การเปรียบเทียบประสิทธิภาพระหว่างอัลกอริทึมอาณานิคมผึ้งเทียม กับวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค

A COMPARISON BETWEEN ARTIFICIAL BEE COLONY ALGORITHM AND PARTICLE SWARM OPTIMIZATION

ลัคนา เพิ่มพูล* และ จักรพันธ์ ปิ่นทอง
Lukkana Poempool* and Jakaphan Pinthong

สาขาวิศวกรรมการจัดการอุตสาหกรรม คณะเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยราชภัฏราชนครินทร์

บทคัดย่อ

อัลกอริทึมอาณานิคมผึ้งเทียมและวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคเป็นหนึ่งในความฉลาดแบบกลุ่มที่เป็นที่นิยมนำมาใช้ในการหาค่าความเหมาะสมทั้งสอง อัลกอริทึมเป็นอัลกอริทึมที่เลียนแบบมาจากพฤติกรรมการหาอาหารของฝูงสัตว์ บทความนี้ได้นำเสนอเทคนิคของอัลกอริทึมอาณานิคมผึ้งเทียมและวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคและนำมาเปรียบเทียบความเหมือนและความต่างพร้อมวัดประสิทธิภาพในการหาค่าที่ดีที่สุดผ่านฟังก์ชันมาตรฐานทั้งหมด 8 ฟังก์ชัน จากการทดลองพบว่าอัลกอริทึมอาณานิคมผึ้งเทียมเหมาะกับปัญหาแบบพหุฐานนิยม ขณะที่วิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคเหมาะกับปัญหาแบบฐานนิยมเดียว

คำสำคัญ: อัลกอริทึมอาณานิคมผึ้งเทียม, วิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค, การหาค่าที่เหมาะสมที่สุด, ฟังก์ชันมาตรฐาน

* ผู้ประสานงาน: ลัคนา เพิ่มพูล
อีเมล์: lukkana.po@gmail.com

Abstract

Artificial Bee Colony Algorithm (ABC) and Particle Swarm Optimization are among popular swarm intelligences for tackling optimization. Both algorithms mimic the behaviors of social swarms' foraging behaviors in nature. ABC mimics the forging behavior of colonies of ants and PSO mimics the forging behavior of flocks of birds. This article presents and compares performances between PSO and ABC with different eight benchmark test functions. The results show that PSO is suitable for solving unimodal problems while ABC is most suitable for multimodal problems.

Keywords: Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), Optimization, Benchmark Test Functions

บทนำ

ความฉลาดแบบกลุ่ม หรือ Swarm Intelligent (บุญเจริญ ศิริเนาวกุล, 2555) เป็นอัลกอริทึมพื้นฐานที่เป็นที่นิยมในปัจจุบันในการนำมาใช้เพื่อแก้ปัญหาที่ความความซับซ้อนทั้งในด้านวิศวกรรมศาสตร์ อุตสาหกรรม การจัดการ วิทยาศาสตร์ และอื่น ๆ อีกมากมาย โดยอัลกอริทึมเหล่านี้ได้รับแรงบันดาลใจมาจากปรากฏการณ์ทางธรรมชาติ หรือ Nature Inspired Algorithms (NIAs) (Sindhya, 2012) ทั้งนี้อัลกอริทึมอาณานิคมผึ้งเทียม (Karaboga, 2005) และวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค (Eberhart & Kennedy, 1995) เป็นความฉลาดแบบกลุ่มที่ได้แรงบันดาลใจมาจากธรรมชาติและเป็นอัลกอริทึมที่เป็นที่นิยมนำมาใช้ในการแก้ปัญหาเพื่อหาค่าคำตอบที่ดีที่สุด โดยทั้งสองวิธีนี้เป็นอัลกอริทึมที่เลียนแบบมาจากพฤติกรรมการหาอาหารของฝูงสัตว์

วิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคถูกพัฒนา โดย Eberhart & Kennedy ในปี 1995 โดยมีแนวคิดมาจากพฤติกรรมการหาอาหารของฝูงนก (Particles) ที่อาศัยการเคลื่อนที่กันเป็นกลุ่ม ฝูงนกเหล่านี้จะมีการส่งสัญญาณระหว่างกันเพื่อแลกเปลี่ยนสื่อสารถึงตำแหน่งที่มีอาหารอยู่ และทำการเคลื่อนที่ไปยังแหล่งอาหารที่ดีที่สุดที่ได้รับข้อมูลมา โดยนกแต่ละตัวจะทำการปรับปรุงตำแหน่งและทิศทางไปเรื่อย ๆ เพื่อค้นหาค่าคำตอบที่ดีที่สุด

อัลกอริทึมอาณานิคมผึ้งเทียม ถูกพัฒนาโดย Karaboga ในปี 2005 ได้รับแรงบันดาลใจมาจากพฤติกรรมการหาน้ำหวานของผึ้ง (Honey Bees) อัลกอริทึมนี้จะแบ่งผึ้งออกเป็นกลุ่มๆ เพื่อช่วยในการหาค่าคำตอบที่ดีที่สุด โดยจะส่งผึ้งงาน (Employed Bees) ไปหาแหล่งอาหารซึ่งแทนด้วยคำตอบที่เป็นไปได้ของปัญหา จากนั้นผึ้งงานจะกลับมารายงานปริมาณน้ำหวานของดอกไม้ในแหล่งอาหารที่พวกเขาค้นพบกับผึ้งเฝ้าดู (Onlooker Bees) ในรังของพวกเขา ทั้งนี้ผึ้งเฝ้าดูจะทำการตรวจสอบแหล่งอาหารที่ถูกเลือกและนำไปเปรียบเทียบกับแหล่งอาหารบริเวณใกล้เคียงด้วย อัลกอริทึมนี้ยังคงปรับปรุงความสามารถในการสำรวจเพื่อหาค่าคำตอบที่ดีที่สุด (Optimization Algorithm) โดยการใช้ผึ้งสำรวจ (Scout Bees) ผึ้งสำรวจจะแนะนำกระบวนการการกลายพันธุ์ให้อัลกอริทึมโดยการค้นหาแหล่งอาหารใหม่ในพื้นที่ที่ไม่เคยสำรวจมาก่อนของพื้นที่สำรวจเพื่อเพิ่มโอกาสในการค้นหาแหล่งอาหารให้กว้างขึ้น

แม้ว่าวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคเป็นหนึ่งในวิธีการที่ดีและนิยมใช้ในการหาคำตอบได้ดีและเร็วกว่าอัลกอริทึมอื่นมาก แต่ก็มีปัญหาในการลู่เข้าก่อนกำหนดซึ่งเป็นสาเหตุหลักที่ทำให้ฝูงนกติดอยู่ในค่าคำตอบที่ดีที่สุดเฉพาะที่ (Local Optima) และไม่สามารถกระโดดออกไปค้นหาในตำแหน่งอื่นได้ ในขณะที่อัลกอริทึมอาณานิคมผึ้งเทียมได้เพิ่มความสามารถในการค้นหาแหล่งอาหารให้กว้างขึ้นโดยการทำงานของผึ้งสำรวจซึ่งเป็นการแก้ปัญหาการติดในค่าคำตอบที่ดีที่สุดเฉพาะที่ (Sharma & Bhambu, 2016)

งานวิจัยนี้ทำการศึกษาการทำงานของอัลกอริทึมอาณานิคมผึ้งเทียมและวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค พร้อมทั้งทดสอบประสิทธิภาพของทั้ง 2 อัลกอริทึมนี้บนฟังก์ชันมาตรฐาน (Benchmark Test Function) 8 ฟังก์ชัน (Molga & Smutnicki, 2005; Hedar, 2018 ; Surjanovic & Bingham, 2017) โดยในบทความนี้จะแบ่งเป็น 5 ส่วน ส่วนแรกจะพูดถึงวิวัฒนาการและภาพรวมของอัลกอริทึม ส่วนที่ 2 จะอธิบายถึงรายละเอียดการทำงานของอัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค ส่วนที่ 3 อธิบายถึงรายละเอียดการทำงานของอัลกอริทึมอาณานิคมผึ้งเทียม ส่วนที่ 4 อธิบายถึงฟังก์ชันมาตรฐานที่นำมาใช้ในการทดสอบ วิธีการทดสอบ อัลกอริทึมที่เลือกมาทดสอบ และผลการทดสอบ ส่วนสุดท้ายจะสรุปผลการทดสอบของงานวิจัยนี้

วิธีดำเนินการวิจัย

1. วิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค (Particle Swarm Optimization: PSO)

วิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค หรือ Particle Swarm Optimization (PSO) เป็นหนึ่งในอัลกอริทึมที่เป็นที่นิยมในการนำมาใช้ในการแก้ปัญหาการหาค่าที่เหมาะสมที่สุด ซึ่งได้แนวคิดมาจากการเลียนแบบพฤติกรรมทางสังคมของฝูงสัตว์ในการหาอาหาร เช่น ฝูงนก หรือฝูงปลา ที่นิยมออกหาอาหารเป็นฝูง โดยนกที่อยู่ใกล้แหล่งอาหารมากที่สุดจะถูกเลือกให้เป็นจ่าฝูง นกที่เหลือจะเคลื่อนที่หรือบินตามจ่าฝูงในการหาอาหารนั่นเอง ในวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคจะเรียกแทนนกแต่ละตัวว่าอนุภาค (Particles) แต่ละอนุภาคจะถูกแทนด้วยความเร็วและตำแหน่ง ทั้งนี้วิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคเริ่มต้นทำงานด้วยการสุ่มอนุภาคลงบนพื้นที่การค้นหา (Search Space) เพื่อสร้างประชากรเริ่มต้น จากนั้นจะทำการเคลื่อนที่บนพื้นที่การค้นหาตามค่าตำแหน่งที่ดีที่สุดของเพื่อนบ้าน (GBEST) และของตัวเองในรอบก่อนหน้า (PBEST) โดยกระบวนการของวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคจะทำการปรับปรุงและค้นค่าคำตอบไปเรื่อย ๆ จนกว่าจะพบค่าคำตอบที่เหมาะสมที่สุด หรือครบรอบที่กำหนดไว้ ค่าตำแหน่งที่ดีที่สุดของเพื่อนบ้านที่ได้ในรอบสุดท้ายจะถูกนำมาใช้เป็นคำตอบของวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค

โดยขั้นตอนการทำงานของวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคอธิบายได้ดังต่อไปนี้

ขั้นตอนที่ 1 ทำการกำหนดค่าเริ่มต้นให้อนุภาค (Initial Particles) ด้วยการสุ่มตำแหน่งและความเร็วเริ่มต้นของอนุภาคทุกตัวลงในพื้นที่การค้นหา ในขั้นตอนนี้จะทำเพียงครั้งเดียวเท่านั้น

ขั้นตอนที่ 2 การประเมินค่าคำตอบของแต่ละอนุภาค (Evaluate) คือ การคำนวณค่าความเหมาะสม (Fitness Value) ผ่านฟังก์ชันวัตถุประสงค์ (Objective Function) เพื่อนำไปใช้ในการปรับปรุงค่าตำแหน่งที่ดีที่สุดที่อนุภาคตัวนั้นเคยพบเจอ (PBEST) และตำแหน่งที่ดีที่สุดที่อนุภาคทุกตัวเคยพบเจอ (GBEST) ในขั้นตอนที่ 3 และ 4 ตามลำดับ

ขั้นตอนที่ 3 การปรับปรุงค่า PBEST (Update PBEST) คือ การที่แต่ละอนุภาคจะทำการเปรียบเทียบค่าความเหมาะสมของอนุภาคกับค่า PBEST ของตัวเอง หากค่าความ

เหมาะสมของอนุภาคไหนดีกว่าจะทำการปรับปรุงค่า PBEST ของตัวเองตามค่าความเหมาะสมของอนุภาคที่ดีกว่า

ขั้นตอนที่ 4 การปรับปรุงค่า GBEST (Update GBEST) คือ การที่แต่ละอนุภาคจะทำการเปรียบเทียบค่าความเหมาะสมของตัวเองกับค่า GBEST ของกลุ่ม หากค่าความเหมาะสมของอนุภาคไหนดีกว่าจะทำการปรับปรุงค่า GBEST ของกลุ่มตามค่าความเหมาะสมของอนุภาคที่ดีกว่า

ขั้นตอนที่ 5 การปรับปรุงความเร็ว (Update Velocity) คือ การปรับปรุงความเร็วของแต่ละอนุภาคเพื่อใช้ในการเคลื่อนที่อนุภาค ด้วยการนำค่า PBEST, GBEST และค่าความเร็วเดิมมาใช้ในการคำนวณ ดังสมการที่ 1

$$\vec{v}_k^{t+1} = \omega\vec{v}_k^t + c_1 r_1(pbest - \vec{x}_k^t) + c_2 r_2(gbest - \vec{x}_k^t) \qquad (1)$$

โดยกำหนดให้

$\vec{v}_k^{t+1}$ คือ ความเร็วในรอบปัจจุบันของอนุภาค
$\vec{v}_k^t$ คือ ความเร็วเดิมของอนุภาค
$pbest$ คือ ตำแหน่งที่ดีที่สุดที่อนุภาคตัวนั้นเคยพบเจอ
$gbest$ คือ ตำแหน่งที่ดีที่สุดที่อนุภาคทุกตัวเคยพบเจอ
c_1, c_2 คือ ค่าคงที่ในการเรียนรู้
ω คือ ค่าการถ่วงน้ำหนัก
r_1, r_2 คือ ค่าที่ได้จากการสุ่มในช่วง [0, 1]
$\vec{x}_k^t$ คือ ตำแหน่งปัจจุบันของอนุภาค

ขั้นตอนที่ 6 การปรับปรุงตำแหน่ง (Update Position) คือ การปรับปรุงตำแหน่งของอนุภาคให้เคลื่อนที่จากตำแหน่งเดิมไปยังตำแหน่งใหม่ โดยนำค่าที่ได้จากการปรับปรุงความเร็วในขั้นตอนที่ 5 มาใช้ในการคำนวณตามสมการข้างล่าง

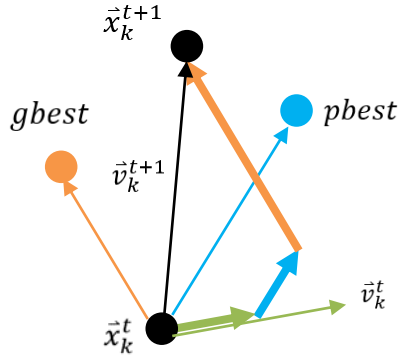
$$\vec{x}_k^{t+1} = \vec{x}_k^t + \vec{v}_k^{t+1} \quad (2)$$

โดยกำหนดให้

$\vec{x}_k^t$ คือ ตำแหน่งปัจจุบันของอนุภาค

$\vec{x}_k^{t+1}$ คือ ตำแหน่งใหม่ของอนุภาค

$\vec{v}_k^{t+1}$ คือ ความเร็วในรอบปัจจุบันของอนุภาค



รูปที่ 1 รูปแสดงการปรับปรุงตำแหน่งและความเร็วของอนุภาคของวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคในรูปแบบ 2 มิติ

ขั้นตอนที่ 7 การตรวจสอบค่าคำตอบหรือค่าความเหมาะสม (Condition Satisfied) ขั้นตอนนี้เป็นการตรวจสอบคำตอบและกำหนดเงื่อนไขในการหยุดค้นหาคำตอบ ถ้าเจอคำตอบหรือครบรอบในการค้นหาคำตอบที่กำหนดให้หยุดการวนซ้ำการทำงาน แต่ถ้าไม่ให้วนซ้ำการทำงานโดยกลับไปทำงานในขั้นตอนที่ 2 นั่นเอง

2. อัลกอริทึมอาณานิคมผึ้งเทียม (Artificial Bee Colony :ABC)

อัลกอริทึมอาณานิคมผึ้งเทียม หรือ Artificial Bee Colony (ABC) ได้แนวคิดมาจากพฤติกรรมของผึ้งในการหาน้ำหวาน โดยในฝูงผึ้งจะแบ่งผึ้งออกเป็น 3 ประเภท คือ ผึ้งงาน (Employed Bee) ผึ้งเฝ้าดู (Onlooker Bee) และผึ้งสำรวจ (Scout Bee) โดยผึ้งงานจะทำหน้าที่ค้นหาแหล่งอาหารและกลับมารายงานปริมาณน้ำหวานในแหล่งอาหารที่ค้นพบกับผึ้งเฝ้าดู จากนั้นผึ้งเฝ้าดูจะเลือกแหล่งอาหารจากข้อมูลของผึ้งงาน ยิ่งแหล่งอาหาร

ใดมีปริมาณน้ำหวานมากก็มีความน่าจะเป็นสูงที่ผึ้งเฝ้าดูจะเลือกแหล่งอาหารนั้นพร้อมทั้งทำการเปรียบเทียบกับแหล่งอาหารอื่น ๆ ในบริเวณใกล้เคียง ขณะที่แหล่งอาหารที่ไม่ได้รับการถูกเลือกจะถูกทิ้งและเปลี่ยนหน้าที่จากผึ้งงานไปเป็นผึ้งสำรวจเพื่อค้นหาแหล่งอาหารใหม่ๆ โดยตำแหน่งของแหล่งอาหารคือค่าคำตอบที่เป็นไปได้ จำนวนของผึ้งงานรวมกับผึ้งสำรวจจะเท่ากับจำนวนของคำตอบที่เป็นไปได้ทั้งหมด (Karaboga & Akay, 2009) โดยกระบวนการค้นหาแหล่งอาหารของผึ้งอธิบายได้ดังต่อไปนี้

ขั้นตอนที่ 1 การสร้างประชากรเริ่มต้นให้กับผึ้งทั้งหมดด้วยการสุ่ม (Initial Phase) ตามสมการที่ (3) พร้อมทั้งคำนวณหาค่าความเหมาะสมของแหล่งอาหารตามสมการที่ (4)

$$\vec{x}_{ij} = \vec{x}_{min\ j} + rand[0,1](\vec{x}_{max\ j} - \vec{x}_{min\ j}) \quad (3)$$

$$fit_{ij}(\vec{x}) = \begin{cases} \frac{1}{1+f_i(\vec{x}_i)} \\ 1 + |f_i(\vec{x}_i)| \end{cases} \quad (4)$$

โดยกำหนดให้

$\vec{x}$ คือ เวกเตอร์ของคำตอบที่ได้จากการสุ่ม

i คือ ขนาดของประชากรทั้งหมด

j คือ จำนวนของพารามิเตอร์ทั้งหมด (มิติ)

$f_i(\vec{x}_i)$ คือ ค่าฟังก์ชันวัตถุประสงค์ของ x

$fit_{ij}(\vec{x})$ คือ ค่าความเหมาะสมของแหล่งอาหาร

ขั้นตอนที่ 2 การทำงานของผึ้งงาน (Employed Bee Phase) ผึ้งงานจะทำการค้นหาแหล่งอาหารใหม่ตามสมการที่ (5) แล้วคำนวณค่าความเหมาะสมตามสมการที่ (4) หากค่าตำแหน่งใหม่ที่คำนวณได้ดีกว่าเดิมจะทำการปรับปรุงตำแหน่งตามค่าใหม่

$$\vec{v}_{ij} = \vec{x}_{ij} + \emptyset \times x_{ij} - x_{rj} \quad (5)$$

โดยกำหนดให้

v_{ij} คือ ตำแหน่งของแหล่งอาหารใหม่

$\emptyset$ คือ ค่าที่ได้จากการสุ่มในช่วง 0 – 1

ขั้นตอนที่ 3 การทำงานของผึ้งเฝ้าดู (Onlooker Bee Phase) ผึ้งเฝ้าดูจะทำการพิจารณาแหล่งอาหารที่ได้จากผึ้งงานจากค่าความน่าจะเป็นที่หาได้จากสมการที่ (6) หากแหล่งอาหารนั้นมีค่าความน่าจะเป็นมากก็มีโอกาสมากที่จะถูกเลือก จากนั้นผึ้งเฝ้าดูจะนำข้อมูลที่เลือกมาทำการคำนวณเพื่อค้นหาแหล่งอาหารที่มีความเหมาะสมมากกว่าเช่นเดียวกับผึ้งงาน

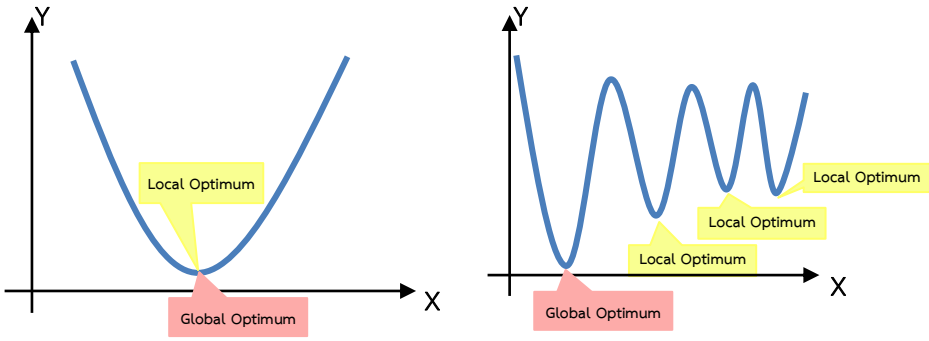
$$P_i = \frac{fit_i(x_i)}{\sum_{i=1}^{Sn} fit_i(x_i)} \quad (6)$$

ขั้นตอนที่ 4 การทำงานของผึ้งสำรวจ (Scout Bee Phase) เริ่มต้นเมื่อแหล่งอาหารเดิมของผึ้งงานไม่ถูกเลือกจากผึ้งเฝ้าดูครบจำนวนที่กำหนดไว้ ผึ้งสำรวจจะทำการคำนวณหาแหล่งอาหารใหม่ด้วยการสุ่มมาแทนที่แหล่งอาหารเดิมที่ไม่ถูกเลือก

ขั้นตอนที่ 5 ขั้นตอนการหยุด (Condition Satisfied) หากผึ้งเจอแหล่งอาหารที่ดีที่สุดหรือครบรอบที่กำหนดจะทำการหยุดการค้นหา แต่ถ้าไม่จะกลับไปทำขั้นตอนที่ 2 อีกครั้ง

3. การทดสอบและวัดประสิทธิภาพ

งานวิจัยนี้ได้ทำการเปรียบเทียบและวัดประสิทธิภาพระหว่างวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคและอัลกอริทึมอาณานิคมผึ้งเทียมผ่านชุดฟังก์ชันมาตรฐานเชิงตัวเลขทั้งหมด 8 ฟังก์ชันซึ่งเป็นตัวแทนของลักษณะของปัญหาที่หลากหลาย ลักษณะของปัญหาที่นำมาทดสอบมีทั้งแบบฐานนิยมเดียวและแบบพหุฐานนิยม โดยที่ปัญหาแบบฐานนิยมเดียว คือ ปัญหาการหาค่าที่เหมาะสมที่สุดบนพื้นที่การค้นหาที่มีเพียงจุดค่าคำตอบหรือค่าที่ดีที่สุดที่แท้จริง (Global Optimum) เพียงจุดเดียวเท่านั้น หรืออาจกล่าวได้ว่า จุดค่าที่ดีที่สุดที่แท้จริงและจุดค่าที่ดีที่สุดเฉพาะที่ (Local Optimum) คือจุดเดียวกัน ดังรูปที่ 2(ก) ในทางกลับกันปัญหาของพหุฐานนิยม คือ ปัญหาการหาค่าที่เหมาะสมที่สุดบนพื้นที่การค้นหาที่มีค่าที่ดีที่สุดเฉพาะที่หลายค่า และมีเพียงค่าเดียวเท่านั้นในค่าที่ดีที่สุดเฉพาะที่ทั้งหมดที่เป็นค่าที่ดีที่สุดที่แท้จริง ดังรูปที่ 2(ข) ทั้งนี้รายละเอียดของฟังก์ชันมาตรฐานทั้ง 8 ฟังก์ชันที่นำมาทดสอบแสดงดังตารางที่ 1



2(ก) ปัญหาแบบฐานนิยมเดียว **2(ข)** ปัญหาแบบพหุฐานนิยม

รูปที่ 2 ตัวอย่างของปัญหาแบบฐานนิยมเดียว และแบบพหุฐานนิยม

ตารางที่ 1 ฟังก์ชันที่ใช้ในการทดสอบ

ชื่อฟังก์ชัน	ชนิดของฟังก์ชัน	สมการ	ขอบเขต
SPHERE	ฐานนิยมเดียว	$f(x)=\sum_{i=1}^{n}x_i^2$	$x\in[-100,100]^n$
SUMSQUARES	ฐานนิยมเดียว	$f(x)=\sum_{i=1}^{n}ix_j^2$	$x\in[-10,10]^n$
ROSENBROCK	ฐานนิยมเดียว	$f(x)=\sum_{i=1}^{n}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$x\in[-2.048,2.048]^n$
DIXON-PRICE	ฐานนิยมเดียว	$f(x)=(x_1-1)^2+\sum_{i=2}^{n}i(2x_i^2-x_{i-1})^2$	$x\in[-10,10]^n$
RASTRIGIN	พหุฐานนิยม	$f(x)=10n+\sum_{i=1}^{n}(x^2-10\cos(2\pi x_i))$	$x\in[-5.12,5.12]^n$
SCHWEFEL	พหุฐานนิยม	$f(x)=418.9829\times n+\sum_{i=1}^{n}(x_i\times sin(\sqrt{\lvert x_i\rvert}))$	$x\in[-500,500]^n$
GRIEWANK	พหุฐานนิยม	$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$x\in[-600,600]^n$
ACKLEY	พหุฐานนิยม	$f(x)=-20\,exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-exp(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i))+20+e$	$x\in[-32.768,32.768]^n$

งานวิจัยนี้ได้ใช้แลปทอปคอมพิวเตอร์โดยใช้ซีพียูรุ่น Intel® core™ i5-5200U @2.20GHz และมีหน่วยความจำ (RAM) 4 GB สำหรับโปรแกรมที่ใช้ในการพัฒนาคือไมโครซอฟท์วิชวลซีพลัสพลัส (Microsoft Visual C++) 2015 และเนื่องจากอัลกอริทึมที่เลือกใช้ในการทดสอบเป็นวิธีการแบบสุ่มอนุภาคเริ่มต้นในตอนแรกจึงทำการทดลองประมวลผลจำนวน 20 ครั้งในแต่ละฟังก์ชันแล้วนำมาหาค่าเฉลี่ย จำนวนประชากรและจำนวนรอบการทำซ้ำที่มากที่สุดถูกกำหนดที่ 30 และ 3,000 ตามลำดับ โดยจะทำการทดสอบกับปัญหาที่มีจำนวนมิติที่ต่างกัน คือ 10, 20 และ 30 มิติ

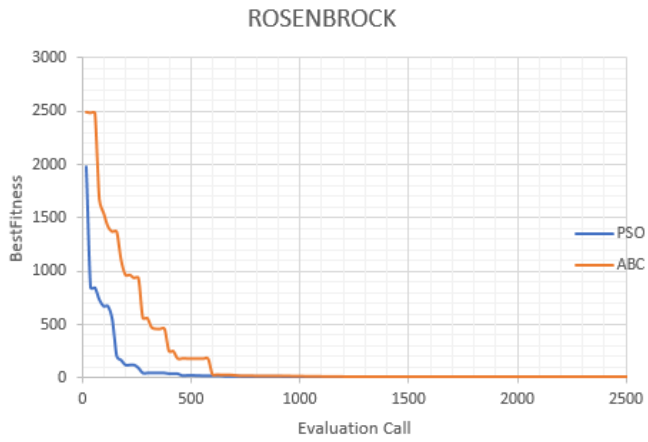
ผลการวิจัยและอภิปรายผล

ในการทดลองนี้มีวัตถุประสงค์เพื่อหาค่าที่น้อยที่สุด ซึ่งค่าคำตอบที่เหมาะสมที่สุดสำหรับทุกฟังก์ชัน (Optimal) คือ 0 ดังนั้นวิธีหรืออัลกอริทึมใดที่หาค่าได้เข้าใกล้ 0 ที่สุด ขั้นตอนนั้นจะมีประสิทธิภาพมากที่สุดซึ่งจะถูกเน้นด้วยเส้นใต้

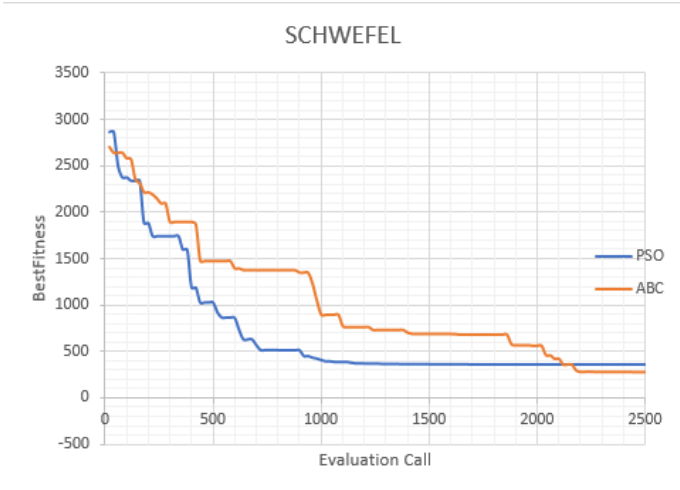
จากตารางที่ 2 แสดงให้เห็นถึงผลการค้นหาค่าคำตอบที่ดีที่สุดของอัลกอริทึมอาณานิคมผึ้งเทียมเทียบกับอัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค ในการทดลองนี้จะเปรียบเทียบผ่านฟังก์ชันมาตรฐานทั้ง 8 ฟังก์ชัน โดย 4 ฟังก์ชันแรกเป็นฟังก์ชันแบบฐานนิยมเดียวและ 4 ฟังก์ชันที่เหลือเป็นฟังก์ชันแบบพหุฐานนิยม โดยวิธีที่เจอคำตอบเร็วที่สุดจะแสดงโดยการเน้นตัวหนาและขีดเส้นใต้ จากตารางที่ 2 จะเห็นว่าในฟังก์ชันที่เป็นแบบฐานนิยมเดียวนั้น อัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคเจอค่าคำตอบได้ดีกว่าอัลกอริทึมอาณานิคมผึ้งเทียม โดยอัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคเจอค่าคำตอบที่ดีที่สุด (Global Optimum) ถึง 2 ฟังก์ชัน คือ ฟังก์ชัน ROSENBROCK ที่ 10 มิติ และฟังก์ชัน DIXON-PRICE ที่ 10 และ 20 มิติ ขณะที่ในฟังก์ชัน SUMSQUARES และฟังก์ชัน SPHERE แม้จะไม่เจอค่าคำตอบที่ดีที่สุดแต่ก็เจอในตำแหน่งที่ดีกว่าอัลกอริทึมอาณานิคมผึ้งเทียม ในขณะที่การทดลองบนฟังก์ชันที่เป็นแบบพหุฐานนิยมนั้น อัลกอริทึมอาณานิคมผึ้งเทียมเจอค่าคำตอบได้ดีกว่าอัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค และเจอค่าคำตอบที่ดีที่สุดในฟังก์ชัน RASTRIGIN ที่ 10 มิติ ขณะที่ฟังก์ชัน SCHWEFEL, ฟังก์ชัน GRIEWANK และ ฟังก์ชัน ACKLEY แม้จะไม่เจอค่าคำตอบที่ดีที่สุด แต่ก็เจอในตำแหน่งที่ดีกว่าอัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค แสดงให้เห็นว่าอัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคมีประสิทธิภาพในการหาค่าคำตอบได้ดีกว่าอัลกอริทึมอาณานิคมผึ้งเทียม ในกรณีการหาค่าคำตอบในฟังก์ชันแบบฐานนิยมเดียวและอัลกอริทึมอาณานิคมผึ้งเทียมมีประสิทธิภาพที่ดีกว่าอัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคในการหาค่าคำตอบในฟังก์ชันแบบพหุฐานนิยม

ตารางที่ 2 ผลการทดลองการเปรียบเทียบประสิทธิภาพในการค้นหาค่าคำตอบของวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค และอัลกอริทึมอาณานิคมผึ้งเทียม

No	ชื่อฟังก์ชัน	จำนวนมิติ	PSO		ABC	
			Best Fitness	Evaluation Call	Best Fitness	Evaluation Call
1	SPHERE	10	**1.67x10^{-130}**	60000	9.95x10^{-17}	60000
		20	**7.12x10^{-71}**	60000	3.93 x10^{-16}	60000
		30	**8.16 x10^{-36}**	60000	8.63 x10^{-16}	60000
2	SUMSQUARES	10	**7.90 x10^{-145}**	60000	7.48 x10^{-17}	60000
		20	**3.41 x10^{-77}**	60000	3.72 x10^{-16}	60000
		30	**6.80 x10^{-40}**	60000	7.33 x10^{-16}	60000
3	ROSENBROCK	10	**0**	17500	0.823935	60000
		20	**7.094 x10^{-29}**	60000	6.09126	60000
		30	**8.394 x10^{-26}**	60000	10.756	60000
4	Dixon-Price	10	**0**	7178	-0.5	60000
		20	**0**	26107	-0.4991	60000
		30	**4.66 x10^{-16}**	60000	-0.498	60000
5	RASTRIGIN	10	11.143518	60000	**0**	13331
		20	51.14071	60000	**1.05 x10^{-8}**	60000
		30	97.15745	60000	**0.049757**	60000
6	SCHWEFEL	10	505.31055	60000	**-0.0288727**	60000
		20	1486.29725	60000	**-0.0531995**	60000
		30	2999.5075	60000	**111.626**	60000
7	GRIEWANK	10	0.0878307	60000	**1.11 x10^{-16}**	60000
		20	0.040449	60000	**4.441 x10^{-16}**	60000
		30	0.0245505	60000	**8.88 x10^{-16}**	60000
8	ACKLEY	10	0.057758	60000	**9.33 x10^{-15}**	60000
		20	1.321441	60000	**3.74 x10^{-14}**	60000
		30	3.171893	60000	**6.78 x10^{-14}**	60000



รูปที่ 3 กราฟแสดงการลู่เข้าหาคำตอบของอัลกอริทึมอาณานิคมผึ้งเทียมและวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคบนฟังก์ชัน ROSENBROCK



รูปที่ 4 กราฟแสดงการลู่เข้าหาคำตอบของอัลกอริทึมอาณานิคมผึ้งเทียมและวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคบนฟังก์ชัน SCHWEFEL

จากรูปที่ 3 เป็นกราฟแสดงการลู่เข้าหาคำตอบบนฟังก์ชัน ROSENBROCK ของอัลกอริทึมอาณานิคมผึ้งเทียมเทียบกับวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค โดยที่แกน Y แทนค่าคำตอบที่ดีที่สุด แกน X แทน Evaluation Call จากกราฟจะเห็นว่าในช่วงแรกของการค้นหาคำตอบวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคหรือกราฟสีน้ำเงิน

ทำการลู่เข้าหาคำตอบได้เร็วกว่าอัลกอริทึมอาณานิคมผึ้งเทียมหรือกราฟสีส้ม จนเมื่อค้นหาไปในระยะเวลาหนึ่งจะเห็นว่า วิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคและอัลกอริทึมอาณานิคมผึ้งเทียมมีการเคลื่อนที่เข้าหาคำตอบใกล้เคียงกัน แต่วิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคสามารถค้นหาคำตอบได้เจอเมื่อครบจำนวนรอบที่กำหนดดังแสดงในตารางที่ 2 ในขณะที่อัลกอริทึมอาณานิคมผึ้งเทียมไม่สามารถค้นหาค่าคำตอบเจอในรอบที่กำหนด จากรูปที่ 4 เป็นการทดสอบบนฟังก์ชัน SCHWEFEL จะเห็นว่าในช่วงแรกวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคหรือกราฟสีน้ำเงินทำการลู่เข้าหาคำตอบได้เร็วกว่าก็จริง แต่เมื่อทำการค้นหาไปในระยะเวลาหนึ่งวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคมีแนวโน้มที่จะคงที่หรือติดในจุดค่าที่ดีที่สุดเฉพาะที่นั่นเอง ในขณะที่อัลกอริทึมอาณานิคมผึ้งเทียมหรือกราฟสีส้ม ในช่วงแรกลู่เข้าหาคำตอบช้ากว่าแต่ก็ลู่เข้าหาคำตอบอย่างต่อเนื่องไปจนได้ค่าคำตอบที่ดีกว่าวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคนั่นเอง

สรุปผลการทดลอง

บทความนี้ได้นำเสนอการทำงานของอัลกอริทึมอาณานิคมผึ้งเทียมและวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค พร้อมทำการเปรียบเทียบประสิทธิภาพระหว่างอัลกอริทึมอาณานิคมผึ้งเทียมและวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคผ่านฟังก์ชันมาตรฐานทั้งหมด 8 ฟังก์ชัน จากการทดสอบพบว่าทั้งสองอัลกอริทึมสามารถค้นหาคำตอบได้และลู่เข้าสู่ค่าคำตอบที่ดีได้ โดยอัลกอริทึมอาณานิคมผึ้งเทียมเหมาะกับการแก้ปัญหาในฟังก์ชันที่มีความซับซ้อนหรือแบบพหุฐานนิยม เนื่องจากอัลกอริทึมอาณานิคมผึ้งเทียมมีการเพิ่มความสามารถในการค้นหาในแนวกว้างทำให้โอกาสในการเจอคำตอบก็มากขึ้นตามไปด้วย แต่เมื่อทำการค้นหาไปในระยะเวลาหนึ่งจะพบว่ายังคงมีปัญหาในการลู่เข้าก่อนกำหนด ในขณะที่อัลกอริทึมวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคเหมาะกับการแก้ปัญหาในฟังก์ชันแบบฐานนิยมเดียว เนื่องจากวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาคนั้นจะทำการค้นหาคำตอบด้วยการเคลื่อนที่ไปบริเวณใกล้เคียงตัวเองเท่านั้นแต่ถ้าคำตอบอยู่ไกลจากตำแหน่งของอนุภาคการค้นหาก็จะทำได้ช้า และมีโอกาสที่จะติดค่าที่ดีที่สุดเฉพาะที่ได้

เอกสารอ้างอิง

บุญเจริญ ศิริเนาวกุล. (2555). *ปัญญาประดิษฐ์: ปัญญาเชิงกลุ่ม*. กรุงเทพมหานคร: ท้อป.

Hedar, A. (2018) *Test Functions for Unconstrained Optimization*, Retrieved from http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm

Karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization*. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey.

Karaboga, D. & Akay B. (2009). A Comparative study of artificial bee colony algorithm. *Applied Mathematics and computation, 214*(1), 108-132.

Karaboga, D. & Akay, B. (2009). A Survey: Algorithms Simulating Bee Swarm Intelligence; *Artificial Intelligence Review, 31* (1), 61-85.

Sindhya, K. (2012). An Introduction to Nature Inspired Algorithms. *Industrial Optimization Group,* 1-36.

Molga, M. & Smutnicki, C. (2005). *Test functions for optimization needs,* Retrieved March 1, 2019, from https://www.robertmarks.org/Classes/ENGR5358/ Papers/functions.pdf

Eberhart, R. C. & Kennedy, J. (1995). *Particle Swarm Optimization*. Proceedings of the 1995 IEEE International Conference on Neural Networks, 4, 1942-1948.

Eberhart, R. C. & Kennedy, J. (1995). *A New Optimizer Using Particle Swarm Theory*. Proceedings of the 6th International Symposium on Micro Machine and Human Science, 6, 39-43.

Sharma, S. & Bhambu, P. (2016). Artificial Bee Colony Algorithm: A Survey. *International Journal of Computer Applications, 149*(4), 11-19.

Surjanovic, S. & D, Bingham, D. (2017). *Optimization Test Functions and Datasets*, Retrieved from http://www.sfu.ca/~ssurjano/index.html