

## กำหนดการเชิงพันธุกรรมและฟังก์ชันซึ่งถูกนิยามโดยอัตโนมัติ

### Genetic Programming and Automatically Defined Function

#### กำหนดการเชิงพันธุกรรม (genetic programming, GP)

กำหนดการเชิงพันธุกรรมเป็นวิธีการหนึ่งที่ใช้สำหรับการเรียนรู้ ซึ่งเป็นวิธีการที่ได้รับการพัฒนามาจากขั้นตอนวิธีเชิงพันธุกรรม (genetic algorithm, GA) ซึ่งคิดค้นโดย Holland (1975) ขั้นตอนวิธีเชิงพันธุกรรมเป็นกระบวนการค้นหาคำตอบโดยการจำลองแบบตามกฎการคัดเลือกโดยธรรมชาติ ซึ่งมีลักษณะที่แตกต่างไปจากเทคนิคอื่นทางปัญญาประดิษฐ์ เนื่องจากเทคนิคทางปัญญาประดิษฐ์ส่วนใหญ่ในการแก้ปัญหาหนึ่งๆ จะเริ่มจากต้องสร้างความรู้ในการแก้ปัญหาขึ้นเสียก่อน จากนั้นจึงทำการค้นหาคำตอบโดยอ้างอิงกับความรู้นั้นๆ ส่วนขั้นตอนวิธีเชิงพันธุกรรมไม่ใช้ความรู้ในการค้นหาคำตอบ โดยในการแก้ปัญหาจะเริ่มจากการสร้างกลุ่มของคำตอบที่เป็นไปได้เป็นขั้นตอนแรกและผลลัพธ์ที่ได้จะมีลักษณะเป็นกลุ่มของคำตอบ ข้อแตกต่างระหว่างกำหนดการเชิงพันธุกรรมและขั้นตอนวิธีเชิงพันธุกรรมคือลักษณะของการแทนคำตอบ กำหนดการเชิงพันธุกรรมจะแทนคำตอบในลักษณะของโปรแกรมคอมพิวเตอร์หรือต้นไม้ (tree) ส่วนขั้นตอนวิธีเชิงพันธุกรรมจะแทนลักษณะของคำตอบเป็นสตริงที่ขนาดคงที่

การค้นหาคำตอบโดยกำหนดการเชิงพันธุกรรม อาจแบ่งออกเป็นขั้นตอนหลัก 4 ขั้นตอนดังนี้

1. **การสร้างประชากรของผลเฉลยเริ่มต้น** เป็นขั้นตอนเริ่มต้นของกำหนดการเชิงพันธุกรรมซึ่งจะทำการสร้างประชากรของผลเฉลยแบบสุ่ม โดยในแต่ละผลเฉลยจะประกอบไปด้วยส่วนประกอบ 2 ชนิดคือ ฟังก์ชันและเทอมินอล

**ฟังก์ชัน** ฟังก์ชันที่ใช้ในผลเฉลยอาจเป็นฟังก์ชันปฏิบัติการพื้นฐาน (เช่น การบวก การลบ) ฟังก์ชันมาตรฐานทางคณิตศาสตร์ (เช่น sin log) ฟังก์ชันมาตรฐานในการโปรแกรม (เช่น if-then-else) หรือฟังก์ชันทางตรรก (เช่น and or) เป็นต้น

**เทอมินอล** โดยปกติจะเป็นเซตของปัจจัยที่เป็นอิสระแก่กันสำหรับปัญหานั้นๆ เช่น ปัญหาการหาทางออกจากทางวงกต เทอมินอลคือคำสั่งในการ เลี้ยวซ้าย เลี้ยวขวา และตรงไป เป็นต้น ในกรณีอื่นๆ เทอมินอลอาจหมายถึงฟังก์ชันที่ไม่มีอาร์กิวเมนต์

2. **การตรวจสอบประสิทธิภาพของผลเฉลย** ในขั้นตอนนี้จะทำการตรวจสอบประสิทธิภาพของผลเฉลยแต่ละผลเฉลยโดยฟังก์ชันความเหมาะสม (fitness function) โดยจะมีผลเฉลยส่วนน้อยเท่านั้นที่มีประสิทธิภาพดีซึ่งจะนำไปสร้างประชากรของผลเฉลยในรุ่น (generation) ใหม่ต่อไป

3. **การสร้างประชากรของผลเฉลยรุ่นใหม่** ขั้นตอนนี้จะเป็นการสร้างประชากรของผลเฉลยรุ่นใหม่โดยการสืบพันธุ์ (reproduction) การไขว้เปลี่ยน (crossover) และการกลาย (mutation) จากนั้นประชากรของผลเฉลยรุ่นใหม่ที่ได้ในขั้นตอนนี้ จะถูกดำเนินการซ้ำในขั้นตอนตรวจสอบประสิทธิภาพใหม่ และนำไปสร้างเป็นประชากรผลเฉลยในรุ่นต่อไป

4. **การหาผลลัพธ์** ขั้นตอนนี้เป็นการหาผลลัพธ์ของกำหนดการเชิงพันธุกรรม โดยผลเฉลยหรือกลุ่มของผลเฉลยที่มีประสิทธิภาพดีที่สุด (ประเมินด้วยฟังก์ชันความเหมาะสม) ที่ปรากฏในขั้นตอนทั้งหมดที่กล่าวมาข้างต้นจะเป็นผลลัพธ์ของกำหนดการเชิงพันธุกรรม

#### ฟังก์ชันซึ่งถูกนิยามโดยอัตโนมัติ (automatically defined functions, เอดีเอฟ)

เอดีเอฟเป็นวิธีที่ได้รับการเสนอโดย Koza (1994) โดยวิธีเอดีเอฟ โครงสร้างของแต่ละผลเฉลยที่ถูกสร้างในกระบวนการเรียนรู้แบบกำหนดการเชิงพันธุกรรมจะประกอบไปด้วยฟังก์ชันหลักและฟังก์ชันย่อย ซึ่งต่างจากกำหนดการเชิงพันธุกรรมปกติซึ่งโครงสร้างของแต่ละผลเฉลยมีเพียงฟังก์ชันหลักเพียงฟังก์ชันเดียว โครงสร้างของผลเฉลยของวิธีเอดี

เอฟฟังก์ชันอยู่กับพารามิเตอร์(parameter)ต่างๆ ได้แก่ จำนวนฟังก์ชัน จำนวนอาร์กิวเมนต์(argument)ของแต่ละฟังก์ชัน และการอ้างอิงระหว่างฟังก์ชันซึ่งต้องมีการกำหนดก่อนหน้า การเลือกใช้พารามิเตอร์ต่างๆนี้จำเป็นต้องมีการศึกษาปัญหาเป็นอย่างดี เนื่องจากปัญหาแต่ละปัญหาจะมีพารามิเตอร์ที่เหมาะสมแตกต่างกัน

วิธีเอดีเอฟเป็นวิธีที่มีประสิทธิภาพในการหาคำตอบของของกำหนดการเชิงพันธุกรรม เนื่องมาจากวิธีเอดีเอฟใช้หลักการของการแยกส่วนของปัญหาในกระบวนการเรียนรู้แบบกำหนดการเชิงพันธุกรรม กล่าวคือปัญหาจะถูกแบ่งออกเป็นส่วยย่อยหลายๆส่วนซึ่งง่ายต่อการแก้ปัญหา อาจเปรียบเทียบได้กับการเขียนโปรแกรมเป็นลำดับขั้นของมนุษย์ซึ่งฟังก์ชันแต่ละฟังก์ชันจะทำหน้าที่แก้ปัญหาย่อยแต่ละปัญหาโดยเฉพาะ เมื่อรวมฟังก์ชันเหล่านี้เป็นโปรแกรมคอมพิวเตอร์ก็จะทำให้สามารถแก้ปัญหาใหญ่ได้ทั้งหมด และยังช่วยลดความซับซ้อนในการเขียนโปรแกรมด้วย โดยสรุปวิธีเอดีเอฟเป็นวิธีที่มีประสิทธิภาพสำหรับการแก้ปัญหาโดยเฉพาะสำหรับปัญหาที่ซับซ้อน และในกรณีที่ปัญหามีความซับซ้อนมากขึ้นวิธีเอดีเอฟจะมีอัตราการเพิ่มของการคำนวณ(จำนวนผลเฉลยที่ต้องถูกประมวลผลจนพบคำตอบ)น้อยกว่ากระบวนการเรียนรู้แบบกำหนดการเชิงพันธุกรรมแบบปกติ

### ความเพียรพยายามเชิงคำนวณ (computational effort)

เนื่องจากกระบวนการของกำหนดการเชิงพันธุกรรมเป็นกระบวนการที่ใช้การสุ่ม ซึ่งทำให้ในแต่ละครั้งของการทดลองได้ผลลัพธ์ที่ไม่เหมือนกัน การวัดประสิทธิภาพของวิธีการเรียนรู้แบบกำหนดการเชิงพันธุกรรมจึงต้องใช้การคำนวณทางสถิติโดยใช้ข้อมูลจากผลของการทดลองซ้ำหลายๆครั้ง

หากกำหนดให้  $Y(M, i)$  แทนความน่าจะเป็นที่จะพบผลเฉลยที่ประสบความสำเร็จในรุ่นที่  $i$  ซึ่งคำนวณได้จากจำนวนรอบของการทดลองที่พบผลเฉลยที่ประสบความสำเร็จในรุ่นที่  $i$  หากด้วยจำนวนรอบของการทดลองทั้งหมดจะสามารถคำนวณหาค่าความน่าจะเป็นสะสมของความสำเร็จ  $P(M, i)$  ได้ โดย  $P(M, i)$  จะเป็นฟังก์ชันที่ไม่ลดลง

หากกำหนดให้  $R(M, i, z)$  แทนจำนวนรอบของการทดลองเพื่อให้สามารถพบ ผลเฉลยที่ประสบความสำเร็จภายในรุ่นที่  $i$  โดยมีความมั่นใจของความสำเร็จเท่ากับ  $z$  (เช่น  $z$  เท่ากับ 99 เปอร์เซ็นต์) จะได้ว่า

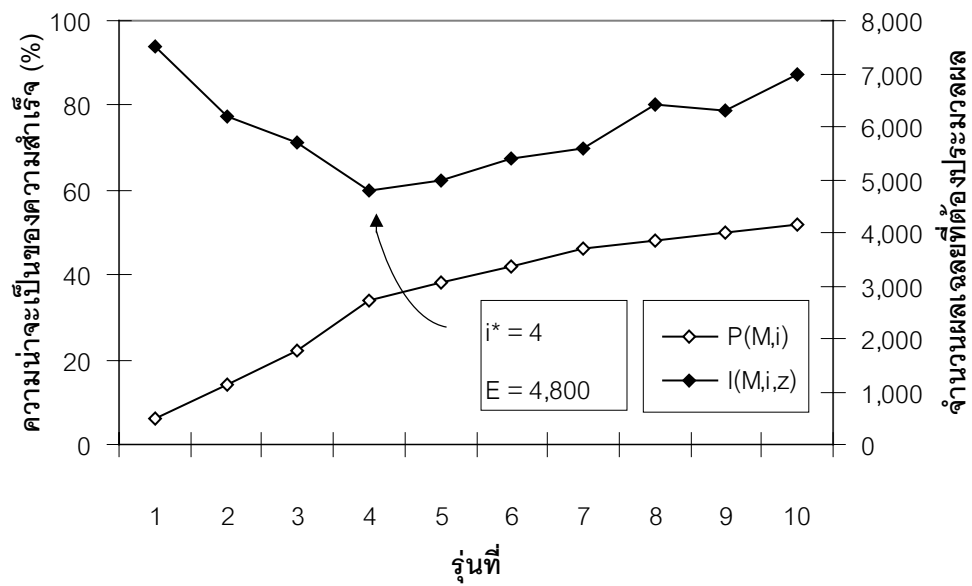
$$z = 1 - [1 - P(M, i)]^{R(M, i, z)}$$

ซึ่งสามารถหา  $R(M, i, z)$  ได้จากสมการ

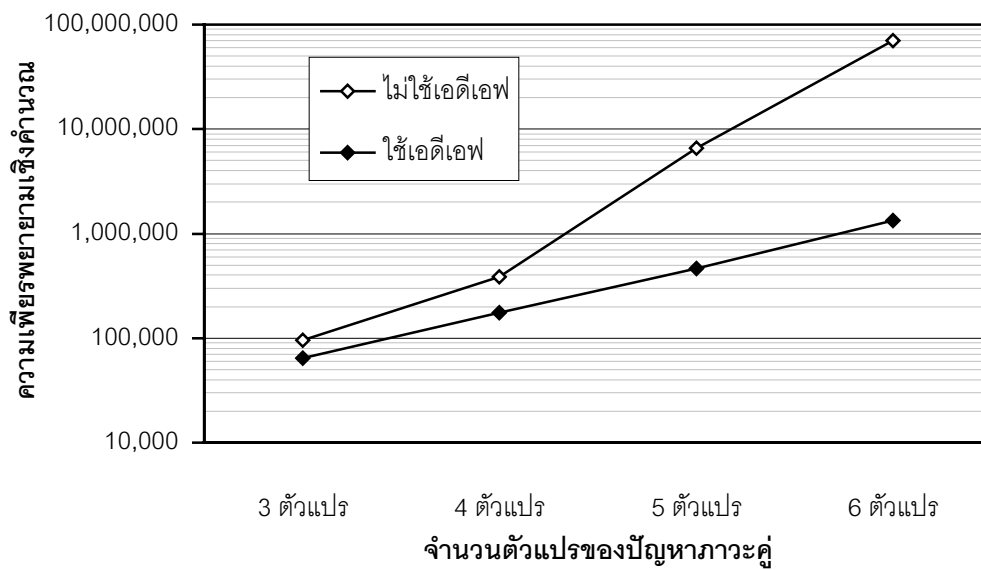
$$R(M, i, z) = \left\lceil \frac{\log(1 - z)}{\log(1 - P(M, i))} \right\rceil$$

หากกำหนดให้  $I(M, i, z)$  แทนจำนวนผลเฉลยอย่างน้อยที่ต้องประมวลผลจนพบ ผลเฉลยที่ประสบความสำเร็จภายในรุ่นที่  $i$  โดยมีค่าความมั่นใจของความสำเร็จ  $z$  จะได้ว่า  $I(M, i, z) = M \times i \times R(M, i, z)$  สำหรับรุ่นที่มีค่า  $I(M, i, z)$  น้อยที่สุดเรียกว่ารุ่นที่ดีที่สุด (best generation,  $i^*$ ) ซึ่งค่า  $I(M, i, z)$  ของรุ่นที่ดีที่สุดนี้เรียกว่า ความเพียรพยายามเชิงคำนวณ ( $E$ ) โดย  $E = I(M, i^*, z) = M \times i^* \times R(M, i^*, z)$

ในการเปรียบเทียบโดยใช้ค่าความเพียรพยายามเชิงคำนวณ หากใช้วิธีในการเรียนรู้เดียวกันในการหาคำตอบของปัญหาที่ต่างกัน ปัญหาที่มีค่าความเพียรพยายามเชิงคำนวณสูงกว่าจะแสดงถึงปัญหานั้นมีความยากกว่าปัญหาที่มีค่าความเพียรพยายามเชิงคำนวณต่ำกว่า และหากใช้วิธีในการเรียนรู้ต่างกันในการหาคำตอบของปัญหาเดียวกัน วิธีการเรียนรู้ที่คำนวณค่าความเพียรพยายามเชิงคำนวณได้ต่ำกว่าแสดงถึงความมีประสิทธิภาพในการเรียนรู้มากกว่าวิธีการเรียนรู้ที่คำนวณค่าความเพียรพยายามเชิงคำนวณได้สูงกว่า



รูปที่ 1 เส้นโค้งสมรรถนะของตัวอย่างปัญหา



รูปที่ 2 ความเพียรพยายามเชิงคำนวณของปัญหาภาวะคู่

### รายการอ้างอิง

Holland, J. H. Adaptation in Natural and Artificial System. Ann Arbor, Michigan : University of Michigan Press, 1975.

Koza, J. R. Genetic Programming II: Automatic Discovery of Reuseable Programs. Massachusetts : MIT Press, 1994.