SWU Engineering Journal (2018) 13(2), 52-71
วารสารวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีที่ 13 ฉบับที่ 2 เดือนพฤษภาคม – สิงหาคม พ.ศ. 2561

การประยุกต์ใช้วิธีวิวัฒนาการด้วยผลต่างเพื่อเปรียบเทียบประสิทธิภาพในการแก้ปัญหาหลายวัตถุประสงค์ที่มีค่าต่อเนื่อง

Application of Differential Evolutionary Algorithm to Compare the Performance of Continuous Multi-Objective Problem Optimization

ตรัยรัตน์ เกิดโภคทรัพย์[1] ปารเมศ ชุติมา[2]*
[1]นิสิตปริญญาโท ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
[2]ศาสตราจารย์ ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
Trirat Kirdphoksap[1] Parames Chutima[2]*
[1]Student, Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University.
[2]Professor, Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University.
*Corresponding author: Email: parames.c@chula.ac.th

บทคัดย่อ

การจำแนกปัญหา (Decomposition) เป็นหลักการพื้นฐานในการแก้ปัญหาแบบหลายวัตถุประสงค์ทั่วไป ซึ่งในปัจจุบันเป็นที่สนใจและนิยมใช้ในการหาค่าที่เหมาะสมอย่างแพร่หลาย ในบทความนี้ จึงนำเสนอวิธีการวิวัฒนาการแบบปรับตัวโดยใช้ผลต่างสำหรับปัญหาหลายวัตถุประสงค์โดยยึดหลักการจำแนก (Adaptive MOEA/D hybridized with Differential Evolution: AMOEA/D-DE) ซึ่งมีแนวคิดพื้นฐานจากวิธีการเชิงวิวัฒนาการแบบหลายวัตถุประสงค์โดยยึดหลักการจำแนก (A Multi-Objective Evolutionary Algorithm based on Decomposition: MOEA/D) และวิธีเชิงวัฒนาการโดยใช้ผลต่างแบบปรับตัว (Adaptive Differential Evolution Algorithm: ADE) โดยอัลกอริทึมนี้ จะทำการจำแนกปัญหาออกเป็นปัญหาย่อย (แบ่งปัญหาหลายวัตถุประสงค์ออกเป็นปัญหาวัตถุประสงค์เดียวหลายปัญหา) และทำการหาค่าที่เหมาะสมของทุกปัญหาย่อยไปพร้อม ๆ กัน ร่วมกับการปรับค่าพารามิเตอร์ควบคุมและกลยุทธ์ที่ใช้ในการหาค่าที่เหมาะสม เพื่อให้เกิดทิศทางการค้นหาคำตอบที่ดีและหลากหลาย โดยแต่ละปัญหาย่อยจะพัฒนาคำตอบรุ่นใหม่ด้วยคำตอบเดิมของปัญหาย่อยข้างเคียง ด้วยวิธีนี้ จึงทำให้ AMOEA/D-DE มีความซับซ้อนในการดำเนินงานที่น้อยและสามารถปรับตัวให้เหมาะสมกับสถานการณ์การค้นหาคำตอบได้ ในบทความนี้ ได้นำ AMOEA/D-DE มาเปรียบเทียบกับ MOEA/D และ MODE/D ด้วยการแก้ปัญหามาตรฐานแบบหลายวัตถุประสงค์ที่มีค่าต่อเนื่อง ซึ่งจากการทดลองพบว่า AMOEA/D-DE มีประสิทธิภาพด้านอัตราการลู่เข้าที่ดีกว่าอัลกอริทึมอื่น ๆ แต่ต้องใช้เวลาการดำเนินงานที่มากกว่า เมื่อเทียบจำนวนในการพัฒนาคำตอบที่เท่ากัน

คำสำคัญ: ปัญหาแบบมากวัตถุประสงค์ การจำแนกปัญหา การหาค่าที่เหมาะสมสำหรับปัญหาหลายวัตถุประสงค์ วิธีการเชิงวิวัฒนาการ วิธีวิวัฒนาการด้วยผลต่างแบบปรับตัว

ABSTRACT

Decomposition is a basic strategy in traditional multi-objective optimization. At present, it has been interested and widely used in multi-objective evolutionary optimization. This article proposes an adaptive MOEA/D hybridized with differential evolution (AMOEA/D-DE). The concept is based on a multi-objective evolutionary algorithm based on decomposition (MOEA/D) and adaptive differential evolution algorithm (ADE). It decomposes a multi-objective problems (MOPs) into a number of scalar optimization subproblems (decomposes a MOPs into several SOPs) and optimizes them simultaneously with adjustment of control parameters and strategies to achieve a diversity of approximated true non-dominated solutions. Each subproblem is optimized by using information from its several neighboring subproblems, which makes AMOEA/D-DE had lower computational complexity and adapt itself to the current search requirements. In experiments, AMOEA/D-DE is compared with a MOEA/D and a multi-objective differential evolution algorithm based on decomposition (MODE/D) in order to solve continuous MOPs. The result from experiments show that AMOEA/D-DE outperforms the others in terms of convergence rate but it takes more time consumption when compared to the equivalent number of function evaluations.

Keyword: Many-objective problems, decomposition, multi-objective optimization, evolutionary algorithm, adaptive differential evolution.

1. บทนำ

การหาค่าที่เหมาะสมของปัญหาจริงในชีวิตประจำวัน ซึ่งเป็นปัญหาที่มีหลายวัตถุประสงค์ (Multi-Objective Optimization Problems: MOPs) จำเป็นต้องประมาณค่าที่เหมาะสมที่สุด เพื่อใช้ในการตัดสินใจดำเนินการต่าง ๆ โดยบางปัญหาอาจมีมากกว่า 3 วัตถุประสงค์ ที่ต้องพิจารณาพร้อมกัน แต่อาจขัดแย้งกันในบางวัตถุประสงค์ เรียกปัญหานี้ว่า ปัญหาแบบมากวัตถุประสงค์ (Many-Objective Optimization Problems: MaOPs) ทำให้ปัญหามีรูปแบบของคำตอบที่เป็นไปได้จำนวนมากหรืออาจเป็นจำนวนอนันต์, ใช้เวลาในการค้นหาคำตอบมาก และเป็นไปได้น้อยที่จะพบคำตอบที่เหมาะสมทั้งหมด ทำให้การใช้วิธีการเชิงวิวัฒนาการแบบหลายวัตถุประสงค์ (A Multi-Objective Evolutionary Algorithms: MOEAs) แบบเดิม (เช่น MOGA, NSGA II, COIN) ในการแก้ปัญหา MaOPs มีประสิทธิภาพลดลง ไม่ดีเท่าที่ควร [1]

การแก้ปัญหา MaOPs อย่างง่ายที่สุด ทำได้ด้วยการลดจำนวนวัตถุประสงค์ลง โดยการรวมวัตถุประสงค์ที่ไม่ขัดแย้งกันไว้เป็นเป้าหมายเดียว [2] อย่างไรก็ตาม วิธีการนี้ไม่สามารถใช้กับหลายปัญหาในทางปฏิบัติได้ ด้วยเหตุนี้วิธีการแบบใหม่จึงถูกพัฒนาขึ้น เรียกว่าวิธีการเชิงวิวัฒนาการแบบมากวัตถุประสงค์ (A Many-Objective Evolutionary Algorithms: MaOEAs) โดยหนึ่งในอัลกอริทึมที่โดดเด่นของ MaOEAs คือวิธีการเชิงวิวัฒนาการแบบหลายวัตถุประสงค์โดยยึดหลักการจำแนก (A Multi-Objective Evolutionary Algorithm based on Decomposition: MOEA/D) ซึ่งถูกพัฒนาโดย Zhang and Li [3] วิธีการดำเนินงานของ MOEA/D จะทำการจำแนก MaOP ออกเป็นปัญหาแบบวัตถุประสงค์เดียว (ปัญหาย่อย) หลายปัญหา และทำการแก้ปัญหาย่อยไปพร้อมกัน แต่ละปัญหาย่อยจะได้คำตอบที่ดีที่สุด

การประยุกต์ใช้อัลกอริทึมในการ MaOPs เป็นการประมาณค่าคำตอบ ซึ่งจำเป็นต้องเลือกคำตอบสุดท้ายที่จะ

ไปดำเนินการกับ MaOPs ซึ่งส่วนใหญ่มีจำนวนคำตอบมากและไม่รู้ขอบเขตที่แน่ชัด อีกทั้งยังใช้เวลามากในการค้นหาคำตอบ MOEA/D ซึ่งเป็นอัลกอริทึมที่โดดเด่นของ MaOEAs จึงเป็นทางเลือกที่ดี เนื่องจาก MOEA/D สามารถค้นหากลุ่มคำตอบที่ดีที่สุด (Pareto optimal solutions) ได้ในเจเนอเรชั่นเดียว ไม่เกิดการค้นหาแบบกระจุกตัว เพราะแบ่งการค้นหาเป็นปัญหาย่อย ทำให้มีการกำหนดทิศทางและขอบเขตของคำตอบในแต่ละปัญหาย่อยที่ชัดเจน รวมถึงความซับซ้อนและเวลาในการค้นหาคำตอบน้อยกว่าอัลกอริทึมอื่น ๆ เช่น วิธีเชิงพันธุกรรมแบบการจัดลำดับที่ไม่ถูกครอบงำ II (Non-Dominated Sorting Genetic Algorithm II: NSGA II) และวิธีการหาค่าเหมาะสมที่สุดแบบกลุ่มอนุภาค (Particle Swarm Optimization: PSO) อีกทั้งยังคงคำตอบที่หลากหลาย โดย MOEA/D จะเห็นผลลัพธ์ที่ดีและแตกต่างกับอัลกอริทึมอื่น ๆ อย่างชัดเจน เมื่อมีวัตถุประสงค์ตั้งแต่ 4 ถึง 15 วัตถุประสงค์ [3]

ถึงแม้ว่า MOEA/D จะสามารถแก้ MaOPs ได้หลากหลายรูปแบบ แต่วิธีการพัฒนาคำตอบเชิงพันธุกรรมแบบทั่วไป (Crossover และ Mutation ของ GA) มีความละเอียดที่น้อย (ให้ค่าที่ไม่ต่อเนื่อง) ในการค้นหาคำตอบเมื่อเทียบกับวิธีการอื่น ๆ ส่งผลให้ได้คำตอบที่ไม่หลากหลายเมื่อใช้กับปัญหาแบบมากวัตถุประสงค์ที่มีค่าต่อเนื่อง (Continuous MaOPs) Li and Zhang [4] จึงออกแบบวิธีการวิวัฒนาการโดยใช้ผลต่างแบบหลายวัตถุประสงค์โดยยึดหลักการจำแนก (A Multi-Objective Differential Evolution Algorithm based on Decomposition: MODE/D) เพื่อใช้สำหรับปัญหา Continuous MaOPs ซึ่งยังคงใช้การคัดเลือกคำตอบที่ดีของแต่ละปัญหาย่อยตามแบบ MOEA/D แต่เปลี่ยนวิธีการพัฒนาคำตอบโดยใช้กระบวนการวิวัฒนาการโดยใช้ผลต่าง (Differential Evolution Operator: DE Operator) แทน โดยกำหนดให้ใช้ผลต่างของ ปัญหาย่อยข้างเคียงของแต่ละปัญหาย่อยในการพัฒนาคำตอบ เนื่องจากปัญหาย่อยข้างเคียงจะมีความสัมพันธ์ของค่าวัตถุประสงค์ที่ใกล้เคียงกัน โดยหลักการของ DE Operator จะใช้การกลายพันธุ์ด้วยผลต่างของคำตอบในการหาคำตอบจุดใหม่บนแกนพิกัด (Coordinate) หรือเรียกว่าระยะห่างระหว่างจุดของแต่ละแกนพิกัด (Distances) ด้วยวิธีการนี้ จึงทำให้อัลกอริทึมมีประสิทธิภาพในการค้นหาคำตอบที่มีค่าต่อเนื่องมากขึ้นได้

ดังนั้นในงานบทความนี้ ผู้วิจัยจึงออกแบบวิธีการวิวัฒนาการแบบปรับตัวโดยใช้ผลต่างสำหรับปัญหาหลายวัตถุประสงค์โดยยึดหลักการจำแนก (Adaptive MOEA/D hybridised with Differential Evolution: AMOEA/D-DE) โดยนำ SaDE, jDE และ JADE มาปรับปรุงและพัฒนา MOEA/D เพื่อให้อัลกอริทึมมีความเหมาะสมกับปัญหาของบทความนี้ และสามารถเรียนรู้ทิศทางของคำตอบที่กำลังเปลี่ยนแปลงไปทางที่ดีและไม่ดีได้ โดยกลยุทธ์ในการพัฒนาคำตอบของ DE Operator จะถูกเลือกใช้ตามสถานการณ์ที่เหมาะสมในการค้นหาคำตอบ และมีการปรับพารามิเตอร์ควบคุม เพื่อให้การค้นหาคำตอบมีความหลากหลายและความต่อเนื่องของคำตอบมากขึ้น อีกทั้งยังคงการค้นหาที่มีทิศทางของคำตอบที่ชัดเจน วิธีในการพัฒนาคำตอบที่ดี และเวลาในการค้นหาคำตอบและความซับซ้อนน้อย

โดยในบทความนี้ จะอธิบายเกี่ยวกับปัญหาที่มีมากกว่าหนึ่งวัตถุประสงค์ ในหัวข้อที่ 2, หลักการพื้นฐานของ MOEA/D ในหัวข้อที่ 3, หลักการพื้นฐานของ DE ในหัวข้อที่ 4, แสดงทฤษฎีและขั้นตอนการดำเนินงานของอัลกอริทึมในงานวิจัย ในหัวข้อที่ 5, ผลการทดลอง ในหัวข้อที่ 6 และสรุปผล ในหัวข้อที่ 7

2. ปัญหาแบบหลายวัตถุประสงค์

ปัญหาที่มีหลายวัตถุประสงค์ (MOPs) เป็นปัญหาที่มีจำนวนฟังก์ชันวัตถุประสงค์มากกว่า 1 ขึ้นไป ซึ่งสามารถเป็นกรณีหาค่าน้อยที่สุดหรือมากที่สุดก็ได้ แสดงรูปแบบพื้นฐานกรณีหาค่าน้อยที่สุดดังนี้

$$\text{Minimize } F(x) = \left(f_1(x), f_2(x), \dots, f_M(x)\right) \quad (1)$$

subject to $x \in R^n$

โดยที่ $x = (x_1, x_2, \dots, x_n)$ เป็นเวกเตอร์ตัวแปรตัดสินใจหรือเวกเตอร์คำตอบ ซึ่งเป็นสมาชิกของ R^n และ

$f_k(x)$ เป็นค่าวัตถุประสงค์ที่ k ของเวกเตอร์คำตอบ x ซึ่งเป็นสมาชิกของ R^M $(k = 1, \dots, M)$

เนื่องจากแต่ละวัตถุประสงค์ของปัญหา MOPs อาจขัดแย้งกัน จึงจำเป็นต้องใช้การคัดเลือกเวกเตอร์คำตอบที่ให้ความสำคัญกับทุกวัตถุประสงค์ไปพร้อมกัน ด้วยวิธีเชิงกลุ่มที่ดีที่สุด (Pareto-optimal approach)

ตัวอย่างการคัดเลือกคำตอบ เช่น $v, u \in R^M$ ซึ่งจะบอกว่า v ครอบงำ u ได้ ก็ต่อเมื่อ $f_k(v) \leq f_k(u)$ ทุกค่า $(k = 1, \dots, M)$ และ $f_k(v) < f_k(u)$ อย่างน้อย 1 ค่า

โดยเวกเตอร์คำตอบ x^* ใด ๆ ที่ไม่มีใครสามารถครอบงำและไม่สามารถครอบงำกันเองได้ จะถูกเรียกว่า Pareto-optimal solution (PS) และเวกเตอร์วัตถุประสงค์ของ x^* จะถูกเรียกว่า Pareto-optimal front (PF) [5]

3. วิธีการเชิงวิวัฒนาการแบบยึดหลักการจำแนก

วิธีการเชิงวิวัฒนาการแบบหลายวัตถุประสงค์โดยยึดหลักการจำแนก (MOEA/D) [3] มีหลักแนวคิดในการรวมรวบคำตอบ [6] โดยจะแบ่ง MOPs ออกเป็นปัญหาแบบวัตถุประสงค์เดียว (Single-objective problems: SOPs) หลายปัญหา (หรือที่เรียกว่าปัญหาย่อย: subproblem) ด้วยการกำหนดค่าถ่วงน้ำหนักเชิงเส้น (λ) ที่แตกต่างกันให้กับฟังก์ชันวัตถุประสงค์ของแต่ละปัญหาย่อย ซึ่งแต่ละปัญหาย่อยจะถูกกำหนดความสัมพันธ์ด้วยระยะห่างของค่าถ่วงน้ำหนักที่ต่างกัน (เรียกว่าปัญหาย่อยข้างเคียง: Neighborhood) อีกทั้งยังพัฒนาคำตอบไปพร้อม ๆ กันโดยใช้คำตอบจากปัญหาย่อยข้างเคียง ซึ่งผลการทดลองพบว่า MOEA/D ให้ประสิทธิภาพที่ดีกับปัญหาหลายรูปแบบ [3, 7, 8] MOEA/D ถูกจัดเป็นเครื่องมือแก้ปัญหาแบบหลายวัตถุประสงค์ที่ดีที่สุดสำหรับปัญหา The Special Session on Performance Assessment of Multi-objective Optimization Algorithms of the CEC 2009 MOEA Competition (ผลถูกแสดงบนเว็บไซต์ http://cswww.essex.ac.uk/staff/zhang/moeacompetition09.htm)

ในปัจจุบัน มีหลากหลายวิธีในการแปลง MOP เป็น SOP โดยในงานนี้ ได้เลือกใช้วิธีเทบบีเชฟฟ์ (Tchebycheff approach) ในการคัดเลือกคำตอบของอัลกอริทึมแบบยึดหลักการจำแนก แสดงสูตรคำนวณดังนี้

$$\text{minimize } g^{te}(x|\lambda^i, z^*) = \max_{1 \leq k \leq M} \{\lambda_k^i | f_k(x) - z_k^*|\} \quad (2)$$

เมื่อ $g^{te}(x|\lambda^i, z^*)$ คือผลต่างมากที่สุดระหว่างค่าวัตถุประสงค์ของสตริง i กับค่าเป้าหมายของเวกเตอร์ค่าถ่วงน้ำหนัก λ^i ของปัญหาย่อยที่ i, z^* คือเวกเตอร์ค่าเป้าหมาย โดยที่ $z^* = z_1^*, z_2^*, \dots, z_M^*$ กรณีหาค่าของคำตอบที่น้อยที่สุด $z_k^* = \min\{f_k(x)\}$ กรณีหาค่าของคำตอบที่มากที่สุด $z_k^* = \max\{f_k(x)\}$, λ^i คือเวกเตอร์ค่าถ่วงน้ำหนักปัญหาย่อยที่ i โดยที่ $\lambda^i = \{\lambda_1^i, \dots, \lambda_M^i\}$, M คือจำนวนฟังก์ชันวัตถุประสงค์

4. วิธีวิวัฒนาการโดยใช้ผลต่าง

วิธีวิวัฒนาการโดยใช้ผลต่าง (Differential Evolution: DE) เป็นเครื่องมือแก้ปัญหาโดยใช้พื้นฐานของประชากร (Population-based Optimizer) ถูกพัฒนาโดย Storn and Price [9] มีทิศทางการค้นหาคำตอบที่แตกต่างและอิสระต่อกันในแต่ละเวกเตอร์คำตอบของประชากร เริ่มต้นปัญหาโดยการสร้างเวกเตอร์เป้าหมาย (Target vector) ในหลายจุดพิกัดแบบสุ่มเลือก ด้วยขอบเขตพารามิเตอร์ที่กำหนดไว้ จากนั้น DE จะเข้าสู่กระบวนการพัฒนาคำตอบ ได้แก่ การกลายพันธุ์ (Mutation), การครอสโอเวอร์ (Recombination) และการคัดเลือกคำตอบ (Selection)

กระบวนการวิวัฒนาการโดยใช้ผลต่าง (DE Operator) ให้ประสิทธิภาพที่ดีกว่ากระบวนการเชิงพันธุกรรมอื่น ๆ (GA Operator เช่น SBX และ UNDX) ในการแก้ปัญหาที่มีวัตถุประสงค์เดียว (Single-Objective Optimization) ด้วยเหตุผลนี้ จึงนำ DE Operator เข้ามาพัฒนาคำตอบแทนวิธี GA Operator เดิมใน MOEA/D [7]

เนื่องจาก DE เป็นอัลกอริทึมที่มีกระบวนการ รูปแบบกลยุทธ์ การคัดเลือกคำตอบ และพารามิเตอร์ควบคุมที่หลากหลาย Venske, et al. [10] ได้แบ่งประเภท DE

ออกเป็น 2 แบบ ได้แก่ วิธีวิวัฒนาการโดยใช้ผลต่างแบบคลาสสิก (Classical DE) และวิธีวิวัฒนาการโดยใช้ผลต่างแบบปรับตัว (Adaptive DE)

4.1 วิธีวิวัฒนาการโดยใช้ผลต่างแบบคลาสสิก

วิธีวิวัฒนาการโดยใช้ผลต่างมีพารามิเตอร์ควบคุม 3 ตัว ได้แก่ N, F และ CR โดย N คือจำนวนหรือขนาดของประชากร ส่วนค่าปัจจัยขยายผลต่าง (F) ใช้ในการปรับขนาดของค่าผลต่างระหว่างสองเวกเตอร์สุ่ม ซึ่งจะถูกนำมาบวกเพิ่มกับเวกเตอร์พื้นฐาน (Base vector) และจะได้เวกเตอร์กลายพันธุ์ (Mutant vector: v) โดยเรียกกระบวนการนี้ว่า การกลายพันธุ์ด้วยผลต่าง (Differential mutation) จากนั้นจะทำการครอสโอเวอร์ระหว่างเวกเตอร์กลายพันธุ์ (v) กับเวกเตอร์เป้าหมาย (Target vector: x) ด้วยอัตราการครอสโอเวอร์ (CR) โดยสุดท้ายจะได้เวกเตอร์ทดลอง (Trial vector: u) เป็นเวกเตอร์คำตอบรุ่นใหม่ เพื่อนำมาเปรียบเทียบและคัดเลือกคำตอบที่ดี ส่วนประกอบของเวกเตอร์คือ จุดพิกัด (Coordinate) และจำนวนพิกัดในแต่ละเวกเตอร์คือ ขนาดเวกเตอร์ (n)

DE Operator เป็นการในการค้นหาคำตอบแบบเวกเตอร์ โดยใช้ผลต่างระหว่างจุดบนแกน (Coordinate) เดียวกันและพารามิเตอร์ควบคุม 2 ตัว เป็นตัวกำหนดทิศทางและขนาดของเวกเตอร์ x ใด ๆ ให้เปลี่ยนแปลงไป [11] จึงทำให้ DE Operator เหมาะในการแก้ปัญหา Continuous MOPs และปัญหาที่มีรูปแบบที่เป็นไปได้ของคำตอบหลากหลายได้ดี DE จะประกอบด้วย 3 ขั้นตอนหลัก ๆ ในการพัฒนาคำตอบ ได้แก่ การกลายพันธุ์ด้วยผลต่าง (Differential mutation), รีคอมบิเนชัน (Recombination) และการคัดเลือก (Selection)

4.2 วิธีวิวัฒนาการโดยใช้ผลต่างแบบปรับตัว

สำหรับวิธีวิวัฒนาการโดยใช้ผลต่างแบบคลาสสิก (Classical DE) จะกำหนดพารามิเตอร์ทั้งหมดเป็นค่าคงที่ไม่มีเปลี่ยนแปลงในระหว่างการดำเนินงานอัลกอริทึม จึงมีการคิดค้นอัลกอริทึมแบบปรับตัว (Adaptive algorithms) ซึ่งสามารถปรับเปลี่ยนค่าพารามิเตอร์หรือวิธีการพัฒนาคำตอบในระหว่างดำเนินงานอัลกอริทึมได้ ตัวอย่างเช่น SaDE, JADE, jDE, EPSDE, ENS-MOEA/D และ FADE โดยสำหรับวิธีวิวัฒนาการโดยใช้ผลต่าง จะแบ่งการปรับตัวออกเป็น 2 รูปแบบ ได้แก่ การปรับตัวด้วยพารามิเตอร์ควบคุม (Parameter adaptation) และการปรับตัวด้วยกลยุทธ์ (Strategy adaptation)

วิธีวิวัฒนาการแบบปรับตัว (Adaptive Differential Evolution) ถูกพิสูจน์ว่าเป็นอัลกอริทึมที่มีการลู่เข้าของคำตอบที่รวดเร็วและน่าเชื่อถือ (เสถียรและความแปรปรวนต่ำ) กว่า Classical DE [12, 13] โดยมีรูปแบบกลยุทธ์ของ DE ซึ่งเป็นโครงสร้างพื้นฐานสำหรับกระบวนการปรับตัวด้วยพารามิเตอร์ (Parameter adaptation operations) เป็นส่วนสำคัญต่อประสิทธิภาพอัลกอริทึมอย่างมาก Qin and Suganthan [12], Teo [14] ได้พัฒนา Adaptive DE จากโครงสร้างของ Classical DE ด้วยกลยุทธ์ DE/rand/1/bin ซึ่งมีความแข็งแรงและเป็นที่นิยมอย่างมาก แต่กลยุทธ์นี้ มีอัตราการลู่เข้าของคำตอบที่ช้า ส่วน Huang, et al. [15] ใช้กลยุทธ์ DE/rand/1/bin ไปพร้อมกับ DE/current-to-best/1/bin ด้วยการสุ่มเลือกหนึ่งกลยุทธ์ในการพัฒนาคำตอบ และทำการปรับค่าความน่าจะเป็นในการเลือกใช้กลยุทธ์จากจำนวนความสำเร็จในการสร้างคำตอบที่ดีของแต่ละกลยุทธ์

SaDE เป็นวิธีวิวัฒนาการโดยใช้ผลต่างแบบปรับตัวด้วยกลยุทธ์ ถูกเสนอโดย Qin and Suganthan [12] ซึ่งวิธีนี้ จะมีสองกลยุทธ์ในการกลายพันธุ์คำตอบ ได้แก่ DE/rand/1 และ DE/current-to-best/1 โดยจะปรับใช้กลยุทธ์ด้วยค่าความน่าจะเป็น ซึ่งได้จากสัดส่วนความสำเร็จที่เกิดขึ้นจากการสร้างคำตอบรุ่นใหม่ที่ดีใน 50 เจเนอเรชั่นก่อนหน้า โดยขั้นตอนการปรับตัวนี้ จะสามารถเรียนรู้และปรับใช้กลยุทธ์ที่เหมาะสมกับสถานการณ์ที่แตกต่างกันได้ ส่วนด้านพารามิเตอร์ ปัจจัยการกลายพันธุ์จะถูกสุ่มค่าอิสระด้วยการแจกแจกแบบปกติ (Normal distribution) ที่ค่าเฉลี่ยเท่ากับ 0.5 และส่วนเบี่ยงเบนมาตรฐานเท่ากับ 0.3 โดยมีขอบเขตในช่วงระหว่าง 0 ถึง 2 ด้วยวิธีนี้ จะสามารถค้นหาคำตอบในบริเวณใกล้เคียงคำตอบเดิม (F_i มีค่าน้อย) และสามารถค้นหาคำตอบที่

แตกต่างจากคำตอบเดิมได้ (F_i มีค่ามาก) ส่วนความน่าจะเป็นในการครอสโอเวอร์ จะถูกสุ่มค่าอิสระด้วยการแจกแจกแบบปกติที่ค่าเฉลี่ยเท่ากับ CR_m และส่วนเบี่ยงเบนมาตรฐานเท่ากับ 0.1 โดย SaDE จะคงค่า F_i และ CR_i เป็นจำนวน 5 เจเนอเรชั่น ก่อนจะสุ่มค่าใหม่จากค่าเฉลี่ยเริ่มต้น CR_m เท่ากับ 0.5 ซึ่งจะถูกปรับปรุงค่าตามทิศทางการค้นหาคำตอบในทุก ๆ 25 เจเนอเรชั่น

Brest, et al. [13] ได้นำเสนอวิธีวิวัฒนาการโดยใช้ผลต่างแบบใหม่ (A new adaptive DE: jDE) ซึ่งมีพื้นฐานมาจาก Classical DE กลยุทธ์ DE/rand/1/bin โดย jDE จะกำหนดจำนวนประชากรคงที่ตลอดการดำเนินงานอัลกอริทึม และปรับค่าพารามิเตอร์ควบคุม F_i และ CR_i ตามความน่าจะเป็น τ_1 และ τ_2 ตามลำดับ ในการเริ่มต้นกระบวนการ จะกำหนดค่า F_i เท่ากับ 0.5 มีขอบเขตในช่วงระหว่าง 0.1 ถึง 1 และ CR_i เท่ากับ 0.9 มีขอบเขตในช่วงระหว่าง 0 ถึง 1 ซึ่ง jDE จะทำการสุ่มค่า F_i และ CR_i ด้วยการแจกแจงแบบยูนิฟอร์ม (Uniform distribution) ผลการทดลองพบว่า jDE ให้ประสิทธิภาพที่ดีกว่า Classical DE กลยุทธ์ DE/rand/1/bin, FEP and CEP [16], the adaptive LEP, Best Levy [17] และ FADE [18]

Zhang and Sanderson [19] ได้นำเสนอ JADE ซึ่งเป็นวิธีวิวัฒนาการโดยใช้ผลต่างที่ทำการพัฒนากลยุทธ์ขึ้นมาใหม่ เพื่อเพิ่มประสิทธิภาพในการค้นหาคำตอบให้หลากหลายและลู่เข้าอย่างรวดเร็ว โดยได้แก่กลยุทธ์ DE/current-to-*p*best/1 และ DE/rand-to-*p*best/1 และแบ่งออกเป็นอย่างละ 2 ประเภท คือ มีตัวเลือกการสุ่มภายนอก (Optional external archive: EA) และไม่มี กลยุทธ์ในการค้นหาคำตอบของ JADE มีหลักการทำงานเช่นเดียวกับ DE/current-to-best/1 แบบปกติ แต่จะสามารถเลือกสุ่มเวกเตอร์ผลต่าง (Difference vector) จาก EA ได้ ทำให้มีทิศทางที่เป็นไปได้ของผลต่างเวกเตอร์หลากหลายมากขึ้น JADE มีการปรับค่าพารามิเตอร์ควบคุมคล้ายคลึงกับ Adaptive DE อื่น ๆ โดย F_i จะถูกสุ่มค่าอิสระด้วยการแจกแจกแบบโคชี (Cauchy distribution) ที่ค่าเฉลี่ยเริ่มต้นเท่ากับ 0.5 และส่วนเบี่ยงเบนมาตรฐานเท่ากับ 0.1 โดยมีขอบเขตในช่วงระหว่าง 0 ถึง 1 และ CR_i จะถูกสุ่มค่าอิสระด้วยการแจกแจกแบบยูนิฟอร์ม ที่ค่าเริ่มต้นเท่ากับ 0.9 โดยมีขอบเขตในช่วงระหว่าง 0 ถึง 1 ผลการทดลองพบว่า JADE มีการลู่เข้าของคำตอบที่รวดเร็วกว่า Adaptive DE ทั้งหมด อีกทั้งยังมีการปรับพารามิเตอร์ได้ด้วยตนเอง

5. อัลกอริทึม AMOEA/D-DE

วิธีการวิวัฒนาการแบบปรับตัวโดยใช้ผลต่างสำหรับปัญหาหลายวัตถุประสงค์โดยยึดหลักการจำแนก (AMOEA/D-DE) มีแนวคิดและแรงบันดาลใจจากเครื่องมือการแก้ปัญหาหลายวัตถุประสงค์แบบดั้งเดิม, MOEA/D และ MODE/D [3, 7, 8] โดยมี MOEA/D เป็นอัลกอริทึมพื้นฐานในการออกแบบและพัฒนาอัลกอริทึม

ในบทความนี้ ได้นำ SaDE, jDE และ JADE ซึ่งเป็นวิธีเชิงวิวัฒนาการโดยใช้ผลต่างแบบปรับตัว (Adaptive DE: ADE) มาประยุกต์ใช้กับ MOEA/D เนื่องจาก DE Operator เป็นวิธีการพัฒนาคำตอบที่ถูกพิสูจน์ว่าดีสำหรับปัญหาแบบวัตถุประสงค์เดียว อีกทั้งการนำ ADE เข้ามาปรับใช้ ยังทำให้อัลกอริทึมสามารถเรียนรู้ทิศทางของระบบว่าเปลี่ยนแปลงไปในทิศทางใด และสามารถปรับตัวให้เข้ากับสถานการณ์ของอัลกอริทึมในการค้นหาคำตอบได้

5.1 การปรับตัวด้วยกลยุทธ์

การปรับตัวด้วยกลยุทธ์ (Strategy Adaptation) มีแนวคิดจาก SaDE ของ Huang, et al. [15] ซึ่งเป็นอัลกอริทึมที่มีการเรียนรู้พฤติกรรมของระบบ และสุ่มเลือกกลยุทธ์ในการค้นหาคำตอบจากค่าความน่าจะเป็นให้เหมาะสมกับสถานการณ์

ในบทความนี้ ได้พัฒนา *s*best (Best so far solution) จาก best (Best solution) แบบเดิม เพื่อให้เกิดประสิทธิภาพที่ดีขึ้นในการประยุกต์ใช้กับ MOEA/D โดยหลักการพื้นฐานของ *s*best จะเหมือนกับ best ทุกอย่าง คือเลือกเวกเตอร์ที่ให้คำตอบที่ดีที่สุดมาเป็นเวกเตอร์เริ่มต้น (Base vector) แล้วพัฒนาคำตอบด้วยผลต่างของเวกเตอร์สุ่ม เมื่อขึ้นเจเนอเรชั่นถัดไป ก็จะทำการเลือกเวกเตอร์ที่ให้คำตอบที่ดี

ใหม่อีกครั้ง ซึ่งอาจจะดีหรือแย่กว่ารอบก่อนหน้าก็ได้ ดังนั้น *s*best จึงถูกเพิ่มการจัดเก็บเวกเตอร์ที่ให้คำตอบที่ดีที่สุดไว้ และจะถูกแทนที่คำตอบเมื่อมีคำตอบที่ดีกว่าเท่านั้น โดยผู้วิจัย ได้พัฒนาและเลือกใช้กลยุทธ์ทั้งหมด 3 แบบ ดังนี้

1) กลยุทธ์ที่ 1: "DE/rand/1/bin"

$$v^i = x^{r1} + F(x^{r2} - x^{r3}) \quad (3)$$

2) กลยุทธ์ที่ 2: "DE/*s*best/1/bin"

$$v^i = sbest^i + F(x^{r1} - x^{r2}) \quad (4)$$

3) กลยุทธ์ที่ 3: "DE/rand-to-*s*best/1/bin"

$$v^i = x^{r1} + \gamma(sbest^i - x^{r1}) + F(x^{r2} - x^{r3}) \quad (5)$$

กลยุทธ์ที่ 1: DE/rand/1/bin เป็นกลยุทธ์แบบดั้งเดิมของวิธีวิวัฒนาการโดยใช้ผลต่าง (DE) และนิยมใช้กันอย่างแพร่หลาย โดยกลยุทธ์นี้ จะทำการสุ่มเลือกเวกเตอร์พื้นฐาน (Base vector) จำนวน 1 เวกเตอร์ และสุ่มเลือกเวกเตอร์ผลต่าง (Difference vector) จำนวน 2 เวกเตอร์ มาใช้ในการคำนวณหาเวกเตอร์คำตอบกลายพันธุ์ (Mutant vector) และทำการยูนิฟอร์มครอสโอเวอร์ จะได้เวกเตอร์คำตอบรุ่นใหม่หรือเวกเตอร์ทดลอง (Trial vector) นำมาใช้ประเมินและคัดเลือกคำตอบ [9]

กลยุทธ์ที่ 2: DE/*s*best/1/bin เป็นกลยุทธ์ที่มีความคล้ายคลึงกับ DE/best/1/bin แบบดั้งเดิม แต่กลยุทธ์นี้ จะจัดเก็บค่าที่ดีที่สุด (Best so far: $sbest$) ของแต่ละปัญหาย่อยไว้ โดยหากไม่มีเวกเตอร์คำตอบที่ดีกว่าเวกเตอร์ $sbest^i$ ของปัญหาย่อยที่ i จะไม่เกิดการแทนที่หรือเปลี่ยนแปลงคำตอบของเวกเตอร์ $sbest$ ดังนั้น กลยุทธ์นี้นอกจากจะมีการลู่เข้าของคำตอบที่รวดเร็วเหมือนกลยุทธ์แบบดั้งเดิมแล้ว อีกทั้งยังคงเก็บค่าที่ดีที่สุด ไม่เกิดการสูญหายของคำตอบที่ดีจากการแทนที่ของปัญหาย่อยข้างเคียง จึงทำให้ทิศทางในการค้นหาคำตอบไม่เกิดการเปลี่ยนแปลงแบบกะทันหัน

กลยุทธ์ที่ 3: DE/rand-to-*s*best/1/bin เป็นกลยุทธ์ที่มีความคล้ายคลึงกับ DE/rand-to-best/1/bin แบบดั้งเดิม โดยใช้หลักการเก็บค่าที่ดีที่สุดไว้เช่นเดียวกับ DE/*s*best/1/bin โดยกลยุทธ์นี้ มีเวกเตอร์พื้นฐาน 2 เวกเตอร์ คือเวกเตอร์สุ่มและเวกเตอร์ $sbest$ ถูกถ่วงน้ำหนักด้วยค่า γ เพื่อหาจุดพิกัดเริ่มต้นในการพัฒนาคำตอบเช่นเดียวกับกลยุทธ์อื่น ๆ โดย DE/rand-to-*s*best/1/bin จะมีการค้นหาคำตอบที่หลากหลายแต่ไม่มากเท่า DE/rand/1/bin และมีการลู่เข้าของคำตอบที่รวดเร็วแต่ไม่เท่า DE/*s*best/1/bin จึงทำให้กลยุทธ์นี้ มีการค้นหาคำตอบอยู่ระหว่าง 2 กลยุทธ์ข้างต้น และด้วยเหตุนี้ อาจทำให้ช่วยสร้างรูปแบบของคำตอบที่หลากหลายและดีขึ้นได้

การเริ่มต้นดำเนินงานของการปรับตัวด้วยกลยุทธ์ ในแต่ละกลยุทธ์จะมีความน่าจะเป็นเท่ากัน โดยจะเกิดการเรียนรู้และปรับความน่าจะเป็นของกลยุทธ์ไปเรื่อย ๆ จากจำนวนความสำเร็จและไม่สำเร็จที่เกิดขึ้น [20] แสดงสูตรคำนวณซึ่งผู้วิจัยได้ดัดแปลงบางส่วนเพื่อให้เหมาะกับอัลกอริทึมที่ใช้ในงานวิจัย ดังนี้

$$p_{s,g+1} = p_{\min} + (1 - S \times p_{\min}) \cdot \left[(1-\alpha)p_{s,g} + \alpha \cdot \left(\frac{Suc_{s,g}}{\sum_{s=1}^{S} Suc_{s,g}}\right)\right] \quad (6)$$

$$Suc_{s,g} = \frac{ns_{s,g}}{ns_{s,g} + nf_{s,g}} + \varepsilon \quad (7)$$

$$ns_{s,g} = \sum_{i=1}^{N}\sum_{j=1}^{N} DSR_{i,j,g}^{s} \quad (8)$$

$$nf_{s,g} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(1 - SR_{i,j,g}^{s}\right) \quad (9)$$

เมื่อ $p_{s,g}$ คือค่าความน่าจะเป็นในการเลือกใช้กลยุทธ์ที่ s ในเจเนอเรชั่นที่ g (The probability of choosing s^{th} strategy) โดยที่ $s = 1,2,\dots,S$, $Suc_{s,g}$ คือสัดส่วนความสำเร็จในการสร้างคำตอบรุ่นใหม่ของกลยุทธ์ที่ s ในเจเนอเรชั่นที่ g (The rate of offspring vectors successfully creating by the s^{th} strategy), $ns_{s,g}$ คือจำนวนความสำเร็จในการสร้างคำตอบรุ่นใหม่ของกลยุทธ์ที่ s เจเนอเรชั่นที่ g (Number of strategy s successfully creating survival offspring vectors in generation g), $nf_{s,g}$ คือจำนวนความไม่สำเร็จในการสร้างคำตอบรุ่นใหม่

ของกลยุทธ์ที่ s ในเจเนอเรชั่นที่ g (Number of strategy s unsuccessfully creating survival offspring vectors in generation g), $SR^s_{i,j,g}$ คือการแทนที่คำตอบ (Solution Replacement) ซึ่งได้จากกลยุทธ์ที่ s จะมีค่าเท่ากับ 1 เมื่อเกิดความสำเร็จในการสร้างคำตอบรุ่นใหม่ในปัญหาย่อยที่ i เวกเตอร์ที่ j เจเนอเรชั่นที่ g (Strategy s successfully creating offspring vector that are better than or equal to the parent vector and its neighboring solutions of subproblem i), $DSR^s_{i,j,g}$ คือการแทนที่คำตอบแบบครอบงำ (Dominated Solution Replacement) ซึ่งได้จากกลยุทธ์ที่ s จะมีค่าเท่ากับ 1 เมื่อเกิดความสำเร็จในการสร้างคำตอบรุ่นใหม่แบบครอบงำคำตอบเดิมในปัญหาย่อยที่ i เวกเตอร์ที่ j เจเนอเรชั่นที่ g (Strategy s successfully creating offspring vector that are better than the parent vector and its neighboring solutions of subproblem i), p_{min} คือค่าความน่าจะเป็นต่ำสุดของกลยุทธ์ (The minimum probability of each strategy), S คือจำนวนกลยุทธ์ (Number of strategy), α คืออัตราการปรับตัว (Adaptation rate) มีค่าระหว่าง 0 ถึง 1, ε คือค่าคงที่เพื่อหลีกเลี่ยงกรณีหาค่าไม่ได้ โดยในงานวิจัยนี้ใช้ค่า 0.0001 (The constant value is used to avoid possible null rates of success), $sbest^i$ คือเวกเตอร์คำตอบที่ดีที่สุด (Best so far vector) สำหรับปัญหาย่อยที่ i โดยที่ $sbest = (sbest_1, \dots, sbest_n)$

5.2 การปรับตัวด้วยพารามิเตอร์ควบคุม

การปรับตัวด้วยพารามิเตอร์ควบคุม (Parameter Adaptation) มีกระทบต่อค่าผลลัพธ์ของคำตอบอย่างมาก เนื่องจาก DE Operator ใช้พารามิเตอร์ควบคุมเป็นตัวกำหนดระยะเคลื่อนของจุดพิกัด ตัวอย่างเช่น หากกำหนดค่า F ที่มาก จะทำให้การเคลื่อนของจุดพิกัดห่างจากจุดเริ่มต้นมาก ส่งผลให้ไม่สามารถหาค่าที่ดีกว่าในบริเวณใกล้เคียงจุดเริ่มต้นได้ หรือหากกำหนดค่า F ที่น้อยจะทำให้การเคลื่อนของจุดพิกัดใกล้จุดเริ่มต้น ทำให้มีโอกาสติดอยู่ในคำตอบที่ดีเฉพาะกลุ่มได้

ในงานวิจัยนี้ จึงได้นำแนวคิดของ jDE และ JADE จาก Brest, et al. [13], Zhang and Sanderson [19], Zhang and Sanderson [21] มาพัฒนาและประยุกต์ใช้กับ MOEA/D ให้เกิดการเรียนรู้และปรับค่าพารามิเตอร์ควบคุมเมื่อคำตอบที่มีในระบบไม่เกิดการเปลี่ยนแปลง แสดงตัววัดการเปลี่ยนแปลงของระบบดังนี้

$$\tau^s_{i,g} = 1 - \left[(1-\beta)\cdot\frac{\sum_{k=1}^{g} ASR^s_{i,j,k}}{\sum_{k=1}^{g} X^s_{i,k}}\right] - \left[\beta\cdot\frac{\sum_{k=1}^{g}\sum_{j=1}^{N} ASR^s_{i,j,k}}{NR\sum_{k=1}^{g} X^s_{i,k}}\right] \quad (10)$$

$$ASR^s_{i,j,g} = \frac{1}{2}\left(SR^s_{i,j,g} + DSR^s_{i,j,g}\right) \quad (11)$$

เมื่อ $\tau^s_{i,g}$ คือค่าสัดส่วนความไม่สำเร็จในการสร้างคำตอบรุ่นใหม่จากการใช้กลยุทธ์ที่ s ในปัญหาย่อยที่ i เจเนอเรชั่นที่ g (The ratio of unsuccessfully creating offspring vector that are better than the parent vector from using strategy s in subproblem i generation g), $ASR^s_{i,j,g}$ คือค่าเฉลี่ยการแทนที่คำตอบกับการแทนที่แบบครอบงำในปัญหาย่อยที่ i เวกเตอร์ที่ j เจเนอเรชั่นที่ g (Average of $SR^s_{i,j,g}$ and $DSR^s_{i,j,g}$), β คือค่าถ่วงน้ำหนักระหว่างความสำเร็จในการสร้างคำตอบรุ่นใหม่ของปัญหาย่อยที่ i กับความสำเร็จในการสร้างคำตอบรุ่นใหม่ของปัญหาย่อยที่ $j \in B(i)$ มีค่าระหว่าง 0 ถึง 1 หากกำหนดค่าเท่ากับ 1 หมายถึงสนใจเวกเตอร์คำตอบทั้งหมดของ $B(i)$ หากเท่ากับ 0 หมายถึงสนใจเฉพาะเวกเตอร์ที่ i ของปัญหาย่อย i (Scaling factor for weighting the successfully creating offspring at the subproblem i and the successfully creating offspring in the neighborhood of subproblem i), $X^s_{i,k}$ คือเลขฐานสอง เท่ากับ 1 เมื่อใช้กลยุทธ์ที่ s ในปัญหาย่อยที่ i เจเนอเรชั่นที่ g, NR คือจำนวนการแทนที่คำตอบสูงสุด (Maximum number of solutions replacement)

5.2.1 การปรับตัวของค่าปัจจัยขยายผลต่าง

การปรับตัวของค่าปัจจัยขยายผลต่าง (Adaptation of scaling factor) ในงานวิจัยนี้ ได้นำแนวคิดการปรับพารามิเตอร์จาก JADE โดยจะทำการปรับพารามิเตอร์ควบคุมในทุกปัญหาย่อยทุกเจเนอเรชั่นด้วยการแจกแจงแบบโคชี (Cauchy Distribution) เมื่อค่าสุ่มน้อยกว่าค่าสัดส่วนความไม่สำเร็จในการสร้างคำตอบรุ่นใหม่ แสดงสูตรการปรับพารามิเตอร์ควบคุมดังนี้

$$F_{i,g+1}^{s} = \begin{cases} \text{cauchy}\left(\mu_{F,i,t+1}^{s}, 0.1\right), & \text{if } rand_1 < \tau_{i,g}^{s} \\ F_{i,g}^{s}, & \text{otherwise} \end{cases} \quad (12)$$

$$\mu_{F,i,g+1}^{s} = \begin{cases} (1-LR)\cdot \mu_{F,i,g}^{s} + LR \cdot F_{i,g}^{s} & \text{if } \sum_{j=1}^{N} DSR_{i,j,g}^{s} > 0 \\ \mu_{F,i,g}^{s} & \text{otherwise} \end{cases} \quad (13)$$

เมื่อ $F_{i,g}^{s}$ คือค่าปัจจัยขยายผลต่าง (Scaling factor) ของกลยุทธ์ที่ s ปัญหาย่อยที่ i ในเจเนอเรชั่นที่ g, $\mu_{F,i,g}^{s}$ คือค่าเฉลี่ยปัจจัยขยายผลต่าง (Mean of scaling factor) ของกลยุทธ์ที่ s ปัญหาย่อยที่ i ในเจเนอเรชั่นที่ g, LR คืออัตราการเรียนรู้ (Learning rate)

5.2.2 การปรับตัวของอัตราการครอสโอเวอร์

การปรับตัวของอัตราการครอสโอเวอร์ (Adaptation of crossover rate) ในงานวิจัยนี้ ใช้สมการถดถอยในการปรับพารามิเตอร์ เนื่องจากค่า CR ที่น้อยจะทำให้ค่าพิกัดไม่เปลี่ยนแปลงจากเดิมมาก ส่งผลให้สามารถค้นหาคำตอบที่ดีในบริเวณใกล้เคียงกับคำตอบที่ดีเดิม ผู้วิจัยจึงกำหนดให้ค่า CR น้อยลงเมื่อเทียบกับจำนวนเจเนอเรชั่นสูงสุดในการดำเนินงานอัลกอริทึม โดยจะทำการปรับค่าพารามิเตอร์ควบคุมในทุกปัญหาย่อยทุกเจเนอเรชั่น เมื่อค่าสุ่มน้อยกว่าค่าสัดส่วนความไม่สำเร็จในการสร้างคำตอบรุ่นใหม่

เนื่องจากค่า CR มีผลต่อการเปลี่ยนแปลงทิศทางของคำตอบที่มาก เมื่ออัลกอริทึมดำเนินงานใกล้ครบจำนวนเจเนอเรชั่นที่กำหนด จะกำหนดให้ค่า CR ลดลงอย่างรวดเร็วเพื่อหาค่าที่ดีที่สุดในบริเวณพิกัดนั้น ๆ ดังนั้น ผู้วิจัยจึงได้นำสมการถดถอยจากแนวคิดของ Lin, et al. [22] มาดัดแปลงให้เหมาะสมกับอัลกอริทึมและปัญหาที่ใช้ในงานวิจัย แสดงสมการถดถอยในการปรับพารามิเตอร์ควบคุมดังนี้

$$CR_{i,g+1}^{s} = \begin{cases} 0.55 + \left[\frac{1}{\pi} \times \arctan\left(\frac{1-\frac{NA_{i,g}^{s}}{RG}-0.8}{0.1}\right)\right], & \text{if } rand_2 < \tau_{i,g}^{s} \\ CR_{i,g}^{s}, & \text{otherwise} \end{cases} \quad (14)$$

$$NA_{i,g}^{s} = \sum_{k=1}^{g} na_{i,k}^{s} \quad (15)$$

เมื่อ $CR_{i,g}^{s}$ คืออัตราการครอสโอเวอร์ (Crossover rate) ของกลยุทธ์ที่ s ปัญหาย่อยที่ i ในเจเนอเรชั่นที่ g, $NA_{i,g}^{s}$ คือจำนวนครั้งที่เกิดการปรับพารามิเตอร์ CR (Cumulative number of crossover probability adjustments) ของกลยุทธ์ที่ s ปัญหาย่อยที่ i ในเจเนอเรชั่นที่ g, $na_{i,g}^{s}$ คือเลขฐานสอง เท่ากับ 1 เมื่อเกิดการปรับพารามิเตอร์ CR ของกลยุทธ์ที่ s ในปัญหาย่อยที่ i เจเนอเรชั่นที่ g, RG คือจำนวนเจเนอเรชั่นที่เหลือ (Remaining number of generations)

5.2.3 การปรับตัวของค่าถ่วงเวกเตอร์พื้นฐาน

การปรับตัวของค่าถ่วงน้ำหนักเวกเตอร์พื้นฐาน (Adaptation of weight of base vectors) ในงานวิจัยนี้ นำแนวคิดการปรับพารามิเตอร์จาก jDE โดยจะทำการปรับพารามิเตอร์ควบคุมในทุกปัญหาย่อยทุกเจเนอเรชั่นด้วยการแจกแจงแบบยูนิฟอร์ม (Uniform Distribution) เมื่อค่าสุ่มน้อยกว่าค่าสัดส่วนความไม่สำเร็จในการสร้างคำตอบรุ่นใหม่

เนื่องจากไม่สามารถบ่งบอกได้ว่าค่าถ่วงน้ำหนักเวกเตอร์พื้นฐานควรเป็นเท่าใด จึงกำหนดให้ปรับตัวแบบกระจายเท่า ๆ กัน เพื่อเพิ่มความหลากหลายของรูปแบบคำตอบให้มากที่สุด แสดงสูตรการปรับพารามิเตอร์ควบคุมดังนี้

$$\gamma_{i,g+1}^{s} = \begin{cases} rand_4, & \text{if } rand_3 < \tau_{i,g}^{s} \\ \gamma_{i,g}^{s}, & \text{otherwise} \end{cases} \quad (16)$$

เมื่อ $\gamma_{i,g}^{s}$ คือค่าถ่วงน้ำหนักเวกเตอร์พื้นฐาน (Weight of base vectors) ของกลยุทธ์ที่ s ปัญหาย่อยที่ i ในเจเนอเรชั่นที่ g

5.3 รหัสเทียมของ AMOEA/D-DE

อัลกอริทึม 1: นำเสนอวิธีการดำเนินงานอัลกอริทึม AMOEA/D-DE ด้วยรูปแบบรหัสเทียม

/* *Initialization*

Generate N weight vectors $\lambda^i = (\lambda_1^i, \lambda_2^i, \dots, \lambda_M^i)$, $i = 1, \dots, N$

For $i = 1, \dots, N$, define the set of indexes $B(i) = \{i_1, \dots, i_{Nb}\}$ where $\{\lambda^{i_1}, \dots, \lambda^{i_{Nb}}\}$ are the Nb closest weight vectors to λ^i (by the Euclidean distance)

Generate an initial population $P_0 = \{x^1, \dots, x^N\}$, $x^i = (x_1^i, x_2^i, \dots, x_n^i)$

Evaluate each individual in the initial population P_0 and associate x^i with λ^i

For $i = 1, \dots, N$, set $sbest^i = x^i$

Initialize $z^* = (z_1^*, \dots, z_M^*)$ by setting $z_k^* = \min_{1 \le i \le N} \{f_k(x^i)\}$, $k = 1,2, \dots, M$

Set $g = 1$, $\tau_{i,0}^s = 0$

For all strategies $s = 1, \dots, S$, set $p_{s,g} = 1/S$

For all $SR_{i,j,g}^s$ and $DSR_{i,j,g}^s$ are set zero

/* *Main computation loop*

repeat

  for each parent vector x^i, $i = 1, \dots, N$ **do**

    Select strategy s from the pool according to $p_{s,g}$

    /* *Parameter adaptation*

    if $rand_1 < \tau_{i,g-1}^s$ **then** //Adaptation of F ($rand$ in $U[0,1]$)

      Generate $F_{i,g}^s = \text{cauchy}(\mu_{F,i,g+1}^s, 0.1)$

    else

      $F_{i,g}^s = F_{i,g-1}^s$

    end if (step 18)

    if $rand_2 < \tau_{i,g-1}^s$ **then** //Adaptation of CR

      Calculate $CR_{i,g}^s = 0.55 + \left[\frac{1}{\pi} \times \arctan\left(\frac{1 - \frac{NA_{i,g}^s}{RG} - 0.8}{0.1}\right)\right]$

    else

      $CR_{i,g}^s = CR_{i,g-1}^s$

    end if (step 23)

    if $rand_3 < \tau_{i,g-1}^s$ **then** //Adaptation of γ

      Generate $\gamma_{i,g}^s = rand$

    else

      $\gamma_{i,g}^s = \gamma_{i,g-1}^s$

    end if (step 28)

    /* *Update real best solution*

    If the tchebycheff value of x^j is better than $sbest^i$, $j \in B(i)$ **then**

      Set $sbest^i = x^j$

    end if (step 35)

    /* *Reproduction*

    Generate a new solution y by DE operator (repair it if necessary)

    Apply polynomial mutation to produce y' (repair it if necessary)

    Update z^*, $z_k^* = \min(z_k^*, f_k(y'))$ and set $n_r = 0$

    /* *Selection*

    for each subproblem $j \in B(i)$ **do**

      if $n_r < NR$ **then**

        if $g^{te}(y'|\lambda^j, z^*) \le g^{te}(x^j|\lambda^j, z^*)$ **then**

          Replace x^j by y', increment n_r and set $SR_{i,j,g}^s = 1$

          if $g^{te}(y'|\lambda^j, z^*) < g^{te}(x^j|\lambda^j, z^*)$ **then**

            Set $DSR_{i,j,g}^s = 1$//true is equal to one

          end if (step 50)

        end if (step 48)

      end if (step 47)

    end for (step 46)

    /* *Update mean of scaling factor*

    if any $DSR_{i,j,g}^s$ of subproblem $j \in B(i)$ be true **then**

      Update $\mu_{F,i,g+1}^s$

60: **else**
61: $\mu^{s}_{F,i,g+1} = \mu^{s}_{F,i,g}$
62: **end if** (step 58)
63:
64: Calculate all $ASR^{s}_{i,j,g}$ and $\tau^{s}_{i,g}$ for each strategy
65: **end for** (step 14)
66: Calculate and update the probability $p_{s,g+1}$ for each strategy
67: $g = g + 1$
68: **until** $g > G$

การเริ่มต้นของ AMOEA/D-DE จะสร้างค่าถ่วงน้ำหนัก λ^{i}_{k} โดยใช้ซิมเพล็กซ์แลตทิซดีไซน์ (Simplex lattice design) เป็นเมทริกซ์ขนาดเท่ากับ จำนวนประชากร (N) x จำนวนวัตถุประสงค์ (M) และคำนวณระยะห่างระหว่างจุดพิกัดของเวกเตอร์ค่าถ่วงน้ำหนัก λ^{i} ทั้งหมด (Euclidean distances) ดังสมการที่ 23 และจัดเก็บเวกเตอร์ข้างเคียงของแต่ละปัญหาย่อยที่ i ลงในเซต $B(i)$

$$d_{ij} = \sqrt{\sum_{k=1}^{M}\left(\lambda^{i}_{k} - \lambda^{j}_{k}\right)^{2}} \qquad (17)$$

เมื่อได้ค่า d_{ij} แล้ว (d_{ij} คือระยะห่างระหว่างจุดของค่าถ่วงน้ำหนักเวกเตอร์ที่ i กับเวกเตอร์ที่ j โดย $i = 1, \dots, N$ และ $j = 1, \dots, N$) ให้สร้างเวกเตอร์คำตอบเริ่มต้นตามจำนวนประชากร โดยวิธีสุ่มเลือกอิสระ ขนาดเวกเตอร์เท่ากับจำนวนผลิตภัณฑ์ทั้งหมด (n) จากนั้นกำหนด g และ i เท่ากับ 1, กำหนดค่าความน่าจะเป็นเริ่มต้นของทุกกลยุทธ์ ($p_{s,1}$) เท่ากับ $1/S$, คำนวณค่าฟังก์ชันวัตถุประสงค์ ($f(x^{i})$) ของเวกเตอร์คำตอบเริ่มต้น และกำหนดค่าเป้าหมาย $z^{*}_{k} = \min_{1 \le i \le N}\{f_{k}(x^{i})\}$ โดยที่ $k = 1,2, \dots, M$

หลังจากจบขั้นตอนการเริ่มต้น อัลกอริทึมจะเข้าสู่ขั้นตอนการปรับปรุง ซึ่งเป็นลูปหลัก (Main loop) ในการดำเนินงาน (ขั้นตอนที่ 12-67) เริ่มต้นโดยสุ่มเลือกกลยุทธ์จากค่าความน่าจะเป็นของกลยุทธ์ในเจเนอเรชั่นที่ g ($p_{s,g}$) และสุ่มค่าระหว่าง 0 ถึง 1 ให้กับพารามิเตอร์ควบคุมทุกตัว หากค่าสุ่มของพารามิเตอร์ควบคุมใด น้อยกว่าค่าสัดส่วนความไม่สำเร็จในการสร้างคำตอบรุ่นใหม่ ($\tau^{s}_{i,g-1}$) ของเจเนอเรชั่นที่ $g-1$ จะทำการปรับค่าพารามิเตอร์ควบคุมนั้นในเจเนอเรชั่นที่ t คำนวณได้จากสมการที่ 18 20 และ 22 จากนั้นคัดเลือก *sbest* จากเวกเตอร์คำตอบข้างเคียง ($j \in B(i)$) ด้วยวิธีเทบบีเชฟฟ์ จากสมการที่ 2 และสุ่มเลือกเวกเตอร์ r1, r2 และ r3 จากเวกเตอร์ข้างเคียงของเวกเตอร์ที่ i (โดย r1 ≠ r2 ≠ r3) เพื่อทำการพัฒนาเวกเตอร์เป้าหมายด้วยกลยุทธ์ที่ s โดย $s = 1,2, \dots S$ สามารถคำนวณได้จากสมการที่ 9 ถึง 11 และทำครอสโอเวอร์ระหว่างเวกเตอร์เป้าหมาย (x^{i}) และเวกเตอร์กลายพันธุ์ (v^{i}) ด้วยวิธียูนิฟอร์มครอสโอเวอร์ โดยมีโอกาสเกิดการแลกเปลี่ยนค่าในแต่ละตำแหน่งที่ c บนเวกเตอร์ i เท่ากับค่าอัตราการครอสโอเวอร์ (CR^{s}_{i}) และทำการสุ่มค่าในช่วง 1 ถึง n นำเวกเตอร์กลายพันธุ์ในตำแหน่งที่สุ่มค่าไปแทนที่เวกเตอร์เป้าหมาย เวกเตอร์ที่ได้จากการครอสโอเวอร์เรียกว่า เวกเตอร์ทดลอง (Trial vector: u^{i}) แสดงดังสมการที่ 24

$$u^{i}_{c} = \begin{cases} v^{i}_{c}, & \text{if } rand_{i,c} \le CR^{s}_{i} \text{ or } c = randi_{i} \\ x^{i}_{c}, & \text{otherwise} \end{cases} \qquad (18)$$

เมื่อได้เวกเตอร์ทดลองแล้ว (u^{i}_{c} คือจุดพิกัดของเวกเตอร์ทดลองตำแหน่งที่ c บนเวกเตอร์ i) จะปรับปรุงเวกเตอร์ด้วยวิธีการกลายพันธุ์แบบพหุนาม (Polynomial Mutation) เพื่อลดโอกาสการเกิดคำตอบที่เหมือนกัน คำนวณได้จากสมการที่ 25 และ 26 (กำหนดให้ $y = u^{i}$ โดยที่ $y = (y_1, \dots, y_n)$)

$$y'_{c} = \begin{cases} y_c + \sigma_c\left(y^{Up}_{c} - y^{Lw}_{c}\right) & \text{with probability } p_m \\ y_c & \text{with probability } 1 - p_m \end{cases} \qquad (19)$$

$$\sigma_c = \begin{cases} (2 \times rand_{\sigma})^{\frac{1}{\eta+1}} - 1 & \text{if } rand_{\sigma} < 0.5 \\ 1 - [2(1 - rand_{\sigma})]^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \qquad (20)$$

เมื่อ y^{Up}_{c} คือค่าขอบเขตบน (Upper bounds) ของเวกเตอร์เป้าหมายตำแหน่งที่ c บนเวกเตอร์รุ่นใหม่, y^{Lw}_{c} คือค่าขอบเขตล่าง (Lower bounds) ของเวกเตอร์เป้าหมายตำแหน่งที่ c บนเวกเตอร์รุ่นใหม่

ทำการคัดเลือกเวกเตอร์ $j \in B(i)$ ด้วยวิธีเทบบีเชฟฟ์ คำนวณได้จากสมการที่ 2 จากนั้นคำนวณค่าสัดส่วนความ

ไม่สำเร็จในการสร้างคำตอบรุ่นใหม่ ($\tau_{i,g}^{s}$) ของปัญหาย่อยที่ i เจเนอเรชั่นที่ g สามารถคำนวณได้จากสมการที่ 16 และ 17 โดยที่ $s = 1,2,\dots,S$ จบการพัฒนาคำตอบของปัญหาย่อยที่ i กำหนดให้ $i = i + 1$

เมื่อพัฒนาคำตอบของปัญหาย่อยครบทุกปัญหา จะจบการพัฒนาคำตอบของเจเนอเรชั่นที่ t ให้คำนวณค่าความน่าจะเป็นของกลยุทธ์ในเจเนอเรชั่นที่ $g + 1$ ($p_{s,g+1}$) สามารถคำนวณได้จากสมการที่ 12 ถึง 15 โดยที่ $s = 1,2,\dots,S$

การหยุดอัลกอริทึม จะหยุดการปรับปรุงคำตอบตามจำนวนเจเนอเรชั่นสูงสุด (G) ที่กำหนด หากต้องการปรับปรุงคำตอบให้กำหนด $g = g + 1$ แล้วนำเวกเตอร์คำตอบที่ได้ในเจเนอเรชั่นก่อนหน้า ไปทำการปรับปรุงต่อในลูปหลัก กำหนด $i = 1$

6. การทดลอง

6.1 ปัญหามาตรฐานหลายวัตถุประสงค์ค่าต่อเนื่อง

ในบทความนี้ ทำการเปรียบเทียบสมรรถนะอัลกอริทึม MOEA/D, MODE/D และ AMOEA/D-DE โดยการทดลองแก้ปัญหา 2 วัตถุประสงค์ ZDT จำนวน 3 ปัญหา [23], ปัญหา 3 วัตถุประสงค์ DTLZ จำนวน 2 ปัญหา และปัญหา 10 วัตถุประสงค์ DTLZ จำนวน 1 ปัญหา [24]

สำหรับทุกปัญหาเป็นกรณีการหาค่าน้อยที่สุด

1) ZDT1

$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \sqrt{\frac{f_1(x)}{g(x)}}\right]$$
$$g(x) = 1 + 9\sum_{i=2}^{n} x_i/n - 1$$

โดยที่ $n = 30$ และ $x_i \in [0,1]$ Pareto-optimal front (PF) จาก $g(x) = 1$

2) ZDT2

$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \left(\frac{f_1(x)}{g(x)}\right)^2\right]$$

โดยที่ $g(x)$ และขนาดเวกเตอร์ x เท่ากับของ ZDT1

3) ZDT3

$$f_2(x) = g(x)\left[1 - \sqrt{\frac{f_1(x)}{g(x)}} - \frac{f_1(x)}{g(x)}sin(10\pi x_1)\right]$$

โดยที่ $g(x)$ และขนาดเวกเตอร์ x เท่ากับของ ZDT1 ปัญหานี้ มี PF แบ่งเป็นช่วง

4) DTLZ1

$$f_1(x) = \frac{1}{2}x_1x_2\dots x_{M-1}\big(1 + g(x_M)\big)$$
$$f_2(x) = \frac{1}{2}x_1x_2\dots(1 - x_{M-1})\big(1 + g(x_M)\big)$$
$$\vdots$$
$$f_{M-1}(x) = \frac{1}{2}x_1(1 - x_2)\big(1 + g(x_M)\big)$$
$$f_M(x) = \frac{1}{2}(1 - x_1)\big(1 + g(x_M)\big)$$
$$g(x_M) = 100\big[|x_M| + \textstyle\sum_{x_i \in x_M}(x_i - 0.5)^2 - cos\big(20\pi(x_i - 0.5)\big)\big]$$

โดยที่ $n = 10$, $x_i \in [0,1]$, M จำนวนวัตถุประสงค์, พิกัด $k = (n - M + 1)$ ตัวสุดท้ายของเวกเตอร์ x แสดงในรูปเวกเตอร์ x_M ($|x_M| = k, M = 3$) และคำตอบที่ดีที่สุด (Pareto-optimal solution: PS) คือ ทุก ๆ $x_i = 0.5$ สำหรับ $x_i \in x_M$

5) DTLZ2

$$f_1(x) = \big(1 + g(x_M)\big)\cos\theta_1\dots\cos\theta_{M-2}\cos\theta_{M-1}$$
$$f_2(x) = \big(1 + g(x_M)\big)\cos\theta_1\dots\cos\theta_{M-2}\sin\theta_{M-1}$$
$$f_3(x) = \big(1 + g(x_M)\big)\cos\theta_1\dots\sin\theta_{M-2}$$
$$\vdots$$
$$f_M(x) = \big(1 + g(x_M)\big)\sin\theta_1$$

โดยที่ $x_i \in [0,1]$, $g(x_M) = \sum_{x_i \in x_M}(x_i - 0.5)^2$ และ $\theta_i = \frac{x_i\pi}{2}$ ในบทความนี้กำหนด $n = 10$ และ $n = 20$ สำหรับปัญหา $M = 3$ และ $M = 10$ ตามลำดับ และ PS คือ $x_i = 0.5$ สำหรับ $x_i \in x_M$

6.2 การตั้งค่าพารามิเตอร์ควบคุม

การทดลองนี้ ใช้การดำเนินงานของ MOEA/D และ MODE/D ตามแบบของ Zhang and Li [3], Li and Zhang [4] โดย MOEA/D จะใช้ Simulated binary crossover (SBX) และการกลายพันธุ์แบบพหุนาม (Polynomial mutation: PM) ในการพัฒนาคำตอบ ส่วน MODE/D จะใช้ DE Operator และ PM

ประชากรเริ่มต้นของทั้ง 3 อัลกอริทึม จะถูกสุ่มเลือกอิสระตามขอบเขตที่เป็นไปได้ของคำตอบในแต่ละปัญหา

และพารามิเตอร์ควบคุมต่าง ๆ อ้างอิงจาก Zhang and Li [3], Venske, et al. [10] แสดงในตารางที่ 1

6.3 ผลการทดลอง

อัลกอริทึม MOEA/D, MODE/D และ AMOEA/D-DE จะถูกนำมาการแก้ปัญหาด้วยการทำซ้ำอิสระอัลกอริทึมละ 30 ครั้ง บนเครื่องคอมพิวเตอร์ Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz RAM 16 GB Windows 10 64 bit operation system และเนื่องจากลักษณะของ MOPs ควรใช้ตัวชี้วัดสมรรถนะหลายรูปแบบเพื่อเปรียบเทียบประสิทธิภาพของอัลกอริทึมต่าง ๆ [25-27] ในบทความนี้ใช้ตัวชี้วัดสมรรถนะ 3 ด้าน ดังนี้

1) การลู่เข้าของคำตอบ

การลู่เข้าของคำตอบ (Convergence metrics) เป็นการวัดระยะห่างที่น้อยที่สุดระหว่างกลุ่มคำตอบที่ได้จากอัลกอริทึม (Obtained Pareto Optimal Solution) กับกลุ่มคำตอบที่แท้จริง (True-Pareto Optimal Solution) โดยกำหนดให้ P เป็นกลุ่มคำตอบที่กระจายตัวสม่ำเสมอบนเส้น PF และ S เป็นกลุ่มคำตอบที่อัลกอริทึมหาได้ แสดงค่าเฉลี่ยระยะห่างระหว่าง S ไปยัง P ดังนี้

$$GD(S,P) = \frac{\sum_{x \in S} d(x,P)}{|S|} \quad (21)$$

เมื่อ $d(x,P)$ คือ Euclidean distance ที่น้อยที่สุดระหว่างคำตอบ x กับจุดคำตอบของ P, $|\cdot|$ คือจำนวนคำตอบภายในเซตใด ๆ

2) การลู่เข้าและความหลากหลายของคำตอบ

การลู่เข้าและความหลากหลายของคำตอบ (Convergence and diversity metrics) เป็นการเปรียบเทียบระยะห่างที่น้อยที่สุดของคำตอบที่แท้จริงที่แต่ละค่ากับกลุ่มคำตอบที่ได้จากอัลกอริทึม แสดงค่าเฉลี่ยระยะห่างระหว่าง P ไปยัง S ดังนี้

$$IGD(P,S) = \frac{\sum_{y \in P} d(y,S)}{|P|} \quad (22)$$

เมื่อ $d(y,S)$ คือ Euclidean distance ที่น้อยที่สุดระหว่างคำตอบ y กับจุดคำตอบของ S

3) ความหลากหลายของคำตอบ

ความหลากหลายของคำตอบ (Diversity Metrics) แบ่งการวัดสมรรถนะเป็นสองด้าน ได้แก่ 1) การกระจายตัวที่สม่ำเสมอของคำตอบที่หาได้ (Distribution Measures) ซึ่งเป็นการวัดผลต่างระยะห่างที่น้อยที่สุดระหว่างคำตอบโดยยิ่งมีค่าผลต่างที่น้อยแสดงถึงคำตอบมีการกระจายตัวที่สม่ำเสมอมาก และ 2) ความกว้างของการแพร่กระจายของคำตอบที่หาได้ (Spread Indicates) เป็นการวัดระยะห่างที่น้อยที่สุดระหว่างคำตอบปลายสุดของคำตอบที่แท้จริงกับคำตอบที่หาได้ เพื่อประเมินว่ากลุ่มคำตอบที่หาได้มีคำตอบที่ครอบคลุมดีหรือไม่ โดยยิ่งมีค่าระยะที่น้อยแสดงถึงคำตอบมีการแพร่กระจายที่ดี แสดงสูตรคำนวณดังนี้

$$\Delta^*(S,P) = \frac{\sum_{c=1}^{C} d(E_c) + \sum_{i=1}^{|S_j|} |d(x_i) - \bar{d}|}{\sum_{c=1}^{C} d(E_c) + |S_j|\bar{d}} \quad (23)$$

เมื่อ $d(x_i)$ คือ Euclidean distance ที่น้อยที่สุดระหว่างคำตอบที่ i กับคำตอบ y ใด ๆ ของ S โดยที่ $y \neq i$, $\bar{d}$ คือค่าเฉลี่ยของ $d(x_i)$, $d(E_c)$ คือระยะห่างระหว่างคำตอบปลายสุดของคำตอบที่แท้จริง (Extreme solution in the set of true Pareto-front) กับคำตอบขอบเขตของคำตอบที่หาได้ (Boundary solution in the set of obtained solution), C คือจำนวนคำตอบปลายสุดของคำตอบที่แท้จริง

การคำนวณค่าตัวชี้วัดสมรรถนะอัลกอริทึม จะนำคำตอบที่ดีที่สุดที่แต่ละอัลกอริทึมหาได้ (ฟรอนเทียร์ที่ 1 ของคำตอบที่หาได้) มาเปรียบเทียบกับเซต P ซึ่งเกิดการการสร้างคำตอบที่กระจายตัวสม่ำเสมอบนเส้น PF ของแต่ละปัญหา โดยจะสร้าง 500 จุดคำตอบ สำหรับปัญหา 2 วัตถุประสงค์, 990 คำตอบ สำหรับปัญหา 3 วัตถุประสงค์ และ 2002 คำตอบ สำหรับปัญหา 10 วัตถุประสงค์

เมื่อได้ค่าตัวชี้วัดสมรรถนะในแต่ละด้านของแต่ละอัลกอริทึมแล้ว ให้นำตัวชี้วัดในด้านเดียวกันมาเปรียบเทียบทางสถิติ โดยในบทความนี้มีทั้งหมด 3 อัลกอริทึม จึงใช้วิธีการวิเคราะห์ความแปรปรวน (ANOVA) และวิธีฟิชเชอร์แพร์ไวซ์ (Fisher pairwise comparisons test) ในการทดสอบความแตกต่างของค่าตัวชี้วัดสมรรถนะอัลกอริทึม

ตารางที่ 1 พารามิเตอร์ควบคุมที่ใช้ในการทดลอง

Parameter	Value	Description
DE parameter		
N	100	Population size (for instances with 2 objectives)
	300	Population size (for instances with 3 objectives)
	715	Population size (for instances with 10 objectives)
CR	1.0	Initial crossover rate
μ_F	0.5	Initial mean of scaling factor
p_m	1/n	Polynomial mutation probability
η	20	Distribution index of polynomial mutation
G	250	Maximum number of generation
MOEA/D parameter		
Nb	20	Number of weight vectors in the neighborhood
NR	5	Maximal number of solutions replaced by each offspring
Adaptive parameter		
α	0.3	Adaptation rate
β	0.3	Scaling factor for weighting the successfully creating offspring at the subproblem i and the successfully creating offspring in the neighborhood of subproblem i
LR	0.8	Learning rate
p_{min}	0.05	Minimum probability value of each strategy

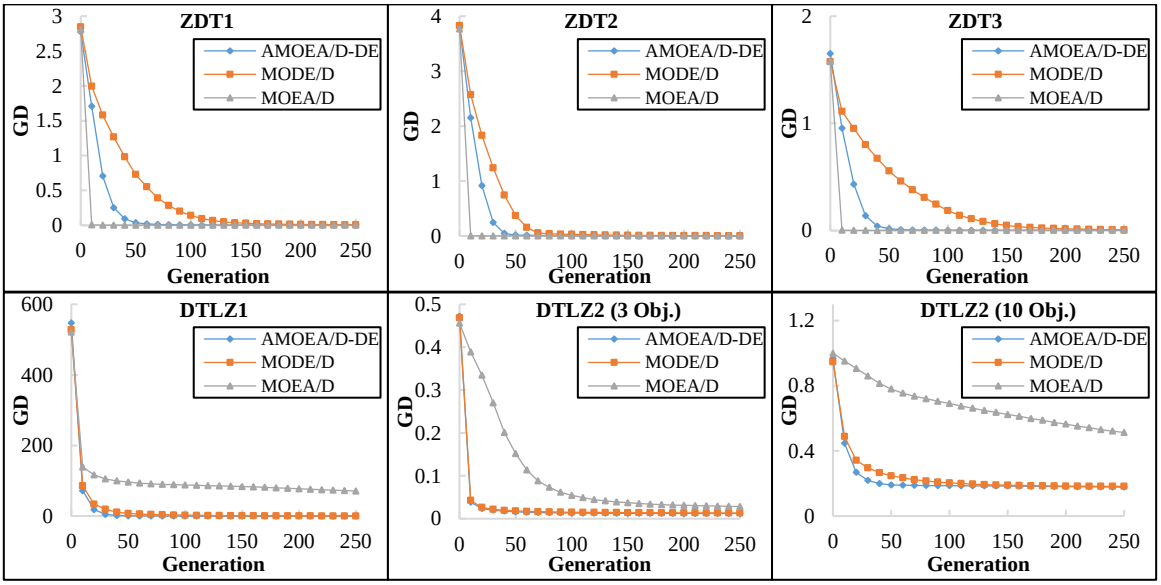
ในตารางที่ 2 เป็นตารางแสดงค่าเฉลี่ยเวลาดำเนินงานที่แต่ละอัลกอริทึมใช้ในแต่ละปัญหา โดยจะเห็นได้ว่า MODE/D มีเวลาดำเนินงานที่น้อยที่สุดในทุกปัญหา ส่วน AMOEA/D-DE ใช้เวลาในการดำเนินงานมากที่สุด

เวลาในการดำเนินงานของแต่ละอัลกอริทึมแปรผันโดยตรงกับจำนวนคำตอบที่ถูกพัฒนา (The number of function evaluations) หรือจำนวนรุ่นของคำตอบที่ถูกพัฒนา (The maximum number of generation) ดังนั้นอัลกอริทึมที่มีการลู่เข้าของคำตอบได้รวดเร็วและมากกว่าจึงช่วยลดต้นทุนของระยะเวลาการค้นหาคำตอบได้ รูปที่ 1 แสดงวิวัฒนาการของค่าเฉลี่ยตัวชี้วัดสมรรถนะด้านการลู่เข้าของคำตอบ (Average GD) ซึ่งพบว่า MOEA/D มีการลู่เข้าของคำตอบที่รวดเร็วที่สุดสำหรับปัญหา 2 วัตถุประสงค์ ส่วนปัญหา 3 และ 10 วัตถุประสงค์ พบว่า AMOEA/D-DE มีการลู่เข้าของคำตอบแต่ที่รวดเร็วที่สุด แต่ยังคงใกล้เคียงกับ MODE/D

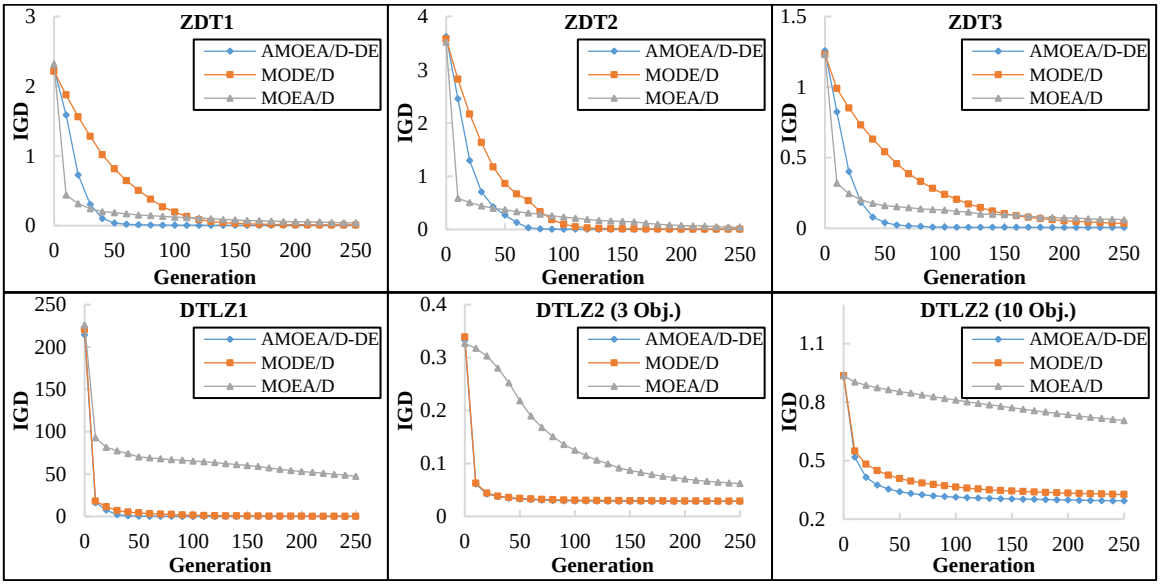
โดยผลลัพธ์สุดท้ายของตัวชี้วัดสมรรถนะด้านการลู่เข้าของคำตอบ (GD) จากการแก้ปัญหา Continuous MOPs แสดงในรูปแบบของค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐาน ดังตารางที่ 3 ซึ่งจะเห็นได้ว่า AMOEA/D-DE มีค่าเฉลี่ย GD ดีที่สุดในทุกปัญหา ยกเว้นปัญหา ZDT3 ที่ MOEA/D มีผลลัพธ์สุดท้ายของการลู่เข้าที่ดีกว่า

นอกจากอัลกอริทึมที่ดีจะให้คำตอบที่มีค่าการลู่เข้าดีแล้ว ยังควรมีความหลากหลายและการกระจายตัวของคำตอบที่สม่ำเสมออีกด้วย รูปที่ 2 แสดงวิวัฒนาการของค่าเฉลี่ยตัวชี้วัดสมรรถนะด้านการลู่เข้าและความหลากหลายของคำตอบ (Average IGD) ซึ่งพบว่า MOEA/D มีการลู่เข้าและความหลากหลายของคำตอบที่ดีและรวดเร็วที่สุดในช่วงต้น สำหรับปัญหา 2 วัตถุประสงค์ ส่วนปัญหา 3 และ 10 วัตถุประสงค์ พบว่า AMOEA/D-DE มีการลู่เข้าและความหลากหลายของคำตอบแต่ที่ดีและรวดเร็วที่สุด แต่ยังคงใกล้เคียงกับ MODE/D เช่นเดียวกับตัวชี้วัด GD และจากผลลัพธ์สุดท้ายของ IGD ในตารางที่ 4 จะเห็นได้ว่า AMOEA/D-DE มีค่าเฉลี่ย IGD ดีที่สุดในทุกปัญหา

ในตารางที่ 5 เป็นตารางแสดงค่าเฉลี่ยตัวชี้วัดด้านความหลากหลายของคำตอบ (Average distribution and spread) โดยจะพบว่า AMOEA/D-DE มีการแผ่กระจายของคำตอบที่กว้างและมีการกระจายตัวของคำตอบที่สม่ำเสมอมากที่สุด



รูปที่ 1 วิวัฒนาการของตัวชี้วัดสมรรถนะด้านการลู่เข้าของคำตอบ



รูปที่ 2 วิวัฒนาการของตัวชี้วัดสมรรถนะด้านการลู่เข้าและความหลากหลายของคำตอบ

รูปที่ 3 ถึง 8 แสดงการกระจายของคำตอบสุดท้ายที่ดีที่สุด ซึ่งได้จากการดำเนินงานของแต่ละอัลกอริทึมในแต่ละปัญหาจำนวน 30 รอบทำซ้ำ (The distribution of the final solutions obtained in the run with the lowest GD value of each algorithm for each test instance) โดยปัญหา ZDT1 ถึง 3, DTLZ1 และ DTLZ2 (3 Obj.) จะถูกแสดงในรูปของกราฟ Scatter plot ส่วน DTLZ2 (10 Obj.) จะถูกแสดงการกระจายด้วยกราฟ 3D-RadVis [28]

จากรูปที่ 3 ถึง 8 จะเห็นได้ว่า การกระจายตัวของคำตอบสุดท้ายของ AMOEA/D-DE มีรูปแบบที่ชัดเจน ซึ่งดีกว่า MOEA/D และ MODE/D ในเกือบทุกปัญหา ยกเว้นปัญหา ZDT3 ที่ MOEA/D มีความหลากหลายมากกว่า

ส่วน MODE/D มีการลู่เข้าของคำตอบสุดท้ายที่ดีน้อยกว่าอัลกอริทึมอื่น ๆ โดยจะเห็นได้ว่าจุดคำตอบอยู่เหนือเส้น PF ในปัญหา ZDT ที่ 1 ถึง 3 แต่ MODE/D มีการกระจายตัวของคำตอบและการลู่เข้าที่ดีกว่า MOEA/D ในปัญหาที่มี 3 วัตถุประสงค์ขึ้นไป

ตารางที่ 2 เวลาการดำเนินงานเฉลี่ย (Avg. CPU time (s))

		AMOEA/D-DE	MODE/D	MOEA/D
Instance	ZDT1	16.66 (0.42)	**9.77 (0.30)**	10.20 (0.32)
	ZDT2	17.97 (0.37)	**10.97 (0.38)**	11.87 (0.43)
	ZDT3	19.57 (0.61)	**12.46 (0.40)**	13.31 (0.43)
	DTLZ1	86.50 (13.49)	**49.79 (6.44)**	**52.42 (6.70)**
	DTLZ2 (3 Obj.)	66.81 (0.71)	**39.16 (1.21)**	41.06 (0.76)
	DTLZ2 (10 Obj.)	250.37 (13.54)	**160.50 (16.84)**	**160.61 (18.75)**

*อักษรตัวหนา คือ กลุ่มของผลลัพธ์ที่ดีที่สุดตามหลักสถิติ (แตกต่างจากผลลัพธ์อื่น ๆ ที่ $p - value < 0.05$)

ตารางที่ 3 ค่าเฉลี่ยตัวชี้วัดสมรรถนะด้านการลู่เข้าของคำตอบ (Average GD)

		AMOEA/D-DE	MODE/D	MOEA/D
Instance	ZDT1	**7.11×10^{-4} (2.84×10^{-5})**	7.36×10^{-3} (1.46×10^{-3})	**9.23×10^{-4} (3.96×10^{-4})**
	ZDT2	**7.67×10^{-4} (4.01×10^{-5})**	3.93×10^{-3} (7.38×10^{-4})	**8.40×10^{-4} (2.03×10^{-4})**
	ZDT3	**5.05×10^{-3} (1.08×10^{-4})**	9.71×10^{-3} (4.66×10^{-3})	**4.15×10^{-3} (4.78×10^{-4})**
	DTLZ1	**1.11×10^{-2} (2.61×10^{-4})**	**1.70×10^{-1} (5.54×10^{-1})**	7.09×10^{1} (2.87×10^{1})
	DTLZ2 (3 Obj.)	**1.23×10^{-2} (1.90×10^{-4})**	1.33×10^{-2} (3.65×10^{-4})	2.85×10^{-2} (3.28×10^{-3})
	DTLZ2 (10 Obj.)	**1.78×10^{-1} (1.45×10^{-3})**	**1.82×10^{-1} (4.20×10^{-3})**	5.13×10^{-1} (7.87×10^{-2})

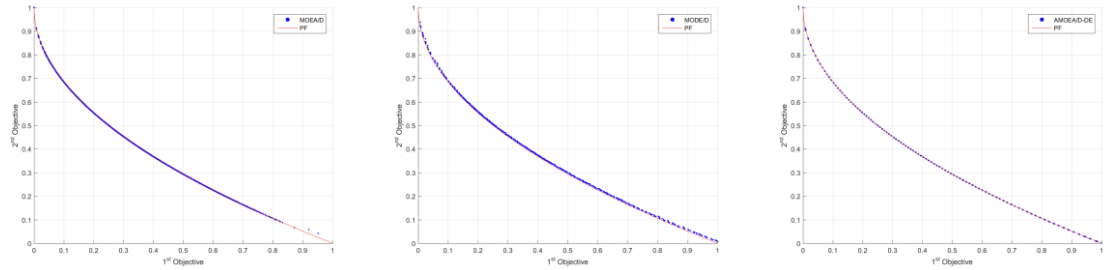
ตารางที่ 4 ค่าเฉลี่ยตัวชี้วัดสมรรถนะด้านการลู่เข้าและความหลากหลายของคำตอบ (Average IGD)

		AMOEA/D-DE	MODE/D	MOEA/D
Instance	ZDT1	**3.87×10^{-3} (8.12×10^{-6})**	9.09×10^{-3} (1.54×10^{-3})	3.90×10^{-2} (1.48×10^{-2})
	ZDT2	**3.81×10^{-3} (1.57×10^{-5})**	5.68×10^{-3} (5.60×10^{-4})	**3.77×10^{-2} (5.67×10^{-2})**
	ZDT3	**1.07×10^{-2} (7.85×10^{-5})**	4.50×10^{-2} (6.34×10^{-2})	7.52×10^{-2} (4.19×10^{-2})
	DTLZ1	**2.68×10^{-2} (2.34×10^{-4})**	**1.47×10^{-1} (4.27×10^{-1})**	4.74×10^{1} (2.33×10^{1})
	DTLZ2 (3 Obj.)	**2.85×10^{-2} (3.18×10^{-4})**	**2.90×10^{-2} (2.98×10^{-4})**	6.21×10^{-2} (1.24×10^{-2})
	DTLZ2 (10 Obj.)	**2.94×10^{-1} (1.65×10^{-3})**	3.26×10^{-1} (2.96×10^{-3})	7.05×10^{-1} (4.38×10^{-2})

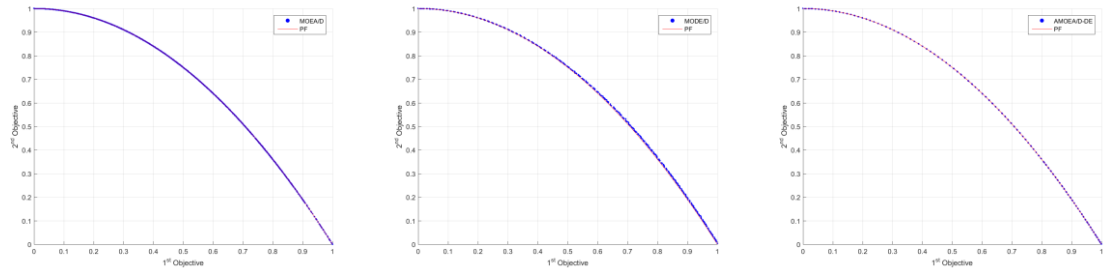
ตารางที่ 5 ค่าเฉลี่ยตัวชี้วัดสมรรถนะด้านความหลากหลายของคำตอบ (Average distribution and spread)

		AMOEA/D-DE	MODE/D	MOEA/D
Instance	ZDT1	**0.2884 (0.0051)**	0.3412 (0.0350)	0.4980 (0.1014)
	ZDT2	**0.1494 (0.0068)**	0.1758 (0.0317)	**0.4285 (0.2496)**
	ZDT3	**0.6136 (0.0122)**	0.7239 (0.0611)	0.7437 (0.0779)
	DTLZ1	**0.3672 (0.0091)**	**0.3841 (0.0664)**	0.7529 (0.1738)
	DTLZ2 (3 Obj.)	**0.4069 (0.0099)**	**0.4161 (0.0131)**	0.5843 (0.0740)
	DTLZ2 (10 Obj.)	**0.4980 (0.0108)**	0.6117 (0.0177)	0.8305 (0.0408)

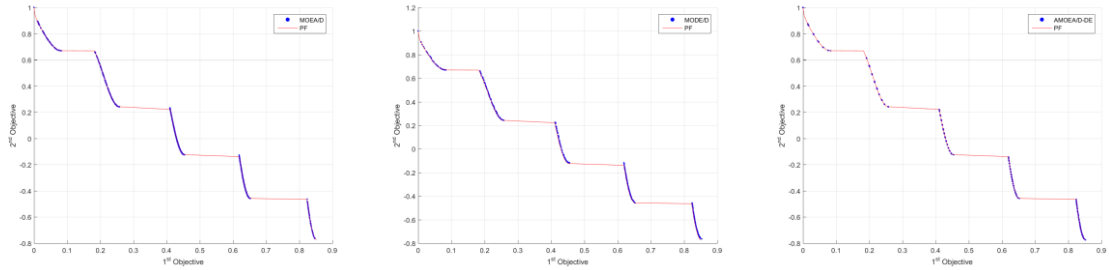
จากผลลัพธ์ที่กล่าวมา สามารถสรุปได้ว่า AMOEA/D-DE ให้คำตอบที่ดีกว่าอัลกอริทึมอื่น ๆ และมีการลู่เข้าของคำตอบที่รวดเร็วกว่า ทำให้สามารถลดจำนวนรอบในการพัฒนาคำตอบลงได้ แต่ยังคงให้ประสิทธิภาพของคำตอบในด้านต่าง ๆ ดีกว่า MOEA/D และ MODE/D



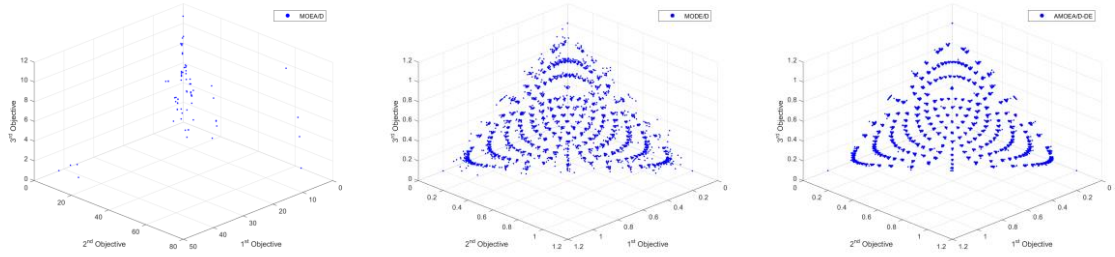
รูปที่ 3 กลุ่มคำตอบที่ดีที่สุด (ฟรอนเทียร์ที่ 1) จากการทำซ้ำทั้งหมด 30 ครั้ง ของแต่ละอัลกอริทึม ในปัญหา ZDT1



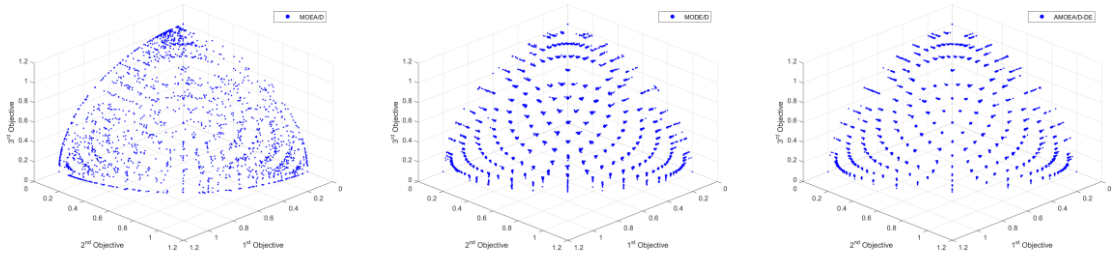
รูปที่ 4 กลุ่มคำตอบที่ดีที่สุด (ฟรอนเทียร์ที่ 1) จากการทำซ้ำทั้งหมด 30 ครั้ง ของแต่ละอัลกอริทึม ในปัญหา ZDT2



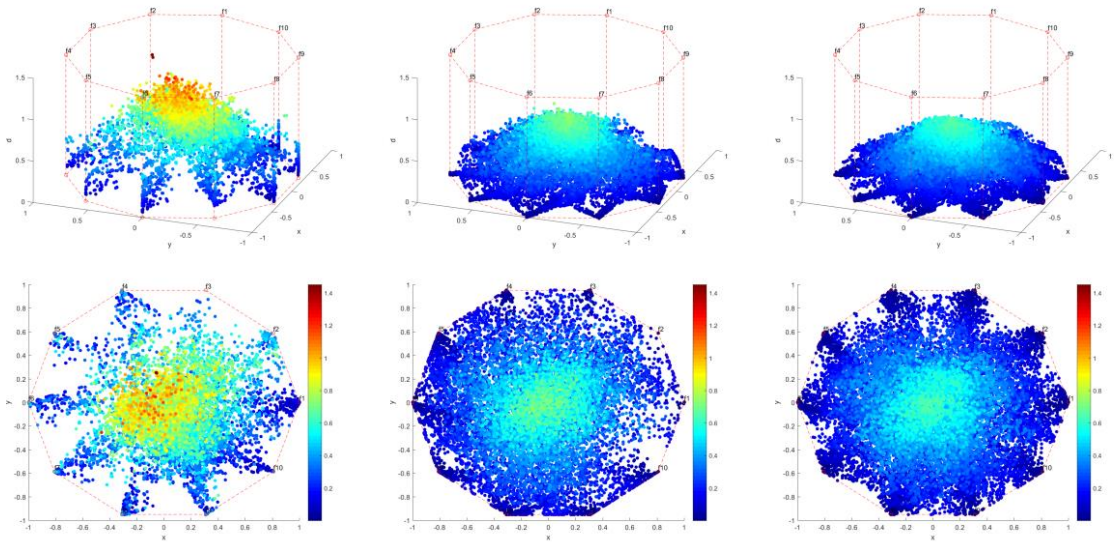
รูปที่ 5 กลุ่มคำตอบที่ดีที่สุด (ฟรอนเทียร์ที่ 1) จากการทำซ้ำทั้งหมด 30 ครั้ง ของแต่ละอัลกอริทึม ในปัญหา ZDT3



รูปที่ 6 กลุ่มคำตอบที่ดีที่สุด (ฟรอนเทียร์ที่ 1) จากการทำซ้ำทั้งหมด 30 ครั้ง ของแต่ละอัลกอริทึม ในปัญหา DTLZ1



รูปที่ 7 กลุ่มคำตอบที่ดีที่สุด (ฟรอนเทียร์ที่ 1) จากการทำซ้ำทั้งหมด 30 ครั้ง DTLZ2 3 วัตถุประสงค์



รูปที่ 8 กลุ่มคำตอบที่ดีที่สุด (ฟรอนเทียร์ที่ 1) จากการทำซ้ำทั้งหมด 30 ครั้ง ของแต่ละอัลกอริทึม ในปัญหา DTLZ2 10 วัตถุประสงค์ เรียงจากลำดับจากซ้ายไปขวาดังนี้ MOEA/D, MODE/D และ AMOEA/D-DE

7. สรุปผล

การแบ่งปัญหาออกเป็นปัญหาย่อยถูกนำมาใช้กันอย่างแพร่หลายในการเขียนโปรแกรมทางคณิตศาสตร์ สำหรับแก้ปัญหา MaOPs ในทางกลับกัน MOEAs ใช้การคัดเลือกคำตอบที่ไม่ถูกครอบงำในการวัดคุณภาพของคำตอบระหว่างการค้นหาเท่านั้น จึงทำให้อัลกอริทึมเหล่านี้ให้ผลลัพธ์การค้นหาคำตอบที่ไม่ดีเท่าที่ควร เพราะเกิดการกระจุกตัวของคำตอบจำนวนมาก ทำให้ MOEA/D เป็นที่นิยมและถูกนำมาใช้อย่างแพร่หลาย

ผู้วิจัย ได้นำเสนอ AMOEA/D-DE ซึ่งมีพื้นฐานมาจาก MOEA/D โดย AMOEA/D-DE จะแบ่ง MaOPs ออกเป็นปัญหาย่อย จากนั้นจะทำการพัฒนาคำตอบของแต่ละปัญหาย่อยไปพร้อม ๆ กันด้วยกระบวนการวิวัฒนาการโดยใช้ผลต่าง (DE Operator) โดยที่แต่ละปัญหาย่อยจะถูกกำหนดปัญหาย่อยข้างเคียงด้วยระยะห่างของจุดพิกัดค่าถ่วงน้ำหนักของแต่ละปัญหาย่อย เพื่อเปรียบเทียบคำตอบที่พัฒนากับปัญหาย่อยในจุดพิกัดที่ใกล้เคียงกัน เนื่องจากปัญหาย่อยที่มีจุดพิกัดของค่าถ่วงน้ำหนักที่ใกล้กันควรมีคำตอบที่ใกล้เคียงกัน

ซึ่งจากการทดลองด้วยค่าพารามิเตอร์เทียบเคียงจากงานวิจัยอื่น ๆ พบว่า AMOEA/D-DE ให้สมรรถนะที่ดีกว่าอัลกอริทึมอื่น ๆ แต่ต้องใช้เวลาในการดำเนินงานที่มากกว่าเมื่อเทียบจำนวนในการพัฒนาคำตอบที่เท่ากัน

อัลกอริทึม AMOEA/D-DE ในงานวิจัยนี้ สามารถปรับตนเองและวิธีการค้นหาคำตอบให้เหมาะสมกับสถานการณ์ ณ ขณะใด ๆ ของการหาคำตอบดีที่สุด ทำให้สามารถค้นหาคำตอบได้หลากหลายและดีขึ้น แต่ยังมีข้อจำกัดในด้านการปรับตัวให้อัลกอริทึมเข้ากับปัญหา เนื่องจากอัลกอริทึมนี้ มีการกำหนดค่าเวกเตอร์ถ่วงน้ำหนักที่ชัดเจนและคงที่ตลอดการดำเนินงานอัลกอริทึม ซึ่งอาจทำให้อัลกอริทึมไม่สามารถค้นหาคำตอบของปัญหาที่มีรูปแบบความเป็นไปได้ของคำตอบกระจุกตัวเป็นกลุ่ม

ในงานวิจัยนี้ เป็นการพัฒนาอัลกอริทึมเพื่อนำมาทดลองแก้ไขปัญหามาตรฐาน ซึ่งเป็นปัญหาที่ไม่พบมากในชีวิตประจำวันหรืออุตสาหกรรมจริงในปัจจุบัน ดังนั้น การนำอัลกอริทึมไปประยุกต์ใช้แก้ไขปัญหาทั่วไปและปัญหาในอุตสาหกรรม อาจช่วยลดเวลาในการค้นหาคำตอบของปัญหาที่มีความซับซ้อนให้ลงและยังคงได้คำตอบที่มีประสิทธิภาพสูงได้

8. เอกสารอ้างอิง

[1] Z. He and G. G. Yen, "Many-Objective Evolutionary Algorithm: Objective Space Reduction and Diversity Improvement," *IEEE Transactions on Evolutionary Computation,* vol. 20, no. 1, pp. 145-160, 2016.

[2] D. Brockhoff and E. Zitzler, "Objective reduction in evolutionary multi-objective optimization: Theory and applications," *Evolutionary Computation,* vol. 17, no. 2, pp. 135-166, 2009.

[3] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation,* vol. 11, no. 6, pp. 712-731, 2007.

[4] H. Li and Q. Zhang, "A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages," in *Parallel problem solving from nature-PPSN IX*: Springer, 2006, pp. 583-592.

[5] K. Miettinen, "Nonlinear Multiobjective Optimization, volume 12 of International Series in Operations Research and Management Science," ed: Kluwer Academic Publishers, Dordrecht, 1999.

[6] M. Ehrgott, "A discussion of scalarization techniques for multiple objective integer programming," *Annals of Operations Research,* vol. 147, no. 1, pp. 343-360, 2006.

[7] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on evolutionary computation,* vol. 13, no. 2, pp. 284-302, 2009.

[8] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Transactions on Evolutionary Computation,* vol. 14, no. 3, pp. 456-474, 2010.

[9] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization,* vol. 11, no. 4, pp. 341-359, 1997.

[10] S. M. Venske, R. A. Gonçalves, and M. R. Delgado, "ADEMO/D: Multiobjective optimization by an adaptive differential evolution algorithm," *Neurocomputing,* vol. 127, pp. 65-77, 2014.

[11] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[12] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Evolutionary Computation,* 2005. *The* 2005 *IEEE Congress on*, 2005, vol. 2, pp. 1785-1791: IEEE.

[13] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE transactions on evolutionary computation,* vol. 10, no. 6, pp. 646-657, 2006.

[14] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Computing,* vol. 10, no. 8, pp. 673-686, 2006.

[15] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm

for constrained real-parameter optimization," in *Evolutionary Computation,* 2006. *CEC* 2006. *IEEE Congress on*, 2006, pp. 17-24: IEEE.

[16] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary computation,* vol. 3, no. 2, pp. 82-102, 1999.

[17] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the Lévy probability distribution," *IEEE Transactions on Evolutionary Computation,* vol. 8, no. 1, pp. 1-13, 2004.

[18] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing,* vol. 9, no. 6, pp. 448-462, 2005.

[19] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation,* vol. 13, no. 5, pp. 945-958, 2009.

[20] S. M. S. Venske, R. A. Goncalves, and M. R. Delgado, "ADEMO/D: Adaptive Differential Evolution for Multiobjective Problems," presented at the 2012 Brazilian Symposium on Neural Networks, 2012.

[21] J. Zhang and A. C. Sanderson, *Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization*. Scientific Publishing Services Pvt. Ltd., Chennai, India., 2009, p. 163.

[22] Q. Lin, Q. Zhu, P. Huang, J. Chen, Z. Ming, and J. Yu, "A novel hybrid multi-objective immune algorithm with adaptive differential evolution," *Computers & Operations Research,* vol. 62, pp. 95-111, 2015.

[23] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation,* vol. 8, no. 2, pp. 173-195, 2000.

[24] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Evolutionary Computation,* 2002. *CEC'*02. *Proceedings of the* 2002 *Congress on*, 2002, vol. 1, pp. 825-830: IEEE.

[25] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[26] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on evolutionary computation,* vol. 7, no. 2, pp. 117-132, 2003.

[27] S. Jiang, Y. S. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *IEEE Trans Cybern,* vol. 44, no. 12, pp. 2391-404, Dec 2014.

[28] A. Ibrahim, S. Rahnamayan, M. V. Martin, and K. Deb, "3D-RadVis: Visualization of Pareto front in many-objective optimization," in *Evolutionary Computation (CEC),* 2016 *IEEE Congress on*, 2016, pp. 736-745: IEEE.