าใช้จ่ายน้อยที่สุด โดยคำนวณความน่าจะเป็นของโครโมโซมแต่ละตัวจากสมการ

$$P_n = \frac{N_{keep} - n + 1}{\sum_{n=1}^{N_{keep}} n}$$

N_{keep} = จำนวนโครโมโซมที่เหลือจากขั้น Natural selection

n = อันดับของ โครโมโซม

2. ถ่วงน้ำหนักโดยดูจากค่าใช้จ่ายของโครโมโซม วิธีนี้จะคำนวณความน่าจะเป็นที่โครโมโซมแต่ละตัวจะถูกสุ่มขึ้นมา จากค่าใช้จ่ายของโครโมโซม ตัวนั้นๆ โดยค่าใช้จ่ายที่นำมาคำนวณนั้น ต้องทำการ Normalize ก่อน ด้วยสมการ

$$C_n = c_n - c_{N_{keep}+1}$$

$$c_n = \text{cost ของ โครโมโซม ตัวที่ } n$$

$c_{N_{keep}+1}$ = cost ของ โครโมโซม ที่มี cost ต่ำที่สุดที่ถูกคัดออกจากชั้น Natural Selection

จากนั้น คำนวณความน่าจะเป็นของ โครโมโซม แต่ละตัวด้วยสมการ

$$P_n = \left| \frac{C_n}{\sum_m^{N_{keep}} C_m} \right|$$

C_n = cost ของ โครโมโซม ตัวที่ n ที่ผ่านการ Normalize แล้ว

C_m = cost ของ โครโมโซม ตัวที่ m ที่ผ่านการ Normalize แล้ว

-การเลือกแบบ Tournament วิธีนี้จะทำการสุ่มโครโมโซม ขึ้นมาจำนวนหนึ่ง 2 ถึง 3 โครโมโซม ก่อน แล้วค่อยเลือกโครโมโซม ที่มีค่าใช้จ่ายน้อยที่สุดในกลุ่มออกมา

5.การจับคู่ Mating

เป็นการนำโครโมโซมคู่ที่ได้เลือกไว้จากชั้น Selection มาสร้างเป็นโครโมโซม ใหม่โดยการ ทำ Crossover ระหว่างโครโมโซม ทั้งสองซึ่งวิธีการในการทำ Crossover มีหลายวิธีดังต่อไปนี้

1. Single Crossover ทำการสุ่มตำแหน่ง crossover ขึ้นมาหนึ่งตำแหน่ง แล้วทำการแลกเปลี่ยน ยีน ที่อยู่ต่อจากตำแหน่ง Crossover ที่อยู่ติดต่อกับตำแหน่ง Crossover เพื่อสร้างเป็นโครโมโซม ใหม่ขึ้นมา 2 โครโมโซม

2. Multipoint crossover ทำการสุ่มตำแหน่ง Crossover ขึ้นมาจำนวนหนึ่งเรียงลำดับจากน้อย ไปหามาก แล้วทำการแลกเปลี่ยนยีน ที่อยู่ระหว่างตำแหน่ง Crossover ที่อยู่ติดกันเพื่อสร้างเป็น โครโมโซมใหม่ขึ้นมา 2 โครโมโซม

3. Uniform crossover mask ซึ่งเป็น bit string ซึ่งมีความยาวเท่ากับโครโมโซมขึ้นมา โดยที่ค่า ของแต่ละบิต Bit ได้มาจากการสุ่ม ซึ่งค่าของแต่ละบิต Bit นี้จะเป็นการกำหนดว่าโครโมโซมที่ สร้างขึ้นใหม่นั้น จะนำค่าของแต่ละยีนมาจากโครโมโซมตัวใด จากโครโมโซม คู่ที่เลือกมาจากชั้น Selection และสร้างอีกโครโมโซมหนึ่งในลักษณะเดียวกันโดยใช้ Inverse ของ Crossover mask ที่ ใช้ข้างต้น

4. Intermediate crossover ใช้สำหรับโครโมโซมที่เป็นแบบ real value string โดยที่ค่าของแต่ละ ยีนในโครโมโซมใหม่จะคำนวณจาก

$$O_1 = P_1 \times \alpha (P_2 - P_1)$$

โดย α เป็น Factor ที่ถูกสุ่มมาจากช่วงที่กำหนดขึ้นช่วงหนึ่ง ซึ่งจะทำการสุ่มใหม่ทุกครั้งที่เปลี่ยนคู่ โครโมโซมและ P_1, P_2 เป็นโครโมโซมจากชั้น Selection

5. Line crossover มีลักษณะคล้ายกับ Intermediate crossover แต่ค่า α ที่ใช้จะคงที่ตลอด

6.การกลายพันธุ์ Mutation

ทำการเปลี่ยนแปลงยีน โดยการสุ่มตำแหน่งของยีนที่จะเปลี่ยนแปลงขึ้นมาตามอัตราส่วนการเกิด Mutation ที่กำหนดไว้ โดยการเปลี่ยนแปลงคือการเปลี่ยนค่าของ Bit จาก 0 เป็น 1 หรือ จาก 1 เป็น 0 ในกรณีที่เป็นแบบ Bit string โดยจะยกเว้นไม่ให้เกิดการเปลี่ยนแปลงกับโครโมโซมที่มีค่าใช้จ่ายน้อยที่สุดในขณะนั้น และจะไม่มี Mutation ในการทำงานรอบสุดท้าย

7.ผลที่ได้เป็นไปตามเกณฑ์หรือไม่

จินเนติก อัลกอริทึม จะทำงานแบบ Iterative นับประชากรในแต่ละ Iteration เป็น Generation ซึ่งจะหยุดทำงานเมื่อคำตอบที่ได้มีค่าใช้จ่ายในระดับที่ต้องการ , ค่าใช้จ่ายที่ต่ำที่สุดในแต่ละรุ่น Generation มีค่าเท่ากัน หรือทำงานครบตามจำนวนรอบที่กำหนดไว้

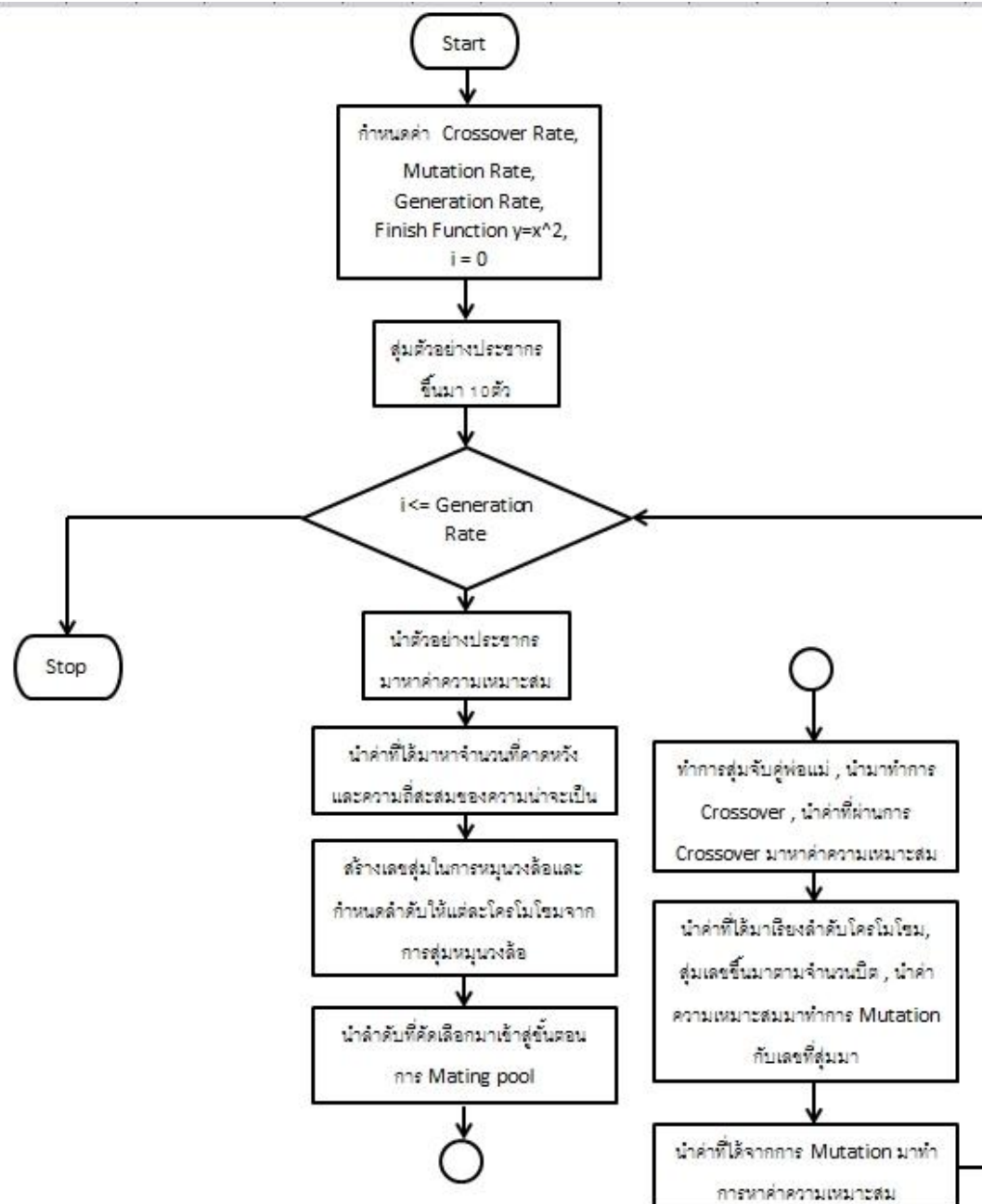
8.จบการทำงาน

เลือกโครโมโซม ที่มีค่าใช้จ่ายน้อยที่สุดเป็นคำตอบของปัญหา

บทสรุป

ปัจจุบันการวิเคราะห์ข้อมูลมีหลายวิธี ซึ่งแต่ละวิธีก็จะให้ผลลัพธ์การวิเคราะห์ที่แตกต่างกันไป ตามความสัมพันธ์ของสมการ ซึ่งถ้าหากกำหนดรูปแบบสมการไม่ถูกต้อง ผลลัพธ์ที่ได้อาจมีความคลาดเคลื่อนจากความเป็นจริงซึ่งส่งผลให้การวิเคราะห์ข้อมูลเกิดความผิดพลาดขึ้นได้ ดังนั้นจึงมีแนวคิดวิธีการของ จินเนติก อัลกอริทึม เพื่อเป็นทางเลือกในการแก้ปัญหาความผิดพลาดในการวิเคราะห์ข้อมูล โดยได้มีการนำกระบวนการจินเนติก อัลกอริทึม ซึ่งเป็นความรู้เกี่ยวกับทฤษฎีทางธรรมชาติมาประยุกต์ใช้ เพื่อหาคำตอบที่เหมาะสมและตรงกับความสัมพันธ์ของข้อมูล โดยการเริ่มสร้างประชากรต้นกำเนิดตามรูปแบบโครโมโซมที่ได้กำหนดไว้ เมื่อได้ประชากรต้นกำเนิดแล้วก็ทำการวัดค่าความเหมาะสมของแต่ละโครโมโซม โครโมโซมทุกตัวที่ได้กำหนดไว้จะถูกนำมาทำขั้นตอน Selection เพื่อคัดเลือกเข้าสู่กระบวนการ Genetic Operator โดยทำการคัดเลือกเฉพาะโครโมโซมที่มีค่าความเหมาะสม Fitness Function เพื่อที่จะได้โครโมโซมที่ยอมรับได้ตามที่กำหนดเอาไว้ โครโมโซมที่คัดเลือกไว้นั้นจะถูกนำมาทำการ Crossover การข้ามสายพันธุ์ และ Mutation การกลายพันธุ์ ได้เป็นโครโมโซมชุดใหม่ ซึ่งเราจะนำโครโมโซมชุดใหม่นี้มาดำเนินการต่อไปจนสิ้นสุดตามเงื่อนไขที่ได้กำหนดไว้ ก็จะได้โครโมโซมที่มีค่าความเหมาะสมที่ดีที่สุด เป็นคำตอบของปัญหา

flowchart



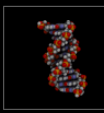
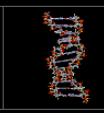
ผลการ RUN PROGRAM

Genetic Algorithm

	ลำดับ	โครโมโซม	x	ค่าความเหมาะสม	ค่าความน่าจะเป็น	จำนวนที่คัดทิ้ง	ความถี่สะสมค่าความน่าจะเป็น	สร้างเลขสุ่มในการหมุนวงล้อแต่ละครั้ง	ลำดับโครโมโซมที่ทุกเลือก
▶	1	00001	1	1	0.000	0.003	0.000	0.059	6
	2	11010	26	676	0.231	2.312	0.232	0.859	2
	3	01000	8	64	0.022	0.219	0.253	0.259	7
	4	10111	23	529	0.181	1.809	0.434	0.742	3
	5	11010	26	676	0.231	2.312	0.666	0.857	3
	6	11000	24	576	0.197	1.970	0.863	0.786	5
	7	01101	13	169	0.058	0.578	0.920	0.445	9
	8	01100	12	144	0.049	0.492	0.970	0.413	6
	9	01000	8	64	0.022	0.219	0.991	0.283	3
	10	00101	5	25	0.009	0.085	1.000	0.163	9

Generate

Resume

Crossover Rate Generation

Mutation Rate Finish Function

ค่า X สูงสุด

	ลำดับที่คัดเลือก	matting_pos	สุ่มจับคู่พอ-ไม่	เลขสุ่ม(r)	ก่อนการสุ่มวงล้อ	สุ่มตำแหน่ง(pos)	หลังการสุ่มวงล้อ	x	ค่าความเหมาะสม	ลำดับโครโมโซม
▶	6	11000	3.6	0.906		ไม่คงที่วงล้อ		24	576	1
	2	11010		> 0.5				26	676	2
	7	01101	3.6	0.377	01101	2	01000	8	64	3
	3	01000		<= 0.5	01000		01101	13	169	4
	3	01000	7.5	0.839		ไม่คงที่วงล้อ		8	64	5
	5	11010		> 0.5				26	676	6
	9	01000	5.7	0.351	01000	2	01000	8	64	7
	6	11000		<= 0.5	11000		11000	24	576	8
	3	01000	5.2	0.971		ไม่คงที่วงล้อ		8	64	9
	9	01000		> 0.5				8	64	10

	ลำดับ	ก่อนการสุ่มพันธุ	เลขสุ่ม (r)	หลังการสุ่มพันธุ	x
▶	1	11000	0.095, 0.043, 0.106, 0.054, 0.177	00010	2
	2	11010	0.192, 0.057, 0.200, 0.156, 0.097	10011	1
	3	01101	0.019, 0.106, 0.079, 0.113, 0.069	11000	2
	4	01000	0.121, 0.078, 0.083, 0.039, 0.153	00110	6
	5	01000	0.146, 0.094, 0.032, 0.005, 0.001	00111	7
	6	11010	0.136, 0.093, 0.090, 0.018, 0.080	10101	2
	7	01000	0.049, 0.150, 0.028, 0.030, 0.103	11110	3
	8	11000	0.078, 0.008, 0.008, 0.003, 0.020	00111	7
	9	01000	0.125, 0.085, 0.010, 0.178, 0.076	00101	5
	10	01000	0.161, 0.033, 0.055, 0.057, 0.141	00110	6

Developed by LS Studios @ RMUTL

CODE PROGRAM

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Collections;

using System.Threading;

namespace Genetic1
{
    public partial class Form1 : Form
    {
        ////////////variable setting//////////

        Random rn = new Random();

        Random radius = new Random();

        static int limit_chrome=10;

        int set_n = 5,bin_n=31; //จำนวน โครโมโซม

        double sum_f = 0, sumpsel = 0;

        int outloop = 0;

        String[] chromosom = new String[limit_chrome];

        int[] x = new int[limit_chrome];

        int[] f = new int[limit_chrome];

        double[] pselect13 = new double[limit_chrome];

        double[] ei = new double[limit_chrome];

        double[] sumpselect13 = new double[limit_chrome];

        int[] qi = new int[limit_chrome];
    }
}

```

```

int[] a = new int[limit_chrome];
int[] cho=new int[limit_chrome];
int b = 0;
double cor_rate = 0.0;
double mtr_rate = 0;
int tog = 0;
int max_a = 0;
ArrayList ar = new ArrayList();
// int loop = 0;
//////////variable setting//////////
public Form1()
{
    InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
    timer1.Enabled = false;
    //gen();
    //MessageBox.Show(cor.Text);
    //cor_rate = Convert.ToDouble(cor.Text + "");
}

public void gen() {

    if (b == 0)
    {
        for (int i = 0; i < limit_chrome; i++)
        {
            cho[i] = (rn.Next(0, bin_n));

```

```

    }
}

```

```

b++;
// for (int b = 0; b < 20; b++)
// {
sum_f = 0;
for (int i = 0; i < limit_chrome; i++)
{

```

```

    String txt_chrom = Convert.ToString(Convert.ToInt32(cho[i].ToString(), 10), 2);

```

```

    txt_chrom = txt_chrom.PadLeft(set_n, '0');
    x[i] = cho[i]; chromosom[i] = txt_chrom;

```

```

//=====ค่า

```

ความเหมาะสม

```

f[i] = cho[i] * cho[i];
sum_f += f[i];

```

```

}

```

```

for (int i = 0; i < limit_chrome; i++)
{

```

```

//=====ค่าความ

```

น่าจะเป็น

```
pselect13[i] = f[i] / sum_f;
String showpselect13 = (f[i] / sum_f).ToString("0.000");
```

```
//=====ค่าความ
```

คาดหวัง

```
ei[i] = pselect13[i] * limit_chrome;
String showE = (pselect13[i] * limit_chrome).ToString("0.000");
```

```
//=====ความถี่
```

สะสม

```
sumpsel += pselect13[i];
sumpselect13[i] = sumpsel;
String showSumpselect13 = sumpsel.ToString("0.000");
```

```
//ค่า r
```

```
//===== เอาค่าลง grid
```

```
grid1.Rows.Add(i + 1, chromosom[i], x[i], f[i], showpselect13, showE,
showSumpselect13);
}
```

```
//////////เอาค่า ที่สุ่มในวงล้อ กับ qi ใส่
```

```
for (int i = 0; i < limit_chrome; i++)
{
    for (int k = 0; k < limit_chrome - 1; k++)
    {
        double ran_r = radius.NextDouble();
```



```

// MessageBox.Show(ran_r+"");
if (ran_r < sumpselect13[k])
{
    qi[i] = k;
    String showRanr = ran_r.ToString("0.000");
    grid1.Rows[i].Cells[7].Value = showRanr;
    grid1.Rows[i].Cells[8].Value = qi[i] + 1;
    break;
}
else {
    qi[i] = rn.Next(0,limit_chrome-1);
    String showRanr = ran_r.ToString("0.000");
    grid1.Rows[i].Cells[7].Value = showRanr;
    grid1.Rows[i].Cells[8].Value = qi[i] + 1;
    break;
}
}
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Random rn3 = new Random();
String[] matting = new String[limit_chrome];
String[] after_cross = new String[limit_chrome];
String[] n_x = new String[limit_chrome];
int[] n_n = new int[limit_chrome];
int[] ran_parent = new int[limit_chrome];
double[] ran_3 = new double[limit_chrome];
int[] pos = new int[limit_chrome];

```

////////////////////ตาราง2

```
for (int i = 0; i < limit_chrome; i++)
```

```
{
```

```
    n_n[i] = qi[i];
```

```
    matting[i] = chromosom[n_n[i]];

```

```
    grid2.Rows.Add((n_n[i] + 1) + "", matting[i]);

```

```
}
```

```
for (int i = 0; i < limit_chrome; i = i + 2)
```

```
{
```

```
    after_cross[i] = matting[i];
```

```
    if (i % 2 == 0)
```

```
    {
```

```
        String condi_r = "";
```

```
        int tmpran1 = Convert.ToInt32(grid2.Rows[rn3.Next(0, 9)].Cells[0].Value + "");
```

```
        int tmpran2 = Convert.ToInt32(grid2.Rows[rn3.Next(0, 9)].Cells[0].Value + "");
```

```
        ran_parent[i] = tmpran1;
```

```
        ran_parent[i + 1] = tmpran2;
```

```
        grid2.Rows[i].Cells[2].Value = tmpran1 + "," + tmpran2;
```

```
        ran_3[i] = rn3.NextDouble();
```

```
        if (ran_3[i] <= cor_rate)
```

```
        {
```

```
            condi_r = "<= " + cor.Text;
```

```
            pos[i] = rn3.Next(1, set_n);
```

```

        after_cross[i] = matting[i].ToString().Substring(0, pos[i]) + matting[i +
1].ToString().Substring(pos[i], set_n - pos[i]);

        after_cross[i + 1] = matting[i + 1].ToString().Substring(0, pos[i]) +
matting[i].ToString().Substring(pos[i], set_n - pos[i]);
    }
    else
    {
        condi_r = ">" + cor.Text;
        pos[i] = 0;
        after_cross[i] = matting[i];
        after_cross[i + 1] = matting[i + 1];
    }
    grid2.Rows[i].Cells[3].Value = ran_3[i].ToString("0.000");
    grid2.Rows[i + 1].Cells[3].Value = condi_r;

    if (pos[i] == 0)
    {
        grid2.Rows[i].Cells[5].Value = "ไม่ครอบคลุมเวอร์";
    }
    else
    {
        grid2.Rows[i].Cells[4].Value = matting[i];
        grid2.Rows[i + 1].Cells[4].Value = matting[i + 1];
        grid2.Rows[i].Cells[5].Value = pos[i] + "";
        grid2.Rows[i].Cells[6].Value = after_cross[i] + "";
        grid2.Rows[i + 1].Cells[6].Value = after_cross[i + 1] + "";

    }

}

n_x[i] = Convert.ToString(Convert.ToInt32(after_cross[i], 2), 10);

```

```

        grid2.Rows[i].Cells[7].Value = n_x[i] + "";
        n_x[i + 1] = Convert.ToString(Convert.ToInt32(after_cross[i + 1], 2), 10);
        grid2.Rows[i + 1].Cells[7].Value = n_x[i + 1] + "";
    }

    for (int i = 0; i < limit_chrome; i++)
    {
        grid2.Rows[i].Cells[8].Value = (Convert.ToInt32(n_x[i]) * (Convert.ToInt32(n_x[i])))
+ "";
        grid2.Rows[i].Cells[9].Value = i + 1;

    }

    double[] tmpr3 = new double[set_n];

    for (int i = 0; i < limit_chrome; i++)
    {

        String showTmpr = "";
        for (int j = 0; j < set_n; j++)
        {
            tmpr3[j] = rn3.NextDouble() / 5;

            if (j != 0)
            {
                showTmpr += " , ";
            }

            showTmpr += tmpr3[j].ToString("0.000");
        }

        grid3.Rows.Add(i + 1 + "", matting[i], showTmpr);
    }

```

```

for (int i = 0; i < limit_chrome; i++)
{
    String tt = grid3.Rows[i].Cells[2].Value.ToString();
    String[] tt2 = tt.Split(',');

    char[] arc = matting[i].ToCharArray();
    String tmpstr = "";
    for (int j = 0; j < set_n; j++)
    {
        double tpr3 = Convert.ToDouble(tt2[j]);

        if (tpr3 < mtr_rate)
        {
            if (arc[j] == '0') { arc[j] = '1'; } else if (arc[j] == '1') { arc[j] = '0'; }
        }

        tmpstr += arc[j];
    }
    matting[i] = tmpstr;
    grid3.Rows[i].Cells[3].Value = matting[i];
    grid3.Rows[i].Cells[4].Value = Convert.ToString(Convert.ToInt32(matting[i], 2), 10);
    a[i] = Convert.ToInt32(Convert.ToString(Convert.ToInt32(matting[i], 2), 10));
    grid3.Rows[i].Cells[5].Value = a[i] * a[i];
    max_a = (a[i] > max_a) ? a[i] : max_a;
    textBo.Text = max_a + "";
    tmpstr = "";

    //ar.Add(a[i]);

```

```

        // ar.Sort();
        //textBo.Text += "," + ar[i];
    }

    //}

    // textBo.Text = "," + ar[limit_chrome-1];

}

private void close_Click(object sender, EventArgs e)
{
    this.Close();
}

private void button1_Click(object sender, EventArgs e)
{
    timer1.Enabled = false;
    cor_rate = Convert.ToDouble(cor.Text);
    mtr_rate = Convert.ToDouble(mtr.Text);
    outloop = 0;
    timer1.Enabled = true;
}

private void timer1_Tick(object sender, EventArgs e)
{
    ArrayList ar = new ArrayList();
    if (outloop == Convert.ToInt32(loopn.Text)) { timer1.Enabled = false; }
}

```

```

textBox5.Text = outloop.ToString();
int chk=0;
int chking=0;
for (int jj = 0; jj < limit_chrome; jj++) {
if(chk!=0){
    if (a[jj] != x[jj])
    {
        chking = 0;
        break;
    }
    else {
        chking = 1;
    }

}
}
if (chking == 1) {
    timer1.Enabled = false;
}

grid1.Rows.Clear();
grid2.Rows.Clear();
grid3.Rows.Clear();
sumpsel = 0;
for (int i = 0; i < limit_chrome; i++) {
    chromosom[i] = "";
x[i] =0;
f[i] = 0;
pselect13[i] =0;
}

for (int i = 0; i < limit_chrome; i++)
{

```

```
        cho[i] = a[i];
    }
    gen();
    chk++;
    outloop++;
}

private void button2_Click(object sender, EventArgs e)
{
    if (tog == 0)
    {
        button2.Text = "Resume";
        tog = 1;
        timer1.Stop();
    }
    else {
        button2.Text = "Pause";
        tog = 0;
        timer1.Start();
    }
}
}
```


เอกสารอ้างอิง

- [1] <http://cs.felk.cvut.cz/~xobitko/ga/>
- [2] http://www.wordiq.com/definition/Genetic_algorithm#References
- [3] http://en.wikipedia.org/wiki/Genetic_algorithm
- [4] <http://garbo.uwasa.fi/pub/cs/report96-1/SCAI06.pdf>