การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 48
The 48th Electrical Engineering Conference (EECON-48)
วันที่ 19-21 พฤศจิกายน 2568 ณ โรงแรมฟูราม่า จังหวัดเชียงใหม่

วิธีหาค่าเหมาะสมที่สุดด้วยไฮเปอร์คิวบ์ที่ใช้กลยุทธ์การเรียนรู้แบบพลวัตของวิธีค้นหาเชิงฝูงอนุภาค

Hypercube Search with Dynamic Strategy Learning in Particle Swarm Optimization

รังสิมันต์ สิทธิกร

สาขาวิชาวิศวกรรมควบคุมอุตสาหกรรมและเครื่องมือวัด สถาบันวิศวกรรมไฟฟ้าและอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์และเทคโนโลยี มหาวิทยาลัยเทคโนโลยีมหานคร rungsimant@mut.ac.th

บทคัดย่อ

บทความนี้เสนอวิธีหาค่าเหมาะสมที่สุดด้วยการสร้างบริเวณค้นหาขนาดเล็กที่เรียกว่าไฮเปอร์คิวบ์ ซึ่งสามารถเคลื่อนที่และขนาดลดลง ที่แปรตามสมรรถนะของการค้นหาในขณะนั้น แล้วอาศัยวิธีค้นหาเชิงฝูงอนุภาคที่ใช้กลยุทธ์การเรียนรู้เชิงพลวัตในไฮเปอร์คิวบ์ สามารถหาคำตอบในฟังก์ชันเปรียบเทียบสมรรถนะแบบดั้งเดิมได้ถึง 6 และ 5 จาก 7 ฟังก์ชันที่จำนวนมิติเท่ากับ 10 และ 30 ภายใต้เงื่อนไขจำนวนการประเมินฟังก์ชันสูงสุด 10000*D วิธีค้นหานี้มีสมรรถนะเหนือกว่าวิธีอื่นที่นำมาเปรียบเทียบ การทดสอบกับฟังก์ชันที่ถูกดัดแปลงให้ตำแหน่งคำตอบในแต่ละมิติที่ต่างกัน วิธีค้นหาที่เสนอยังคงหาคำตอบได้เช่นเดิม

คำสำคัญ: ไฮเปอร์คิวบ์ การหาค่าเหมาะสมที่สุดวิธีเชิงฝูงอนุภาค

Abstract

This paper proposes an optimization method by constructing a small search area called a hypercube, which can move and shrink based on the current search performance. The approach uses particle swarm optimization with a dynamic learning strategy within the hypercube. The method successfully found solutions for 6 and 5 out of 7 classical benchmark test functions with 10 and 30 dimensions, respectively, under the maximum function evaluation limit of 10,000*D. The proposed method outperformed other comparative optimization techniques. Moreover, the proposed method maintained strong performance even when we modified the solution of benchmark functions to position differently across dimensions.

Keywords: Hypercube, Particle Swarm Optimization

1. บทนำ

การหาค่าเหมาะสมที่สุดวิธีเชิงฝูงอนุภาค (Particle Swarm Optimization: PSO) โดย Eberhart และ Kennedy [1] อาศัยการสุ่มตำแหน่งของอนุภาคจำนวนหนึ่ง แล้วนำไปทดสอบในปัญหา ถ้ายังไม่ได้ค่าเหมาะสมที่สุด จะปรับปรุงตำแหน่งของอนุภาค โดยเรียนรู้จากตำแหน่งที่ดีที่สุดของตัวเองในอดีต (personal best: *pbest*) และตำแหน่งที่ดีที่สุดในฝูงอนุภาคจากการค้นหาที่ผ่านมา (global best: *gbest*) การทดสอบสมรรถนะวิธีค้นหาจะอาศัยฟังก์ชันเปรียบเทียบสมรรถนะ ซึ่งรู้ค่าเหมาะสมที่สุดของตำแหน่งคำตอบ วิธี PSO สามารถหาคำตอบของฟังก์ชันฐานนิยมเดียว (Unimodal function) ที่จำนวนมิติ (Dimension: D) ไม่สูง แต่การค้นหาจะแย่ลงหรือเรียกว่าติดกับดักค้นหา (trap) เมื่อทดสอบในฟังก์ชันพหุฐานนิยม (Multimodal function) แต่ด้วยจุดเด่นของวิธีค้นหาที่ไม่ซับซ้อน จึงถูกนำไปพัฒนาต่อมากมาย วิธี CLPSO โดย Liang และคณะ [2] เสนอวิธีปรับปรุงตำแหน่งในแต่ละมิติของอนุภาค ให้สามารถเรียนรู้จาก *pbest* ของอนุภาคตัวอื่นได้ สามารถหาค่าเหมาะสมที่สุดได้อย่างรวดเร็วในฟังก์ชันจำนวน 10 มิติ แต่การค้นหากลับแย่ลงในฟังก์ชันจำนวน 30 มิติ ด้วยอัลกอริทึมที่ไม่ซับซ้อน วิธี CLPSO จึงถูกนำไปพัฒนาต่อโดย Lynn และ Suganthan [3] เรียกว่า HCLPSO-EE โดยแบ่งอนุภาคออกเป็นสองกลุ่ม ที่วัตถุประสงค์การค้นหาของแต่ละกลุ่มต่างกัน คือกลุ่มที่เน้นการค้นหาในวงกว้าง (Exploration search) และกลุ่มที่เน้นการค้นหาในวงแคบ (Exploitation search) แม้ว่าสมรรถนะโดยรวมดีกว่าวิธีค้นหาอื่นที่นำมาเปรียบเทียบ แต่ทดสอบในฟังก์ชันเปรียบเทียบสมรรถนะ CEC2005 ซึ่งไม่ใช่ฟังก์ชันเดิมที่วิธี CLPSO ใช้ทดสอบ จึงไม่สามารถพิสูจน์ได้ว่าวิธี HCLPSO-EE มีสมรรถนะดีกว่าวิธี CLPSO หรือไม่ สำหรับวิธี CLPSO ยังถูกนำไปพัฒนาโดย Xia และคณะ [4] เรียกว่า TAPSO อาศัยการสร้างคลังข้อมูลอนุภาค 3 ชุด สำหรับเลือกใช้เรียนรู้การปรับปรุงตำแหน่ง ผลทดสอบในฟังก์ชันเปรียบเทียบสมรรถนะแบบดั้งเดิม ผลการค้นหาดีกว่าวิธี CLPSO และวิธีค้นหาอื่นที่นำมาเปรียบเทียบ แต่ก็ยังติดกับดักค้นหา นอกจากนี้ผลทดสอบในฟังก์ชันเปรียบเทียบสมรรถนะแบบดั้งเดิม ที่ดัดแปลงโดยใช้วิธีการเลื่อนและการหมุนให้กับค่าค้นหา ก่อนจะนำค่าไปทดสอบในฟังก์ชัน จะเป็นการเพิ่มความยากอย่างมากในการค้นหา ส่งผลให้สมรรถนะแย่ลงอย่างมาก ดังนั้น B. Zhan และคณะ [5] ที่เสนอวิธีค้นหา MASPE-HGSA ได้เสนอการดัดแปลงตำแหน่งคำตอบของฟังก์ชัน เพียงแค่การเลื่อนในหลายรูปแบบ วิธีนี้ง่ายและยังคงลักษณะเดิมของฟังก์ชัน เพียงแต่ตำแหน่งคำตอบแต่ละมิติที่ต่างกัน เป็นการค่อยเพิ่มระดับความยากให้กับฟังก์ชัน ซึ่งสามารถตรวจสอบวิธีค้นหาที่สามารถหาคำตอบกรณีตำแหน่งคำตอบแต่ละมิติเหมือนกัน จะยังคงหาคำตอบได้ดังเดิมหรือไม่ถ้าตำแหน่งคำตอบเปลี่ยนไป

วิธีค้นหา Hc-ACPSO โดย [6] สามารถหาคำตอบในฟังก์ชันเปรียบเทียบสมรรถนะแบบดั้งเดิมได้ถึง 5 ฟังก์ชัน แต่ต้องใช้อนุภาคจำนวนมากและความเร็วการค้นหาต่ำ ดังนั้นบทความนี้จึงปรับปรุงวิธี

ดังกล่าวโดยเสนอวิธี Hc-DSPSO ซึ่งจะอธิบายอัลกอริทึมทั้งสองวิธีในหัวข้อที่ 2 และ 3 ผลการทดสอบและการเปรียบเทียบสมรรถนะค้นหากับวิธีค้นหาอื่นแสดงในหัวข้อที่ 4 และสรุปผลการทดลองในหัวข้อสุดท้าย

2. วิธี Hc-ACPSO

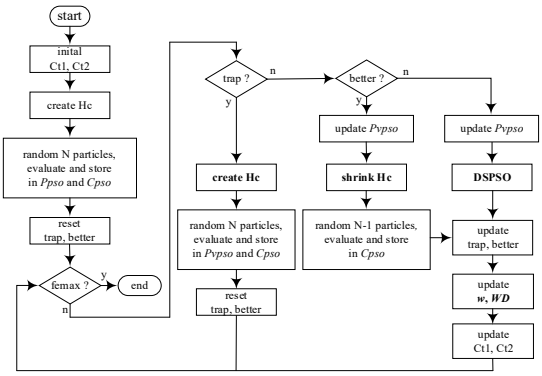
วิธีหาค่าเหมาะสมที่สุดด้วยการจำกัดบริเวณค้นหาให้เล็กลงที่เรียกว่าไฮเปอร์คิวบ์ (Hypercube: Hc) โดย [6] สามารถหาคำตอบของฟังก์ชันเปรียบเทียบสมรรถนะจำนวนมิติสูงได้รวดเร็ว แต่ไม่ได้อธิบายวิธีการสร้าง Hc ซึ่งเป็นกระบวนการสำคัญ ดังนั้นวิธี Hc-ACPSO โดย [7] เสนอวิธีการสร้าง Hc และคลังข้อมูลเก็บตำแหน่งศูนย์กลาง Hc ไว้สำหรับสร้าง Hc ใหม่ของอัลกอริทึม โดยที่ข้อมูลในคลังได้จากกระบวนการสุ่มอนุภาคในขอบเขตของฟังก์ชัน แล้วนำไปทดสอบและเลือกตำแหน่งของอนุภาคที่ดีที่สุด ซึ่งกระบวนการดังกล่าวจะยังเกิดขึ้นในวงรอบค้นหา สำหรับขนาดความกว้างของ Hc (Hypercube width: WD) และค่าถ่วงน้ำหนัก (inertia weight: w) ของอัลกอริทึม ACPSO จะแปรตามขนาดการลดลงของค่าเหมาะสมที่สุด เทียบกับค่าจากในวงรอบค้นหาก่อนหน้า วิธีนี้สามารถหาคำตอบของฟังก์ชันจำนวน 20 มิติ แต่จะต้องใช้อนุภาคจำนวนมากถึง 200 ใน 10,000 วงรอบซ้ำของการค้นหา เมื่อคำนวณจำนวนการประเมินฟังก์ชันมากที่สุด (Maximum function evaluation: femax) ได้เท่ากับ $4x10^6$ จึงไม่เหมาะกับการค้นหาของฟังก์ชันที่จำนวนมิติสูง

3. วิธี Hc-DSPSO

หลักการของวิธีค้นหาที่เสนอ จะเริ่มจากการสร้างคลังข้อมูล Ct1 และ Ct2 ไว้สำหรับสร้าง Hc ซึ่งจะอธิบายในหัวข้อ 3.1 จากนั้นสร้าง Hc ให้ความกว้าง (WD) แต่ละมิติเท่ากับ 1.5% ของขอบเขตค้นหาของปัญหาหรือฟังก์ชัน สุ่มตำแหน่งของอนุภาคจำนวน N ในบริเวณ Hc ประเมินค่าฟังก์ชัน เก็บข้อมูลอนุภาคลงใน *pvPso* และ *cPso* รีเซ็ตตัวบ่งชี้การติดกับดักค้นหา (trap) และตัวบ่งชี้การค้นหาดีขึ้น (better) ในระหว่างการค้นหาก่อนเข้าสู่การตรวจสอบตัวบ่งชี้และเมื่อมีการประเมินค่าฟังก์ชัน นอกจากนี้จะตรวจสอบการเกินจำนวน femax เพื่อสิ้นสุดการค้นหา

เส้นทางการค้นหาในวงรอบมี 3 วิธี โดยถ้า trap ถูกเซ็ต จะไปสู่การสร้าง Hc ใหม่ ด้วยวิธีการเดียวกันกับก่อนเข้าสู่วงรอบค้นหา แต่ถ้า trap ถูกรีเซ็ต และ better ถูกเซ็ต จะลดขนาด Hc ลง โดยจะแทนที่ข้อมูล *pvPso* ด้วย *cPso* แล้วใช้ตำแหน่งของอนุภาคที่ค่าของฟังก์ชันดีที่สุดเป็นจุดศูนย์กลางใหม่ของ Hc คำนวณความกว้าง WD ใหม่ ซึ่งจะอธิบายในหัวข้อ 3.2 จากนั้นสุ่มอนุภาคใหม่จำนวน N-1 ประเมินค่าฟังก์ชัน เก็บข้อมูลของอนุภาคใหม่และอนุภาคที่เป็นจุดศูนย์กลางลงใน *cPso* แต่ถ้า better ถูกรีเซ็ต จะแทนที่ข้อมูล *pvPso* ด้วย *cPso* แล้วเข้าสู่อัลกอริทึม DSPSO ซึ่งจะอธิบายในหัวข้อ 3.3

หลังผ่านการค้นหาจากการลดขนาด Hc หรืออัลกอริทึม DSPSO จะปรับปรุงค่า trap กับ better และ w ในอัลกอริทึม DSPSO กับ WD จากนั้นปรับปรุงคลังข้อมูล Ct1 และ Ct2 แล้ววนกลับไปตรวจสอบการเกินจำนวน Femax อีกครั้ง



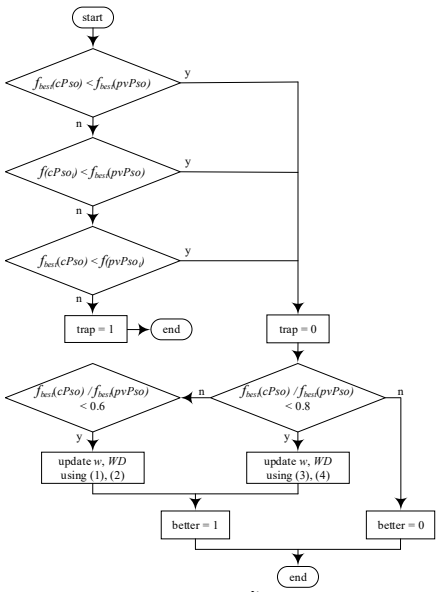
รูปที่ 1 โครงสร้าง Hc-DSPSO

3.1 วิธีการสร้างและปรับปรุงคลังข้อมูลของ Hc

ข้อมูลในคลัง Ct1 และ Ct2 ได้จากการสุ่มตำแหน่งของอนุภาคจำนวน 100 ในขอบเขตของฟังก์ชัน ประเมินค่าฟังก์ชัน แล้วเลือกตำแหน่งของอนุภาคที่ค่าฟังก์ชันดีที่สุด มาสร้างอนุภาคนี้ซ้ำจำนวน 300 จากนั้นจะแทนที่ตำแหน่งในมิติแรกของอนุภาคทั้งหมด ด้วยการสุ่มตำแหน่งใหม่เพียงมิติเดียว ประเมินค่าฟังก์ชัน ตำแหน่งในมิติแรกของอนุภาคที่ค่าฟังก์ชันดีที่สุด จะถูกเก็บใน Ct1 และเก็บตำแหน่งทุกมิติของอนุภาคนี้และค่าฟังก์ชันใน Ct2 จากนั้นเลื่อนไปทำด้วยกระบวนการเดียวกันในมิติถัดไปจนครบทุกมิติ แล้ววนกลับมาทำซ้ำอีก 2 ครั้ง ดังนั้นคลัง Ct1 จะมีอนุภาคจำนวน 3 ตำแหน่ง และคลัง Ct2 จะมีอนุภาคจำนวน 3*D ตำแหน่ง จากนั้นคำนวณค่าฟังก์ชันของอนุภาคใน Ct1 แล้วเก็บค่าไว้

ในระหว่างการค้นหา ถ้า trap ไม่ถูกเซ็ตและค่าฟังก์ชันที่ดีที่สุดของอนุภาคใน *cPso* น้อยกว่า 10 เท่า ของค่าฟังก์ชันที่แย่ที่สุดของอนุภาคในคลังข้อมูลนั้น ก็จะแทนที่อนุภาคที่แย่ที่สุดในคลังข้อมูล ด้วยอนุภาคที่ดีที่สุดใน *cPso*

3.2 การปรับปรุงค่าพารามิเตอร์



รูปที่ 2 วิธีกำหนดค่าตัวบ่งชี้ trap และ better

ในระหว่างการค้นหาถ้าค่าฟังก์ชันลดลงน้อยมาก หมายความว่าตำแหน่งคำตอบอาจไม่อยู่ใน Hc ก็จะไปสร้าง Hc ใหม่ โดยตรวจสอบจากตัวบ่งชี้การติดกับดักคือ trap ว่าถูกเซ็ตหรือไม่ โดยถ้าเงื่อนไขทั้ง 3 เป็นจริง คือ ค่าฟังก์ชันของอนุภาคที่ดีที่สุดใน *cPso* ไม่น้อยกว่าค่าของอนุภาคที่ดีที่สุดใน *pvPso* ค่าฟังก์ชันของอนุภาคที่ดีในลำดับ j ใน *cPso* ไม่น้อยกว่าค่าของอนุภาคที่ดีที่สุดใน *pvPso* และค่าฟังก์ชันของอนุภาคที่ดีที่สุดใน *cPso* ไม่น้อยกว่าค่าของอนุภาคที่ดีในลำดับ j ใน *pvPso* โดยที่ลำดับ j กำหนดให้เท่ากับ 40% ของจำนวนอนุภาค N

ดังนั้นถ้าไม่ติดกับดักการค้นหา แล้วค่าฟังก์ชันของอนุภาคที่ดีที่สุดใน *cPso* น้อยกว่า 60% ของค่าจากอนุภาคที่ดีที่สุดใน *pvPso* จะปรับปรุง w และ WD ด้วยสมการ (1) และ (2) แต่ถ้าค่าน้อยกว่ากว่า 80% จะปรับปรุงค่าทั้งสองด้วยสมการ (3) และ (4) นอกเหนือจากเงื่อนไขนี้ ค่าทั้งสองจะเท่าเดิม จากนั้นเซ็ตค่าตัวบ่งชี้ better

$$w \leftarrow \left(0.5 + 0.1\times\left(\frac{f_{best}(cPso)}{f_{best}(pvPso)}\right)^2\right)\times w \quad (1)$$

$$WD \leftarrow \left(0.5 + 0.1\times\left(\frac{f_{best}(cPso)}{f_{best}(pvPso)}\right)\right)^2 \times WD \quad (2)$$

$$w \leftarrow \left(1 - 0.1\times e^{\left(-\frac{0.5\times f_{best}(cPso)}{f_{best}(pvPso)}\right)}\right)\times w \quad (3)$$

$$WD \leftarrow \left(1 - 0.1\times e^{\left(-\frac{0.5\times f_{best}(cPso)}{f_{best}(pvPso)}\right)}\right)\times WD \quad (4)$$

เมื่อ $f_{best}(cPso)$ คือค่าฟังก์ชันที่ดีที่สุดของอนุภาคใน *cPso*

$f_{worst}(cPso)$ คือค่าฟังก์ชันที่แย่ที่สุดของอนุภาคใน *cPso*

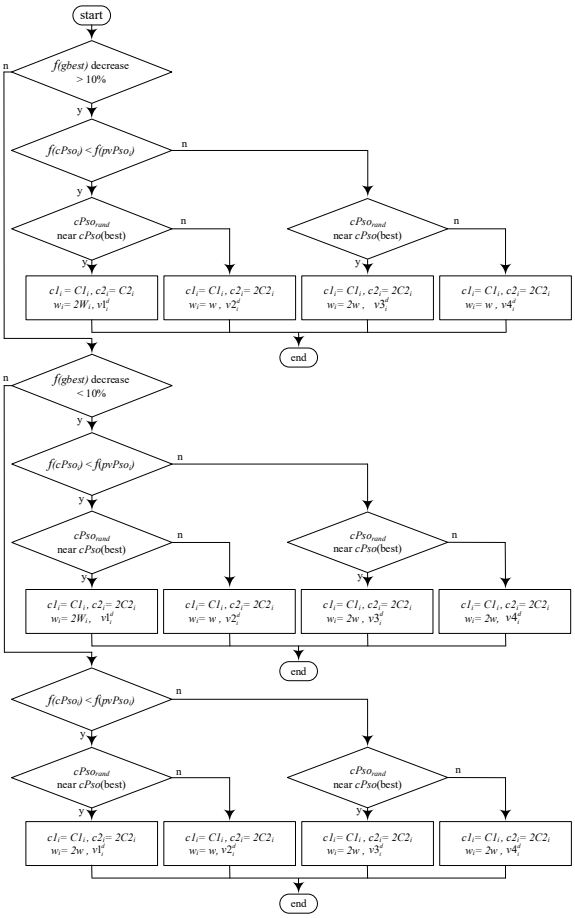
w มีค่าเริ่มต้นเท่ากับ 0.3

3.3 อัลกอริทึม DSPSO

การปรับปรุงตำแหน่งอนุภาคใน DSPSO จะพิจารณาจากขนาดการลดลงของค่าฟังก์ชันของ *gbest* เทียบกับในวงรอบค้นหาก่อนหน้า แสดงดังรูปที่ 3 เพื่อไปสู่เงื่อนไขชั้นถัดไป ซึ่งจะตรวจสอบการลดลงของค่าฟังก์ชันของอนุภาคตัวนั้นเปรียบเทียบกับวงรอบค้นหาก่อนหน้า แล้วไปสู่เงื่อนไขชั้นสุดท้าย ซึ่งก็จะสุ่มอนุภาคมาตรวจสอบว่าอยู่ใกล้อนุภาคที่ค่าฟังก์ชันดีที่สุดหรือแย่ที่สุด ด้วยสมการ (5) จากเงื่อนไขเหล่านี้ จะไปสู่การเลือกวิธีคำนวณค่าสัมประสิทธิ์ $c1_i$ $c2_i$ และ w_i ในอัลกอริทึม DSPSO ด้วยสมการ (6)-(8) และเลือกรูปสมการคำนวณความเร็วจากตัวเลือกในสมการ (9)-(12) สำหรับปรับปรุงตำแหน่งของอนุภาคด้วยสมการ (13) ถ้าหลังการปรับปรุงแล้วตำแหน่งของอนุภาคในมิติใดอยู่นอกขอบเขต Hc จะถูกทำให้กลับมาอยู่ในบริเวณ Hc ด้วยวิธีเช่นเดียวกับใน [7]

$$\frac{f(cPso_i)}{f_{best}(cPso)} < \frac{f_{worst}(cPso)}{f(cPso_i)} \quad (5)$$

$$C1_i = 1 - 0.1\times e^{\left(-\frac{0.5\times f(cPso_i)}{f_{best}(cPso)}\right)} \quad (6)$$



รูปที่ 3 วิธีกำหนดค่าสัมประสิทธิ์สำหรับการปรับปรุงความเร็วใน DSPSO

$$C2_i = 1 - 0.1\times e^{\left(-\frac{0.5\times f(cPso_i)}{f(gbest)}\right)} \quad (7)$$

$$W_i = w\times\frac{f(cPso_i)}{f_{worst}(cPso)} \quad (8)$$

เมื่อ $f(cPso_i)$ คือค่าฟังก์ชันของอนุภาคตัวที่ i ใน *cPso*,

$$v1_i^d \leftarrow w_i\times rand_i^d\times v_i^d + c1_i\times rand_i^d\times\left(pbest_i^d - x_i^d\right) + c2_i\times rand_i^d\times\left(gbest^d - x_i^d\right) \quad (9)$$

$$v2_i^d \leftarrow w_i\times rand_i^d\times v_i^d - c1_i\times rand_i^d\times gbest^d + c2_i\times rand_i^d\times\left(gbest^d - x_i^d\right) \quad (10)$$

$$v3_i^d \leftarrow w_i\times rand_i^d\times v_i^d + c2_i\times rand_i^d\times\left(gbest^d - x_i^d\right) \quad (11)$$

$$v4_i^d \leftarrow w_i\times rand_i^d\times v_i^d - c1_i\times rand_i^d\times gbest^d + c2_i\times rand_i^d\times\left(gbest^d - x_i^d\right) \quad (12)$$

$$x_i^d \leftarrow x_i^d + vk_i^d \;;\; k = 1,2,3,4 \quad (13)$$

4. ผลการทดลอง

การทดสอบจำนวนของอนุภาคที่เหมาะสมของวิธีค้นหา กำหนดจำนวนอนุภาคตั้งแต่ 10-50 และ 30-70 ทดสอบในฟังก์ชัน *f1-f7* [7] จำนวน 10 และ 30 มิติ ขอบเขตค้นหาเช่นเดียวกับ [4] เงื่อนไข FEmax เท่ากับ 10000*D โดยใช้โปรแกรม Matlab ทดสอบซ้ำแต่ละฟังก์ชันเป็นจำนวน 30 ครั้ง แต่ละครั้งจะถูกควบคุมให้รูปแบบการสุ่มเริ่มต้นที่ต่างกันด้วยคำสั่ง rng (Control random number generator) แสดงผลทดสอบ

ด้วยแผนภาพกล่องดังรูปที่ 4-10 โดยเพิ่มข้อมูลของค่าเฉลี่ยแบบ Mean แสดงด้วยสัญลักษณ์รูปเพชรที่ถูกแรเงา

ผลทดสอบในฟังก์ชันจำนวน 10 มิติ สามารถหาคำตอบได้ 30 ครั้ง ในฟังก์ชัน $f3, f5$ และ $f6$ ทุกจำนวนของอนุภาคที่ใช้ทดสอบ แต่ใน $f1$ เมื่อจำนวนอนุภาคเท่ากับ 30 ส่วน $f4$ และ $f7$ หาคำตอบได้อยู่ในช่วง 22-29 ครั้ง ส่วน $f2$ มีค่าฟังก์ชันเฉลี่ยแบบ Mean และ Median ใกล้เคียงกันเมื่อจำนวนอนุภาคระหว่าง 30-50 ส่วนผลทดสอบในฟังก์ชันจำนวน 30 มิติ สามารถหาคำตอบได้ 30 ครั้ง ในฟังก์ชัน $f4, f5$ และ $f6$ ทุกจำนวนของอนุภาคที่ใช้ทดสอบ ส่วน $f3$ และ $f7$ หาคำตอบได้อยู่ในช่วง 17-27 ครั้ง ในขณะที่สมรรถนะการค้นหาใน $f3$ และ $f7$ จะตรงกันข้ามกันเมื่อเปลี่ยนแปลงจำนวนอนุภาค

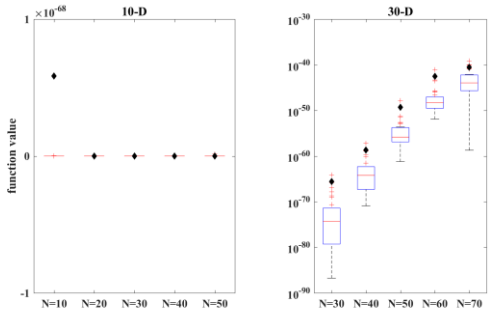
ความเร็วของการค้นหารูปที่ 11 มาจากการทดสอบ กรณีรูปแบบการสุ่มเริ่มต้นการค้นหาของทุกฟังก์ชันเหมือนกัน โดยใช้จำนวนอนุภาคเท่ากับ 30 และ 50 ในฟังก์ชันจำนวน 10 และ 30 มิติ สามารถหาคำตอบได้ 6 และ 5 ฟังก์ชัน และเมื่อเรียงลำดับความเร็วของการค้นหาในฟังก์ชัน จากมากไปน้อยจะเหมือนกันคือ $f6, f4, f7, f3, f5, f1$ และ $f2$

การเปรียบเทียบสมรรถนะระหว่างวิธีค้นหาค่าเหมาะสมที่สุดจะใช้เงื่อนไข Femax เท่ากัน โดยเลือกวิธีค้นหาที่ผลการค้นหาดีที่สุดในฟังก์ชัน $f1$-$f7$ จำนวน 10 มิติ จาก [4, 7] และเพิ่มวิธีค้นหา MASPE-HGSA ในการเปรียบเทียบฟังก์ชันจำนวน 30 มิติ แสดงดังตารางที่ 1 และ 2 สำหรับการเปรียบเทียบสมรรถนะ มักใช้ค่าเฉลี่ยของฟังก์ชันแบบ Mean เป็นเกณฑ์ประเมิน แต่ถ้าค่าฟังก์ชันซึ่งมาจากการทดสอบซ้ำ ไม่ได้มีลักษณะการกระจายแบบปรกติ (Normal distribution) เช่น ผลทดสอบ $f4$ ในตารางที่ 1 ค่าเฉลี่ยของฟังก์ชันแบบ Mean ของวิธีค้นหาทั้งหมดจะใกล้เคียงกัน แต่ค่าเฉลี่ยแบบ Median ของวิธี Hc-DSPSO เท่ากับ 0.00E+00 โดยที่ตัวเลขในวงเล็บคือจำนวน 22 ครั้งที่หาคำตอบได้ จะเห็นว่าด้วยข้อมูลค่าเฉลี่ยแบบ Median จะเห็นสมรรนะค้นหาที่แท้จริง ซึ่งจะทำให้การเปรียบเทียบสมรรถนะของวิธีค้นหามีความถูกต้องมากขึ้น

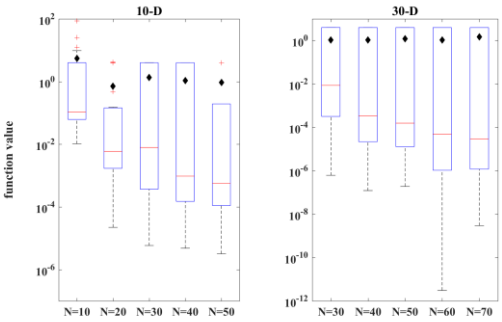
สมรรถนะวิธีค้นหาของฟังก์ชัน 10 มิติ ในตารางที่ 1 ถ้าใช้ค่าเฉลี่ยแบบ Mean เป็นเกณฑ์ประเมิน วิธี Hc-DSPSO ดีที่สุด ซึ่งดีกว่าวิธี TAPSO เพียงเล็กน้อย แต่เมื่อพิจารณาค่าเฉลี่ยแบบ Median และจำนวนครั้งที่ได้คำตอบ วิธีที่เสนอดีกว่าอย่างชัดเจน ส่วนวิธี Hc-ACPSO สามารถหาคำตอบได้ 4 ฟังก์ชัน โดยเฉพาะ $f2$ หาค่าได้ดีที่สุด แต่จำนวน Femax มากถึง $6x10^5$

สมรรถนะวิธีค้นหาของฟังก์ชัน 30 มิติ ในตารางที่ 2 วิธี TAPSO สมรรถนะค้นหาดีที่สุดในฟังก์ชันฐานนิยมเดียว แม้ว่าจะด้อยกว่า MASPE-HGSA เพียงเล็กน้อยในฟังก์ชัน $f1$ ส่วนคำตอบจะค้นหาได้เพียงฟังก์ชันเดียวคือ $f6$ สำหรับวิธี Hc-DSPSO หาคำตอบได้ 5 ฟังก์ชัน และถ้าใช้ค่าเฉลี่ยแบบ Median ประกอบการพิจารณา วิธีที่เสนอไม่ติดกับดักค้นหาในฟังก์ชัน $f2$ ในขณะที่สมรรถนะค้นหาลดลงในฟังก์ชัน $f1$

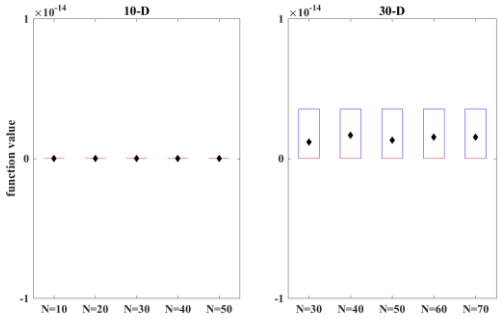
วิธี MASPE-HGSA แม้ว่าจะค้นหาได้ดีเฉพาะ $f1$ แต่ได้เพิ่มการทดสอบการค้นหาในฟังก์ชันที่ดัดแปลง ให้ตำแหน่งคำตอบซึ่งเดิมทุกมิติค่าเดียวกันให้ต่างออกไป แต่ลักษณะของฟังก์ชันยังเหมือนเดิม จึงสามารถตรวจสอบวิธีค้นหา ที่ถูกออกแบบให้หาคำตอบได้ดีเฉพาะตำแหน่งของคำตอบในทุกมิติเหมือนกัน วิธีการดัดแปลงฟังก์ชันจะแสดงในตารางที่ 3 และผลทดสอบในตารางที่ 4 สำหรับวิธี MASPE-HGSA ได้ทดสอบเฉพาะฟังก์ชันจำนวน 30 มิติ ซึ่งดัดแปลงด้วยวิธี Simple shift เท่านั้น ซึ่งยังคงสมรรถนะเดิมใน $f1$ แต่ด้อยลงใน $f2$ ส่วนวิธีที่เสนอหาคำตอบได้ 100% ทั้งในปัญหา 10 มิติ และ 30 มิติ ส่วนการค้นหาใน $f2$ ดีขึ้นเล็กน้อย



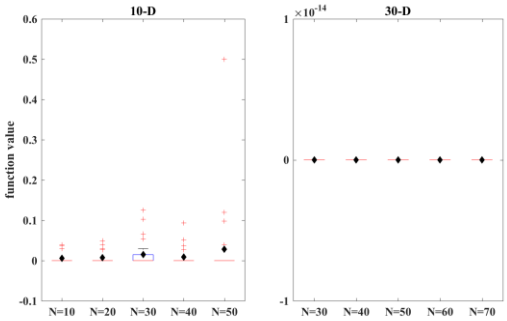
รูปที่ 4 ผลทดสอบฟังก์ชัน $f1$



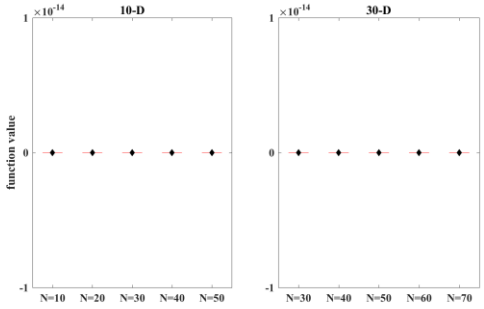
รูปที่ 5 ผลทดสอบฟังก์ชัน $f2$



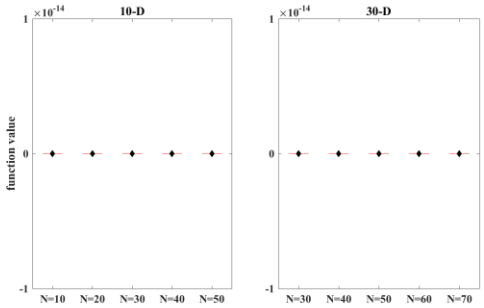
รูปที่ 6 ผลทดสอบฟังก์ชัน $f3$



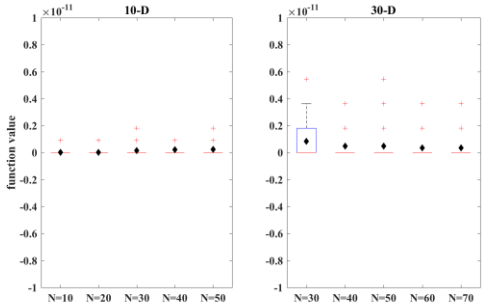
รูปที่ 7 ผลทดสอบฟังก์ชัน $f4$



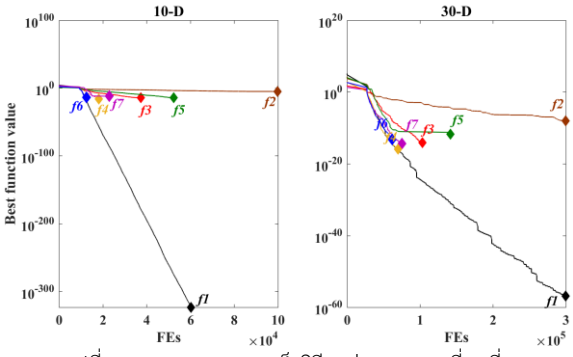
รูปที่ 8 ผลทดสอบฟังก์ชัน *f*5



รูปที่ 9 ผลทดสอบฟังก์ชัน *f*6



รูปที่ 10 ผลทดสอบฟังก์ชัน *f*7



รูปที่ 11 ผลทดสอบความเร็ววิธีหาค่าเหมาะสมที่สุดที่เสนอ

5. สรุป

บทความนี้เสนอวิธีหาค่าเหมาะสมที่สุดด้วยไฮเปอร์คิวบ์ร่วมกับวิธีค้นหาเชิงฝูงอนุภาคที่อาศัยกลยุทธ์การเรียนรู้เชิงพลวัต โดยระหว่างการค้นหาถ้าได้ค่าฟังก์ชันที่ดีขึ้น ไฮเปอร์คิวบ์สามารถเคลื่อนที่และขนาดลดลงได้ จากนั้นจะสุ่มอนุภาคใหม่จำนวน N-1 ในไฮเปอร์คิวบ์ และเก็บอนุภาคที่ดีที่สุดไว้ สามารถหาคำตอบอย่างรวดเร็วในหลายฟังก์ชัน และสมรรถนะดีกว่าวิธีค้นหา Hc-ACPSO รวมถึงดีกว่าวิธีค้นหาอื่นที่นำมาเปรียบเทียบอย่างไรก็ตามยังติดกับดักค้นหา 8 ครั้ง ในฟังก์ชัน f4 จำนวน 10 มิติ เมื่อพิจารณาตำแหน่งของค่าฟังก์ชันที่ดีที่สุดจากการค้นหา มี 1-2 มิติของตำแหน่งที่อยู่ห่างจากคำตอบ ดังนั้นถ้าเพิ่มการตรวจสอบมิติที่ติดกับดักในไฮเปอร์คิวบ์ แล้วนำไปใช้เป็นข้อมูลสำหรับสร้างตำแหน่งศูนย์กลางของไฮเปอร์คิวบ์ อาจได้สมรรถนะค้นหาที่ดีขึ้น

เอกสารอ้างอิง

[1] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39-43.

[2] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281-295, June 2006.

[3] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm and Evolutionary Computation*, vol. 24, pp. 11-24, Oct. 2015.

[4] X. Xia *et al.*, "Triple archives particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 50, no. 12, pp. 4862-4875, Dec. 2020, doi: 10.1109/TCYB.2019.2943928.

[5] B. Zhan and W. Gu, "A multi-stage adaptive sequential parameter exploration hunger games search algorithm for solving complex optimization problems," *IEEE Access*, vol. 11, pp. 100919-100947, 2023, doi: 10.1109/ACCESS .2023.3308690.

[6] M. Tunay, "Evolutionary search algorithm based on hypercube optimization for high-dimension functions," *Int. J. of Comput. And Exp. Sci. and Eng.*, vol. 6, no. 1, pp. 42-62, Mar. 2020.

[7] S. Rungsimant, "Optimization methods through selecting suitable search techniques in adaptive hypercubes," *Thailand Electrical Engineering Journal*, vol. 4, no. 3, pp. 12-18, Sep. 2024.

รังสิมันต์ สิทธิกร คณะวิศวกรรมศาสตร์และเทคโนโลยี มหาวิทยาลัยเทคโนโลยีมหานคร สนใจงานวิจัยด้าน วงจรรวม เครื่องมือวัดชนิดแม่นยำสูง เซนเซอร์ และวิธีหาค่าเหมาะสมที่สุด

ตารางที่ 1 การเปรียบเทียบสมรรถนะการค้นหาในฟังก์ชันจำนวน 10 มิติ

Algorithm	*f1*		*f2*		*f3*		*f4*		*f5*		*f6*		*f7*	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median
F_PSO	1.21E-148	-	8.97E-01	-	3.55E-15	-	4.28E-02	-	0.00E+00	-	2.48E-01	-	7.14E+02	-
HCLPSO	6.81E-45	-	1.01E+00	-	4.50E-15	-	9.85E-03	--	0.00E+00	-	0.00E+00	-	3.95E+00	-
EPSO	3.51E-46	-	1.37E-01	-	4.38E-15	-	2.18E-02	-	0.00E+00	-	0.00E+00	-	1.18E+01	-
TAPSO	4.48E-273	-	1.42E-11	-	0.00E+00	-	5.98E-02	-	1.62E-03	-	0.00E+00	-	0.00E+00	-
Hc-DSPSO	0.00E+00	0.00E+00 (30)	1.34E+00	7.90E-03 (0)	0.00E+00	0.00E+00 (30)	1.48E-02	0.00E+00 (22)	0.00E+00	0.00E+00 (30)	0.00E+00	0.00E+00 (30)	1.51E-13	0.00E+00 (26)
Hc-ACPSO (Hc2)	2.61E-70	- (0)	1.93E-25	- (0)	2.96E-15	- (5)	0.00E+00	- (30)	8.02E-07	- (0)	0.00E+00	- (30)	0.00E+00	- (30)

ตารางที่ 2 การเปรียบเทียบสมรรถนะการค้นหาในฟังก์ชันจำนวน 30 มิติ

Algorithm	*f1*		*f2*		*f3*		*f4*		*f5*		*f6*		*f7*	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median
F_PSO	5.19E-58	-	2.24E+01	-	6.39E-15	-	2.48E-07	-	0.00E+00	-	1.80E01	-	2.39E+03	-
HCLPSO	3.57E-43	-	3.65E+00	-	2.46E-14	-	9.14E-16	-	2.37E-15	-	2.55E-14	-	2.37E+01	-
EPSO	1.04E-44	-	9.93E-01	-	1.59E-14	-	1.11E-16	-	1.89E-15	-	5.92E-10	-	9.70E-13	-
TAPSO	9.31E-155	-	2.86E-10	-	5.45E-15	-	9.35E-03	-	1.27E-02	-	0.00E+00	-	1.82E-13	-
Hc-DSPSO	4.77E-50	1.49E-56 (0)	1.20E+00	1.56E-04 (0)	1.30E-15	0.00E+00 (19)	0.00E+00	0.00E+00 (30)	0.00E+00	0.00E+00 (30)	0.00E+00	0.00E+00 (30)	4.84E-13	0.00E+00 (27)
MASPE-HGSA	2.70E-176	-	3.09E+00	-	5.15E-14	-	8.20E-04	-	-	-	2.08E-13	-	1.36E+03	-

ตารางที่ 3 วิธีดัดแปลงคำตอบของฟังก์ชันด้วยการเลื่อน

Function	Original	Simple shift	Step shift	Random shift
$f1(x)=\sum_{i=1}^{D} z_i^2,\ z=x-o$ $z=[z_1,z_2,\ldots,z_D],\ x=[x_1,x_2,\ldots,x_D]$	$o=[0,0,\ldots,0]$	$o=[10,10,\ldots,10]$	$o=[2,4,\ldots,2\cdot D]$	$o=[rand_1,rand_2,\ldots,rand_D]$
$f2(x)=\sum_{i=1}^{D} 100\cdot(z_{i+1}-z_i^2)^2-(z_i-1)^2$ $z=x-o,\ z=[z_1,z_2,\ldots,z_D],$ $x=[x_1,x_2,\ldots,x_D]$	$o=[0,0,\ldots,0]$	$o=[10,10,\ldots,10]$	$o=[0.5,1,\ldots,0.5\cdot(D+1)]$	$o=[rand_1,rand_2,\ldots,rand_D]$

ตาราง 4 ผลทดสอบในฟังก์ชันที่ดัดแปลง

Algorithm		Hc-DSPSO (10-D)		Hc-DSPSO (30-D)		MASPE-HGSA (30-D)	
		Mean	Median	Mean	Median	Mean	Median
f1	Original	0.00E+00	0.00E+00 (30)	4.77E-50	1.49E-56	2.70E-176	-
	Simple	0.00E+00	0.00E+00 (30)	0.00E+00	0.00E+00 (30)	2.70E-176	-
	Step	0.00E+00	0.00E+00 (30)	8.91E-28	0.00E+00 (29)	-	-
	Random	0.00E+00	0.00E+00 (30)	0.00E+00	0.00E+00 (30)	-	-
f2	Original	1.34E+00	7.90E-03 (0)	1.20E+00	1.56E-04 (0)	3.09E+00	-
	Simple	6.69E-01	3.00E-03 (0)	1.20E+00	4.50E-03 (0)	1.16E+00	-
	Step	1.26E+00	2.37E-02	9.33E-01	2.57E-04	-	-
	Random	1.22E+00	1.04E-02	9.34E-01	3.72E-04	-	-