การคำนวณเชิงวิวัฒนาการระหว่าง เจเนติกอัลกอริทึม กับ พาติคอลสวอมออปติมัยเซชัน

Evolutionary computation between Genetic Algorithm and Particle Swarm Optimization

สุภกิจ นุตยะสกุล

คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Email: supakit@it.kmitl.ac.th

บทคัดย่อ – Genetic algorithm (GA) สามารถแก้ปัญหาและค้นหาผลลัพธ์ได้ดี แต่กระนั้นเจเนติกอัลกอริทึมยังขาดในด้านการแบ่งข้อมูลระหว่างประชากรจึงทำให้ GA ค้นหาผลลัพธ์ค่อนข้างช้า วิธีการใหม่ที่ได้รับการกล่าวถึงคือ Particle Swarm Optimization (PSO) ซึ่งจัดอยู่ในกลุ่มทฤษฎีการวิวัฒนาการกลุ่มเดียวกับ GA โดย PSO ใช้การจำลองหลักการค้นหาอาหารหรือเคลื่อนย้ายของสิ่งมีชีวิต ที่อยู่เป็นกลุ่ม เช่นฝูงนกหรือฝูงปลา ไม่เหมือนกับ GA ที่ใช้การแข่งขันของประชากรค้นหาผลลัพธ์ บทความนี้นำเสนอเทคนิคของ PSO โดยเปรียบเทียบกับ GA เพื่อเชื่อมโยงความเหมือนและความต่างระหว่าง GA กับ PSO และสามารถนำไปประยุกต์ใช้งานเพื่อแก้ปัญหาที่ซับซ้อนได้

คำสำคัญ – เจเนติกอัลกอริทึม, พาติคอลสวอมออปติมัยเซชัน, อีโวลูชันนารีคอมพิวเตชัน

ABSTRACT – Genetic algorithm (GA) proved by many researchers that can solve optimization problems. However, GA lack on sharing information between the populations in consequent GA finds the solution quite slow. The new technique that is mentioned is particle swarm optimization (PSO). PSO is a technique in a group of evolutionary computation like GA. PSO find the best result by simulation moving of bird or fish to find foods or living, unlike GA. The GA uses the candidate of population to find the solution. This article presents techniques of PSO comparing with GA to point out the difference the process finding the solution.

KEYWORDS – Evolutionary Computation, Genetic algorithm, Particle swarm optimization

---

1. บทนำ

ในทางวิทยาศาสตร์และวิศวกรรมใช้ การคำนวณเชิงวิวัฒนาการ (evolutionary computation) เพื่อค้นหาผลลัพธ์ของปัญหาที่มีความซับซ้อน เช่น การจัดรูปเส้นทางการจัดส่งสินค้าให้ได้ปริมาณสูงและค่าใช้จ่ายเชื้อเพลิงต่ำสุด การคำนวณสัดส่วนของ สารเคมีเพื่อให้ได้องค์ประกอบของสารที่มีประสิทธิภาพ การบริหารทรัพยากรการในผลิตให้มีประสิทธิภาพ วิธีการค้นหาผลลัพธ์ที่ดีสุดเหล่านี้นิยมใช้การคำนวณเชิงวิวัฒนาการทั้งสิ้น การคำนวณเชิงวิวัฒนาการมีต้นกำเนิดจากทฤษฎีวิวัฒนาการของสิ่งมีชีวิต ผู้คิดค้นคือ Charles Darwin ปี 1842 ต่อจากนั้นนักวิจัยได้นำทฤษฎีมาประยุกต์กับการคำนวณของคอมพิวเตอร์จึงเป็นวิธีการคำนวณเชิงวิวัฒนาการ ซึ่งเมื่อจัดกลุ่มแล้วใช้หลักการค้นหาผลลัพธ์จัดอยู่ในกลุ่มวิธีเมต้าฮิวริสติก (meta-heuristic) ซึ่งใช้หลักการค้นหาผลลัพธ์โดยอาศัยวนซ้ำสุ่มค่าพารามิเตอร์เพื่อหาผลลัพธ์ที่ดีสุด ผลลัพธ์ในการค้นหาแต่ละครั้งจะไม่ซ้ำกัน

การคำนวณเชิงวิวัฒนาการมีวิธีการหลากหลายแบบที่ใช้กันแพร่หลายคือ เจเนติกอัลกอริทึม (genetic algorithm : GA) [1-3] พาติคอลสวอมออปติมัยเซชัน (particle swarm optimization : PSO) การทำงานของ GA ใช้การจำลองปัญหาให้อยู่ในรูปแบบเหมือนสิ่งมีชีวิตที่ประกอบด้วยโครโมโซม (chromosome) และยีน (gene) การค้นหาผลลัพธ์ของปัญหาใช้กระบวนการทางพันธุ์ศาสตร์ประกอบด้วย การสืบพันธุ์ (reproduction) การกลายพันธุ์ (mutation) การแลกเปลี่ยนยีน (reproduction) และการคัดเลือกประชากร (selection) วิธีการทางพันธุศาสตร์เหล่านี้ทำหน้าที่ปรับปรุงประชากรจากรุ่นหนึ่งสู่อีกรุ่นหนึ่ง เพื่อให้อยู่รอดต่อสภาพแวดล้อมที่เปลี่ยนแปลงไป ดังเช่นการวิวัฒนาการของสิ่งมีชีวิตในอดีตมาถึงปัจจุบันมีการปรับปรุงลักษณะทั้งภายในและภายนอกให้เหมาะสมกับสภาพแวดล้อม การค้นหาผลลัพธ์ของปัญหาด้วย GA จึงเปรียบเสมือนกับการนำพารามิเตอร์ของปัญหาแทนในรูปแบบสิ่งมีชีวิต และกำหนดให้สภาพแวดล้อมเป็นปัญหาหรือฟังก์ชันที่ต้องการค้นหาค่าพารามิเตอร์ที่เหมาะสม

การคำนวณเชิงวิวัฒนาการอีกกลุ่มหนึ่งที่กำลังได้รับความนิยมและมีงานวิจัยต่างๆนำมาทดสอบและปรับปรุงคือ วิธีหาค่าที่ดีสุดแบบกลุ่มอนุภาค หรือ PSO [4] ใช้หลักการวิวัฒนาการ โดยนำมาจากพฤติกรรมทางสังคม การเคลื่อนย้ายของฝูงนกหรือปลาเพื่อค้นหาอาหาร PSO ใช้หลักของการแทนปัญหาในรูปประชากรคล้าย GA แต่ PSO มีสิ่งที่เด่นกว่าและน่าสนใจตรงที่การแบ่งปันข้อมูลระหว่างประชากร เป็นข้อมูลที่ใช้ร่วมกันเพื่อช่วยให้ประชากรกันในการค้นหาผลลัพธ์และแก้ปัญหาที่ต้องการได้อย่างรวดเร็วขึ้น การแบ่งปันข้อมูลนี้แตกต่างจาก GA ที่ประชากรต้องแข่งขันกันเพื่อค้นหาผลลัพธ์และไม่มีการแบ่งปันข้อมูลของทิศทางการค้นหาผลลัพธ์ของปัญหา

บทความนี้จึงนำเสนอวิธีการทำงานของ PSO เปรียบเทียบกับ GA เพื่อให้เห็นถึงกระบวนการทำงานและการใช้งาน โดยยกตัวอย่างการแก้ปัญหาการหาผลลัพธ์ที่ดีสุดของฟังก์ชันทางคณิตศาสตร์ (numerical optimization) นำมาประกอบการอธิบาย เนื้อหาเริ่มต้นโดยส่วนที่ 2 แนะนำให้รู้จักกับ GA และกระบวนการทำงาน จากนั้นในส่วนที่ 3 นำเสนอการทำงานของ PSO ส่วนที่ 4 วิจารณ์เปรียบเทียบ GA PSO และ Local Search ต่อจากนั้นในส่วนที่ 5 แสดงตัวอย่างการประยุกต์ใช้ GA และ PSO และบทสรุปแสดงในส่วนสุดท้าย

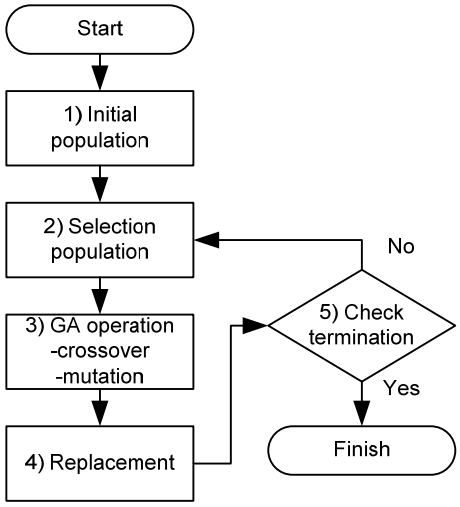
2. Genetic Algorithm

GA ใช้หลักการมาจากทฤษฎีการวิวัฒนาการของ Charles Darwin โดยทาง John Holland [1] นำมาเสนอหลักการ และจากนั้น Goldberg [6] ได้นำ GA มาปรับปรุงและประยุกต์ใช้กับงานทางวิศวกรรมเป็นผลให้ตั้งแต่นั้นเป็นต้นมา GA จึงเป็นที่รู้จักและนำมาประยุกต์ใช้ในงานกันแพร่หลาย สังเกตได้จากจำนวนงานวิจัยมีการตีพิมพ์มากตั้งแต่ปี 1990 เป็นต้นมา ในปัจจุบัน GA ได้ปรับปรุงเพื่อเพิ่มประสิทธิภาพการค้นหาให้มีความสามารถและประสิทธิภาพสูงขึ้น วิธีการปรับปรุงเช่น การนำ GA มาผสมผสานกับวิธีการค้นหาแบบโลคอล (local search) เพื่อนำจุดเด่นของทั้งสองวิธีการ โดยที่ GA สามารถกระจายการค้นหาผลลัพธ์ได้กว้างลักษณะเหมือนการค้นหาแบบกว้าง ส่วนการค้นหาแบบโลคอลสามารถค้นหาในลักษณะลงลึกเข้าถึงผลลัพธ์ที่ดีสุดในช่วงแคบ การผสมจุดเด่นทั้งสองทำให้ประสิทธิภาพของ GA แบบผสมผสานทำงานได้ดีขึ้น ในบทความนี้นำเสนอโครงสร้างและรูปแบบของ GA แบบดั้งเดิมที่ยังไม่มีการผสมผสาน หรือประยุกต์เทคนิคการค้นหาแบบอื่นๆเข้าไว้ภายใน เพื่อให้ผู้อ่านเห็นถึงการทำงาน และเปรียบเทียบการทำงานระหว่าง GA แบบดั้งเดิมกับการทำงานของ PSO ในรูปแบบเดิมที่ไม่มีการปรับปรุงประสิทธิภาพ ซึ่ง

จะสะท้อนเห็นถึงกระบวนการทำงาน ความแตกต่างของทั้งสองวิธีการได้อย่างดี

ก่อนที่จะรู้จักกระบวนการทำงานของ GA จำเป็นต้องรู้จักพารามิเตอร์และตัวแปรต่างๆ ภายใน GA แบ่งประชากรเป็นสองชนิดคือ ประชากรเก่า *Pold* และ ประชากรใหม่ *Pnew* เปรียบเสมือนกลุ่มของพ่อแม่กับกลุ่มลูก ประชากรทั้งหมดมีโครงสร้างเหมือนกันประกอบด้วยโครโมโซมและยีนเป็นส่วนประกอบภายในหรือลักษณะภายใน ส่วนประกอบอีกส่วนคือลักษณะภายนอกซึ่งเรียกว่าค่าความเหมาะสมของประชากร Fitness value เปรียบได้กับความสูงหรือน้ำหนักของประชากร ดังนั้นเมื่อนำ GA มาใช้เพื่อแก้ไขปัญหาจะทำให้ประชากรหนึ่งตัวประกอบพารามิเตอร์ P_N = {*Chromosome*, *Fitness value*} มีความหมายดังนี้

- *Chromosome* คือกลุ่มพารามิเตอร์ที่เป็นส่วนประกอบใช้ค้นหาหรือคำนวณ เพื่อค้นหาผลลัพธ์ พารามิเตอร์สามารถสร้างได้สองแบบคือ สร้างจากเลขไบนารี ซึ่งเป็นการสร้างจากเลขฐานสอง และแบบสร้างจากเลขจำนวนจริง [5] ตัวอย่างเช่น พารามิเตอร์ x เป็นเลขจำนวนจริงในช่วง -5 ถึง 5 ระยะห่างระหว่างเลขอยู่ในช่วง 0.01 (ดังนั้นจาก -5 ถึง 5 แบ่งเป็น 1000 ขั้น) หากแทนให้อยู่ในรูปไบนารีจะต้องออกแบบให้มีขนาด 10 บิต หากแทนด้วยเลขจำนวนจริงสามารถใช้พารามิเตอร์หนึ่งตัว การเลือกใช้การเข้ารหัสขึนอยู่กับลักษณะของปัญหาที่แก้ไข อย่างไรก็ตามลักษณะของเลขไบนารีอาจจะมีการคำนวณมากกว่าเพราะใช้กระบวนการแปลงเลข โครโมโซมที่มีพารามิเตอร์หลายตัวหรือยีนหลายตัวเขียนได้เป็น $Chromosome_D = \{Ch_1, Ch_2, \ldots, Ch_d\}$
- *D คือจำนวนมิติของปัญหา หรือจำนวนพารามิเตอร์ที่ใช้ในการคำนวณเพื่อค้นหาผลลัพธ์*
- *Fitness value คือค่าความเหมาะสมของประชากรคำนวณได้จาก fitness value = objective function (Chromosome)*
- *N คือจำนวนประชากร*



รูปที่ 1. แผนผังการทำงานของ GA

กระบวนการทำงานของ GA แสดงดังรูปที่ 1 ประกอบด้วยกระบวนการหลักๆ 5 ขั้นตอนโดยที่

1) Initial population การสร้างประชากรเริ่มต้นโดยใช้การสุ่มค่าให้ยีน ค่าของยีนต้องสังเกตว่าไม่ให้มีค่าไม่เกินขอบเขตของปัญหา การสุ่มที่ดีค่าที่ได้ต้องกระจายและไม่มีรูปแบบเลขซ้ำกัน ดังสมการ (1) ฟังก์ชันสุ่มส่งค่าโครโมโซมของประชากรที่ได้สุ่มค่าแล้วให้แก่ประชากรเก่า

$$Pold_i = random\ generator\ (\ ) \quad (1)$$

2) Selection population การคัดเลือกประชากรเพื่อนำไปทำกระบวนทางพันธุกรรม (genetic operation) โดยวิธีการคัดเลือกประชากรทำได้หลายวิธีเช่น การเลือกโดยการหมุนวงล้อ, การคัดเลือกโดยการสุ่ม และการเลือกโดยแข่งขัน [7] แต่ละวิธีการให้น้ำหนักการเลือกประชากรแตกต่างกัน วิธีการสุ่มเป็นวิธีการกระจายโอกาสให้แก่ประชากรมากสุด ส่วนวิธีการแข่งขันประชากรในกลุ่มที่มีค่าความเหมาะสมสูงมีโอกาสได้คัดเลือกมากที่สุด สมการ (2) กลุ่มประชากรเก่าส่งให้ฟังก์ชันคัดเลือกประชากรซึ่งจะเลือกประชากรเก่าจำนวนสองตัว คัดลอกเป็นประชากรใหม่คือ $Pnew_i$ และ $Pnew_j$

$$[Pnew_i, Pnew_j] = function\ selection\ (Pold) \quad (2)$$

3) Genetic operation การกระทำทางพันธุกรรมประกอบด้วยการครอสโอเวอร์ (crossover) คือการสลับค่าของโครโมโซมระหว่างประชากรที่ได้คัดเลือก วิธีการสลับค่ามีหลายแบบเช่น การสุ่มตำแหน่งยีนที่ต้องการสลับค่า และการสลับค่าแบบระบุตำแหน่งยีน [7] จำนวนของการสลับค่าของยีนขึ้นอยู่กับค่าความน่าจะเป็นในการแลกเปลี่ยนยีน (crossover probability) เรียกย่อ *pc* เมื่อทำการครอสโอเวอร์เสร็จแล้วจึงเข้าสู่กระบวนการกลายพันธ์ (mutation) เป็นกระบวนการสุ่มค่าพารามิเตอร์ใหม่ให้ยีน โดยยีนในตำแหน่งใดที่จะถูกกำหนดค่าให้กำหนดจากโอกาสจากค่าความน่าจะเป็นในการกลายพันธ์ (mutation probability) เรียกย่อ *pm* สมการ (3) ประชากร $Pnew_i$ และ $Pnew_j$ ส่งค่าให้ฟังก์ชันเพื่อแลกเปลี่ยนโครโมโซมระหว่างกันโดยโอกาสในการสลับกำหนดจาก *pc* ผลลัพธ์ที่ได้ส่งกลับแทนให้แก่ประชากรใหม่ซึ่งโครโมโซมของประชากรทั้งสองมีลักษณะแตกต่างไปจากเดิม เพราะมีบางส่วนที่ได้จากประชากรอีกตัวมาผสมอยู่ สมการ (4) ทั้งประชากร $Pnew_i$ หรือ $Pnew_j$ ส่งค่าให้ฟังก์ชันกลายพันธ์ ฟังก์ชันทำหน้าที่เปลี่ยนค่ายีนใหม่ด้วยการสุ่ม โอกาสที่จะได้ค่ายีนใหม่ขึ้นอยู่กับ *pm* ผลลัพธ์ที่ได้ส่งกลับให้ประชากรใหม่ตัวเดิม

$$[Pnew_i, Pnew_j] = function\ crossover\ (Pnew_i, Pnew_j, pc) \quad (3)$$

$$Pnew = function\ mutation\ (Pnew, pm) \quad (4)$$

4) Replacement การแทนค่าประชากรที่มีค่าเหมาะสมเป็นประชากรในรุ่นต่อไป สมการ (5) ประชากรทั้งหมดใน *Pold* และ *Pnew* จะนำมาพิจารณาเลือกเฉพาะที่มีค่าความเหมาะสมสูงส่งให้แก่ *Pold* ใช้เป็นประชากรรุ่นต่อไป

$$Pold = function\ replacement\ (Pold, Pnew) \quad (5)$$

5) Termination การตรวจสอบการสิ้นสุดการค้นหาผลลัพธ์ การค้นหาผลลัพธ์ใช้ระยะเวลานานหลายรุ่นประชากร ดังนั้นเงื่อนไขที่ตรวจสอบอาจกำหนดได้สองลักษณะคือ เมื่อผลลัพธ์ที่ได้มีค่าสูงกว่าหรือเท่ากับผลลัพธ์ที่ต้องการจึงกำหนดให้หยุดการทำงาน หรือจำนวนรอบการค้นหาใช้ระยะเวลานานถึงค่าที่กำหนดไว้ จึงหยุดการทำงาน

ในส่วนต่อไปแนะนำให้รู้จักส่วนประกอบและการทำงานของ PSO จากนั้นการนำ GA และ PSO มาช่วยค้นหาผลลัพธ์จุดต่ำสุดของฟังก์ชันคณิตศาสตร์

3. Particle Swarm Optimization

ในปี 1995 J.Kenndy and R.Eberthart [4] ได้นำเสนอ PSO โดยมีแนวคิดนำมากจากการศึกษาพฤติกรรมทางสังคมของสิ่งมีชีวิต เช่นลักษณะการเคลื่อนที่ของฝูงนก ฝูงปลา โดยนกหรือปลาแต่ละตัวที่อยู่ในกลุ่ม เรียกว่า อนุภาค (particle) ในกรณีฝูงนก นกแต่ละตัวทำหน้าที่บินค้นหาอาหารและอาจย้ายที่อยู่ไปด้วยพร้อมๆกัน จุดใดที่มีอาหารอยู่ สามารถพักอาศัยได้หรือเป็นจุดที่มีอันตราย นกจะสื่อสารกันในกลุ่มเพื่อแลกเปลี่ยนข้อมูล (Sharing Information) ระหว่างกัน

การบินของนกหากมองเป็นพารามิเตอร์จะประกอบด้วยตำแหน่งที่นกบินอยู่ (position) และความเร็วของการเคลื่อนที่ (velocity) ดังนั้นในหนึ่งอนุภาคเมื่อแทนด้วยพารามิเตอร์ของปัญหาหนึ่งจะประกอบด้วยตัวแปร ดังสมการ (6)

$$P_K = \{X, V, Fitness\ value, Gbest, Pbest\} \quad (6)$$

โดยที่

P คืออนุภาคประกอบด้วย $P = \{P_1, P_2, .., P_K\}$

K คือจำนวนอนุภาค

X คือตำแหน่งที่อนุภาคที่อยู่ในปัญหา เขียนในรูป $X_D = \{x_1, x_2, \ldots, x_D\}$

D คือจำนวนมิติของปัญหา

V คือค่าความเร็วในการเคลื่อนที่ของอนุภาค ประกอบด้วยเวกเตอร์ตามขนาดของมิติของปัญหา $V_D = \{v_1, v_2, \ldots, v_D\}$

Fitness value คือค่าความเหมาะสมของอนุภาคนั้น สามารถคำนวณได้จากฟังก์ชันเป้าหมายซึ่งได้จาก *Fitness value* = *objective function* (X)

$Gbest_D$ คือค่าตำแหน่งที่อนุภาคนั้นได้ค่าความเหมาะสมสูงสุด โดย $Gbest_D = \{Gbest_1, Gbest_2, \ldots, Gbest_D\}$

Pbest คือค่าความเหมาะสมที่ดีสุดที่กลุ่มอนุภาครุ่นปัจจุบันได้มา

การทำงานของ PSO แสดงดังแผนผังในรูปที่ 2 ประกอบด้วยกระบวนการทั้งหมด 7 ขั้นตอน

1) Initial particle การสร้างประชากรเริ่มต้นโดยใช้การสุ่มค่าให้ X และค่าความเร็วการเคลื่อนที่ให้ V ของอนุภาคแต่ละตัว ดังสมการ (7) และ (8) สุ่มค่าพารามิเตอร์ให้ตัวแปรภายของ X และ V ตามจำนวนของมิติของปัญหา ลักษณะนี้คล้ายกับการสุ่มค่าให้แก่ประชากรเริ่มต้นใน GA แต่ PSO จะมีการเก็บค่า V ซึ่งเป็นพารามิเตอร์เสริมคำนวณความเร็วการเคลื่อนที่

$$X_D = function\ random\ value\ (\ ) \quad (7)$$

$$V_D = function\ random\ value\ (\ ) \quad (8)$$

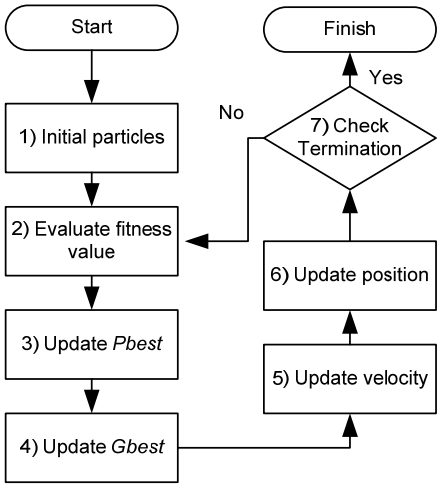
2) Evaluate fitness value การคำนวณค่าความเหมาะสมของอนุภาค ฟังก์ชันความเหมาะสมดังสมการ (9) คำนวณโดยส่งค่า X ให้แก่ฟังก์ชันเป้าหมาย การคำนวณค่าความเหมาะสมนี้มีลักษณะเหมือนกับการคำนวณใน GA

$$fitness\ value = objective\ function(X_D) \quad (9)$$

3) Update *Pbest* การเก็บค่าความเหมาะสมที่ดีสุดซึ่งเป็นค่าความเหมาะสมที่ดีสุดในรอบการทำงานปัจจุบัน มีเงื่อนไขการปรับปรุงตามสมการ (10) ถ้าค่าความเหมาะสมของอนุภาคปัจจุบันมีค่าดีกว่า *Pbest* จะทำการปรับปรุงค่า *Pbest* ลักษณะนี้ใน GA ทั่วไปไม่มีการเก็บประชากรค่าความเหมาะสมสูงสุดของประชากร แต่จะเก็บไว้ในกลุ่มประชากรเก่า

IF *fitness value* > *Pbest* THEN

Pbest = *fitness value* ENDIF (10)



รูปที่ 2. แผนผังการทำงานของ PSO

4) Update *Gbest* การเก็บค่าตำแหน่งให้ผลลัพธ์ดีที่สุด เป็นตำแหน่งที่ได้ผลลัพธ์ที่ดีสุดพิจารณาจากการวนรอบตั้งแต่รอบแรกถึงปัจจุบันซึ่งแตกต่างจาก *Pbest* ที่พิจารณาการเก็บค่าที่ดีสุดเฉพาะรอบปัจจุบัน Gbest จึงเหมือนการเก็บค่าความเหมาะสมที่ดีสุดในประวัติศาสตร์ เงื่อนไขการปรับปรุงดังสมการ (11) ลักษณะการจัดเก็บเช่นนี้ไม่มีใน GA แบบทั่วไป โดย GA ใช้กลุ่มประชากรเก่าเก็บค่าต่อเนื่อง โดยผ่านการคัดเลือกด้วยค่าความเหมาะสมและการอยู่รอดจากอดีตถึงปัจจุบัน

$$\text{IF } \textit{fitness value} > \textit{Gbest} \text{ THEN}$$
$$\textit{Gbest}_D = X_D \text{ ENDIF} \qquad (11)$$

5) Update velocity การคำนวณความเร็วการเคลื่อนที่ของแต่ละอนุภาค ดังสมการ (12)

$$V_D' = \text{weight} * V_D + n_1 * \textit{random}() * (\textit{Pbest} - X_D) + n_2 * \textit{random}() * (\textit{Gbest} - X_D) \qquad (12)$$

โดยที่

V_D' คือ ค่าความเร็วที่คำนวณได้จากอนุภาครุ่นปัจจุบัน

V_D คือ ค่าความเร็วที่คำนวณได้จากอนุภาครุ่นก่อน

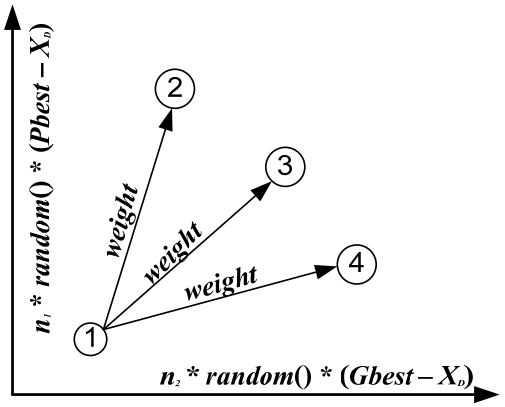
weight คือ ค่าการถ่วงน้ำหนัก

n_1 และ n_2 คือ ค่าคงที่ความเร็วในการค้นหา

6) Update position การปรับปรุงตำแหน่งของแต่ละอนุภาค โดยใช้ผลลัพธ์จากการคำนวณในสมการ (12) ทำให้ได้ตำแหน่งใหม่ที่อนุภาคจะเคลื่อนที่ไป จากนั้นสมการ (13) นำผลลัพธ์มาปรับปรุง ซึ่งการปรับปรุงด้วยการบวกค่าเพิ่มนั้นจำเป็นต้องตรวจสอบขอบเขตของปัญหา ป้องกันไม่ให้ตำแหน่งเกินค่าขอบเขตของปัญหา

$$X_D = X_D + V_D' \qquad (13)$$

7) Termination Check การตรวจสอบการสิ้นสุดการวนรอบ เงื่อนไขที่ตรวจสอบแบ่งได้สองลักษณะคือ เมื่อผลลัพธ์ที่ได้มีค่าที่สูงหรือดีกว่าตำตอบที่ต้องการให้หยุดการทำงาน หรือการค้นหาใช้ระยะเวลานาน มีจำนวนรอบถึงค่าที่กำหนดไว้ จึงหยุดการทำงาน หากยังไม่สิ้นสุดให้กระโดดไปทำงานในลำดับที่ 2



รูปที่ 3. การคำนวณการเคลื่อนที่ของแต่ละอนุภาค

การเคลื่อนที่ของอนุภาคตามการคำนวณในขั้นตอนที่ 5 จากสมการ (12) เมื่อนำมาพิจารณาเขียนเป็นกราฟแสดงการเคลื่อนที่ดังรูปที่ 3 พิจารณาการเคลื่อนที่ในลักษณะหนึ่งมิติจากสมการ ค่าการถ่วงน้ำหนักใช้กำหนดระยะทางในการเคลื่อนที่ ช่วงสมการส่วนที่มี *Pbest* ประกอบอยู่ใช้คำนวณทิศทางตามแนวแกน *y* และ ช่วงสมการส่วนที่มี *Gbest* ประกอบอยู่ใช้คำนวณทิศทางตามแนวแกน *x* เมื่อสมการสามส่วนรวมกัน เราสมมุติให้อนุภาคอยู่ในตำแหน่งที่ 1 หากส่วนของ *Pbest* มีค่าที่ได้จากการสุ่มมากและเป็นทิศทางที่ได้ผลลัพธ์ดี จะทำให้อนุภาคเคลื่อนที่จากตำแหน่งที่ 1 ไปตำแหน่งที่ 2 ถ้าเราสมมุติให้ส่วนของ *Pbest* ได้ผลลัพธ์ด้อยกว่าเดิม สมการในส่วนของ *Gbest* จะดึงให้อนุภาคเคลื่อนที่จากตำแหน่งที่ 1 ไปตำแหน่งที่ 4 จากปัญหาที่สมมุตินี้เปรียบเทียบได้กับ การที่นกเคลื่อนที่จาก 1 ไป 2 แสดงให้เห็นว่านกตัวนี้พบอาหารมากขึ้นในตำแหน่งที่ 2 และอาหารมีมากกว่าที่นกในกลุ่มบอกกัน แต่หากนกเคลื่อนที่จาก 1 ไป 4 แสดงให้เห็นว่านกตัวนี้เคลื่อนลักษณะย้ายจากตำแหน่งที่ในนกในรุ่นบอกไปยังตำแหน่งเดิมเพราะตำแหน่งใหม่มีอาหารลดลง

4. เปรียบเทียบวิธีการค้นหาของ GA, PSO และ Local search

PSO มีส่วนที่คล้ายกับ GA โดยที่ PSO แทนปัญหาให้อยู่ในรูปอนุภาค การเก็บพารามิเตอร์ของปัญหาคล้ายกับ GA ที่ใช้ประชากรและโครโมโซม ด้านการค้นหาผลลัพธ์ PSO ใช้การสุ่มตำแหน่งลักษณะการสุ่มจึงคล้ายกับกระบวนการทางพันธุกรรมของ GA อย่างไรก็ตามส่วนที่ PSO แตกต่างจาก GA คือ PSO มีการคำนวณความเร็วทิศทางการเคลื่อนที่ โดยอาศัย *V*, *Pbest* และ *Gbest* เพื่อใช้ในการการแลกเปลี่ยนข้อมูลของทิศทางการเคลื่อนที่ระหว่างอนุภาค อย่างไรก็ตามข้อมูลที่ใช้กำหนดทิศทางของกลุ่มอนุภาคยังคงมีฟังก์ชันสุ่มการทำงาน ซึ่งจะทำให้ปริมาณเปลี่ยนแปลงตามโอกาสที่สุ่ม ซึ่งลักษณะการสุ่มที่ผสมเข้ามานี้จะทำให้ PSO สามารถหลุดจากการที่กลุ่มอนุภาคลู่เข้าสู่โลคอลออปติมัมได้

ลักษณะพิเศษของ PSO ที่ใช้การคำนวณการหาเคลื่อนย้ายอนุภาค โดยการเก็บข้อมูลของกลุ่มคือ *Gbest* และของรุ่น *Pbest* คำนวณทิศทางและความเร็วการเคลื่อนที่มีลักษณะไม่เหมือนกับการคำนวณด้วยวิธีการ์เด็น (gradient method) หรือควอนไซนิวตัน (quasi-Newton method) [8] ซึ่งเป็นวิธีที่ใช้หลักการของการลู่ การคำนวณจากทิศทางการลาดเข้าสู่ค่าโลคอล อย่างไรก็ตามหลักการเคลื่อนย้ายอาจจะมีลักษณะหลักการคล้ายกับวิธีนิวเดอเมท (Nelder-Mead method) [9] ที่ทำงานโดยใช้การคำนวณเวกเตอร์โดยใช้ทิศทางการเคลื่อนที่ สามจุดกำหนดตำแหน่งใหม่ที่ต้องการเคลื่อนที่ไป

ข้อสังเกตการเคลื่อนทิศทางเคลื่อนที่ของ PSO โดยการนำตำแหน่งที่ได้ผลลัพธ์ดีสุดมาคำนวณอาจเกิดปัญหาได้ [10-13] เพราะความหลากหลายของประชากรที่จำกัด และโดยบังคับด้วยทิศทางเดิมที่ได้ผลลัพธ์ดี ย่มทำให้ความหลากหลายของประชากรลดลง เปรียบได้กับอนุภาคอยู่ในกลุ่มกันค่อนข้างแน่น ความหลากหลายของประชากรที่ลดลงนี้เป็นปัญหาที่สำคัญในการค้นหาผลลัพธ์ของวิธีการในลักษณะการสุ่ม (stochastic optimization) การที่ประชากรไม่มีความหลากหลายจะเป็นผลทำให้ได้มีโอกาสได้ผลลัพธ์ในลักษณะโลคอลมีสูง อย่างก็ดีปัญหาเหล่านี้ในกรณีของ GA หากมีการกำหนดพารามิเตอร์โอกาสการแลกเปลี่ยนยีน โอกาสการกลายพันธ์ และการรูปแบบการคัดเลือกประชากรที่ไม่เหมาะสม ก็จะเกิดการกระจุกตัวของประชากรเช่นเดียวกัน

5. ทดสอบ GA และ PSO ค้นหาผลลัพธ์ของฟังก์ชันทางคณิตศาสตร์

การเปรียบเทียบความสามารถของการค้นหาผลลัพธ์ด้วย GA และ PSO นักวิจัยส่วนมากนิยมทดสอบกับฟังก์ชันทางคณิตศาสตร์ ซึ่งเป็นฟังก์ชันที่รู้ค่าที่ดีสุดและตำแหน่งที่ได้ผลลัพธ์ดีที่สุด การวัดผลหากวิธีการใดค้นหาผลลัพธ์ที่ดีสุดหรือใกล้เคียงมากที่สุดในเวลาที่จำกัดถือว่าเป็นวิธีการที่ให้ผลลัพธ์ได้อย่างรวดเร็ว อย่างไรก็ตามฟังก์ชันทางคณิตศาสตร์มีลักษณะความซับซ้อนแตกต่างกัน บางฟังก์ชันตอบสนองได้ดีต่อปัญหาแบบหนึ่ง แต่เมื่อทดสอบพบกับปัญหาอีกรูปแบบอาจจะมีการตอบสนองด้อยกว่าเดิม

การทดสอบการทำงานของ PSO กับ GA ใช้ฟังก์ชันทางคณิตศาสตร์พื้นฐาน [12] ซึ่งมีฟังก์ชันที่ทดสอบจำนวนมาก บทความนี้นำมาสองฟังก์ชันที่ลักษณะความซับซ้อนแตกต่างกันคือ ฟังก์ชัน Sphere ดังรูปที่ 4 และฟังก์ชัน Peaks ดังรูปที่ 5

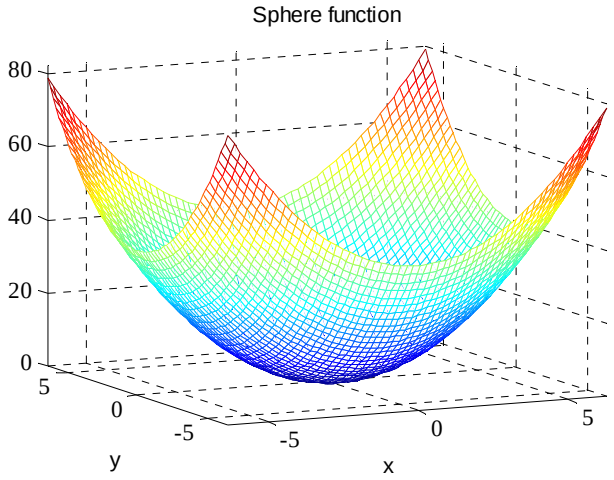
ฟังก์ชัน Sphere มีลักษณะส่วนโค้งลาดเอียงลู่เข้าสู่ตำแหน่งที่ต่ำสุดที่ x = 0 และ y =0 มีค่า z น้อยที่สุดที่ 0 สมการของฟังก์ชันนี้แสดงดังสมการที่ (14)

$$f(z) = x^2 + y^2 \tag{14}$$

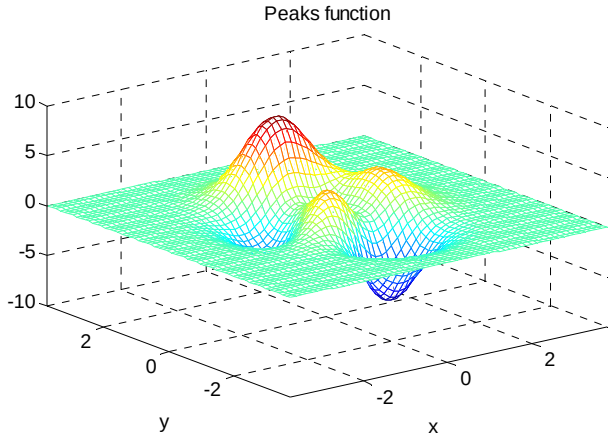
ฟังก์ชัน Peaks [14] มีลักษณะส่วนสูงต่ำสลับไปมา มีความซับซ้อนมากว่าฟังก์ชันแรก Peaks มีส่วนที่เป็นโลคอลออปติมัม 2 ตำแหน่งและส่วนที่เป็นโกบอลออปติมัม 1 อยู่ในตำแหน่งที่ x = -0.5 และ y = -1.5 มีค่า z น้อยที่สุดที่ประมาณ -6 สมการของฟังก์ชันนี้แสดงดังสมการที่ (15)

$$f(z) = 3 * (1- x)^2 \, exp(-x^2 - (y+1)^2) - 10 *(x/5 - x^3 - y^5) * exp(-x^2-y^2) - 1/3 * exp(-(x+1)^2-y^2) \tag{15}$$

การแทนปัญหาให้ GA และ PSO กำหนดให้โครโมโซมหรือพารามิเตอร์ภายในอนุภาคเป็นแบบเลขจำนวนจริง เพื่อให้ GA คำนวณค่าความเหมาะสมเร็วยิ่งขึ้น หากใช้ไบนารีโครโมโซมจะทำให้เสียเวลาในการแปลงผลลัพธ์ ฟังก์ชันเป้าหมายกำหนดได้ดังสมการที่ 14 และ 15



รูปที่ 4. ฟังก์ชัน Sphere



รูปที่ 5. ฟังก์ชัน Peaks

การทดสอบใช้ GA และ PSO ค้นหาผลลัพธ์ของฟังก์ชัน Sphere และ Peaks เมื่อจำเป็นต้องวันประสิทธิภาพจำเป็นต้องทดสอบที่เวลาการคำนวณเท่ากัน การทดสอบนี้กำหนดที่เวลา 2 วินาที และกำหนดให้ปัญหามีความซับซ้อนที่ 10 มิติโดยการเพิ่มสมการจากเดิมที่มีสองมิติ ใช้พารามิเตอร์สองตัว เพิ่มจำนวนสมการเป็นห้าเท่า และนำผลลัพธ์ทั้งหมดมารวมกันหาค่าเฉลี่ย การโปรแกรมเขียนด้วยภาษา Matlab ผลลัพธ์ที่ได้จากการทดสอบแสดงดังตารางที่ 1 ในตารางตัวเลขที่ขีดเส้นใต้คือค่าที่เข้าใกล้ตำแหน่งออปติมัม ผลลัพธ์พบว่า GA ค้นหาผลลัพธ์ของฟังก์ชัน Sphere ช้ากว่า PSO ซึ่งหาผลลัพธ์ลู่เข้าอย่างเร็วมาก แต่ในกรณีของ Peaks ฟังก์ชันแล้ว PSO ค้นหาผลลัพธ์ช้ากว่า GA ซึ่งสอดคล้องกับ [13] โดย PSO จะมีปัญหาเกี่ยวกับการเกาะกลุ่มของประชากรอยู่ในตำแหน่งโลคอลออปติมัม ดังนั้น PSO จำเป็นต้องอาศัยการเพิ่มความแตกต่างของอนุภาคเพื่อให้เกิดการกระจายในการค้นหาผลลัพธ์ได้มีประสิทธิภาพยิ่งขึ้น

ตาราง 1. ผลการทดสอบ GA และ PSO ด้วยฟังก์ชัน Sphere และ Peaks

ฟังก์ชันคณิตศาสตร์	GA	PSO
Sphere	3E-5	8.3E-100
Peaks	-5.08	-4.5

6. บทสรุป

บทความนี้เสนอการทำงานของ GA และ PSO ซึ่งทั้งสองวิธีจัดอยู่ในกลุ่มทฤษฎีวิวัฒนาการ วิธีการทำงานจะเห็นได้ว่า PSO มีคุณสมบัติที่เหนือกว่า GA โดย PSO สามารถนำตำแหน่งที่ได้ผลลัพธ์ดีสุดในรุ่นและตำแหน่งที่ได้ผลลัพธ์ดีที่สุดจากอดีตถึงปัจจุบัน เป็นข้อมูลแบ่งปันให้แต่ละอนุภาคใช้ประกอบการคำนวณทิศทางการเคลื่อนที่ ซึ่งแตกต่างจาก GA ที่มีการแบ่งข้อมูลของตำแหน่งระหว่างประชากรในขอบเขตที่จำกัด การทดสอบการทำงานสะท้อนให้เห็นว่าวิธีการทั้งสองสามารถลู่เข้าสู่จุดที่ได้ผลลัพธ์ที่ดีได้

เอกสารอ้างอิง

[1] John H. Holland, “Adaptation in Natural and Artificial System, University of Michigan press, Ann Arbor, 1979

[2] M. Srinivas, L. M. Patnaik, “Genetic algorithm: a survey”, IEEE computer society, Vol. 27, pp. 17-26, 1994

[3] Carlos A. Coello, “An Updated Survey of GA-Based Multiobjective Optimization Techniques”, ACM Computing Survey (CSUR), Vol. 32, June 2000

[4] Kennedy J. and Eberthart R. “Particle Swarm Optimization”, Proc. IEEE Int. Conf. Neural Network, Vol. 4, pp 1942-1948, 1995

[5] Cezary Z. Janikow and Zbigniew Michalewicz, “An Experimental Comparison of Binary and Floating Point Representation in Genetic Algorithm”, Proceeding of the 4th International Conference on Genetic Algorithm, pp. 31-36, 1991

[6] David E. Goldberg, “Genetic Algorithm in Search, Optimization and Machine Learning”, Addison Wesley, New York, 1989

[7] Z. Michalewicz, “Genetic Algorithm + Data Structures = Evolution Programs”, Springer-Verlag, New York, 1992

[8] P. E. Gill and W. Murray, “Quasi-Newton method for Unconstrained Optimization”, IMA Journal of Applied Mathematics, Vol. 9, pp. 91-108, 1972

[9] Nelder J. A. and Mead R. , “A simplex method for function optimization”, The Computer Journal, Vol.7, pp. 308-313, 1965

[10] A. Ratnawera, S. K. Halgamuge, and H. C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Transaction on Evolutionary Computation, Vol.8, No.3, pp. 240-255, 2004

[11] Angline P. J., “Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference,” The 7th Annual Conference on Evolutionary Programming, San Diego, USA 1998

[12] J. Vesterstrom, R. Thomsen, “A Comparative study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithm on Numerical Benchmark Problem,” in Proc. IEEE Congr Evolutionary Computation, Protland, pp. 1980-1987, 2004

[13] Deniel W. Boeringer and Douglas H. Werner, “Particle Swarm Optimization Versus Genetic Algorithms for Phased Array Synthesis”, IEEE Trans. On Antennas and Proparation, Vol. 52, No. 3, pp 771-779, March 2004

[14] The Mathworks, “Genetic Algorithm and Direct Search Toolbox User’s Guide”, The Mathworks Inc, p.222, 2005

ภาคผนวก

ตัวอย่างโปรแกรม PSO พัฒนาด้วย Matlab

```
%parameter
dimension = 10;
weight = 0.729844;
swarmsize = 10;
c1 = 1.496180;
c2 = 1.496180;
maxgen = 10000;

%inital parameters
X = zeros(10,dimension);
V = zeros(10,dimension);
fitness = zeros(1,10);
pbest = zeros(10,dimension);
pbest_fit = zeros(1,10);
```

```
bestfit = 1000000.0;
gbest = zeros(1,dimension);

%function sphere
lowli = -5.12;%lower limit
upli = 5.12;  %upper limit
vmax = upli-lowli;

for i = 1:swarmsize
    %initail particle
    for j = 1:dimension
      X(i,j)=lowli+(rand()/2 )*vmax;
      V(i,j)=-vmax+(rand()/2 )*(2*vmax);
      pbest(i,j) = X(i,j);
    end


%evaluate fitness value
[fit(i),bestfit] = evaluate(X,i,dimension,
gbest, bestfit);
pbest_fit(i) = fit(i);
end
for g = 1:maxgen
   for i = 1:swarmsize
       for j = 1:dimension
          %update velocity
          r1 = rand()/2;
          r2 = rand()/2;
          V(i,j)=weight * V(i,j)+ c1*r1 *
(pbest(i,j)-X(i,j)) + c2*r2 * (gbest(j) -
X(i,j));
          %check boundary
          if V(i,j) > vmax
             V(i,j) = vmax;
          end
          if V(i,j) < -vmax
             V(i,j) = -vmax;
          end

          %update position
          X(i,j) = X(i,j) + V(i,j);
          if X(i,j) > upli
             X(i,j) = upli - 0.001;
             V(i,j) = 0;
          end
          if X(i,j) < lowli
             X(i,j) = lowli + 0.001;
             V(i,j) = 0;
          end
     end

%evaluate & update gbest
[fit(i),bestfit,gbest]  =  evaluate(X,  i,
dimension, gbest, bestfit);
     %update pbest
     if fit(i) < pbest_fit(i)
        for j = 1:dimension
            pbest(i,j) = X(i,j);
        end
        pbest_fit(i) = fit(i);
      end
    end
    disp(bestfit)
end

function[fitness,bestfit,gbest]=
evaluate(X, i, dimension, gbest, bestfit)
%--- Sphere function
    sum = 0;
    for j = 1:dimension
        sum = sum + X(i,j)^2;
    end
    fitness = sum;
    %update gbest
    if fitness < bestfit
        bestfit = fitness;
        for j = 1:dimension
            gbest(j) = X(i,j);
        end
    end
end
```