กรรมวิธีการสร้าง

โปรแกรมคอมพิวเตอร์อย่างอัตโนมัติที่อาศัยการเลียนแบบการถ่ายทอดทางพันธุกรรมของสิ่งมีชีวิตตามธรรมชาติ (Genetic Programming)

ร.อ.ดร.วิทยา ปัญญาวรญาณ
อาจารย์ฝ่ายศึกษา โรงเรียนนายเรือ

Genetic Programming เป็นรูปแบบหนึ่งของเทคนิคแบบ Evolutionary Computation โดยมีจุดประสงค์ ให้คอมพิวเตอร์สามารถสร้างโปรแกรมคอมพิวเตอร์ที่ต้องการได้ด้วยตนเอง Genetic Programming ทำงานโดยเลียนแบบวิวัฒนาการของสิ่งมีชีวิต ที่อาศัยกลไกการคัดเลือกโดยธรรมชาติตามทฤษฎีของ C. Darwin และการถ่ายทอดทางพันธุกรรมด้วยวิธีต่าง ๆ เช่น Reproduction, Crossover และ Mutation เป็นต้น ในบทความนี้เป็นการอธิบายพื้นฐานหลักการทำงานของ Genetic Programming และแสดงตัวอย่างประกอบการสร้างโปรแกรมคอมพิวเตอร์ที่สามารถจำลองการทำงานของ ๔ : ๑ Multiplex

๑ บทนำ

ในปัจจุบันได้มีการพัฒนากรรมวิธีต่าง ๆ ที่เลียนแบบธรรมชาติขึ้นมาเพื่อใช้ในการแก้ปัญหาซับซ้อน กรรมวิธีเลียนแบบธรรมชาติที่สำคัญอย่างหนึ่งคือ Evolutionary Computation หรือ EC ใน [๑, ๒, ๓]

EC จะใช้ทฤษฎีทางชีววิทยาที่อธิบายถึงวิวัฒนาการของสิ่งมีชีวิตที่อาศัยกลไกการคัดเลือกโดยธรรมชาติและการถ่ายทอดทางพันธุกรรมเป็นหลักและแนวทางในการทำงาน รูปแบบหลัก ๆ ของ EC ในปัจจุบันมี ๔ ประเภทคือ Genetic Algorithms (GA), Evolution Strategies (ES), Evolutionnary Programming (EP) และ Genetic Programming (GP)

สามรูปแบบแรกของ EC ถูกพัฒนาอย่างอิสระต่อกัน โดยที่ GA และ EP ถูกพัฒนาขึ้นในสหรัฐอเมริกา GA เหมาะสำหรับใช้ในการหาค่าคงที่ ๆ เหมาะสมของฟังก์ชันทางคณิตศาสตร์หรือที่เรียกกันว่า Optimization EP ใช้ในการทดลองสร้างสมองสังเคราะห์ (Artificial Intelligence) ES ถูกพัฒนาขึ้นในสหพันธ์สาธารณรัฐเยอรมันและใช้ในการหาค่าคงที่ ๆ เหมาะสมของฟังก์ชันทางคณิตศาสตร์เหมือนกับ GA ส่วน GP นั้นพัฒนาต่อมาจาก GA และจะใช้ในการสร้างโปรแกรมคอมพิวเตอร์หรือฟังก์ชันทางคณิตศาสตร์

ในบทความนี้จะขออธิบายหลักการพื้นฐานของ {GP} เท่านั้น ซึ่งสามารถแบ่งออกเป็นหัวข้อย่อย ๆ ได้ดังต่อไปนี้ หัวข้อที่ ๒ จะขอกล่าวถึงทฤษฎีที่สามารถอธิบายการวิวัฒนาการของสิ่งมีชีวิตและประวัติความเป็นมาของ EC หัวข้อที่ ๓ เป็นการอธิบายการทำงานของ GP หัวข้อที่ ๔ แสดงตัวอย่าง

การทำงานของ GP หัวข้อที่ ๕ เป็นการวิเคราะห์การทำงานของ GP และ หัวข้อที่ ๖ คือการสรุป

๒ บทเรียนจากธรรมชาติ

เมื่อปี พ.ศ. ๒๔๐๒ {C. Darwin} ได้อธิบายทฤษฎีการคัดเลือกโดยธรรมชาติไว้ใน On the Origin of Species by Means of Natural Selection เขาได้กล่าวไว้ว่า

- สิ่งมีชีวิตต่าง ๆ จะให้กำเนิดลูกเท่าที่จำเป็นเพื่อให้เผ่าพันธุ์หรือชนิดของตนเองดำรงอยู่ได้
- ลูกที่เกิดมาจะมีลักษณะและความสามารถแตกต่างจากพ่อหรือแม่ไปบ้าง
- สิ่งมีชีวิตชนิดใดที่มีความสามารถในการหาอาหาร การปรับตัวให้เข้ากับสิ่งแวดล้อม และขยายพันธุ์ได้ดีจะสามารถดำรงชีวิตอยู่รอด และจะผลักดันหรือทำลายสิ่งมีชีวิตประเภทอื่นๆ

ด้วยเหตุผลต่าง ๆ เหล่านี้ จะเกิดการคัดเลือกสิ่งมีชีวิตที่เหมาะสมกับสภาพแวดล้อมมากที่สุด และการคัดเลือกเช่นนี้จะเกิดกับสิ่งมีชีวิต อย่างต่อเนื่องจนทำให้เกิดการวิวัฒนาการ

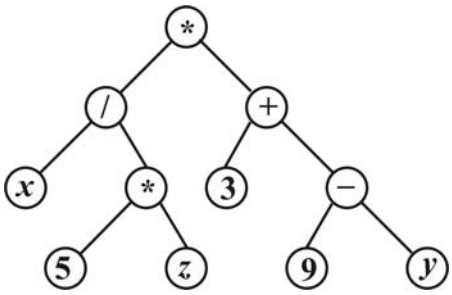
EC จำลองการวิวัฒนาการของสิ่งมีชีวิตแบบง่าย ๆ โดย EC ทำงานกับประชากร (Population) ที่ประกอบด้วย Individual หลาย ๆ ตัว Individual เหล่านี้ก็คือตัวอย่างคำตอบที่น่าจะเป็นไปได้ที่เราต้องการนั้นเอง ซึ่งเทียบได้กับสิ่งมีชีวิตตัวหนึ่งตามทฤษฎีการคัดเลือกโดยธรรมชาติ Individual แต่ละตัวจะมีค่า Fitness กำหนดอยู่ ค่า Fitness นี้ก็คือค่าที่ใช้วัดความสามารถของ Individual ว่าเหมาะสมกับการแก้ปัญหามากน้อยเท่าไร

มีการพัฒนา EC มาตั้งแต่ราวปี พ.ศ. ๒๔๙๐ แต่เนื่องจากขาดอุปกรณ์ในการทำงานที่มีความเร็วเพียงพอ จึงทำให้ EC ไม่ได้รับการพัฒนาเท่าที่ควร ประมาณปี พ.ศ. ๒๕๐๐ ได้มีผลงานจากนักวิจัยหลายท่านเช่น Holland [4], Rechenberg [5], Fogel [6] และ Schwefel [7] ที่ช่วยผลักดันให้ EC มีการพัฒนาอย่างรวดเร็วจนถึงปัจจุบัน

ข้อดีที่สำคัญของ EC คือ มีลักษณะการแก้ปัญหาแบบ Global Search คือมีโอกาสหาคำตอบที่แท้จริงที่เหมาะสมกับทุก ๆ เงื่อนไขหรือขอบเขตได้ สามารถปรับให้เหมาะสมกับลักษณะของงานประเภทต่าง ๆ ได้โดยการปรับแต่งค่า Parameter ที่ควบคุมการทำงาน และสามารถใช้ร่วมกับกรรมวิธีการแก้ปัญหาประเภทอื่น ๆ ได้ เป็นต้น

๓ Genetic Programming

J. R. Koza [8, 9, 10] ได้พัฒนา GP ขึ้นมาโดยมีจุดประสงค์ที่ต้องการให้คอมพิวเตอร์สามารถเขียนหรือสร้างโปรแกรมคอมพิวเตอร์ด้วยตัวคอมพิวเตอร์เอง ในระหว่างที่เราทำงานกับ GP จะมีการสร้างโปรแกรมคอมพิวเตอร์ขึ้นมามากมายและโปรแกรมคอมพิวเตอร์ที่สร้างขึ้นมานี้จะถูกเรียกว่า Individual ซึ่งจะเขียนย่อในที่นี้ว่า **Ind**



รูปที่ ๑ : ตัวอย่าง Individaul : $\frac{x}{(5z)} \cdot (3 + (9 - y))$

Ind ใน GP ส่วนใหญ่มีลักษณะโครงสร้างคล้ายกับต้นไม้ (Expression Tree) ดังที่แสดงในรูปที่ ๑ **Ind** แต่ละตัวจะมีค่า Fitness กำหนดอยู่ ค่า Fitness เป็นค่าใช้แสดงความสามารถของ **Ind** ว่าเหมาะสมกับการแก้ปัญหามากน้อยเท่าไร กระบวนการทำงานของ GP สามารถอธิบายได้ดังต่อไปนี้

๑. เริ่มแรกหรือที่เรียกกันว่า Generation ที่ศูนย์ (G = 0) จะมีการสร้างโปรแกรมคอมพิวเตอร์หรือ **Ind** แบบสุ่มหรืออย่างไม่เจาะจงจาก Terminal Set (T) และ Function Set (F) ตามจำนวน **Ind** ต่อหนึ่ง Generation ซึ่งเราเรียกกันว่า Population Size ที่ต้องกำหนดไว้ตอนแรก

๒. หลังจากนั้นเราจะหาค่า Fitness หรือคะแนนความเหมาะสมและความสามารถในการแก้ปัญหาของ **Ind** แต่ละตัว

๓. ตรวจสอบเงื่อนไขในการหยุดทำงานของ GP ซึ่งโดยปกติมี ๒ อย่างคือ

(ก) GP ได้สร้าง **Ind** ที่ต้องการ ซึ่งหมายถึงได้ **Ind** ที่มีค่า Fitness ตามที่กำหนด

(ข) ครบจำนวนรอบการทำงานของ GP หรือที่เราเรียกว่า Generation ตามที่กำหนดไว้

๔. ถ้าเงื่อนไขหนึ่งในสองอย่างของการหยุดทำงานของ GP ถูกต้อง GP จะหยุดทำงานและเราจะนำ **Ind** ที่มีค่า Fitness ดีที่สุดใช้เป็นผลลัพธ์ของ GP

๕. ถ้าเงื่อนไขในการหยุดทำงานของ GP ไม่ถูกต้อง GP จะต้องทำงานต่อไป

๖. GP จะเลือกวิธีการสร้าง **Indลูก** ซึ่งก็คือ **Ind** ในรุ่นหรือ Generation ต่อไป จากสองวิธีคือ Crossover และ Reproduction

๗. หลังจากนั้นจะมีการคัดเลือก **Ind** หนึ่งหรือสองตัวให้เป็น **Indผู้ให้กำเนิด** ที่ถูกใช้ในการสร้าง **Indลูก** GP จะใช้ค่า Fitness ของ **Ind** เป็นหลักในการคัดเลือก ซึ่งหมายความว่า **Ind** ที่มีค่า Fitness ดีกว่าจะมีโอกาสถูกเลือกมากกว่า **Ind** ที่มีค่า Fitness ด้อยกว่า

๘. ต่อจากนั้น GP จะสร้าง **Indลูก** ตามวิธีที่ได้เลือกไว้ในข้อ ๖

(ก) Reproduction คือการสร้าง **Indลูก** หนึ่งตัวที่เหมือนกับ Ind ผู้ให้กำเนิด ทุกอย่าง

(ข) Crossover คือการสร้าง **Indลูก** สองตัว ด้วยการแลกเปลี่ยนโครงสร้างของ **Indผู้ให้กำเนิด** ๒ ตัว

๙. หลังจากที่ได้สร้าง **Indลูก** **Indลูก** อาจจะถูกเปลี่ยนแปลงโครงสร้างบางส่วนโดยวิธี Mutation

๑๐. เมื่อสร้าง **Indลูก** ใน Generation ที่ศูนย์ครบตามจำนวนที่ต้องการ **Indลูก** ที่ถูกสร้างนี้จะถูกจัดให้เป็น **Ind** ใน Generation ที่หนึ่ง $(G = 1)$ จำนวน **Ind** ในแต่ละ Generation หรือที่เราเรียกกันว่า Population size จะต้องคงที่เสมอ

๑๑. หลังจากนั้นจะมีการหาค่า Fitness ของ **Ind** ใน Generation ที่หนึ่ง ต่อไป

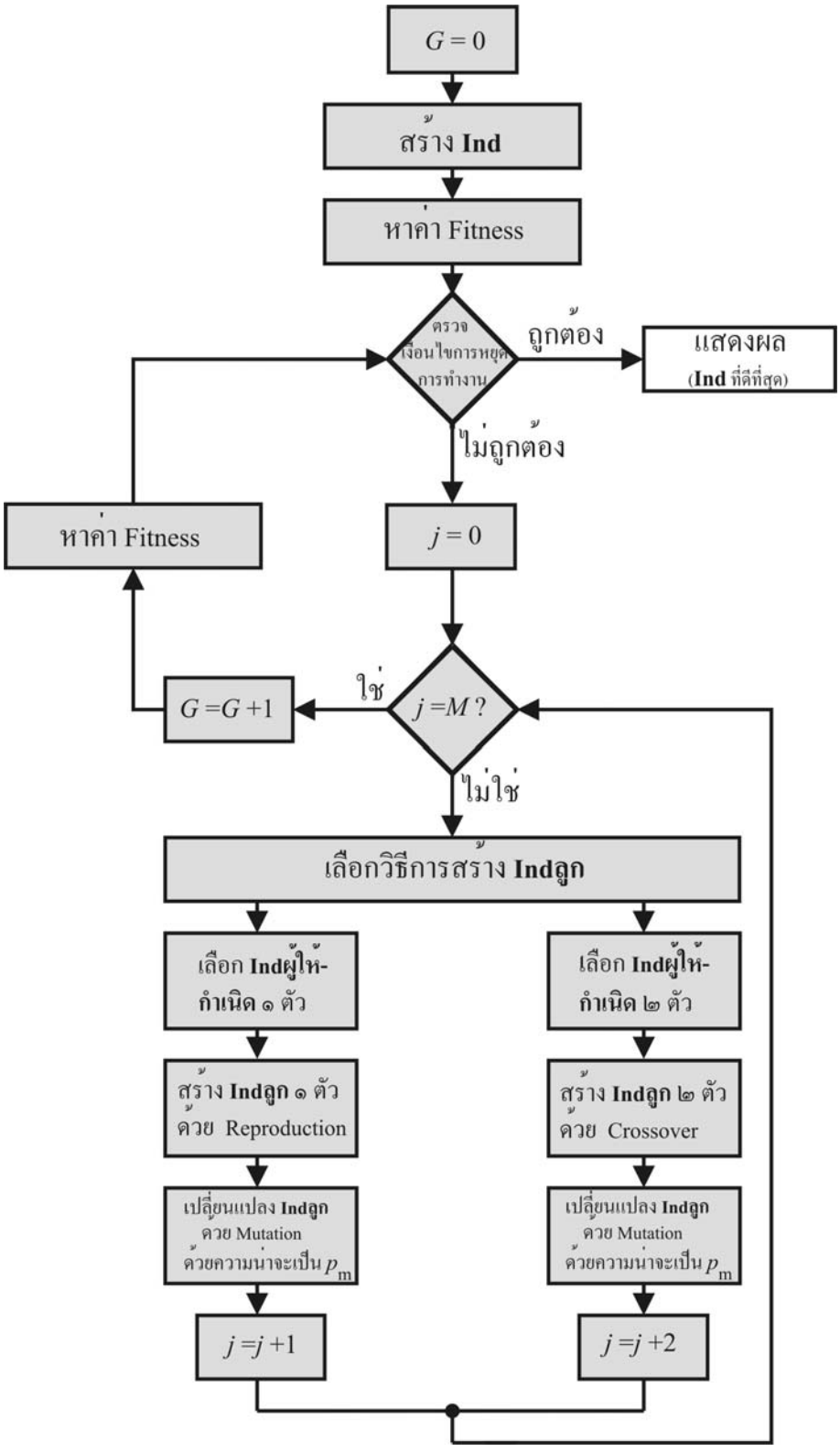
๑๒. GP จะทำเช่นนี้ต่อไปเรื่อย ๆ จนกว่าเงื่อนไขในการหยุดทำงานของ GP จะถูกต้อง

รูปที่ ๒ ได้สรุปกระบวนการทำงานของ GP ให้เราได้เห็นภาพอีกครั้ง ต่อไปนี้เป็นการอธิบายส่วนที่สำคัญของ GP

Indผู้ให้กำเนิด คือ **Ind** ที่ถูกเลือกมาเพื่อใช้สร้าง **Indลูก**

Indลูก คือ **Ind** ที่ถูกสร้างจาก **Indผู้ให้กำเนิด** และจะทำหน้าที่เป็น **Ind** ใน generation ต่อไป

Generation คือ จำนวนรอบในการทำงานของ GP



รูปที่ ๒ : สรุปหลักการทำงานของ Genetic Programming โดยที่ G คือ หมายเลขของ Generation, M คือ จำนวน **Ind** ในหนึ่ง Generation หรือ Population Size, j คือ จำนวน **Indลูก** ที่ถูกสร้างขึ้น

Function Set F สมาชิกใน F คือฟังก์ชันต่าง ๆ ที่เป็นส่วนประกอบของ **Ind** เช่น

- ฟังก์ชันพื้นฐานทางคณิตศาสตร์ เช่น sin, cos, $+, -, \times, \div$
- ฟังก์ชันลอจิก เช่น not, or, and
- ฟังก์ชันแสดงเงื่อนไข เช่น if-then-else
- ฟังก์ชันการทำงานซ้ำ เช่น for, do-until เป็นต้น

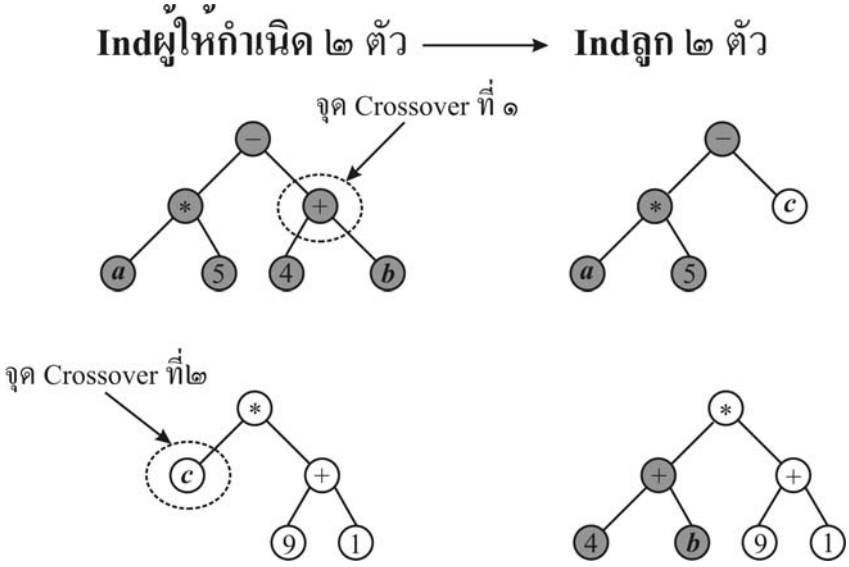
Terminal Set (T) สมาชิกใน T ประกอบด้วย ค่าคงที่และตัวแปรต่าง ๆ ที่จะเป็น Input ให้กับ F

การสร้าง Individual ในครั้งแรก สำหรับ **Ind** ใน Generation ที่ศูนย์ $(G = 0)$ จะถูกสร้างแบบไม่เจาะจงจากสมาชิกใน Function Set และ Terminal Set

Fitness คือค่าที่ใช้วัดความสามารถของ **Ind** ว่าเหมาะสมกับการแก้ปัญหามากน้อยเท่าไร ค่า Fitness ขึ้นอยู่กับลักษณะของปัญหาที่ต้องการแก้หรือคำตอบที่ต้องการ เช่น ถ้าเราต้องการหาฟังก์ชันทางคณิตศาสตร์เพื่อใช้ในการทำนาย ค่า Fitness จะคำนวณได้จากค่าผิดพลาดในการทำนายหรือผลต่างระหว่างค่าจริงและค่าทำนาย ดังนั้น **Ind** ที่มีค่าผิดพลาดในการทำนายน้อยจะมีค่า Fitness ที่ดีกว่า

การคัดเลือก (Selection) คือการคัดเลือก **Indผู้ให้กำเนิด** ที่จะเป็นผู้ให้กำเนิด **Indลูก** การคัดเลือกจะใช้ค่า Fitness เป็นหลักในการพิจารณา สำหรับ GP ส่วนใหญ่จะใช้การคัดเลือกแบบ Tournament Selection ซึ่งมีกรรมวิธีดังนี้ ขั้นแรกจะเลือกค่าคงที่ในการเลือกมาค่าหนึ่งซึ่งเราเรียกว่า Tournament Size เช่น ๑๐ เป็นต้น หลังจากนั้นจะเลือก **Ind** แบบไม่เจาะจงมา ๑๐ ตัว เฉพาะ **Ind** ที่มีค่า Fitness ดีที่สุด จะถูกเลือกให้เป็น **Indผู้ให้กำเนิด**

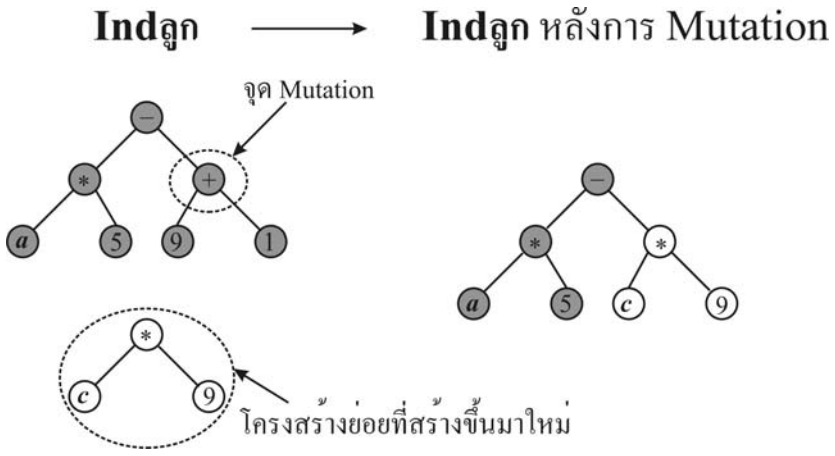
Crossover เป็นการสร้าง **Indลูก** โดยเลียนแบบการสืบพันธุ์ตามธรรมชาติ การสร้าง **Indลูก** ด้วยวิธี Crossover จะมีการแลกเปลี่ยนโครงสร้างระหว่าง **Indผู้ให้กำเนิด** สองตัวอย่างไม่เจาะจง ดังที่แสดงในรูปที่ ๓



รูปที่ ๓ : ตัวอย่างการ Crossover ระหว่าง **Indผู้ให้กำเนิด** สองตัว

ตอนแรกจะมีการเลือก **Indผู้ให้กำเนิด** สองตัวและจะมีการเลือกจุด Crossover ใน **Ind ผู้ให้กำเนิด** ทั้งสองอย่างสุ่มหรือไม่เจาะจง หลังจากนั้นจะทำการแลกเปลี่ยนโครงสร้างย่อยระหว่าง **Ind ผู้ให้กำเนิด** ที่จุด Crossover ที่เลือกไว้ ซึ่งจะทำให้เกิด **Indลูก** สองตัว ดังนั้น **Indลูก** แต่ละตัวที่ได้จะมีโครงสร้างจาก **Indผู้ให้กำเนิด** สองตัว วิธีนี้คล้ายกับการถ่ายทอดทางพันธุกรรมจากพ่อและแม่ไปสู่ลูก

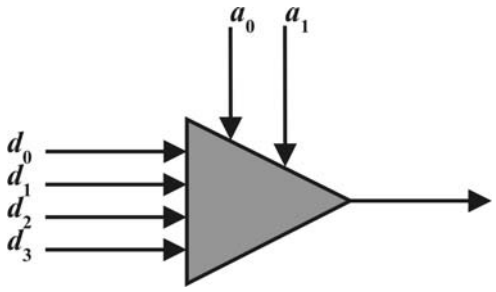
Mutation เป็นการเปลี่ยนโครงสร้างของ **Indลูก** โดยมีการเปลี่ยนโครงสร้างบางส่วนแบบสุ่มหรือไม่เจาะจง เป็นการทำให้ลูกมีความแตกต่างจากพ่อและแม่ดังที่แสดงในรูปที่ ๔ วิธีนี้คล้ายกับกับการกลายพันธุ์ของสิ่งมีชีวิตตามธรรมชาติ



รูปที่ ๔ : ตัวอย่างการ Mutation

๔ ตัวอย่างการทำงานของ Genetic Programming}

ในหัวข้อนี้จะเป็นการแสดงตัวอย่างการทำงานของ GP ในการสร้างโปรแกรมเพื่อแก้ปัญหากล่าวคือ เราต้องหาโปรแกรมที่สามารถจำลองการทำงานของ 4 : 1 Multiplex ดังรูปที่ ๔



รูปที่ ๕ : 4 : 1 Multiplex

4 : 1 Multiplex จะประกอบด้วย สัญญาณเข้า ๔ สัญญาณ คือ d_0, d_1, d_2, d_3 และ สัญญาณควบคุม 2 สัญญาณ คือ a_0, a_1 ที่ทำหน้าที่ควบคุมสัญญาณออกจาก 4 : 1 Multiplex ให้เป็นตามตารางที่ ๑

สัญญาณควบคุม a_1	สัญญาณควบคุม a_0	สัญญาณออก
0	0	d_0
0	1	d_1
1	0	d_2
1	1	d_3

ตารางที่ ๑ : สัญญาณออกจาก 4 : 1 Multiplex

ตัวอย่างโปรแกรมที่สามารถจำลองการทำงานของ 4 : 1 Multiplex ได้คือ

$$(if(a_0)then(if(a_1)then(d_3)else(d_1)else(if(a_1)then(d_2)else(d_0))))$$

โดยที่ $if-then-else$ คือ ฟังก์ชันเงื่อนไข เช่น $(if(x_0)then(y_0)else(y_1))$

- ถ้า $x_0 = 1$ จะได้ผลลัทธ์ y_0
- ถ้า $x_0 = 0$ จะได้ผลลัทธ์ y_1

๔.๑ การเตรียม Genetic Programming

ก่อนที่จะทำงานกับ GP เราจะต้องปฏิบัติดังนี้

๑. **เลือก Function Set** ตามลักษณะของปัญหา F สามารถประกอบด้วยสมาชิกดังนี้

$$F = \{if - then - else, and, or, not\} \quad (๑)$$

โดยที่ :

- if-then-else คือ ฟังก์ชันเงื่อนไข
- and คือ ฟังก์ชันลอกิจ **และ**
- or คือ ฟังก์ชันลอกิจ **หรือ**
- not คือ ฟังก์ชันลอกิจ **นิเสธ**

๒. เลือก Terminal Set ซึ่งควรประกอบด้วยสมาชิกดังนี้

$$T = \{a_0, a_1, d_0, d_1, d_2, d_3\} \quad (๒)$$

โดยที่ : a_0, a_1 คือ สัญญาณควบคุม และ d_0, d_1, d_2, d_3 คือ สัญญาณเข้า

๓. เลือกวิธีหาค่า Fitness ค่า Fitness ในตัวอย่างนี้คือการตรวจสอบว่า สัญญาณออกจาก 4 : 1 Multiplex ถูกต้องตามสัญญาณควบคุมมากน้อยเท่าใด เนื่องจาก 4 : 1 Multiplex ประกอบด้วยสัญญาณควบคุมและสัญญาณเข้ารวมทั้งสิ้น ๖ สัญญาณ ดังนั้น จึงต้องมีการตรวจสอบทั้งสิ้น 2^6 หรือ ๖๔ กรณี ดังสมการต่อไปนี้

$$f_r = \sum_{i=1}^{64} \left(f_{4:1MUX_i}(a_0, a_1, d_0, d_1, d_2, d_3) \oplus fGP_i(a_0, a_1, d_0, d_1, d_2, d_3) \right) \quad (๓)$$

โดยที่ :

- f_r คือ ค่า Fitness
- $f_{4:1MUX}(a_0, a_1, d_0, d_1, d_2, d_3)$ คือ โปรแกรมที่สามารถจำลองการทำงานของ 4 : 1 Multiplex ได้
- $fGP(a_0, a_1, d_0, d_1, d_2, d_3)$ คือ **Ind** ที่สร้างโดย GP
- $f_{4:1MUX_i}(a_0, a_1, d_0, \ldots)$ และ $fGP_i(a_0, a_1, d_0, \ldots)$ คือ ผลลัพธ์ ที่ได้จาก $f_{4:1MUX}(a_0, a_1, d_0, \ldots)$ และ $fGP(a_0, a_1, d_0, \ldots)$ เมื่อ $a_0, a_1, d_0, d_1, d_2, d_3$ มีค่าในกรณีที่ i
- $\oplus$ ฟังก์ชันลอจิก xor กล่าวคือ ถ้า $fGP(a_0, a_1, d_0, \ldots)$ และ $f_{4:1MUX}(a_0, a_1, d_0, \ldots)$ ในกรณีที่ i แสดงค่าตรงกันจะได้ผลลัพธ์เท่ากับ ๑

จากสมการ (3) **Ind** ที่ต้องการควรมีค่า Fitness $f_r = 64$

๔. กำหนดเงื่อนไขการหยุดทำงานของ GP GP จะหยุดการทำงานเมื่อเงื่อนไขใดต่อไปนี้ถูกต้อง

- **Ind** ที่มีค่า Fitness : $f_r = 64$ ได้ถูกสร้างขึ้น หรือ
- จำนวนรอบการทำงานของ GP ซึ่งก็คือ Generation $G = 50$

๕. กำหนดค่า Parameter สำหรับควบคุมการทำงานของ GP ซึ่งสามารถกำหนดได้ตามตารางที่ ๒

จำนวน Individual ต่อหนึ่ง Generation (Population Size)	100
จำนวน Generation (Maximum Number of Generation)	50
ความน่าจะเป็นของการ Crossover	90%
ความน่าจะเป็นของการ Reproduction	10%
ความน่าจะเป็นของการ Mutation	20%
วิธีการสร้าง **Ind** สำหรับ Generationที่ศูนย์	แบบสุ่มหรือไม่เจาะจง
วิธีการคัดเลือก (Selection Method)	Tournament
ค่าคงที่ในการเลือกโดยวิธี Tournament (Tournament Size)	10

ตารางที่ ๒ : Parameter สำหรับควบคุมการทำงานของ Genetic Programming

๔.๒ ผลการทดลอง

จากการทดลองใน Generation ที่ศูนย์ GP ยังไม่สามารถสร้างคำตอบได้ตามต้องการซึ่งเป็นเรื่องปกติเพราะว่า **Ind** ทุกตัวได้ถูกสร้างแบบไม่เจาะจง **Ind** ที่ดีที่สุดมีค่า Fitness เท่ากับ ๔๘ หลังจากนั้น GP ได้สร้าง **Ind** ที่ดีขึ้นเรื่อย ๆ จนภายใน Generation ที่ ๑๐ GP สามารถสร้าง **Ind** หรือโปรแกรมคอมพิวเตอร์ที่สามารถจำลองการทำงานของ 4 : 1 Multiplex ได้ **Ind** ที่ดีที่สุดของแต่ละ Generation มีดังนี้คือ

Generation เริ่มต้น หรือ Generation ที่ศูนย์ Ind ที่ดีที่สุดมีค่า Fitness : $f_r = 48$ คือ

$$(if(not(a_1))then(d_1 else(d_3)))$$

Generationที่ ๑ - ๔ Ind ที่ดีที่สุดเหมือนกับ Generation ที่ศูนย์

Generationที่ ๕ Ind ที่ดีที่สุดมีค่า Fitness $f_r = 56$ คือ

$$(if(a_1)then(if(a_o)then(d_3)else(d_2)else(d_1)))$$

Generationที่ ๖ และ ๗ Ind ที่ดีที่สุดเหมือนกับ Generationที่ ๕

Generationที่ ๘ Ind ที่ดีที่สุดมีค่า Fitness : $f_r = 60$ คือ

$$(if(a_1)then(if(a_1)then(if(a_0)then(d_3)else(d_2))else(d_0))$$
$$else(if(a_0)then(if(d_0)then(d_2)else(d_1)else)(d_0)))$$

Generationที่ ๙ Ind ที่ดีที่สุดเหมือนกับ Generationที่ ๘

Generationที่ ๑๐ Ind ที่ดีที่สุดมีค่า Fitness : $f_r = 64$ คือ

$$(if(a_1)then(if(a_1)then(if(a_0)then(d_3)else(d_2))else(d_0))$$
$$else(if(a_0)then(if(d_0)then(d_2)else(d_1)else)(d_0)))$$

๕ บทวิเคราะห์

GP ไม่ใช้การแก้ปัญหาแบบสุ่มหาคำตอบหรือที่เรียกว่า Random Search แต่ GP เป็นกรรมวิธีการแก้ปัญหาแบบฉลาดที่อาศัยการทำงานแบบสุ่มหรือไม่เจาะจง การกำหนด T และ F เป็นการกำหนดขอบเขตการค้นหาคำตอบของ GP ดัง T และ F ไม่ควรกำหนดให้กว้างมากเกินไปเพราะจะทำให้ GP ใช้เวลานานในการค้นหาคำตอบ นอกจากนี้ T และ F ไม่ควรแคบจนเกินไปเพราะจะทำให้ GP ไม่สามารถสร้างคำตอบได้ ที่ Generation ที่ศูนย์ $(G = 0)$ GP สร้าง **Ind** ขึ้นมาจำนวนมากนั้นหมายความว่า GP เริ่มหาคำตอบพร้อมกันจากหลายจุด เนื่องจาก **Ind** ทุกตัวได้ถูกสร้างแบบไม่เจาะจงจาก T และ F **Ind** ส่วนใหญ่จะมีค่า Fitness ไม่ดีเท่าที่ควร

Fitness เป็นตัวแสดงค่าความเหมาะสมในการแก้ปัญหาของ **Ind** ที่ถูกสร้างขึ้นมา การที่ให้ **Ind** ที่ค่า Fitness ดีมีโอกาสมากในการสร้าง **Indลูก** นั้นก็คือการกำหนดทิศทางการค้นหาคำตอบของ GP % ว่าให้ดำเนินต่อจาก **Ind** ดี

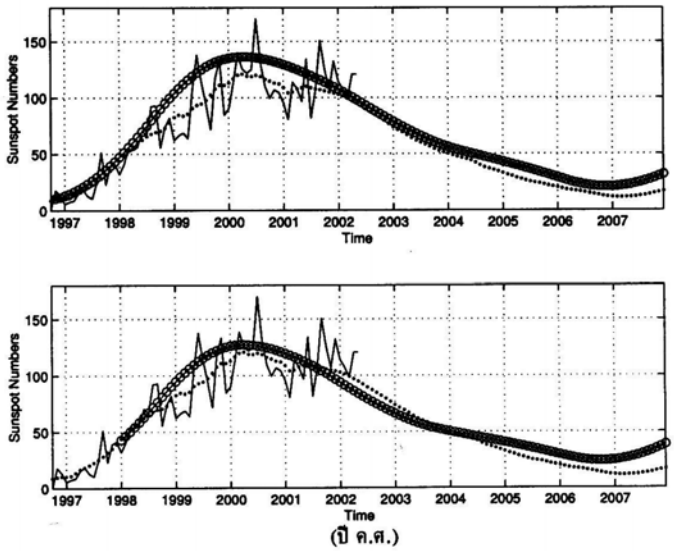
การ Crossover ระหว่าง **Ind** ที่ดีสองตัวเพื่อให้เกิด **Indลูก** **Indลูก** ที่เกิดมานั้นจะไม่เหมือนกับ **Indผู้ให้กำเนิด** แต่ตัวเองจะประกอบด้วยส่วนต่าง ๆ จากผู้ให้กำเนิดทั้งสองและอาจจะมีค่า Fitness ที่ดีกว่าด้วย ส่วนใหญ่ GP จะใช้การ Crossover เป็นเครื่องมือหลักในการค้นหาคำตอบใหม่

Reproduction คือการสร้าง **Indลูก** ให้เหมือน **Indผู้ให้กำเนิด** อย่าง ๑๐๐% เราอาจเรียกวิธีนี้ว่า Cloning ก็ได้ การ Reproduction เป็นการป้องกันไม่ให้ **Ind** ดีที่ GP เคยสร้างไว้ไม่ถูกทำลายไปและยังมีโอกาสให้กำเนิดบุตรใน Generation ต่อ ๆ ไป ตามปกติ GP จะสร้าง **Indลูก** ด้วยวิธี Crossover ประมาณ ๘๐ - ๙๐% ส่วน **Indลูก** ที่เหลือจะถูกสร้างโดยวิธีแบบ Reproduction

ด้วยการ Mutation ทำให้ **Indลูก** มีความหลายหลากมากขึ้นจากเดิมหรือเราอาจเรียกว่าการกลายพันธุ์ก็ได้ เป็นการป้องกันไม่ให้ GP ไม่สามารถสร้าง **Indลูก** ที่ดีกว่าผู้ให้กำเนิดได้ ซึ่งจะเหมือนกับการเปลี่ยนทิศทางการค้นหาคำตอบของ GP เพื่อป้องกันไม่ให้ GP หยุดอยู่ที่ Local Extreme หรือ จุดที่เหมาะสมเฉพาะในบริเวณใกล้เคียง โดยปกติจำนวน **Indลูก** ที่ผ่านการ Mutation จะมีเพียงเล็กน้อยเท่านั้นคือ ๐ - ๒๐%

GP มี Parameter หลายอย่างที่สามารถปรับหรือกำหนดให้เหมาะสมกับปัญหาได้ แต่การกำหนด Parameter เหล่านี้ไม่มีหลักเกณฑ์ที่แน่นอนจะขึ้นอยู่กับการทดลองและประสบการณ์ของผู้ทำงานกับ GP เอง มีนักวิจัยหลายท่านได้พยายามอธิบายการทำงานของ GP และการกำหนดค่าของ Parameter ด้วยวิธีการทางคณิตศาสตร์แต่ก็ยังไม่ประสบผลสำเร็จเท่าที่ควร Parameter หลัก ๆ ของ GP คือ จำนวน Individual ต่อ Generation (Population Size) และ จำนวน Generation ค่าทั้งสองค่านี้ควรกำหนดให้สูงขึ้นตามความซับซ้อนของปัญหา แต่ไม่ควรกำหนดให้สูงมากเกินไปจนคอมพิวเตอร์ที่เราใช้ทำงานด้วยใช้เวลานานมากในการแสดงผล

ใน [๙, ๑๑] ได้แสดงให้เห็นว่าในปัจจุบันมีการนำ GP มาใช้ประโยชน์ในหลายด้านและมีการพัฒนาอย่างต่อเนื่อง ตัวอย่างการนำ GP มาใช้ประโยชน์ที่จะขอกล่าวในที่นี้คือการทำนายที่เรียกว่า Time Series Prediction โดยปกติการทำนายประเภทนี้ผู้ทำนายจะต้องศึกษาและใช้กระบวนการทางคณิตศาสตร์ที่ซับซ้อนเพื่อสร้างสมการคณิตศาสตร์มาใช้ในการทำนาย ใน [12, 13, 14] ได้มีการพัฒนาวิธีการทำนาย Time Series Prediction โดยให้ GP เป็นเครื่องมือหลักในการสร้างสมการคณิตศาสตร์เพื่อใช้ในการทำนาย นอกจากนี้ยังได้เปรียบเทียบการทำนายค่าเฉลี่ยของจำนวนจุดด่างบนดวงอาทิตย์ (Sunspot Number) ด้วยวิธีต่าง ๆ การนำ GP มาใช้จะสามารถลดกระบวนการทางคณิตศาสตร์ที่ซับซ้อนได้ ทั้งยังมีผลการทำนายค่อนข้างดี



รูปที่ ๖ : การทำนายค่าเฉลี่ยของจำนวนจุดด่างบนดวงอาทิตย์ในแต่ละเดือนจนถึง เดือนสิงหาคม พ.ศ. ๒๕๕๐ (ค.ศ. ๒๐๐๗) จากจุดเริ่มต้นที่ต่างกัน *เส้นต่อเนื่อง* : ค่าเฉลี่ยจริงที่วัดได้ จุด : ค่าทำนายโดย NOAA, *วงกลม* : ค่าทำนายโดยใช้ GP ใน [14]

รูปที่ ๖ แสดงผลการทำนายค่าเฉลี่ยของจำนวนจุดด่างบนดวงอาทิตย์ในแต่ละเดือนเป็นระยะยาวจนถึงเดือนสิงหาคม พ.ศ. ๒๕๕๐ (ค.ศ ๒๐๐๗) จากจุดเริ่มต้นที่ต่างกันโดย US National Oceanic and Atmospheric Administration (NOAA)} และ GP ใน [14] ค่าทำนายที่ได้จากทั้งสองวิธีมีลักษณะที่

ใกล้เคียงกัน นอกจากนี้ยังแสดงให้เห็นว่าค่าเฉลี่ยของจำนวน จุดด่างบนดวงอาทิตย์ ในแต่ละเดือนจะลดลงเรื่อย ๆ จนถึงประมาณปลายปี พ.ศ. ๒๕๔๙ หรือ ต้นปี พ.ศ. ๒๕๕๐

๖ สรุป

GP เป็นกรรมวิธีหนึ่งที่ถูกพัฒนาขึ้นมาเพื่อให้คอมพิวเตอร์สามารถสร้างโปรแกรมขึ้นด้วยตัวเองได้ โดยเลียนแบบการวิวัฒนาการตามธรรมชาติที่อาศัยกลไกในการถ่ายทอดทางพันธุกรรม เริ่มแรกของการทำงานเราต้องกำหนดสมาชิกใน Function Set และ Terminal Set เพื่อเป็นการกำหนดขอบเขตและแนวทางในการสร้างโปรแกรมคอมพิวเตอร์ของ GP ที่เรียกกันว่า Individual หลังจากนั้น GP จะสร้าง Individual ขึ้นมาแบบสุ่มหรือไม่เจาะจงจาก Terminal Set และ Function Set ที่กำหนดไว้หลาย ๆ ตัว Individual ที่ดี ๆ จะถูกเลือกเพื่อนำไปสร้าง Individual ใหม่ต่อไปด้วยวิธีต่าง ๆ เช่น Crossover หรือ Reproduction Individual ใหม่ที่ถูกสร้างขึ้นอาจถูกเปลี่ยนแปลงโครงสร้างบางส่วนโดยวิธี Mutation วิธีทั้งสามแบบคือ Crossover, Reproduction และ Mutation เป็นการเลียนแบบการถ่ายทอดทางพันธุกรรมในสิ่งมีชีวิต Individual ที่สร้างใหม่ที่ดี ๆ ก็จะถูกเลือกเพื่อนำไปสร้าง Individual ใหม่ต่อไป GP จะทำงานไปเรื่อย ๆ จนกว่าจะครบจำนวนรอบการทำงานหรือได้ Individual ที่เราต้องการ ปัจจุบันมีการนำ GP มาใช้ประโยชน์และได้ผลดีในหลายด้าน เช่น การทำนายแบบ Time Series Prediction เป็นต้น

หนังสืออ้างอิง

๑. T. Bäck, U. Hammel, H. Schwefel : Evolutionary Computation : Comments on the History and Current State, In : IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, หน้า ๓ - ๑๖, April, 1997.

๒. T. Bäck, D. Fogel, Z. Michalewicz : *Handbook of Evolutionary Computation*, Oxford University Press, Oxford 1997.

๓. D. B. Fogel : *What is evolutionary computation?*, In : *IEEE Spectrum*, Vol. 37, No. 2, หน้า ๒๖ - ๓๒, February, 2000.

๔. J. H. Holland : *Outline for a logical theory of adaptive systems*, In : *Journal of the Association for Computing Machinery*, Vol. 3, หน้า๒๙๗ - ๓๑๔, 1962.

๕. I. Rechenberg : *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishment, Library translation No. 1122, Farnborough, Hants., U.K., August 1965.

๖. L. J. Fogel, A. J. Owens, M. J. Walsh : *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, 1966.

๗. H. -P. Schwefel : *Projekt MHD-Staustrahlrohr : Experimentelle Optimierung einer Zweiphasend\"use. Teil* , Technischer Bericht 11.034/68, 35, AEG Forschungsinstitut, Berlin, Germany, Oktober 1968.

๘. J. R. Koza : Genetic Programming : On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.

๙. W. Banzhaf, P. Nordin, R. E. Keller, F.D. Francone : Genetic Programming An Introduction : On the Automatic Evolution of Computer Programs and its Applications}, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998.

๑๐. W. Banzhaf, J. R. Koza, C. Ryan, L. Spector, C. Jacob : *Genetic progamming*, In : *IEEE Intelligent Systems*, Vol. 15, No. 13, หน้า ๗๔ - ๘๔, 2000.

๑๑. http://www.genetic-programming.org

๑๒. W. Panayaworayan and G. Wuetschner : *Time Series Prediction Using a Recursive Algorithm of a Combination of Genetic Programming and Constant Optimization*, In : Mendel 2002 8th International Conference on Soft Computing, หน้า 68-73, Brno, Czech Republic, June 5-7, 2002.

๑๓. W. Panayaworayan and G. Wuetschner : *Time Series Prediction Using a Recursive Algorithm of a Combination of Genetic Programming and Constant Optimization*, In : GECCO 2002 : Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, หน้า ๑๐๑ - ๑๐๗, New York City, New York, USA, 9 July 2002.

๑๔. W. Panayaworayan, G. Wuetschner : *Time Series Prediction Using a Recursive Algorithm of a Combination of Genetic Programming and Constant Optimization*, In : Facta Universitatis, Series: Electronics and Energetics, Vol. 15, No. 2, หน้า ๒๖๕ - ๒๗๙, August 2002.