หน่วยการสร้างในการคำนวณเชิงวิวัฒน์เพื่อแก้ปัญหาหลายจุดประสงค์

นายจิระเดช พลสวัสดิ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2552

BUILDING BLOCKS IN EVOLUTIONARY COMPUTATION FOR SOLVING MULTI-OBJECTIVE PROBLEMS

Mr. Jiradej Ponsawat

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Engineering
Department of Computer Engineering
Chulalongkorn University
Academic Year 2009

หัวข้อวิทยานิพนธ์	หน่วยการสร้างในการคำนวณเชิงวิวัฒน์เพื่อแก้ปัญหาหลายจุดประสงค์
โดย	นายจิระเดช พลสวัสดิ์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาดุษฎีบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.สาธิต วงศ์ประทีป)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

..... กรรมการ
(ศาสตราจารย์ ดร.บุญเสริม กิจศิริกุล)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

..... กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.ณชล ไชยรัตนะ)

จิระเดช พลสวัสดิ์ : หน่วยการสร้างในการคำนวณเชิงวิวัฒน์เพื่อแก้ปัญหาหลายจุดประสงค์ (BUILDING BLOCKS IN EVOLUTIONARY COMPUTATION FOR SOLVING MULTI-OBJECTIVE PROBLEMS). อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ศ. ดร.ประภาส จงสถิตย์วัฒนา, 90 หน้า.

วิทยานิพนธ์ฉบับนี้ ได้ศึกษาการใช้การระบุหน่วยการสร้างโดยเมทริกซ์ไคกำลังสอง (BICM) เพื่อปรับปรุงวิธีการไขว้เปลี่ยนให้เป็นแบบใช้หน่วยการสร้าง เพื่อนำไปแก้ปัญหาหลายจุดประสงค์ การทดลองในวิทยานิพนธ์นี้ได้ทำการศึกษาทั้งในกรณีจุดประสงค์เดียวและหลายจุดประสงค์ ในกรณีปัญหาจุดประสงค์เดียวได้ทำการทดลองนำกลวิธีระบุหน่วยการสร้างไปใช้ปรับปรุงการไขว้เปลี่ยนของขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย และได้พบว่าสามารถแก้ปัญหากับดักได้ดีกว่าขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย โดยเฉพาะปัญหาขนาดใหญ่และมีหน่วยการสร้างหลวม จากนั้นได้ทดลองแก้ปัญหาหลายจุดประสงค์ โดยปรับวิธีการไขว้เปลี่ยนของขั้นตอนวิธี NSGA-II ให้เป็นวิธีการไขว้เปลี่ยนแบบใช้หน่วยการสร้าง และได้ออกแบบวิธีการหาและใช้หน่วยการสร้างสำหรับปัญหาหลายจุดประสงค์ที่ต่างกัน 4 แบบ คือ 1) การใช้หน่วยการสร้างร่วมจากคำตอบในพาเรโตฟรอนท์ 2) การผสมหน่วยการสร้างแบบเชื่อมั่น 3) การผสมหน่วยการสร้างแบบขยาย และ 4) ใช้หน่วยการสร้างจากแต่ละจุดประสงค์แยกกัน จากผลการทดลองพบว่าสำหรับปัญหาขนาดใหญ่ขั้นตอนวิธีที่นำเสนอทั้งสี่แบบ ให้ผลการทดลองดีกว่าขั้นตอนวิธี NSGA-II เดิมซึ่งใช้วิธีการไขว้เปลี่ยนแบบสองจุด

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2552

4771851121 : MAJOR COMPUTER ENGINEERING
KEYWORDS : BUILDING BLOCK / MULTI-OBJECTIVE / GENETIC ALGORITHM

JIRADEJ PONSAWAT : BUILDING BLOCKS IN EVOLUTIONARY COMPUTATION FOR SOLVING MULTI-OBJECTIVE PROBLEMS. THESIS ADVISOR : PROF. PRABHAS CHONGSTITVATANA, Ph.D., 90 pp.

This thesis studies the use of Building Block identification in conjunction with Chi-square matrix to solve multi-objective problems. The canonical crossover operator in the evolutionary algorithm is substituted by the Building-block-wise crossover operator. The experiments in this thesis are carried out on both single and multi-objective problems. In case of single objective problems, the crossover operator in Simple Genetic Algorithm is replaced by the BB-wise one. The results show that for the trap problems, BB-wise Genetic Algorithm has better performance especially for difficult problems such as the large problems and the problems that have loose linkage. For multi-objective problems, the well-known algorithm, NSGA-II is used as the base algorithm. Four different Building-block-wise crossover operators are proposed and investigated: 1) BB based on solutions in Pareto front 2) BB from all objectives in conjunctive form 3) BB from all objectives in disjunctive form 4) BB by mixing BB from each objective. The results of the experiments show that for the large problems, all proposed operators out-perform the two-point crossover operator in the original NSGA-II.

Department: Computer Engineering. Student's Signature Jiradej Ponsawat
Field of Study: Computer Engineering. Advisor's Signature
Academic Year: 2009

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา ศ. ดร. ประภาส จงสถิตย์วัฒนา ที่ได้ให้คำแนะนำ ให้แนวทางการแก้ปัญหา แนะนำวิธีคิด และให้กำลังใจยามที่ผู้วิจัยรู้สึกท้ออย่างยิ่ง ตลอดหลายปีที่ผู้วิจัยได้มีโอกาสทำงานกับท่าน เป็นประสบการณ์อันมีค่ายิ่ง หลายสิ่งหลายอย่างที่ได้เรียนรู้จากท่านทำให้ผู้วิจัยมีวันนี้ได้

ขอขอบคุณกรรมสอบวิทยานิพนธ์ทุกท่าน ได้แก่ รศ.ดร.สาธิต วงศ์ประทีป ศ.ดร. บุญเสริม กิจศิริกุล ผศ.ดร.อาทิตย์ ทองทักษ์ และ รศ. ดร.ณชล ไชยรัตนะ ในการตรวจแก้และให้คำแนะนำอันเป็นประโยชน์ยิ่งต่องานวิจัย

ตลอดระยะเวลาการศึกษา ผู้วิจัยได้รับทุนสนับสนุนจากสำนักงานคณะกรรมการการอุดมศึกษา ทุนพัฒนาอาจารย์สาขาขาดแคลน สังกัดมหาวิทยาลัยขอนแก่น จึงขอขอบคุณไว้ ณ ที่นี้ด้วย

วิทยานิพนธ์นี้สำเร็จได้จากความช่วยเหลือของบุคคลหลายท่านที่คอยให้ความช่วยเหลือ ให้คำแนะนำ ข้อคิดเห็นต่างๆ และให้กำลังใจด้วยดีตลอดมา ดังนี้ พี่คทา ประดิษฐ์วงศ์ (สำหรับคำแนะนำแนวทางในการวิจัย) คุณเฉลิมทรัพย์ สังขวิจิตร คุณสุนิสา ริมเจริญ และคุณธนัสนี เพียรตระกูล(สำหรับคำปรึกษาในทุก ๆ เรื่อง) น้องมิ้ง SPACE Lab (สำหรับความช่วยเหลือในการใช้ server ในการทดลอง) คุณสาคร เมฆรักษาวนิช คุณมหศักดิ์ เกตุฉ่ำ (สำหรับความช่วยเหลือในการเตรียมเล่มวิทยานิพนธ์) ขอขอบคุณ พี่ ๆ เพื่อน ๆ น้อง ๆ ทุกคนที่ไม่สามารถเอ่ยชื่อในที่นี้ได้หมด สำหรับน้ำใจที่มีให้ตลอดมา

สุดท้ายนี้ ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ พ่อตา แม่ยาย พี่เอ๋ พี่หน่อย พี่ศักดิ์-พี่อ้อ พี่เทพ-พี่หญิงและญาติพี่น้องทุกท่านที่คอยเป็นห่วงเป็นใย ให้กำลังใจ และให้การสนับสนุนในทุกด้านตลอดมา และที่ขาดไม่ได้ขอขอบคุณสำหรับกำลังใจที่มีให้ตลอดเวลาจาก อังคณา พลสวัสดิ์ น้องฟางและน้องทิว (ภรรยาและลูกๆ ผู้เป็นกำลังใจที่มีค่ายิ่ง)

สารบัญ

บทที่ หน้า

สารบัญตาราง

สารบัญภาพ

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

โดยปกติปัญหาในโลกจริงมักจะเป็นปัญหาหลายจุดประสงค์ และแต่ละจุดประสงค์ก็มักจะขัดแย้งกันเอง ถ้าคำตอบมีค่าในจุดประสงค์หนึ่งดี ค่าในจุดประสงค์อื่นมักจะแย่ลง ดังนั้นคำตอบที่ได้มักจะมีหลายคำตอบ ซึ่งไม่สามารถชี้ชัดได้ว่าคำตอบใดดีกว่ากัน แต่จะเป็นการถ่วงดุลของแต่ละจุดประสงค์นั่นเอง ตัวอย่างเช่น ปัญหาการออกแบบวงจรที่ต้องการลดเวลาในการทำงานให้สามารถทำงานได้เร็วขึ้นโดยใช้ทรัพยากรให้น้อยลง [1] สองจุดประสงค์นี้มักจะไม่ค่อยไปด้วยกัน ถ้าต้องการให้ทำงานเร็วมักจะใช้ทรัพยากรมาก และถ้าต้องการใช้ทรัพยากรน้อยการทำงานก็มักจะช้าไปด้วย ปัญหาการแทรกลายน้ำ [2] เป็นอีกตัวอย่างหนึ่งซึ่งต้องการความทนทานของลายน้ำ (Robustness) และในขณะเดียวกันก็ต้องการซ่อนลายน้ำไม่ให้กระทบกระเทือนกับคุณภาพของต้นฉบับด้วย (Distortion) สองสิ่งนี้มักไม่ไปด้วยกัน ถ้าต้องการความทนทานมากก็ต้องแทรกลายน้ำในปริมาณมาก แต่ก็จะทำให้สังเกตได้หรืออาจทำให้คุณภาพต้นฉบับเสียไป แต่ถ้าซ่อนลายน้ำไว้น้อยความทนทานของลายน้ำก็จะลดลง เมื่อถูกโจมตีลายน้ำก็จะหายไปได้ง่าย ลักษณะเช่นนี้เป็นธรรมชาติของปัญหาหลายจุดประสงค์ซึ่งแต่ละจุดประสงค์มักจะขัดแย้งกันเอง

ขั้นตอนวิธีเชิงวิวัฒน์ (Evolutionary Algorithms: EAs) ถูกนำไปใช้สำหรับแก้ปัญหาหลายจุดประสงค์ (Multi-objective Problem: MOP) อย่างแพร่หลาย [3] [4] [5] [6] [7] ส่วนขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithm: GA) [8] [9] เป็นวิธีการหนึ่งในกลุ่มนี้ที่ได้รับความนิยม เนื่องจากสามารถนำไปแก้ปัญหาได้หลากหลาย นอกจากนี้ในการทำงานแต่ละครั้งทำให้ได้คำตอบหลายคำตอบพร้อมกัน มีการปรับปรุงขั้นตอนวิธีเชิงพันธุกรรมออกมาหลากหลายรูปแบบเพื่อใช้ในการแก้ปัญหาหลายวัตถุประสงค์ [3]

ความสำเร็จของขั้นตอนวิธีเชิงพันธุกรรมถูกอธิบายไว้ด้วยความสามารถในการประกอบหน่วยการสร้าง (Building Blocks) หน่วยการสร้างในที่นี้ คือ ส่วนย่อย ๆ ที่จะรวมกันเป็นคำตอบ ที่ผ่านมาได้มีความพยายามที่จะระบุหน่วยการสร้างเรื่อยมา หนึ่งในความพยายามเหล่านี้คืองานวิจัยที่ทำการระบุหน่วยการสร้างโดยใช้เมทริกซ์ความพร้อมเพรียง (Simultaneity Matrix) [10] [11] และได้นำ เสนอเทคนิคสำหรับระบุหน่วยการสร้าง (Building Block Identification: BBI) เพื่อนำมาแก้ปัญหาที่มีหน่วยการสร้างเด่นชัดในกลุ่มปัญหาที่แบ่งแยกย่อยได้ (Additively Decomposable Functions: ADFs) เช่น กลุ่มปัญหาหลอก (Deceptive Problems) วิธีการนี้ใช้เมทริกซ์ความพร้อมเพรียงสำหรับวัดความสัมพันธ์ของตัวแปรในคำตอบ

จากนั้นใช้ขั้นตอนวิธีการแบ่งส่วน (PAR algorithm) สำหรับแบ่งกลุ่มตัวแปรที่ขึ้นต่อกัน เพื่อนำไปใช้ในระหว่างกระบวนการไขว้เปลี่ยน (Crossover) ทำให้บิตที่มีความสัมพันธ์กันไม่ถูกแยกออกจากกัน เพื่อช่วยป้องกันไม่ให้หน่วยการสร้างของคำตอบที่ดีอยู่แล้วถูกทำลายโดยตัวดำเนินการไขว้เปลี่ยน ในขณะที่ขั้นตอนวิธีเชิงพันธุกรรมแบบธรรมดา (Simple GA) ไม่สามารถป้องกันได้ นำไปเปรียบเทียบผลกับขั้นตอนวิธี BOA (Bayesian Optimization Algorithm) [12] และขั้นตอนวิธี hBOA (hierarchical BOA) [13] ซึ่งเป็นขั้นตอนวิธีประมาณการแจกแจงอย่างหนึ่ง (Estimation of Distribution Algorithms: EDAs) [14] [15] กลุ่มของขั้นตอนวิธีประมาณการแจกแจงจะสร้างแบบจำลอง (model) ของคำตอบเพื่อเป็นตัวแทนของคำตอบและใช้แบบจำลองนี้สุ่มสร้างคำตอบใหม่ จากนั้นคัดเลือกคำตอบดี ๆ เพื่อนำกลับมาสร้างเป็นแบบจำลองอีก ทำเช่นนี้วนไปเรื่อย ๆ จนกว่าจะได้คำตอบที่พอใจ แต่เนื่องจากกระบวนการเรียนรู้แบบจำลองใช้เวลาในการสร้างแบบจำลองที่ดีที่สุดนาน ตัวอย่างเช่น กระบวนการสร้างเครือข่ายของเบส์ (Bayesian Believe Network) [16] [17] ผลการเปรียบเทียบสรุปว่าขั้นตอนวิธี hBOA ใช้จำนวนครั้งการประเมินค่าคำตอบน้อยกว่า แต่การคำนวณเมทริกซ์เทียบกับการสร้างโครงข่ายของเบย์ ใช้เวลาน้อยกว่าถึง 10 เท่าและใช้หน่วยความจำน้อยกว่า 10 เท่าด้วย นอกจากตัววัดที่ใช้เมทริกซ์ความพร้อมเพรียงแล้ว ได้มีงานวิจัยอื่นที่ปรับปรุงให้ใช้เมทริกซ์ไคกำลังสอง [18] ซึ่งเป็นตัววัดทางสถิติที่น่าเชื่อถือมาใช้แทนเมทริกซ์ความพร้อมเพรียงได้ โดยที่ยังสามารถใช้หลักการอื่น ๆ เช่นเดียวกับงานวิจัยเดิมได้

นอกจากนี้ขั้นตอนวิธีการระบุหน่วยการสร้างโดยใช้เมทริกซ์ไคกำลังสองมีความบกพร่องอยู่ประการหนึ่งคือ จะทำงานได้ดีเมื่อคำคอบมีความหลากหลาย (diversity) อยู่ระดับหนึ่ง ทำให้ค่อนข้างจะมีปัญหากับการใช้งานในปัญหาจุดประสงค์เดียว (Single Objective Problem) ในช่วงที่คำตอบลู่เข้าสู่คำตอบเดียว การระบุหน่วยการสร้างจะไม่แสดงถึงโครงสร้างจริงของปัญหา ใน BOA ก็มีคุณลักษณะนี้เช่นเดียวกัน [19] เนื่องมาจากขั้นตอนการสุ่มตัวอย่างในปริมาณจำกัด แม้ว่าโดยภาพรวมแล้วจะสามารถหาคำตอบได้เร็วขึ้น แต่ก็ไม่อาจปฏิเสธได้ว่าประสิทธิภาพของการผสมคำตอบเพื่อให้ได้คำตอบที่ดีขึ้นลดลง และอาจจะไม่สามารถชักนำ (guide) ให้มีคุณภาพเพิ่มขึ้น เนื่องจากหน่วยการสร้างที่ตรวจสอบได้ระบุตำแหน่งการไขว้เปลี่ยนที่ไม่ก่อประโยชน์ แต่เนื่องด้วยธรรมชาติของปัญหาหลายจุดประสงค์ที่คำตอบในพาเรโตฟรอนท์มีความหลากหลายของคำตอบอยู่แล้วทำให้ผลของการระบุหน่วยการสร้างด้วยเมทริกซ์ไคกำลังสองน่าจะสามารถระบุหน่วยการสร้างได้ดีกว่า แม้จะยังไม่ได้ใช้ขั้นตอนวิธีสำหรับรักษาความหลากหลายของคำตอบ (maintain diversity) อย่างไรก็ดีในการแก้ปัญหาหลายจุดประสงค์ยังคงต้องการรักษาความหลากหลายของคำตอบเพื่อประโยชน์ในการกระจายตัวของคำตอบในพาเรโตฟรอนท์ [20]

งานวิจัยเกี่ยวกับการแก้ปัญหาด้วยหน่วยการสร้างส่วนใหญ่ยังสนใจอยู่ในปัญหาจุดประสงค์เดียว เมื่อไม่นานมานี้เริ่มมีคนสนใจที่จะศึกษาหน่วยการสร้างในปัญหาหลาย

จุดประสงค์ ตัวอย่างเช่น ในวิทยานิพนธ์ของ Day [4] และในงานวิจัย [6] ได้มีการออกแบบปัญหาหลายจุดประสงค์ที่มีหน่วยการสร้างเด่นชัดไว้ด้วย และจากการที่กระบวนการระบุหน่วยการสร้างสามารถแก้ปัญหาจุดประสงค์เดียวได้ดี จึงต้องการศึกษาความเป็นไปได้ และประสิทธิภาพของการนำขั้นตอนวิธีระบุหน่วยการสร้างโดยใช้เมทริกซ์ไคกำลังสองที่ชัชวิทย์และประภาสได้เสนอไว้ใน [10] [11] และ [18] มาแก้ปัญหาหลายจุดประสงค์และปรับปรุงให้เหมาะสมกับปัญหาหลายจุดประสงค์

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อประยุกต์ใช้การจำแนกส่วนประกอบที่เรียกว่าหน่วยการสร้างในขั้นตอนวิธีเชิงพันธุกรรมให้สามารถแก้ปัญหาหลายจุดประสงค์ได้ดีและเหมาะสม

1.3 ขอบเขตของการวิจัย

1. ขั้นตอนวิธีที่จะพัฒนาอยู่ในกลุ่มหน่วยการสร้างชัดแจ้ง (Explicit Building Blocks) คือมีการระบุหน่วยการสร้างเพิ่มเติมจากขั้นตอนวิธีเชิงพันธุกรรมโดยทั่วไป
2. ปัญหาที่ใช้ทดสอบเป็นปัญหาในโดเมนไม่ต่อเนื่อง (Discrete Domain)
3. ใช้แนวคิดพาเรโตร่วมในการค้นหาคำตอบ (Pareto Based Approach)

1.4 ขั้นตอนและวิธีดำเนินการวิจัย

1. ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง
2. วิเคราะห์และออกแบบการทดลอง
3. พัฒนาขั้นตอนวิธีที่นำเสนอ
4. ทดสอบประสิทธิภาพและปรับปรุงวิธีการที่นำเสนอ
5. วิเคราะห์และสรุปผล
6. จัดทำเอกสารงานวิจัย

1.5 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

1. ได้ขั้นตอนวิธีการประยุกต์ใช้หน่วยการสร้างในปัญหาจุดประสงค์เดียวให้เหมาะสำหรับปัญหาหลายจุดประสงค์
2. สามารถแก้ปัญหาหลายจุดประสงค์ที่มีหน่วยการสร้างเด่นชัดได้

3. มีความรู้ความเข้าใจในการใช้หน่วยการสร้างมากขึ้น เพื่อนำไปใช้ศึกษาและพัฒนาสำหรับแก้ปัญหาหลายจุดประสงค์โดยทั่วไปได้มากยิ่งขึ้น

1.6 เนื้อหาในวิทยานิพนธ์นี้

รายละเอียดต่าง ๆ ในวิทยานิพนธ์จะนำเสนอเป็นลำดับดังนี้ ในบทที่ 2 จะกล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง ได้แก่ ขั้นตอนวิธีเชิงพันธุกรรม หน่วยการสร้าง และขั้นตอนวิธีเชิงพันธุกรรมสำหรับแก้ปัญหาหลายจุดประสงค์ บทที่ 3 จะเสนอวิธีการนำขั้นตอนวิธีเชิงพันธุกรรมที่ใช้หน่วยการสร้างมาใช้ในการแก้ปัญหาจุดประสงค์เดียว

ในบทที่ 4 เสนอวิธีการใช้งานการระบุหน่วยการสร้างในปัญหาหลายจุดประสงค์แบบต่างๆ และจะนำวิธีที่เสนอไปทำการทดลองเปรียบเทียบผลในบทถัดไป บทที่ 5 นำเสนอปัญหาที่ใช้ในการทดลอง และผลการทดลองเปรียบเทียบขั้นตอนวิธีเชิงวิวัฒน์ที่ใช้หน่วยการสร้างในการแก้ปัญหาหลายจุดประสงค์ รวมทั้งวิเคราะห์ผลการทดลอง สุดท้ายในบทที่ 6 จะสรุปผลการวิจัย แนะแนวทางการนำหน่วยการสร้างไปใช้ในการแก้ปัญหาหลายจุดประสงค์ที่เหมาะสม และแนวทางการทำวิจัยเพิ่มเติมในอนาคต

1.7 งานตีพิมพ์

ในระหว่างการศึกษาได้มีการตีพิมพ์ผลงานวิจัยดังนี้

Ponsawat J., Punyaporn W., Nachol N., and Chongstitvatana P., ”Solving Multi-Objective Problems with Building Blocks Identification”. 7th International Symposium on Communications and Information Technologies (ISCIT2007), Australia, October 16-19, 2007.

Punyaporn, W., Ponsawat, J., and Chongstitvatana, P., "Solving Additively Decomposable Functions by Building Blocks Identification", International Joint Conference on Computer Science and Software Engineering, Thailand, May 2-4, 2007, pp.23-26.

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง ได้แก่ ขั้นตอนวิธีเชิงพันธุกรรม หน่วยการสร้าง ขั้นตอนวิธีระบุหน่วยการสร้างแบบต่างๆ นิยามของปัญหาหลายจุดประสงค์และขั้นตอนวิธีเชิงพันธุกรรมสำหรับแก้ปัญหาหลายจุดประสงค์ ตามลำดับดังต่อไปนี้

2.1 ขั้นตอนวิธีเชิงพันธุกรรม

ขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithm) [8] เป็นกระบวนการค้นหาคำตอบโดยอาศัยหลักการการวิวัฒนาการตามธรรมชาติ มีขั้นตอนดังแสดงในรูปที่ 1

```
gen = 0
Initialize n individuals as P
Evaluate fitness of individuals in P
While ( terminate condition is not met ) and ( gen < MAXGEN) Do
  Select n/2 mates from P
  Produce P' using crossover over each mate
  Perform mutation of P' result P"
  Evaluate P"
  Select new P from (P ∪ P") based on fitness
  gen = gen + 1
End While
```

รูปที่ 1 ขั้นตอนวิธีเชิงพันธุกรรม

กระบวนการของขั้นตอนวิธีเชิงพันธุกรรมมีส่วนที่ต้องพิจารณาดังต่อไปนี้

ก. การเข้ารหัส (Encoding)

การค้นหาคำตอบของขั้นตอนวิธีเชิงพันธุกรรมไม่จำเป็นต้องรู้เกี่ยวกับหน้าตาของคำตอบจริง โดยจะทำการค้นหาคำตอบที่ดีจากคำตอบที่เข้ารหัส (encode) ไว้ มักจะอยู่ในรูปสายอักขระ ทั้งที่เป็นทวิภาค (binary), ตัวเลข (integer) หรือจะเป็นตัวอักษรทั่วไปก็ได้ การทำเช่นนี้ ทำให้เราสามารถย่นย่อปริภูมิของคำตอบจริง ให้เหลือเพียงปริภูมิ ของการเข้ารหัสนั้น ๆ แต่ในทางกลับกัน ถ้าการเข้ารหัสนั้นไม่เหมาะสมกับปัญหาแทนที่จะทำให้ปริภูมิในการหาคำตอบเล็กลง ก็อาจจะทำให้ปริภูมิโตขึ้นได้

ข. การกำหนดค่าเริ่มต้น (Initialization)

โดยปกติแล้วกระบวนการของขั้นตอนวิธีเชิงพันธุกรรมจะเริ่มจากการสุ่มสร้างประชากรที่เป็นคำตอบขึ้นมาจำนวนหนึ่งก่อน แล้วทำการวัดคุณภาพคำตอบ และเลือกคำตอบที่ดีๆ ไปเป็นต้นแบบสำหรับสร้างคำตอบใหม่ที่คาดว่าจะดียิ่งขึ้น อย่างไรก็ดี ในหลายๆ ปัญหามีการคัดเลือกประชากรที่มีแนวโน้มที่ดีจากขั้นตอนวิธีง่ายๆ และรวดเร็ว เพื่อนำมาใช้เป็นประชากรเริ่มต้น เพื่อให้ขั้นตอนวิธีเชิงพันธุกรรมใช้เวลาสั้นลงในการหาคำตอบที่ดีขึ้นกว่าที่มีอยู่ นอกจากนี้ในงานวิจัยที่เป็นการคำนวณแบบขนาน ก็มักจะมีการนำประชากรที่ดีจากหน่วยประมวลผลอื่นๆ มารวมกันเพื่อใช้เป็นประชากรเริ่มต้นในการประมวลผลครั้งต่อไป

ค. การประเมินค่า (Evaluation)

ค่าที่จะใช้บอกความดีของแต่ละคำตอบในที่นี้เรียกว่า ค่าความเหมาะสม (Fitness Value) ซึ่งใช้สำหรับเปรียบเทียบคุณภาพของแต่ละคำตอบ ว่าคำตอบใดดีกว่าคำตอบใดมากน้อยเพียงไร เพื่อนำมาใช้ในกระบวนการเลือกคำตอบดี ๆ มาสร้างคำตอบใหม่ ฟังก์ชันวัดคุณภาพ (Fitness Function) ที่ดีจะสามารถแยกแยะคำตอบที่ดีออกจากคำตอบที่ไม่ดีได้ทำให้กระบวนการค้นหาของขั้นตอนวิธีเชิงพันธุกรรมสามารถทำงานได้ผลดี ในปัญหาหลายจุดประสงค์ปัจจุบันนิยมใช้ค่าความเหมาะสมที่เป็นผลจากการจัดเรียงความเด่นของคำตอบตามแนวคิดพาเรโต (Pareto Dominance) และจะไม่ได้ใช้ค่าความดีในแต่ละจุดประสงค์โดยตรง

ง. การคัดเลือก (Selection)

หลังจากการกำหนดประชากรเริ่มต้นแล้ว กระบวนการคัดเลือกเป็นกระบวนสำคัญที่ใช้เพื่อเลือกคำตอบที่ดีมาเป็นต้นแบบสำหรับสร้างคำตอบที่ดีขึ้นในประชากรรุ่นถัดไป กระบวนการคัดเลือกสามารถทำได้หลายแบบขึ้นอยู่กับแต่ละปัญหาว่ามีคุณลักษณะอย่างไร เช่น ในปัญหาที่ต้องการความหลากหลายของประชากรสูง ก็อาจจะใช้การคัดเลือกแบบแข่งขัน (Tournament Selection) ที่มีปริมาณการแข่งขันน้อย ๆ เช่น สุ่มเลือกประชากรขึ้นมาเพียงสองตัวและเลือกใช้ตัวที่มีค่าความเหมาะสมสูงกว่า

กระบวนการคัดเลือกในปัญหาหลายจุดประสงค์จะมีความพิเศษมากกว่าปัญหาจุดประสงค์เดียว เนื่องจากหนึ่งคำตอบมีค่าความเหมาะสมหลายค่าตามจำนวนจุดประสงค์ และโดยปกติมักจะมีค่าขัดแย้งกันคือ ในจุดประสงค์หนึ่งมีค่ามากส่วนอีกจุดประสงค์หนึ่งอาจจะมีค่าน้อยเมื่อเทียบกับคำตอบอื่น การเปรียบเทียบว่าคำตอบใดดีกว่าจึงต้องพิจารณาจากหลายจุดประสงค์ร่วมกัน

จ. การไขว้เปลี่ยน (Crossover)

การไขว้เปลี่ยนโดยทั่วไปเป็นการรวมสองคำตอบเข้าด้วยกันโดยคาดหวังว่าจะได้คำตอบใหม่ซึ่งมีส่วนประกอบที่ดีของทั้งสองคำตอบทำให้คุณภาพของคำตอบดีขึ้น ในทางกลับกันก็สามารถทำให้เกิดการทำลายคุณภาพคำตอบของทั้งสองคำตอบทำให้คำตอบแย่ลงได้ ในบางกรณี การเลือกตำแหน่งในการไขว้เปลี่ยนที่เหมาะสมจะทำให้สามารถหาคำตอบที่

ต้องการเจอและส่งผลให้แก้ปัญหาได้รวดเร็วยิ่งขึ้น งานวิจัยนี้สนใจที่จะปรับปรุงกระบวนการนี้ให้เหมาะสำหรับปัญหาหลายจุดประสงค์

ฉ. การกลายพันธุ์ (Mutation)

กระบวนการกลายพันธุ์ มีจุดประสงค์หลักในการสร้างความหลากหลายให้กับประชากร เพื่อไม่ให้เกิดการลู่เข้าที่เร็วเกินไป (Premature Convergence) เพราะถ้าความหลากหลายของประชากรลดลงเร็วเกินไปอาจทำให้ได้คำตอบที่ไม่ใช่คำตอบเหมาะที่สุด (Optimal Solution) ส่วนใหญ่แล้วกระบวนการกลายพันธุ์มักจะเป็นกระบวนการที่ทำลายคำตอบ เนื่องจากเป็นการสุ่มเพื่อเปลี่ยนแปลงคำตอบทำให้ไม่สามารถควบคุมหรือทราบผลที่ตามมาได้

ช. การควบคุมความหลากหลายของคำตอบ (Diversity Control)

โดยปกติแล้วการควบคุมความหลากหลายของประชากรจะอยู่ในกระบวนการคัดเลือกคำตอบ ซึ่งจะส่งผลต่อคุณภาพคำตอบที่ได้ โดยเฉพาะในปัญหาหลายจุดประสงค์จะมีคำตอบเป็นกลุ่มของคำตอบที่เรียกว่า พาเรโตฟรอนต์ (Pareto Front) หากขาดความหลากหลายของคำตอบก็จะได้จำนวนจุดของคำตอบน้อยหรือไม่กระจายตัวในพาเรโตฟรอนต์ [20] นอกจากนี้การระบุหน่วยการสร้างที่ใช้เป็นเครื่องมือในงานวิจัยนี้ต้องการให้ประชากรมีความหลากหลายอยู่ปริมาณหนึ่ง [10]

2.1.1 ทฤษฎีบทเค้าร่าง (Schema Theorem)

ทฤษฎีบทเค้าร่างใช้อธิบายพฤติกรรมของขั้นตอนวิธีเชิงพันธุกรรมแบบธรรมดา (Simple Genetic Algorithm) ทฤษฎีบทเค้าร่างอธิบายได้ด้วยสมการต่อไปนี้ กำหนดให้ H คือ เค้าร่าง (Schema) $H = h_0 \ldots h_{l-1}, \quad h_i \in \{0, 1, *\}$

$$m(H,t+1) \geq \frac{m(H,t)\, f(H)}{f_{avg}} \left(1 - p_c \frac{\delta(H)}{l-1} - p_m o(H)\right)$$

เมื่อ $m(H,t)$ คือ จำนวนประชากรที่เข้ากันได้กับเค้าร่าง H ที่เวลา t

$f(H)$ คือ ค่าความเหมาะสมเฉลี่ยของประชากรที่เข้ากันได้กับเค้าร่าง H ที่เวลา t

f_{avg} คือ ค่าความเหมาะสมเฉลี่ยของประชากรทั้งหมด

$\delta(H)$ คือ ความยาวของเค้าร่าง (Length of a Schema)

$o(H)$ คือ อันดับของเค้าร่าง (Order of a Schema)

p_c คือ ความน่าจะเป็นในการไขว้เปลี่ยน

p_m คือ ความน่าจะเป็นในการกลายพันธุ์

l คือ ความยาวของโครโมโซม

<u>หมายเหตุ</u> *ความยาวของเค้าร่าง คือ ระยะห่างมากที่สุดของเลขโดด (digit) ที่ไม่ใช่ $*$*

อันดับของเค้าร่าง คือ จำนวนเลขโดดที่ไม่ใช่ $*$

จากสมการข้างต้นจะได้ว่าจำนวนเค้าร่างที่สั้นและมีอันดับต่ำของประชากรที่มีค่าความเหมาะสมสูงกว่าค่าเฉลี่ยความเหมาะสมของประชากรจะมีปริมาณเพิ่มขึ้นเป็นฟังก์ชันเลขชี้กำลัง Goldberg [8] กล่าวไว้ว่า "เค้าร่างที่สั้น มีอันดับต่ำ และมีค่าความเหมาะสมสูงกว่าค่าความเหมาะสมเฉลี่ยจะมีปริมาณเพิ่มขึ้นเป็นเลขชี้กำลังในประชากรรุ่นถัดๆ ไป" ("short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations") ทฤษฎีบทเค้าร่างนี้ถูกใช้สำหรับนิยามหน่วยการสร้าง

2.1.2 หน่วยการสร้าง (Building Block)

สมมติฐานเริ่มแรกของหน่วยการสร้าง Goldberg [8] และ Grefenstette [21] กล่าวไว้ว่าขั้นตอนวิธีเชิงพันธุกรรมประสบผลสำเร็จด้วยการประกอบเค้าร่างที่สั้นมีอันดับต่ำและมีความเหมาะสมสูง ซึ่งถูกเรียกว่า "หน่วยการสร้าง" (Building Blocks)

Anil Menon ได้ให้สมมติฐานของหน่วยการสร้างประสิทธิผล (Effective Building Block Hypothesis) [22 หน้า.145] ไว้ดังนี้ "ขั้นตอนวิธีเชิงวิวัตน์ทำงานโดยการรวมเค้าร่างอันดับต่ำ ๆ ที่มีค่าความเหมาะสมสูงกว่าค่าเฉลี่ยเพื่อให้ได้เค้าร่างอันดับสูงขึ้นเรื่อย ๆ"

โดยสรุปแล้ว หน่วยการสร้าง ก็คือส่วนของคำตอบที่ดีที่รวมกันแล้วทำให้ได้คำตอบที่ดียิ่งขึ้น หน่วยการสร้างในปัญหาง่ายมักจะไม่ขึ้นต่อกันหรือไม่มีความสัมพันธ์กัน ทำให้การประกอบเข้าด้วยกันทำได้ง่ายและอิสระกว่า แต่ในบางกลุ่มปัญหา เช่น กลุ่มปัญหาหลอก (Deceptive Problems) จะประกอบด้วยหน่วยการสร้างย่อย ๆ ที่ขึ้นต่อกันหรือมีความสัมพันธ์กัน ทำให้การระบุหน่วยการสร้าง (Building Block Identification) มีความสำคัญในการแก้ปัญหาที่มีความเชื่อมโยงระหว่างบิต (Linkage Learning Problem)

2.1.3 การระบุหน่วยการสร้าง (Building Block Identification)

การระบุหน่วยการสร้าง หรือบางครั้งจะรู้จักในนามการเรียนรู้ความเชื่อมโยง (Linkage Learning) [23] หรือปัญหาการเรียงลำดับ (Ordering Problem)

ปัญหาที่มีความเชื่อมโยงระหว่างบิตเป็นปัญหาที่จำเป็นต้องรู้ความสัมพันธ์ระหว่างแต่ละบิตของคำตอบเพื่อใช้ในการปรับปรุงคำตอบ เนื่องจากค่าความเหมาะสมของคำตอบขึ้นอยู่กับกลุ่มของบิตที่ขึ้นต่อกัน หากสามารถระบุหน่วยการสร้างและป้องกันการทำลายหน่วยการสร้างในระหว่างการไขว้เปลี่ยนได้ จะทำให้ขั้นตอนวิธีเชิงพันธุกรรมมีประสิทธิภาพดีขึ้นอย่างมีนัยสำคัญ [24]

มีหลายงานวิจัยที่พยายามแก้ปัญหาโดยอาศัยความรู้เกี่ยวกับความสัมพันธ์ระหว่างบิตของคำตอบโดยเฉพาะในกลุ่มของขั้นตอนวิธีประมาณการแจกแจง (Estimation of Distribution Algorithms: EDAs) ซึ่งจะทำการเรียนรู้แบบจำลองที่ครอบคลุมกลุ่มคำตอบที่ถูกเลือก จากนั้นจะสุ่มสร้างคำตอบใหม่จากแบบจำลองที่ได้ เช่น MIMIC [25], BMDA [26], BOA [12], และ eCGA [23] เป็นต้น

การหากลุ่มเชื่อมโยง (Linkage Groups) หรือหน่วยการสร้าง สามารถทำได้หลายแนวทาง ซึ่งสามารถแบ่งประเภทได้เป็น 3 กลุ่มดังนี้

1. การตรวจสอบโดยการเปลี่ยนแปลงค่าทีละน้อย (Perturbation)

ขั้นตอนวิธีเชิงพันธุกรรมในกลุ่มนี้ตรวจสอบการเชื่อมโยงโดยการเปลี่ยนค่ายีนของคำตอบทีละนิดแล้วตรวจสอบการเปลี่ยนแปลงของค่าความเหมาะสมที่เปลี่ยนแปลงไปหลังจากการปรับค่ายีน ขั้นตอนวิธีในกลุ่มนี้มักจะแบ่งการทำงานออกเป็นสองขั้นตอน ในขั้นตอนแรกจะทำการปรับค่ายีนและตรวจหาความเชื่อมโยง (หน่วยการสร้าง) ในขั้นที่สองจะผสมหน่วยการสร้างที่ได้จากขั้นตอนแรก ตัวอย่างขั้นตอนวิธีที่อยู่ในกลุ่มนี้ได้แก่ mGA [27], fmGA [28], gemGA [29], LINC [30] และ LIMD [31] ซึ่งจะมีรายละเอียดในหัวข้อถัดไป

2. การสร้างแบบจำลอง (Model Building)

การสร้างแบบจำลองอยู่ในจำพวกขั้นตอนวิธีประมาณการแจกแจง ตัวอย่างขั้นตอนวิธีที่อยู่ในกลุ่มนี้ได้แก่ ECGA [23], BOA [32], hBOA [12], และ DSMGA [33] [34] โดยปกติขั้นตอนวิธีในกลุ่มนี้จะมีการสร้างแบบจำลองของปัญหาจากคำตอบที่ถูกคัดเลือก โดยจะประกอบด้วยขั้นตอนวิธีดังต่อไปนี้

1) สุ่มสร้างประชากรใหม่
2) เลือกกลุ่มคำตอบที่ดีกว่า
3) สร้างแบบจำลองตัวแทนคำตอบที่เลือกมา
4) สร้างประชากรรุ่นถัดไปจากแบบจำลองที่ได้
5) ทำซ้ำขั้นตอน 2) ถึง 5) จนกว่าจะเจอเงื่อนไขการหยุด

3. การหาความเชื่อมโยงแบบปรับตัว (Interaction Adaptation)

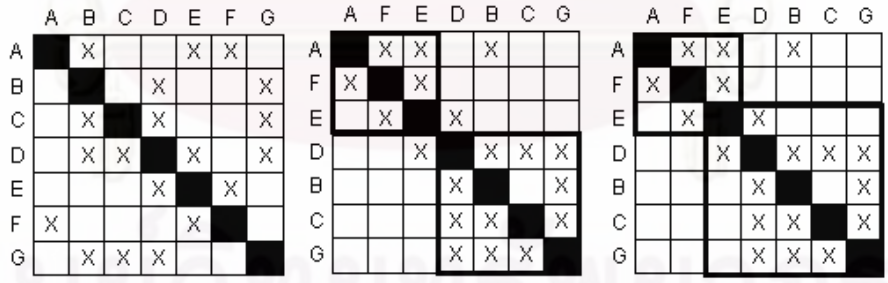
เป็นการทำโดยอ้อมในระหว่างการทำงานของขั้นตอนวิธีเชิงพันธุกรรม การหาความเชื่อมโยงจะแฝงอยู่ในการเข้ารหัส โครโมโซมที่มียีนซึ่งมีความเชื่อมโยงกันสูงอยู่ใกล้กันมากกว่าก็จะมีโอกาสรอดไปยังประชากรรุ่นถัดไปมากกว่า ตัวอย่างขั้นตอนวิธีในกลุ่มนี้คือ LEGO [35] และ LLGA [36]

2.1.4 ขั้นตอนวิธีสำหรับระบุหน่วยการสร้าง

ในหัวข้อนี้จะแสดงขั้นตอนวิธีการระบุหน่วยการสร้างแบบต่างๆ ของแต่ละกลุ่มที่ได้กล่าวถึงในหัวข้อที่ผ่านมา

ก. ขั้นตอนวิธี DSMGA

ขั้นตอนวิธี DSMGA (Dependency Structure Matrix Genetic Algorithm) อยู่ในกลุ่มการสร้างแบบจำลอง (model building) ถูกเสนอโดย Tian-Li Yu [33] [34] สำหรับใช้แก้ปัญหาที่มีการเหลื่อมซ้อนกันของหน่วยการสร้าง ขั้นตอนวิธีนี้ใช้เมทริกซ์ในการแทนความสัมพันธ์ภายในกราฟเพื่อช่วยวิเคราะห์และแยกย่อยปัญหา เครื่องหมาย ‘X’ ในตารางบ่งบอกถึงความสัมพันธ์ระหว่างจุดภายในกราฟ เมื่อดูรูปที่ 2 (a) ประกอบ A กับ B มีความสัมพันธ์กันโดยมีลูกศรชี้จาก A ไปยัง B แนวเส้นทแยงมุมในเมทริกซ์จะไม่มีความหมาย (ในที่นี้สามารถแทนเครื่องหมาย ‘X’ ด้วยหนึ่งและช่องว่างแทนด้วย 0) ในรูปที่ 2 (b) แสดงตัวอย่างการแบ่งกลุ่ม (clustering) เพื่อให้ภายในกลุ่มมีความสัมพันธ์กันเองและสัมพันธ์กับภายนอกกลุ่มให้น้อยที่สุด ทำได้โดยการสลับตำแหน่งแถวและสดมภ์ ในรูป (b) จะได้กลุ่มที่แบ่งแล้วเป็น 2 กลุ่มคือ {A,F,E} และ {D,B,C,G} แต่ก็จะเห็นได้ว่ายังมีความสัมพันธ์บางส่วนยังอยู่ภายนอกกลุ่มที่แบ่งได้ ในรูปที่ 2 (c) แสดงรูปแบบการแบ่งกลุ่มอีกแบบที่มีการเหลื่อมกันของกลุ่มที่แบ่งได้ เกิดจากการเพิ่มขนาดของกลุ่มหนึ่งขึ้นทำให้สองความสัมพันธ์ที่อยู่ภายนอกกลุ่มได้อยู่ภายในกลุ่ม

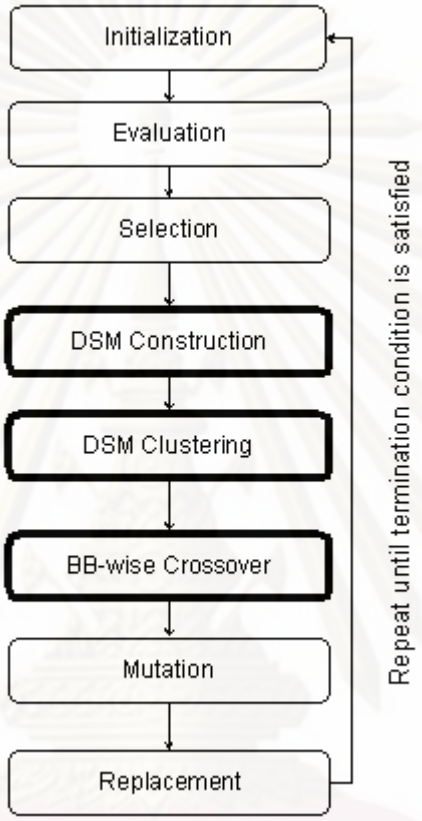


(a) ความสัมพันธ์ดั้งเดิม (b) หลังแบ่งกลุ่ม (c) แบ่งกลุ่มอีกแบบ

รูปที่ 2 การใช้เมทริกซ์แทนความสัมพันธ์ในกราฟของขั้นตอนวิธี DSMGA

กระบวนการแบ่งกลุ่มเป็นขั้นตอนที่ยุ่งยากสำหรับปัญหานี้ มีหลายงานวิจัยที่พยายามจะทำให้ขั้นตอนนี้เป็นอัตโนมัติ เช่นในงานวิจัยของ Fernadez [37] ในปี 1998 ใช้ simulated annealing search เพื่อหาการแบ่งกลุ่ม ในปี 2002 Whitfield และคณะ [38] ใช้ขั้นตอนวิธีเชิงพันธุกรรมเพื่อแบ่งกลุ่ม แต่ก็ยังประสบปัญหาค่าเหมาะสมเฉพาะส่วน (local

optima) ผู้เสนอขั้นตอนวิธีนี้ได้เปลี่ยนมาใช้แบบจำลองความยาวในการอธิบาย (model description length) และแบบจำลองอธิบายข้อมูลที่ไม่สอดคล้อง (the mismatched data) ร่วมกับขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายเพื่อหาการแบ่งกลุ่ม ตัวอย่างเช่น ความสัมพันธ์ที่ไม่อยู่ในกลุ่มจะถูกมองเป็นข้อมูลที่ไม่สอดคล้อง ขั้นตอนการแบ่งกลุ่มนี้มีชื่อว่า DSM Clustering เป็นส่วนหนึ่งของขั้นตอนวิธี DSMGA ดังแสดงในรูปที่ 3



รูปที่ 3 ขั้นตอนการทำงานของ DSMGA

การทำงานหลักๆ ของขั้นตอนวิธีนี้มี 3 ส่วน คือ 1) DSM Construction 2) DSM Clustering และ 3) BB-wise Crossover ขั้นตอนวิธี DSMGA จึงกลายเป็นปัญหาการหาค่าเหมาะสุด (optimization) สองปัญหา ปัญหาแรกเป็นปัญหาที่ต้องการแก้ไข และอีกปัญหาคือปัญหาในการแบ่งกลุ่ม (DSM Clustering) ซึ่งจะเป็นส่วนที่ใช้หาหน่วยการสร้าง โดยมองแต่ละกลุ่มว่าเป็นหน่วยการสร้าง เมื่อได้หน่วยการสร้างแล้วจะใช้วิธีการไขว้เปลี่ยนแบบใช้หน่วยการสร้าง (BB-wise crossover) ที่เสนอโดย Harik ในปี 1999 [23] ซึ่งจะคล้ายกับการไขว้เปลี่ยนแบบบิตในกรณีที่หน่วยการสร้างไม่มีการเหลื่อมกัน สิ่งที่ต่างกันก็คือ การแลกเปลี่ยนจะแลกเปลี่ยนทั้งก้อนหน่วยการสร้าง

สำหรับขั้นตอน DSM Construction จะใช้หาความสัมพันธ์ระหว่างตัวแปร ในการหาความสัมพันธ์ดังกล่าวผู้วิจัยได้เปรียบเทียบวิธีวัดความสัมพันธ์ 3 แบบ คือ 1) การตรวจสอบความเป็นและไม่เป็นเชิงเส้น (Linear/nonlinear Detection) 2) เมทริกซ์ความพร้อมเพรียง

(Simultaneity Matrix) และ 3) การวัดความไร้ระเบียบ (Entropy-based) ในข้อที่ 1) จะได้กล่าวถึงในขั้นตอนวิธีระบุหน่วยการสร้างหัวข้อถัดไป ข้อ 2) เป็นงานก่อนหน้าของขั้นตอนวิธีที่เลือกใช้ในวิทยานิพนธ์ฉบับนี้ และข้อ 3) เป็นพื้นฐานการวัดความสัมพันธ์ในหลายวิธีเช่น BOA, ecGA และ BMDA สำหรับขั้นตอนวิธี DSMGA นี้ Tian-Li Yu [34] เลือกใช้ตัววัดที่ 3) นี้ด้วย คือการวัดความไร้ระเบียบ

ข. ขั้นตอนวิธี LINC และ LIMD

Munetomo และ Goldberg [30] [31] เสนอเทคนิคการระบุหน่วยการสร้างโดยการตรวจสอบความไม่เป็นเชิงเส้นและความไม่ราบเรียบ ขั้นตอนวิธีนี้ใช้วิธีการเปลี่ยนแปลงค่าทีละนิด (perturbation) ใน LINC (Linkage Identification by Nonlinearity Check) ถ้าตำแหน่งบิตคู่ใดที่ถูกตรวจสอบพบว่าไม่เป็นเชิงเส้นอย่างน้อย 1 ครั้ง คู่บิตนั้นก็จะถูกรวมเข้าไปในกลุ่มเชื่อมโยง (หรือหน่วยการสร้าง) เมื่อกำหนดให้ $s_1\ s_2\ s_3 \ldots s_l$ เป็นโครโมโซมของคำตอบ การหาค่าการเปลี่ยนแปลงจะเป็นดังนี้

$$\Delta f_i(s) = f(\cdots \bar{s}_i \cdots\cdots) - f(\cdots s_i \cdots\cdots)$$

$$\Delta f_j(s) = f(\ldots\ldots \bar{s}_j \ldots) - f(\ldots\ldots s_j \ldots)$$

$$\Delta f_{ij}(s) = f(\ldots \bar{s}_i . \bar{s}_j \ldots) - f(\ldots \bar{s}_i . s_j \ldots)$$

เมื่อ $\bar{s}_i = 1 - s_i$ และ $\bar{s}_j = 1 - s_j$ $\bar{s}_i = 1 - s_i$ และ $\bar{s}_j = 1 - s_j$; ในกรณีสายอักขระแบบไบนารี 0, 1

ถ้า $\Delta f_{ij}(s) = \Delta f_i(s) + \Delta f_j(s)$ นั่นคือการเปลี่ยนแปลงเป็นแบบเชิงเส้น (linearity)

แต่ถ้า $\Delta f_{ij}(s) \neq \Delta f_i(s) + \Delta f_j(s)$ แสดงถึงการไม่เป็นเชิงเส้น

จะเห็นว่าการเปลี่ยนแปลงค่านี้จะต้องทำกับทุกคู่บิตที่เป็นไปได้จากคำตอบทั้งหมดในประชากร LINC สามารถตรวจสอบได้เฉพาะปัญหาที่หน่วยการสร้างไม่เหลื่อมกัน ขั้นตอนวิธี LINC จึงถูกปรับให้เป็นขั้นตอนวิธี LIMD (Linkage Identification by non-Monotonicity Detection) ซึ่งจะตรวจสอบว่าการเปลี่ยนแปลงค่าความเหมาะสมอยู่ในทิศทางเดียวหรือไม่ ถ้าการเปลี่ยนแปลงค่าความเหมาะสมไม่เป็นไปตามเงื่อนไขต่อไปนี้จะถือว่าคู่บิตนั้นมีความเชื่อมโยงกัน

ถ้า $(f_i(s) > 0 \text{ และ } \Delta f_i(s) > 0)$ แล้ว $(\Delta f_{ij}(s) > \Delta f_i(s) \text{และ} \Delta f_{ij}(s) > \Delta f_j(s))$
เป็นการตรวจสอบความเรียบแบบเพิ่มค่า

ถ้า $(f_i(s) < 0 \text{ และ } \Delta f_i(s) < 0)$ แล้ว $(\Delta f_{ij}(s) < \Delta f_i(s) \text{และ} \Delta f_{ij}(s) < \Delta f_j(s))$
เป็นการตรวจสอบความราบเรียบแบบลดค่า

จากเงื่อนไขการตรวจสอบข้างต้น ถ้าทิศทางการเปลี่ยนแปลงค่าความเหมาะสมทั้งกรณีเปลี่ยนค่า *i* อย่างเดียว เปลี่ยนค่า *j* อย่างเดียว และเปลี่ยนทั้ง *i* และ *j* อยู่ทิศทางเดียวกันจะถือว่าเป็น

แบบราบเรียบ (monotonicity) แต่ถ้าไม่สอดคล้องตามนี้จะถือว่าเป็นแบบไม่ราบเรียบ (non-monotonicity) ในกรณีที่ไม่ราบเรียบจะถือว่ามีความเชื่อมโยงและอยู่ในหน่วยการสร้างเดียวกัน

ค. ขั้นตอนวิธี messyGA

messy GA [27] เข้ารหัสคำตอบเป็นคู่ลำดับ (p,v) p คือตำแหน่งหรือชื่อของบิต v คือค่าของบิต ทำให้สามารถย้ายตำแหน่งของบิตไปได้เรื่อย ๆ โดยไม่เสียความหมาย เมื่อผสมคำตอบจะทำให้มีกรณีตำแหน่งเกินกำหนด (over-specified) คือตำแหน่งที่ค่าบิตมากกว่า 1 ค่า และมีตำแหน่งต่ำกว่ากำหนด (under-specified) คือไม่มีค่าของบิตในตำแหน่งนั้น มีหลายแนวทางในการตีความตำแหน่งเหล่านี้ ตัวอย่างเช่น ตำแหน่งเกินกำหนดจะใช้ค่าของบิตที่มีจำนวนมากกว่า (Majority Voting) หรือจะเลือกค่าที่เจอก่อน (First Come, First Serve) ส่วนบิตที่ต่ำกว่ากำหนดจะเลือกจากค่าเฉลี่ยของแผ่นแบบแข่งขัน (competitive templates)

ง. ขั้นตอนวิธี BOA

ขั้นตอนวิธี BOA [32] [12] ถูกเสนอโดย Pelikan ซึ่งจะอยู่ในกลุ่มการสร้างแบบจำลอง โดยจะใช้เครือข่ายของเบยส์เป็นแบบจำลอง มีขั้นตอนวิธีการทำงานดังนี้

1) สุ่มสร้างประชากรเริ่มต้น
2) เลือกคำตอบที่ดีจากประชากรรุ่นปัจจุบัน
3) สร้างเครือข่ายของเบยส์จากประชากรที่เลือกมา
4) สร้างประชากรรุ่นถัดไปจากเครือข่ายเบยส์ที่ได้
5) ทำซ้ำขั้นตอน 2. ถึง 4. จนกระทั่งพบเงื่อนไขการหยุด

เครือข่ายของเบยส์เป็นกราฟมีทิศทางแบบไม่มีวงวน แต่ละโนดแทนตัวแปร เส้นเชื่อมแสดงความขึ้นต่อกันของตัวแปร นอกจากนี้ยังมีตารางความน่าจะเป็นกำกับแต่ละโนด ตารางนี้จะมีขนาดขึ้นกับจำนวนโนดก่อนหน้าที่ชี้มายังโนดนั้น ถ้ามีเส้นเชื่อมเข้าหา k เส้น และแต่ละตัวแปรมีค่าต่างๆ กันได้ m รูปแบบ จะได้ตารางมีความน่าจะเป็นกำกับจำนวน $m \cdot m^k$ ค่าปรากฏในตาราง (เท่ากับจำนวนรูปแบบที่เป็นไปได้ทั้งหมดของโนดก่อนหน้าคูณกับจำนวนรูปแบบที่เป็นไปได้ของโนดนั้น) ขั้นตอนวิธีนี้จะจำกัดขนาดของเครือข่ายด้วยค่า k คือจำกัดจำนวนเส้นเชื่อมที่วิ่งเข้าหาแต่ละโนด

เครือข่ายที่ดีที่สุดคือ เครือข่ายที่สอดคล้องกับข้อมูลประชากรที่เลือกมามากที่สุด ในการใช้งานจริงขั้นตอนวิธีนี้ใช้วิธีการเชิงละโมบ (greedy) ในการหาเครือข่ายที่เหมาะสุด การค้นหาแบ่งเป็นสองส่วน

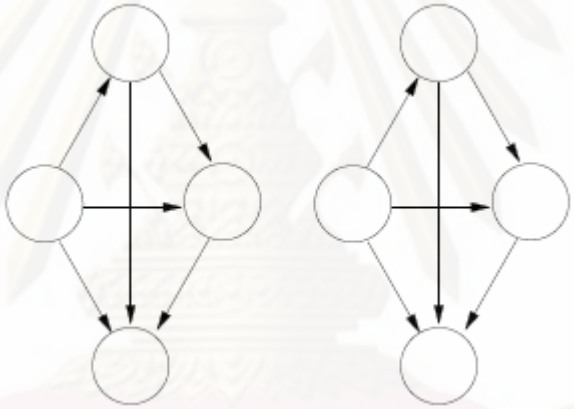
1. **การหาโครงสร้าง** การหาโครงสร้างจะใช้วิธีเชิงละโมบในการหาเครือข่ายเริ่มต้นจากเครือข่ายว่างเปล่าคือ แต่ละโนดไม่มีเส้นเชื่อม จากนั้นทำ

กระบวนการต่อไปนี้ 1) เพิ่มเส้นเชื่อม 2) ลดเส้นเชื่อม 3) กลับทิศเส้นเชื่อม จนกว่าจะไม่สามารถเพิ่มคุณภาพของเครือข่ายตามตัววัดที่เลือกใช้ [16]

2. **การหาความน่าจะเป็น** การหาความน่าจะเป็น จะนำประชากรที่ถูกเลือกมาทำการคำนวณหาความน่าจะเป็นสำหรับแต่ละโนด

การสุ่มสร้างประชากรใหม่ จะสุ่มตามค่าของความน่าจะเป็น ในเครือข่ายโดยเริ่มจากโนดที่ไม่มีเส้นเชื่อมเข้าหา จากนั้นจึงสุ่มสร้างโนดถัดไปที่โนดก่อนหน้าถูกสุ่มค่าเรียบร้อยแล้ว

เครือข่ายที่ได้ของขั้นตอนวิธีนี้ ถือว่าเป็นหน่วยการสร้าง เส้นเชื่อมแสดงความเชื่อมโยงระหว่างตัวแปร ตัวแปรที่มีเส้นเชื่อมโยงต่อเนื่องไปถึงกันถือว่าอยู่ในหน่วยการสร้างเดียวกัน เช่น ในปัญหากับดักขนาด 4 บิต จะได้เครือข่ายเป็นกลุ่มๆ ละ 4 โนด แยกกันดังแสดงในรูปที่ 4



รูปที่ 4 ตัวอย่างโครงสร้างเครือข่ายเบย์สของปัญหากับดักขนาด 4 บิต

จ. ขั้นตอนวิธี ECGA

ขั้นตอนวิธี ECGA (Extended Compact Genetic Algorithm) ถูกเสนอโดย Harik [23] เป็นขั้นตอนวิธีที่อยู่ในกลุ่มการสร้างแบบจำลอง โดยมีแบบจำลองเป็นความน่าจะเป็นแบบกั้นเขต (marginal probability) ดังแสดงในตารางที่ 1 ขั้นตอนวิธีนี้มีขั้นตอนการทำงานดังนี้

1) สุ่มสร้างประชากรเริ่มต้น
2) เลือกประชากรแบบแข่งขัน
3) สร้างแบบจำลองของประชากรโดยใช้การค้นหาแบบละโมบ
4) สิ้นสุดการทำงานถ้าแบบจำลองลู่เข้า
5) กรณียังไม่ลู่เข้าจะสร้างประชากรใหม่จากแบบจำลองที่ได้
6) จากนั้นกลับไปเริ่มขั้นตอนที่ 2

ตารางที่ 1 ความน่าจะเป็นแบบกั้นเขต

[0,3]		[1]		[2]	
ค่าบิต	p	ค่าบิต	p	ค่าบิต	p
00	0.5	0	0.5	0	0.6
01	0	1	0.5	1	0.4
10	0				
11	0.5				

ตารางที่ 1 แสดงความน่าจะเป็นแบบกั้นเขต แต่ละเขตจะเป็นหน่วยการสร้าง ซึ่งจะมีความน่าจะเป็นของแต่ละเขต มีลักษณะพิเศษ คือ เป็นความน่าจะเป็นของบิตมากกว่า 1 บิต ในที่นี้บิต 0 และ 3 อยู่ในเขตเดียวกัน ความน่าจะเป็นของ 2 บิตนี้จะขึ้นต่อกัน ส่วนบิต 1 และบิต 2 เป็นบิตอิสระ การสุ่มแค่เพียงครั้งเดียวจะได้ค่าของบิตที่อยู่ในเขตเดียวกันทั้งหมด

ความน่าจะเป็นของบิต 0 และ 3 มีค่าที่ไม่เป็น 0 อยู่สองแบบ คือ “00” และ “11” นั่นคือไม่มีโอกาสที่บิต 0 จะมีค่าเป็น 1 แล้วบิต 3 มีค่าเป็น 0 นั่นคือสามารถเก็บข้อมูลของทั้งสองบิตนี้ได้ด้วยบิตเดียวคือ 0 แทน “00” และ 1 แทน “11” ทำให้ข้อมูลที่จะใช้เก็บมีขนาดเล็กลง

ขั้นตอนวิธี ECGA จะเลือกการกั้นเขตที่มีค่าของสมการต่อไปนี้น้อยสุด คือ ต้องการเอนโทรปี (entropy) น้อยลง

$$\log n \sum_{I} 2^{S(I)} + n \sum_{I} E(M_I)$$

เมื่อ n เป็นขนาดประชากร

I เป็นเซตย่อยของการกั้นเขต

$S(I)$ เป็นขนาดของเซต I

$E(M_I)$ คือค่าเอนโทรปีของความน่าเป็นแบบกั้นเขตของเซตย่อย I ดังสมการต่อไปนี้

$$E(M_I) = \sum_{p} \begin{cases} -p \log p & ;if \ \ p \neq 0 \\ 0 & ;if \ \ p = 0 \end{cases}$$

เมื่อ p เป็นค่าความน่าจะเป็นในเซต I

ขั้นตอนวิธีสร้างแบบจำลอง เริ่มจากให้ทุกบิตเป็นอิสระจากกันจากนั้นใช้การค้นหาเชิงละโมบ คือ จะพยายามรวมบิตทีละ 2 เซตเข้าด้วยกัน โดยหาจากทุกรูปแบบที่เป็นไปได้ แล้วเลือกเอาการรวมเซตที่ให้ผลรวมในสมการด้านบนน้อยที่สุด ถ้าการรวมยังสามารถลดค่าได้ก็จะรวมเซตเข้าด้วยกัน จนกระทั่งไม่เหลือเซตย่อย เหลือเพียงเซตใหญ่เซตเดียว

ฉ. ขั้นตอนวิธี BICM

การระบุหน่วยการสร้างด้วยเมทริกซ์ไคกำลังสอง (Building-block Identification by Chi-square Matrix: BICM) ประกอบด้วยการทำงาน 2 ส่วนหลักๆ คือ 1) ขั้นตอนการสร้างเมทริกซ์ไคกำลังสอง (Chi-square Matrix Construction: CMC) และ 2) ขั้นตอนการแบ่งกลุ่มความสัมพันธ์ (Partitioning)

BICM เป็นการระบุหน่วยการสร้างที่ใช้ในงานวิจัยนี้มีรายละเอียดในหัวข้อ 3.1 การระบุหน่วยการสร้างโดยเมทริกซ์ไคกำลังสอง

2.2 ปัญหาหลายจุดประสงค์ (Multi-objective Optimization Problem: MOP)

ปัญหาหลายจุดประสงค์โดยทั่วไปสามารถนิยามได้ตามบทนิยาม 1

บทนิยาม 1 ปัญหาหลายจุดประสงค์ (Multi-objective Problem)

คือ ปัญหาที่ต้องการหาเวกเตอร์ $\vec{x}^* = \left[x_1^* \; x_2^* \; x_3^* \; \ldots \; x_n^*\right]^T$ ที่สอดคล้องกับ *m* อสมการข้อบังคับ

$$g_i(\vec{x}) \geq 0 \qquad i = 1, 2, 3, \ldots, m$$

และ *p* สมการข้อบังคับ

$$h_i(\vec{x}) = 0 \qquad i = 1, 2, 3, \ldots, p$$

และทำให้สมการจุดประสงค์ต่อไปนี้มีค่าจุดประสงค์เหมาะที่สุด

$$\vec{f}(\vec{x}) = \left[f_1(\vec{x}) \; f_2(\vec{x}) \; f_3(\vec{x}) \ldots \; f_k(\vec{x})\right]^T$$

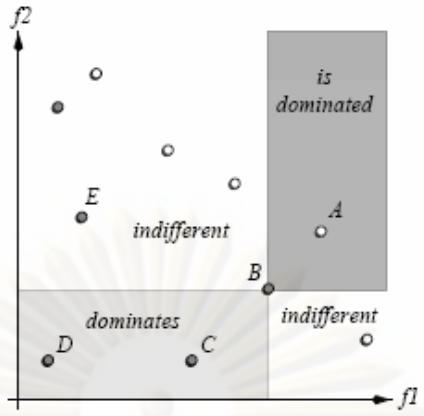
เมื่อ *k* เป็นจำนวนจุดประสงค์

บทนิยาม 2 การครอบงำแบบพาเรโต (Pareto Dominance) [3]

เวกเตอร์ $\vec{u} = (u_1, \ldots, u_k)$ ครอบงำ $\vec{v} = (v_1, \ldots, v_k)$ ก็ต่อเมื่อ u มากกว่า v บางส่วน

$\forall i \in \{1, \ldots, \mathrm{k}\}, u_i \geq v_i \; \wedge \; \exists i \in \{1, \ldots, \mathrm{k}\}: u_i > v_i$

นิยามเกี่ยวกับการครอบงำดูจากค่าของจุดประสงค์ เวกเตอร์ $\vec{u}$ ครอบงำเวกเตอร์ $\vec{v}$ ก็ต่อเมื่อในแต่ละจุดประสงค์ค่าของ *u* ดีกว่าหรือเท่ากับค่าของ *v* และ มีอย่างน้อยหนึ่งกรณีที่ค่าของ *u* มีค่าดีกว่า *v*



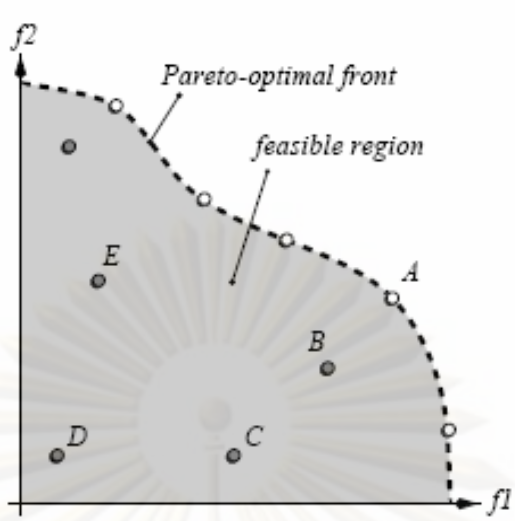
รูปที่ 5 การครอบงำแบบพาเรโต

ในรูปที่ 5 เป็นค่าในแกนจุดประสงค์ และต้องการคำตอบที่ให้ค่าจุดประสงค์มากสุด รูปนี้แสดงให้เห็นการครอบงำ เมื่อพิจารณาคำตอบ B บริเวณสีเทาอ่อนด้านล่างแสดงส่วนที่ B ครอบงำหรือจะเรียกว่าบริเวณที่ถูกครอบงำโดย B บริเวณสีเทาเข้มด้านบนขวาเป็นบริเวณที่ B ถูกครอบงำ ในที่นี้ B ถูกครอบงำโดย A ส่วนพื้นที่อื่นๆ ที่เหลือไม่มีใครครอบงำ B และจะได้ว่า B และ E ต่างก็ไม่ครอบงำซึ่งกันและกัน

บทนิยาม 3 คำตอบเหมาะที่สุดแบบพาเรโต (Pareto Optimality) [3]

$\vec{x}^* \in \mathsf{F}$ จะเป็นคำตอบเหมาะที่สุดแบบพาเรโต ถ้าไม่สามารถหาคำตอบ $\vec{x} \in \mathsf{F}$ ซึ่ง $f_i(\vec{x}) \geq f_i(\vec{x}^*)$ สำหรับทุก i = 1, ..., k และ $f_j(\vec{x}) > f_j(\vec{x}^*)$ สำหรับค่า j อย่างน้อยหนึ่งค่า

บทนิยาม 3 เป็นนิยามคำตอบเหมาะที่สุดของปัญหาหลายจุดประสงค์ คำตอบเหมาะที่สุดแบบพาเรโต คือ คำตอบที่ไม่สามารถหาคำตอบอื่นมีค่าในจุดประสงค์หนึ่งดีกว่าได้โดยที่จุดประสงค์อื่นไม่แย่กว่า นิยามคำตอบเหมาะที่สุดแบบพาเรโตสัมพันธ์กับนิยามการครอบงำแบบพาเรโตคือ คำตอบเหมาะที่สุดแบบพาเรโตเป็นคำตอบที่ไม่ถูกครอบงำโดยคำตอบอื่น รูปที่ 6 เป็นค่าเดียวกับรูปที่ 5 ส่วนสีเทา แสดงให้เห็นบริเวณที่เป็นไปได้ในแกนจุดประสงค์ของคำตอบทั้งหมด คำตอบในบริเวณที่แรเงาเป็นคำตอบที่ถูกครอบงำ คำตอบที่อยู่ตามแนวเส้นประเป็นคำตอบที่ไม่ถูกครอบงำหรือคำตอบเหมาะที่สุดแบบพาเรโต คำตอบที่ไม่ถูกครอบงำทั้งหมดรวมเรียกว่า “พาเรโตฟรอนต์” (Pareto Front)



รูปที่ 6 พาเรโตฟรอนต์

โดยธรรมชาติจะพบว่าปัญหาหลายจุดประสงค์จะมีความขัดแย้งกันในแต่ละจุดประสงค์ คือ เมื่อดีในด้านหนึ่งแล้วก็จะแย่ในอีกด้านหนึ่ง ในปี ค.ศ. 1896 Vilfredo Pareto ได้ให้นิยามคำตอบของปัญหาหลายจุดประสงค์มีชื่อว่า “Edgeworth-Pareto Optimum” (ดูบทนิยาม 3) ซึ่งแตกต่างจากปัญหาจุดประสงค์เดียวคือ ไม่ใช่คำตอบที่ดีที่สุดเพียงค่าเดียว แต่เป็นคำตอบที่ต้องถ่วงดุลระหว่างแต่ละจุดประสงค์ คำตอบในปัญหาหลายจุดประสงค์จึงมีได้หลายคำตอบ และจะเรียกกลุ่มของคำตอบเหล่านี้ว่า เซตคำตอบเหมาะที่สุดแบบพาเรโต (Pareto Optimal Set)

2.3 ขั้นตอนวิธีเชิงวิวัฒน์แบบหลายจุดประสงค์

ขั้นตอนวิธีเชิงวิวัฒน์แบบหลายจุดประสงค์จะหาคำตอบเป็นกลุ่มประชากร (Population) ทำให้ได้เซตของคำตอบหลายคำตอบพร้อมกันในการทำงานหนึ่งครั้ง แทนที่จะต้องประมวลผลหลาย ๆ ครั้งเพื่อให้ได้ทีละคำตอบ และได้รับผลกระทบจากความไม่ต่อเนื่องของพาเรโตฟรอนต์น้อยกว่าวิธีกำหนดการเชิงเส้น (Linear Programming) ซึ่งจะได้รับผลกระทบอย่างมากจากความไม่ต่อเนื่องและลักษณะเว้า (Concave) ของพาเรโตฟรอนต์ [3]

ขั้นตอนวิธีเชิงวิวัฒน์ได้ถูกเสนอให้นำไปใช้กับปัญหาหลายจุดประสงค์ครั้งแรกในงานวิจัยของ Rosenberg ในปี 1967 [39] แต่การใช้งานจริงเริ่มขึ้นในปี 1984 ในระยะแรกยังไม่ค่อยได้รับความสนใจจากนักวิจัย แต่ 10 ปีให้หลังก็เริ่มได้รับความนิยมและปรากฏเทคนิคต่างๆ ตามมา

เนื่องจากมีขั้นตอนวิธีสำหรับแก้ปัญหาหลายจุดประสงค์เป็นจำนวนมาก จึงเป็นไปได้ยากที่จะกล่าวถึงทุกขั้นตอนวิธี ณ ที่นี้ จึงขอกล่าวถึงเพียงบางส่วนที่ได้รับความนิยม โดยแบ่งได้เป็น 2 กลุ่มหลัก ๆ คือ กลุ่มที่ใช้แนวคิดพาเรโตและกลุ่มที่ไม่ใช้แนวคิดพาเรโต

2.3.1 แบบไม่ใช้พาเรโต (Non-Pareto Based Approach)

ช่วงแรกที่เริ่มใช้ขั้นตอนวิธีเชิงวิวัฒน์แก้ปัญหาหลายจุดประสงค์มักจะอยู่ในแนวทางนี้ ในระหว่างกระบวนการหาคำตอบของขั้นตอนวิธีในกลุ่มนี้จะไม่ได้ใช้แนวคิดแบบพาเรโตร่วมด้วยคือไม่ได้ทำการจัดเรียง (Sort) เพื่อหาอันดับแบบพาเรโตของคำตอบที่หาได้ในขณะนั้นๆ ทำให้ไม่สามารถรับประกันว่าจะได้คำตอบที่เหมาะที่สุดแบบพาเรโต แต่ทำงานเร็วและง่ายต่อการใช้งาน เหมาะสมกับปัญหาที่มีจำนวนจุดประสงค์น้อย ตัวอย่างวิธีในกลุ่มนี้ได้แก่ แบบแบ่งกลุ่มประชากรย่อยและแบบผลรวมคิดน้ำหนัก

ก. แบบแบ่งกลุ่มประชากรย่อย (Subpopulation Approach)

ตัวอย่างเช่น Vector Evaluated Genetic Algorithm: VEGA [40] ขั้นตอนวิธีจำพวกนี้จะแบ่งกลุ่มประชากรเป็นกลุ่มย่อย ๆ ตามจำนวนจุดประสงค์ ถ้ามี k จุดประสงค์ก็จะแบ่งเป็น k กลุ่ม ขนาด n/k การประเมินคุณภาพคำตอบจะกำหนดค่าความเหมาะสมให้กับประชากรแต่ละกลุ่มแยกกันตามจุดประสงค์นั้น ๆ การเลือกคำตอบจะอ้างอิงเฉพาะภายในกลุ่ม ไม่สนใจว่าจุดประสงค์อื่นจะมีคุณภาพอย่างไร

ข้อดีของขั้นตอนวิธีกลุ่มนี้คือง่ายและเร็ว แต่มีข้อเสียคือไม่ค่อยจะเจอคำตอบที่ดีกลาง ๆ ระหว่างแต่ละจุดประสงค์ซึ่งจะไม่ถูกจัดว่าดีเด่นในจุดประสงค์ใดโดยเฉพาะ

ข. แบบผลรวมคิดน้ำหนัก (Aggregation Approach)

วิธีการนี้ใช้การหาผลรวมแบบคิดน้ำหนักของหลายจุดประสงค์

$$\min \sum_{i=1}^{k} w_i f_i(\vec{x}) \text{ , เมื่อ } w_i \geq 0$$

w_i เป็นน้ำหนักที่แสดงถึงความสำคัญของจุดประสงค์นั้น ๆ โดยปกติจะกำหนดให้ผลรวมของน้ำหนักมีค่าเป็น 1 การกำหนดน้ำหนักสำหรับแต่ละจุดประสงค์มี 3 วิธีด้วยกัน [41] ได้แก่ CWA (Conventional Weighted Aggregation), BWA (Bang-bang Weighted Aggregation) และ DWA (Dynamic Weighted Aggregation) [42] ใน CWA จะกำหนดค่าน้ำหนักคงที่ตลอดเวลา ส่วนใน BWA และ DWA จะมีค่าเปลี่ยนแปลงตามฟังก์ชันที่กำหนด

ข้อดีคือ ง่ายทั้งแง่การทำความเข้าใจและการนำไปใช้ การคำนวณค่าผลรวมจุดประสงค์สามารถทำได้รวดเร็วซึ่งจะเหมาะกับปัญหาที่มีข้อจำกัดด้านเวลาในการคำนวณ แต่ก็มีข้อเสียคือ การรวมกันแบบเชิงเส้น (Linear) ใช้งานได้ไม่ดีกับพาเรโตฟรอนต์ที่มีรูปเว้าแต่สามารถใช้งานได้กับปัญหาที่มีโดเมนของปัญหาง่ายและรู้สัดส่วนของค่าในแต่ละจุดประสงค์

2.3.2 แบบใช้พาเรโต (Pareto Based Approach)

การใช้แนวคิดพาเรโตในขั้นตอนวิธีเชิงพันธุกรรมถูกเสนอแนะโดย Goldberg [8] หลังจากนั้นขั้นตอนวิธีเชิงพันธุกรรมสำหรับแก้ปัญหาหลายจุดประสงค์ส่วนใหญ่ก็จะประกอบด้วยการจัดเรียงหรือเปรียบเทียบคำตอบตามแนวคิดพาเรโต ซึ่งมีหลายวิธีดังที่จะกล่าวถึงต่อไปนี้

ก. Multiobjective Genetic Algorithm (MOGA)

Fonseca and Fleming เสนอ MOGA [43] ในปี ค.ศ. 1993 อยู่บนพื้นฐานของขั้นตอนวิธีเชิงพันธุกรรมแบบธรรมดา และได้ให้ค่าความเหมาะสมแผนใหม่โดยคิดจากลำดับที่ (rank) ในพาเรโตฟรอนต์ คำตอบที่อยู่ในลำดับที่เดียวกันจะมีค่าความเหมาะสมเท่ากัน คำตอบที่ไม่ถูกครอบงำจะได้ค่าความเหมาะสมมากสุด และเพื่อให้คำตอบกระจายตัวได้ใช้ niche count and shared fitness ช่วยทำให้คำตอบในพื้นที่เบาบางมีโอกาสถูกเลือกสูงกว่าพื้นที่หนาแน่น

ข. Niched-Pareto Genetic Algorithm 1 and 2 (NPGA)

Horn และคณะ [44] เสนอ Niched-Pareto Genetic Algorithm (NPGA) ในปี ค.ศ. 1993 โดยใช้แนวคิดพาเรโตและการคัดเลือกแบบใหม่เป็นแบบแข่งขัน (Tournament selection) ร่วมกับการใช้เซตเปรียบเทียบ ซึ่งจะทำการสุ่มเลือกสองคำตอบและเซตสำหรับเปรียบเทียบจากประชากรโดยปกติใช้ขนาด 10% ของประชากร แล้วเปรียบเทียบสองคำตอบกับเซตสำหรับเปรียบเทียบ ถ้ามีกรณีที่คำตอบหนึ่งในสองตัวที่เลือกมาไม่ถูกครอบงำโดยเซตสำหรับเปรียบเทียบก็จะเลือกคำตอบนั้น นอกเหนือจากนี้จะเลือกคำตอบจากการปันค่าความเหมาะสม (fitness sharing) [45] โดยคำตอบที่อยู่ในบริเวณความหนาแน่นน้อยกว่าจะถูกเลือก

Erickson และคณะเสนอ NPGA2 ในปี ค.ศ. 2001 ซึ่งใช้การจัดลำดับแบบพาเรโต แต่ก็ยังใช้ การคัดเลือกแบบแข่งขันเหมือนกับ NPGA และแก้ปัญหากรณีสองคำตอบเสมอกันด้วยการปันค่าความเหมาะสมแบบปรับปรุงต่อเนื่อง (Continuously Updated Fitness Sharing) [46] ซึ่งแตกต่างจากการปันค่าความเหมาะสมปกติคือ การคำนวณหาสมาชิกสำหรับปันค่าความเหมาะสมคิดจากคำตอบบางส่วนของประชากรรุ่นถัดไปแทนที่จะใช้คำตอบในประชากรรุ่นปัจจุบัน

ค. Non-dominated Sorting Genetic Algorithm I and II (NSGA)

ในหัวข้อนี้จะกล่าวถึงขั้นตอนวิธี NSGA และ NSGA-II อย่างคร่าวๆ สำหรับขั้นตอนวิธี NSGA-II จะลงในรายละเอียดในหัวข้อ 4.1.1

NSGA ถูกพัฒนาขึ้นในปี ค.ศ. 1993 [47] โดย Srinivas และ Deb ซึ่งจะทำการจัดเรียงประชากรออกเป็นกลุ่มย่อย ๆ เรียกว่า ฟรอนต์ (fronts) ค่าความเหมาะสมของสมาชิกในฟรอนต์เดียวกันมีค่าเท่ากัน การปันค่าใช้ระยะห่างแบบยุคลิด (Euclidean distance) ระหว่างสมาชิกในฟรอนต์ หลังจากแบ่งชั้นของคำตอบและแยกคำตอบที่เด่นออกจากกลุ่มประชากรจะทำการสุ่มเลือกจากคำตอบเด่นที่แยกออกมาสำหรับใช้ในกระบวนการปรับปรุงคุณภาพคำตอบต่อไป

Deb และคณะเสนอขั้นตอนวิธีของ NSGA-II [5] โดยเริ่มจากการจัดเรียงคำตอบตามความเด่น สมาชิกในฟรอนต์เดียวกันจะได้ค่าความเหมาะสมเท่ากัน เรียกว่า ลำดับที่ (Rank) คำตอบที่ไม่ถูกครอบงำจะเป็นลำดับที่ 1 จากนั้นใช้กระบวนการคัดเลือกแบบใหม่เรียกว่า “ตัวดำเนินการคัดเลือกแบบแข่งขันความหนาแน่น” (crowded tournament selection operator) กล่าวคือ การแข่งขันจะเปรียบเทียบลำดับที่ของสองคำตอบ ลำดับที่น้อยกว่าจะถูกเลือก ถ้าลำดับที่เท่ากันจะเปรียบเทียบระยะเบียด (crowding distance) คำตอบที่มีระยะเบียดมากกว่าจะถูกเลือก ระยะเบียดในที่นี้คือ การประมาณความหนาแน่นของคำตอบซึ่งได้จากการหาค่าเฉลี่ยระยะห่างจากสองคำตอบที่ใกล้สุดสองด้านในแกนจุดประสงค์ คำตอบที่ถูกเลือกจะใช้ในการสร้างประชากรใหม่ (offspring) ด้วยตัวดำเนินการปกติของขั้นตอนวิธีเชิงพันธุกรรม (recombination and mutation) เสร็จแล้วค่อยรวมประชากรใหม่เข้ากับประชากรรุ่นปัจจุบันและจัดเรียงประชากรทั้งหมด สมาชิกในลำดับที่ 1 จะกลายเป็นประชากรรุ่นถัดไปทั้งหมด ถ้ายังไม่เต็มจำนวนประชากร จะรวมเอาลำดับที่ต้น ๆ มาร่วมด้วย แต่ถ้าประชากรในลำดับที่ 1 มีมากกว่าจำนวนประชากรที่ต้องการ คำตอบที่มีระยะเบียดน้อยที่สุดจะถูกตัดออก ขั้นตอนวิธี NSGA-II เป็นที่รู้จักกันแพร่หลาย ถูกใช้ในการเปรียบเทียบ MOEA ในหลายงานวิจัย และใช้เป็นพื้นฐานในการออกแบบขั้นตอนวิธีอื่นที่ประสบความสำเร็จเช่น mBOA [48]

ง. Strength Pareto Evolutionary Algorithm I and II (SPEA)

Zitzler และ Thiele นำเสนอ Strength Pareto Evolutionary Algorithm (SPEA) [49] ในปี 1998 มีการใช้ประชากรภายนอกที่เรียกว่า หน่วยเก็บถาวร (Archive) และควบคุมขนาดของหน่วยเก็บถาวรโดยใช้การจับกลุ่ม (clustering) การเก็บคำตอบที่ไม่ถูกครอบงำมีวิธีการง่ายๆ และตรงไปตรงมาคือใช้การคัดลอกคำตอบเข้าไปไว้ในหน่วยเก็บถาวรและคำตอบที่ถูกครอบงำจะถูกลบทิ้ง

การจับกลุ่มจะใช้เมื่อมีจำนวนสมาชิกในหน่วยเก็บถาวรมากเกินเก็บ ก่อนอื่นจะกำหนดให้แต่ละคำตอบอยู่คนละกลุ่ม หลังจากนั้นจะรวมสองกลุ่มที่ใกล้กันมากที่สุดเข้าเป็นกลุ่มเดียวกันจนกระทั่งเหลือจำนวนกลุ่มเท่ากับขนาดของหน่วยเก็บถาวร จากนั้นหาตัวแทนของกลุ่มซึ่งเป็นสมาชิกที่มีระยะห่างเฉลี่ยแบบยุคลิดไปยังคำตอบอื่น ๆ สั้นที่สุด ตัวแทนของกลุ่มจะถูกเก็บไว้ สมาชิกอื่น ๆ ในกลุ่มจะถูกลบออกจากหน่วยเก็บถาวร การจับกลุ่มเป็นแบบไม่มี

พารามิเตอร์ มีผลดีคือผู้ใช้ไม่ต้องกำหนดพารามิเตอร์ใดเพิ่มเติม แต่ก็มีข้อเสียคือ คำตอบปลายสุดของพาเรโตฟรอนต์จะถูกลบไปด้วยและการลบก็ส่งผลให้จำนวนคำตอบในพาเรโตฟรอนต์ที่หาเจอลดลง การให้ค่าความเหมาะสมอิงกับแนวคิดพาเรโต

Zitzler และคณะได้พัฒนา SPEA เป็น SPEA2 ใน [50] ส่วนที่แตกต่างจากเดิมได้แก่ การกำหนดค่าความเหมาะสม การประมาณความหนาแน่น (Density Estimation) และการตัดปลายหน่วยเก็บถาวร (Archive Truncation) โดยเป็นผู้เริ่มใช้ระยะห่าง *k* คำตอบใกล้สุด (*k*-th Nearest Neighbour) เมื่อคำตอบในหน่วยเก็บถาวรมีมากเกินกำหนดจะทำการประมาณความหนาแน่นสำหรับนำไปตัดปลายหน่วยเก็บถาวร คำตอบที่มีระยะห่างจากคำตอบอื่น *k* ตัวที่ใกล้สุดเป็นระยะทางน้อยที่สุดจะถูกตัดออกจนกระทั่งเหลือจำนวนคำตอบเท่ากับขนาดที่กำหนด วิธีการวัดระยะห่างจากคำตอบใกล้สุด *k* ตัว ช่วยป้องกันการหดตัวของพาเรโตฟรอนต์ แต่ก็ขึ้นอยู่กับค่า k ที่กำหนดด้วยว่าเหมาะสมหรือไม่

จ. Pareto Archived ES (PAES)

PAES ถูกพัฒนาขึ้นโดย Knowles และ Corne ในปี 1999 [51] ใช้การคัดเลือกโดยอาศัยแนวคิดพาเรโตร่วมกับกลยุทธ์เชิงวิวัฒน์ (Evolutionary Strategies: ES) แบบ (1+1)-ES ขั้นตอนวิธีหลัก ๆ เป็นกลยุทธ์เชิงวิวัฒน์ ส่วนที่เพิ่มเติมคือ การคัดเลือกคำตอบและหน่วยเก็บถาวร เริ่มต้นจากการสุ่มคำตอบเพียงคำตอบเดียว วัดคุณภาพคำตอบแล้วเก็บเข้าหน่วยเก็บถาวร จากนั้นใช้ตัวดำเนินการกลายพันธุ์เพื่อสร้างคำตอบใหม่หนึ่งคำตอบแล้วคัดเลือกว่าจะใช้คำตอบไหนเป็นต้นแบบในการสร้างคำตอบต่อไป ผลการคัดเลือกคำตอบแบ่งเป็น 3 กรณี ได้แก่ 1) ถ้าคำตอบใหม่ครอบงำคำตอบเดิม จะเลือกคำตอบใหม่และทิ้งคำตอบเดิม 2) คำตอบเดิมครอบงำคำตอบใหม่จะทิ้งคำตอบใหม่ไปแล้วสร้างคำตอบใหม่ต่อไป 3) ถ้าทั้งสองคำตอบไม่ครอบงำซึ่งกันและกัน จะเปรียบเทียบคำตอบใหม่กับคำตอบในหน่วยเก็บถาวร ถ้าคำตอบใหม่สามารถครอบงำคำตอบในหน่วยเก็บถาวร จะเก็บคำตอบใหม่เข้าไปในหน่วยเก็บถาวรและลบคำตอบที่ถูกครอบงำทิ้ง แต่ถ้าคำตอบใหม่ถูกคำตอบในหน่วยเก็บถาวรครอบงำก็จะทิ้งคำตอบใหม่ไป แต่ถ้าไม่มีคำตอบไหนครอบงำกันและกันเลยจะรับคำตอบใหม่เข้าไปในหน่วยเก็บถาวร และถ้าจำนวนคำตอบในหน่วยเก็บถาวรเต็มอยู่แล้ว ก็จะพิจารณาตัดคำตอบโดยพิจารณาความหนาแน่นของคำตอบ โดยทำการแบ่งพื้นที่ในแกนจุดประสงค์แบบเวียนเกิดไปเรื่อย ๆ คำตอบอยู่ในบริเวณหนาแน่นมากจะถูกลบทิ้ง และเลือกคำตอบที่หนาแน่นน้อยสุดมาเป็นคำตอบต้นแบบต่อไป

ฉ. Pareto Envelop-based Selection Algorithm I and II (PESA)

Corn และคณะเสนอ PESA [52] ในปี 2000 ซึ่งเป็นการรวม SPEA และ PAES ไว้ด้วยกัน คือมีหน่วยเก็บถาวรสำหรับเก็บเซตของคำตอบที่ไม่ถูกครอบงำและใช้วิธีการแบ่งกริด

หลายมิติ (Hyper-grid Division) ในการรักษาความหลากหลายของหน่วยเก็บถาวร และยังนำตัววัดนี้ไปใช้เพื่อรักษาความหลากหลายของการคัดเลือกคำตอบซึ่งใช้ในโดเมนลักษณะทางกายภาพ (Phenotype Space) ไม่ได้ใช้ประชากรเพียงตัวเดียวอย่างที่ใช้ใน PAES แต่ใช้ประชากรขนาดเล็กและมีหน่วยเก็บถาวรขนาดใหญ่

Corne และคณะปรับปรุง PESA เป็น PESA-II ในปี ค.ค. 2001 [53] โดยเสนอการคัดเลือกแบบอิงบริเวณ (Region-based Selection) ซึ่งทำการเลือกบริเวณแทนการเลือกเป็นคำตอบ คือทำการเลือกบริเวณก่อนแล้วค่อยสุ่มเลือกคำตอบจากบริเวณนั้น ช่วยทำให้คำตอบที่ได้กระจายทั่วพาเรโตฟรอนต์จริง ๆ แรงจูงใจของวิธีนี้คือช่วยลดภาระของการจัดลำดับแบบพาเรโต

ช. Multiobjective Struggle GA (MOSGA)

Andersson เสนอ MOSGA [54] [55] ซึ่งรวมการจัดลำดับแบบพาเรโตเข้ากับ Struggle Crowding Genetic Algorithm [56] สุ่มเลือกคำตอบขึ้นมา 2 คำตอบ แล้วใช้ตัวดำเนินการปกติ คือ ไขว้เปลี่ยนและกลายพันธ์เพื่อสร้างคำตอบใหม่ จากนั้นหาคำตอบที่เหมือนคำตอบใหม่มากที่สุดจากประชากรทั้งหมดมาหนึ่งคำตอบเพื่อนำมาเปรียบเทียบกับคำตอบใหม่ ถ้าคำตอบใหม่ดีกว่าก็จะแทนที่คำตอบที่คล้ายกันมากที่สุดนั้น แต่วิธีการนี้ก่อให้เกิดลอยเลื่อนทางพันธุกรรม (Genetic Drift) จุดเด่นของงานวิจัยนี้คือ การนำเสนอแบบจำลองที่ใช้วัดความทนทาน (robustness) ของคำตอบที่สร้างขึ้น มีการดึงข้อมูลเกี่ยวกับอายุ (generational information) ออกมาสร้างเป็นพื้นผิวตอบสนอง (response surface) และประมาณการความทนทานของคำตอบในพาเรโตฟรอนต์

ซ. Multiobjective messy GA (MOMGA)

ขั้นตอนวิธี messy GA [27] เป็นพื้นฐานของขั้นตอนวิธีนี้ มีรายละเอียดในหัวข้อ 2.1.4 ค. ขั้นตอนวิธี messy GA ซึ่งเป็นขั้นตอนวิธีสำหรับแก้ปัญหาจุดประสงค์เดียว MOMGA ถูกเสนอโดย Van Veldhuizen ในปี ค.ศ. 1999 [57] โดยได้พยายามปรับเพิ่ม messy GA ให้สามารถแก้ปัญหาหลายจุดประสงค์ ด้วยการเพิ่มแผ่นแบบแข่งขันให้เท่ากับจำนวนจุดประสงค์ การทำงานแบ่งเป็น 3 ขั้นตอน 1) ขั้นการกำหนดค่าเริ่มต้น (Initialization Phase) 2) ขั้นแรกเริ่ม (Primordial Phase) และ 3) ขั้นการวางเคียง (Juxtapositional Phase) ในขั้นการกำหนดค่าเริ่มต้น MOMGA จะผลิตทุกหน่วยการสร้างตามขนาดที่ระบุด้วยกรรมวิธีเชิงกำหนด (Deterministic Process) ที่รู้จักกันในนามการกำหนดค่าเริ่มต้นแจงนับบางส่วน (Partially Enumerative Initialization) ตามด้วยขั้นแรกเริ่มซึ่งจะทำการคัดเลือกแบบแข่งขันจากประชากร และถ้าจำเป็นก็จะลดขนาดของประชากรด้วย จากนั้นขั้นวางเคียงจะสร้างประชากรด้วยตัวดำเนินการรวมกันใหม่แบบตัดและต่อ (Cut and Splice Recombination Operator)

ในการผสมคำตอบ ตัวดำเนินการตัดจะตัดสายอักขระด้วยความน่าจะเป็นที่เพิ่มขึ้นตามความยาวของสายอักขระ โครโมโซมที่ยาวกว่ามีโอกาสถูกตัดมากกว่า ตัวดำเนินการต่อจะต่อหัวของสายอักขระเข้ากับหางของสายอักขระอื่นด้วยความน่าจะเป็นคงที่ซึ่งจะทำให้การทำงานคล้ายกับการไขว้เปลี่ยนหนึ่งจุด (one-point crossover) แต่ในช่วงแรก ๆ ที่คำตอบยังสั้นอยู่โอกาสที่จะตัดยังน้อยทำให้เกิดการต่อกันของสองคำตอบมากกว่า ผลจากการใช้สองตัวดำเนินการนี้ทำให้โครงสร้างคำตอบเปลี่ยนไปและจะกลายเป็นโครงสร้างที่แน่นขึ้น

ฌ. Multiobjective mGA II and IIa (MOMGA-II and MOMGA-IIa)

Zydallis และคณะได้เสนอ MOMGA-II ในปี ค.ศ. 2001 [58] ซึ่งปรับปรุงจาก MOMGA และ fast-messy GA [28] การทำงานแบ่งออกเป็น 3 ขั้นตอนคือ 1.การกำหนดค่าเริ่มต้น 2. การกรองหน่วยการสร้าง (Building Block Filtering) 3. ขั้นการวางเคียง มีข้อแตกต่างจาก MOMGA ที่สำคัญอยู่ใน 2 ส่วนแรก คือปรับจากที่ใช้กรรมวิธีเชิงกำหนดเป็นการกำหนดค่าเริ่มต้นบริบูรณ์เชิงสุ่ม (Probabilistically Complete Initialization: PCI) และขั้นการกรองหน่วยการสร้างจะลดจำนวนหน่วยการสร้างด้วยกระบวนการกรอง (Filtering Process) และเก็บหน่วยการสร้างที่ดีที่สุดที่เคยเจอ สำหรับขั้นการวางเคียงเหมือนกันกับ MOMGA

Day, R. O. และคณะได้พัฒนา MOMGA-II ให้เป็น MOMGA-IIa ในปี ค.ศ. 2005 [4] [59] มีการเปลี่ยนแปลงที่สำคัญได้แก่ 1. หน่วยเก็บถาวรแบบกัมมันต์ (Active Archive) 2. ระบบจัดการแผ่นแบบแข่งขัน (Competitive Template Management System) 3. ระบบการติดตามหน่วยการสร้างของคำตอบ (BB solutions tracing system)

ญ. Multi-objective Bayesian Optimization Algorithm (mBOA)

Khan และคณะเสนอ mBOA ในปี ค.ศ. 2002 [48] มีการทำงานเหมือนกันกับ BOA [12] ยกเว้นส่วนของการคัดเลือกซึ่งจะใช้กระบวนการคัดเลือกและการจัดเรียงเพื่อหาลำดับของการครอบงำแบบเดียวกับ NSGA-II คือ BOA จะทำการรวมประชากรใหม่และประชากรรุ่นพ่อแม่เข้าด้วยกันแล้วจัดเรียงหาลำดับแบบพาเรโต ทำการเลือกโดยอาศัยลำดับแบบพาเรโตและระยะเบียดเพื่อนำกลับมาสร้างเป็นแบบจำลองใหม่ตามวิธีการในกลุ่มขั้นตอนวิธีประมาณการแจกแจง [14] [15] คือ จะสร้างแบบจำลอง (model) ของคำตอบเพื่อเป็นตัวแทนของคำตอบและใช้แบบจำลองนี้สุ่มสร้างคำตอบใหม่ จากนั้นคัดเลือกคำตอบดี ๆ เพื่อนำกลับมาสร้างเป็นแบบจำลองอีก ทำเช่นนี้วนไปเรื่อย ๆ จนกว่าจะได้คำตอบที่พอใจ ในวิทยานิพนธ์ฉบับนี้จะนำขั้นตอนวิธี mBOA ไปเปรียบเทียบผลกับขั้นตอนวิธีที่นำเสนอ

ฎ. Intelligent Multi-objective Evolutionary Algorithms (IMOEA)

Ho และคณะเสนอ IMOEA ในปี 2004 [60] ใช้ตัวรวบรวมยีนอัจฉริยะ (Intelligent Gene Collector: IGC) เป็นตัวหลักในการทำงาน ใช้หลักการแบ่งแยกและเอาชนะ (divide and conquer) ซึ่งจะแบ่งคำตอบพ่อและแม่ออกเป็นส่วนย่อย ๆ จำนวน *N* คู่ และหาว่าแต่ละคู่มีชิ้นไหนที่ดีกว่าจากการประกอบส่วนย่อย ๆ เหล่านี้ให้ได้คำตอบใกล้เคียงคำตอบดีที่สุดที่เป็นได้ภายในจำนวนการประเมินคุณภาพสูงสุด 2*N* ครั้ง โดยมีปริมาณของแต่ละส่วนเกิดขึ้นเท่า ๆ กัน จัดเรียงคำตอบที่ประกอบได้ตามค่าความเหมาะสมและนับจำนวนการปรากฏของแต่ละส่วนในคำตอบที่ดีว่าส่วนไหนมีมากกว่าจะถือว่าเป็นส่วนที่ดีกว่า การระบุและประกอบหน่วยการสร้างตามแนวคิดของงานวิจัยนี้คาดหวังว่าแต่ละบิตไม่ขึ้นต่อกันหรือขึ้นต่อกันน้อย

2.4 สรุป

รายละเอียดในบทนี้ได้กล่าวถึงทฤษฎีแบ่งเป็น 2 ส่วนหลักๆ ส่วนแรกคือขั้นตอนวิธีเชิงพันธุกรรมและหน่วยการสร้าง ส่วนที่สองคือปัญหาหลายจุดประสงค์และขั้นตอนวิธีสำหรับแก้ปัญหาหลายจุดประสงค์ ในส่วนของหน่วยการสร้างได้กล่าวถึงทฤษฎีและนิยามของหน่วยการสร้างตามด้วยขั้นตอนวิธีต่างๆ ที่ใช้ระบุหน่วยการสร้าง

ขั้นตอนวิธีที่ใช้แก้ปัญหาหลายจุดประสงค์ที่ได้กล่าวถึงในบทนี้ มีบางส่วนที่ได้นำขั้นตอนวิธีระบุหน่วยการสร้างในปัญหาจุดประสงค์เดียวมาปรับใช้ให้สามารถแก้ปัญหาหลายจุดประสงค์ เช่น mBOA ปรับปรุงมาจาก BOA ขั้นตอนวิธี MOMGA-II และ MOMGA-IIa ปรับปรุงมาจาก mGA และ fmGA ขั้นตอนวิธีที่ยกตัวอย่างมานี้เป็นขั้นตอนวิธีสำหรับแก้ปัญหาหลายจุดประสงค์ที่ใช้หน่วยการสร้าง ซึ่งจะนำมาเปรียบเทียบผลกับขั้นตอนวิธีที่นำเสนอ

ในงานวิจัยนี้ได้เลือกขั้นตอนวิธีพื้นฐานที่แก้ปัญหาหลายจุดประสงค์โดยไม่ใช้หน่วยการสร้าง (NSGA-II) เพื่อนำมาปรับปรุงให้มีการระบุหน่วยการสร้างและทดสอบประสิทธิภาพการทำงานต่อไป

บทที่ 3

การแก้ปัญหาจุดประสงค์เดียว

ขั้นตอนวิธีเชิงพันธุกรรมได้กลายเป็นขั้นตอนวิธีทางเลือกหนึ่งที่ใช้แก้ปัญหาได้หลากหลาย โดยเฉพาะปัญหาที่ขั้นตอนวิธีทั่วๆ ไปไม่สามารถหาคำตอบได้หรือหาคำตอบได้ไม่ดีพอ เช่น ปัญหาที่มีพื้นที่การค้นหาคำตอบเป็นหลุมบ่อลึกและเขาสูงสลับเป็นจำนวนมาก หรือบริเวณพื้นที่ค้นหาคำตอบขรุขระมากและมีบริเวณคำตอบเหมาะสุดเฉพาะที่ที่หลากหลาย

มีการนำเสนอหลากหลายวิธีในระยะเวลาไม่นานมานี้ เพื่อปรับปรุงประสิทธิภาพของขั้นตอนวิธีเชิงวิวัฒน์ทั่วไป โดยเฉพาะการศึกษาการค้นหาและใช้ประโยชน์จากความรู้เกี่ยวกับการเชื่อมโยงเพื่อเพิ่มประสิทธิภาพของขั้นตอนวิธีเชิงวิวัฒน์

แนวคิดเกี่ยวกับการเชื่อมโยงเริ่มต้นจากการศึกษาทางด้านพันธุศาสตร์ (Genetics) และใช้อธิบายความสัมพันธ์ทางกายภาพระหว่างยีนสองตำแหน่ง พบว่า ยีนที่อยู่ใกล้ๆ กันในโครโมโซมเดียวกัน จะถูกถ่ายทอดไปด้วยกันเสมือนเป็นหน่วยเดียวกันด้วยความน่าจะเป็นที่สูง เรียกรูปแบบนี้ว่า กลุ่มเชื่อมโยง (linkage group) แต่เนื่องจากการที่ยีนสามารถโยกย้ายตำแหน่งภายในโครโมโซมได้ ทำให้บางกลุ่มมีความเชื่อว่าเป็นการดึงดูดกันของยีนหนึ่งกับยีนอื่น แนวคิดดังกล่าวนี้ถูกนำไปใช้ในการคำนวณเชิงวิวัฒน์ โดยที่ความเชื่อมโยงถูกใช้ในการอธิบายความสัมพันธ์ระหว่างตัวแปรในปัญหาที่พึ่งพาอาศัยกัน (interdependency) การพึ่งพาอาศัยกันในที่นี้หมายถึงสถานะเหมาะที่สุดของตัวแปรหนึ่ง S_i (มีค่าเป็น 0 หรือ 1) ขึ้นอยู่กับสถานะของตัวแปรอื่น S_j

การวัดปริมาณความเชื่อมโยงโดยปกติจะทำได้ยาก แต่จะสามารถทำได้อย่างตรงไปตรงมาถ้าปัญหานั้นสามารถแยกเป็นปัญหาย่อยได้ ถ้าเซต **S** = $\{S_1, S_2, \ldots, S_n\}$ เป็นเซตตัวแปร n ตัวแปร ฟังก์ชันค่าความเหมาะสมที่เป็นแบบแบ่งแยกย่อยได้ (Additively Decomposable Function: ADF) จะสามารถเขียนให้อยู่ในรูปผลรวมของฟังก์ชันย่อยๆ ที่มีขนาดเล็กกว่าได้ดังนี้

$$f(S) = \sum_{i=1}^{m} f_i(S_{v_i})$$

เมื่อ m เป็นจำนวนฟังก์ชันย่อย f_i เป็นฟังก์ชันย่อยที่ i และ S_{v_i} เป็นเซตย่อยของตัวแปรซึ่งประกอบไปด้วยตัวแปรที่มีดัชนีปรากฏในเซต V_i ตัวอย่างเช่น กำหนดให้ V_i = {1,2,3,4} จะได้ S_{v_i} = $\{S_1, S_2, S_3, S_4\}$ ตัวแปรที่อยู่ภายในเซต V_i เดียวกัน จะเป็นตัวแปรที่พึ่งพาซึ่งกันและกันที่เรียกว่า กลุ่มเชื่อมโยง (linkage group) หรือโดยทั่วไปเรียกว่า หน่วยการสร้าง

(Building Block) แต่ปัญหาในโลกจริงส่วนใหญ่จะเป็นความเชื่อมโยงที่เหลื่อมกัน (overlapping linkage sets)

ในระยะเวลาไม่นานมานี้มีหลายงานวิจัยพยายามใช้หลากหลายวิธีในการแก้ปัญหาแบบแบ่งแยกย่อยได้ ใน [61] ได้ปรับปรุงขั้นตอนวิธีเชิงพันธุกรรมให้สามารถปรับขนาดของประชากรได้ระหว่างการทำงาน และยังเปรียบเทียบขั้นตอนวิธีแบบปรับขนาดประชากรได้กับขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย นอกจากนี้มีการใช้ปัญหากับดัก (Trap) ในการประเมินประสิทธิภาพของขั้นตอนวิธีในงานวิจัยอื่นๆ ตัวอย่างเช่น งานวิจัย [62] และ [63] บางงานวิจัยสนับสนุนให้หาหน่วยการสร้างเพื่อปรับปรุงความสามารถของขั้นตอนวิธีเชิงพันธุกรรม ตัวอย่างเช่น การเรียนรู้ความเชื่อมโยง (Linkage Learning) ใน [23] และ [36] และอีกหลายเทคนิคที่หาหน่วยการสร้างโดยตรงเช่น messyGA [27]

ขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายจะประสบปัญหายุ่งยากในการใช้ ถ้ากลุ่มเชื่อมโยงมีบิตที่กระจายตำแหน่งและไม่ได้อยู่ต่อเนื่องกัน และการใช้การไขว้เปลี่ยนแบบจุดเดียว (one point crossover) ก็จะทำลายหน่วยการสร้างของคำตอบดีๆ ในปัญหานี้ได้ ดังนั้น ในบทนี้ได้นำเสนอกลวิธีการระบุหน่วยการสร้างและทำการไขว้เปลี่ยนในตำแหน่งที่เหมาะสมซึ่งจะรักษาหน่วยการสร้างของคำตอบดีๆ และยังผสมให้ได้คำตอบดีๆ ขึ้นไปอีก

นิยามของหน่วยการสร้างมี 2 คุณลักษณะสำคัญ ส่วนแรกคือ หน่วยการสร้างปรากฏในคำตอบดีๆ และอีกส่วนคือ การประกอบหน่วยการสร้างอย่างเหมาะสมซึ่งจะทำให้คุณภาพของคำตอบที่ได้ดีขึ้น

อย่างไรก็ตามการระบุหน่วยการสร้างก็ไม่สามารถทำได้อย่างง่ายๆ หรือตรงไปตรงมาขึ้นอยู่กับแต่ละปัญหา ในงานวิจัยนี้จะเสนอแนวทางสำหรับระบุหน่วยการสร้างในรูปของ การแบ่งกลุ่มบิตที่มีความสัมพันธ์กันสูง ความรู้ที่ได้จะถูกใช้เพื่อป้องกันการทำลายหน่วยการสร้างจากการไขว้เปลี่ยนในคำตอบที่ดีๆ โดยบิตที่อยู่ภายในหน่วยการสร้างเดียวกันไม่ควรจะถูกแยกออกจากกัน ในการไขว้เปลี่ยนควรจะไปด้วยกัน

ในบทนี้ได้ทำการทดลองเพื่อแสดงให้เป็นประสิทธิภาพของขั้นตอนวิธีที่ใช้และไม่ใช้การระบุหน่วยการสร้าง โดยใช้ปัญหาที่สามารถแบ่งแยกย่อยได้ ซึ่งจะปรากฏหน่วยการสร้างเด่นชัด วิธีการที่เสนอจะทำการระบุหน่วยการสร้างและนำหน่วยการสร้างหรือผลการแบ่งส่วนที่ได้มาใช้สำหรับหาตำแหน่งไขว้เปลี่ยนที่เหมาะสม โดยได้นำเสนอการเปรียบเทียบเทคนิคนี้กับขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย และการระบุหน่วยการสร้างทำได้โดยการสร้างเมทริกซ์ไคกำลังสองและใช้ขั้นตอนวิธีแบ่งส่วนที่นำเสนอใน [64] ค่าในเมทริกซ์ไคกำลังสองแสดงถึงปริมาณความสัมพันธ์ระหว่างสองบิตจากคำตอบดีๆ ที่ถูกเลือก ส่วนขั้นตอนวิธีแบ่งส่วนทำหน้าที่แบ่งกลุ่มบิตที่มีความสัมพันธ์กันสูงให้อยู่ด้วยกัน

การไขว้เปลี่ยนของขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายซึ่งใช้การไขว้เปลี่ยนแบบจุดเดียวมักจะทำลายคำตอบดีๆ ในขณะที่การไขว้เปลี่ยนที่นำเสนอจะช่วยรักษาหน่วยการสร้างของคำตอบดีๆ และยังผสมให้ได้คำตอบดียิ่งขึ้นไปอีก

3.1 การระบุหน่วยการสร้างโดยไคกำลังสอง

การระบุหน่วยการสร้างที่จะกล่าวถึงในที่นี้ถูกนำไปใช้ทั้งในปัญหาจุดประสงค์เดียวและหลายจุดประสงค์ ส่วนที่แตกต่างกันคือที่มาของข้อมูลที่จะนำมาหาหน่วยการสร้างซึ่งจะได้อธิบายในหัวข้อนั้นๆ โดยเฉพาะอีกครั้งหนึ่ง ขั้นตอนวิธีสำหรับหาหน่วยการสร้างในงานวิจัยนี้จะเรียกชื่อว่า การระบุหน่วยการสร้างด้วยเมทริกซ์ไคกำลังสอง (Building-block Identification by Chi-square Matrix: BICM) ประกอบด้วยการทำงาน 2 ส่วนหลักๆ คือ 1. ขั้นตอนการสร้างเมทริกซ์ไคกำลังสอง (Chi-square Matrix Construction) และ 2. ขั้นตอนการแบ่งกลุ่มความสัมพันธ์ (Partitioning)

3.1.1 ขั้นตอนการสร้างเมทริกซ์ไคกำลังสอง

ไคกำลังสอง (Chi-square: χ^2) เป็นตัววัดทางสถิติ ถูกใช้ในการแสดงปริมาณความสัมพันธ์ของตัวแปร แบบที่ใช้เรียกว่า ไคกำลังสองของเปียสัน (Pearson's chi-square) มีสมการคำนวณดังนี้

$$\chi^2 = \frac{(observed - expected)^2}{expected}$$

เมื่อ *observed* เป็นจำนวนตัวอย่างที่สังเกตได้
expected เป็นจำนวนตัวอย่างที่คาดว่าจะเกิดขึ้น

ตัวอย่างการวัดความสัมพันธ์ของสองตัวแปร (บิตที่ i กับ บิตที่ j) ในขั้นตอนวิธีประมาณการแจกแจงแบบ BMDA (Bivariate Marginal Distribution Algorithm) [26] จะดูจากค่าไคกำลังสองดังสมการต่อไปนี้

$$\chi^2_{i,j} = \sum_{b_i, b_j} \frac{(n \cdot p(b_i, b_j) - n \cdot p(b_i) \cdot p(b_j))^2}{n \cdot p(b_i) \cdot p(b_j)}$$

เมื่อ n เป็นจำนวนคำตอบที่เลือกมา
p แสดงถึงค่าความน่าจะเป็น
b_i และ b_j เป็นตัวแปรที่ต้องการวัดความสัมพันธ์ (บิตที่ i และบิตที่ j)
$1 \le i, j \le l$, $i \ne j$; l เป็นความยาวของคำตอบ

และมีเงื่อนไขว่า ถ้า χ^2 มีค่าน้อยกว่า 3.84 จะได้ว่า บิตที่ i ไม่ขึ้นกับบิต j ด้วยความเชื่อมั่น 95% แต่ในงานวิจัยของชัชวิทย์ [18] ถือว่าแต่ละบิตที่มีความสัมพันธ์กันจะมีความน่าจะเป็นของ

การปรากฏแต่ละรูปแบบ (00, 01, 10, 11) ไม่เป็นแบบสุ่มคือมีค่าเป็นค่าใดค่าหนึ่งมากหรือน้อยกว่าปกติ ไม่ได้เกิดเท่า ๆ กัน ซึ่งจะทำให้ค่าไคกำลังสองมีค่าสูง แต่ถ้าแต่ละรูปแบบเกิดขึ้นเท่าๆ กัน จะทำให้ไคกำลังสองมีค่าน้อย

กำหนดให้ $\boldsymbol{M}$ เป็นเมทริกซ์ไคกำลังสอง มีค่าในแนวเส้นทแยงมุมเป็น 0 นอกนั้นเป็นค่าไคกำลังสองระหว่างบิตในลำดับที่เดียวกับแถวที่ i และสดมภ์ที่ j โดยใช้สมการคำนวณดังนี้

$$Chi-square(i,j)=\sum_{xy}\frac{(Count_S^{xy}(i,j)-n/4)^2}{n/4}\ ;\ xy\in\{00,01,10,11)$$

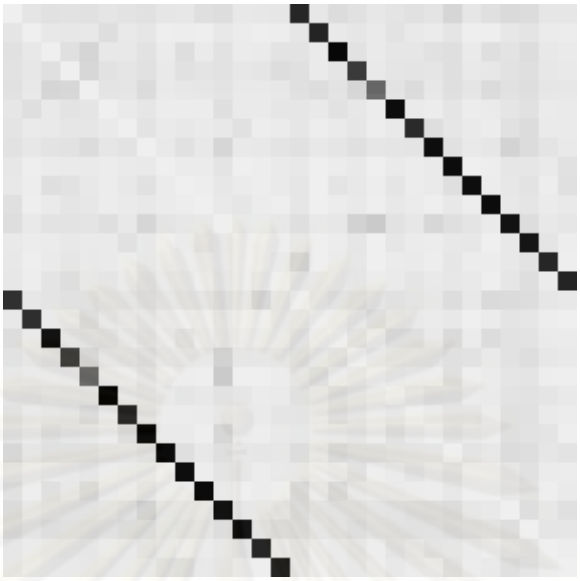
เมื่อ $Count_S^{xy}(i,j)$ เป็นจำนวนคำตอบที่มีบิตที่ i มีค่าตรงกับ x และบิตที่ j มีค่าตรงกับ y ค่าความคาดหวังของ ‘00’ ‘01’ ‘10’ และ ‘11’ มีค่าเป็น $n/4$ เท่ากัน เมื่อ n เป็นจำนวนคำตอบที่ถูกเลือกมาสร้างเมทริกซ์ไคกำลังสอง ความซับซ้อนของการคำนวณเมทริกซ์ไคกำลังสองเป็น $O(l^2n)$ เมื่อ l เป็นความยาวของคำตอบ

ขั้นตอนวิธีสำหรับสร้างเมทริกซ์ไคกำลังสอง (CMC algorithm) แสดงไว้ในรูปที่ 7 โดยมีข้อมูลนำเข้า S เป็นประชากรที่มีคุณภาพดีที่ต้องการนำมาหาหน่วยการสร้าง จะได้ผลลัพธ์เป็นเมทริกซ์ M

Algorithm CMC(S)
1. **for** i = 0 **to** l – 1 **do**
   $m_{ii} \leftarrow 0$;
   for $j = i + 1$ to $l - 1$ **do**
   $m_{ij} \leftarrow$ **Chi-square**(i,j);
2. **for** i = 0 **to** l – 1 **do**
   for $j = i + 1$ to $l - 1$ **do**
   $m_{ji} \leftarrow m_{ij}$;
3. **return** $M \leftarrow (m_{ij})$;

รูปที่ 7 ขั้นตอนวิธีการสร้างเมทริกซ์ไคกำลังสอง

รูปที่ 8 แสดงตัวอย่างของเมทริกซ์ไคกำลังสองของปัญหา MOP1



รูปที่ 8 ตัวอย่างปริมาณไคกำลังสองของปัญหา MOP1

บริเวณสีเข้มแสดงถึงคู่บิตมีความสัมพันธ์มาก ส่วนบริเวณสีจางมีความสัมพันธ์น้อย แต่ละช่องแทนตำแหน่งบิตหนึ่งตำแหน่งเรียงจากซ้ายไปขวาและจากบนลงล่างแบบคู่อันดับ

3.1.2 ขั้นตอนการแบ่งกลุ่มความสัมพันธ์

เมทริกซ์ไคกำลังสองที่ได้จะถูกแบ่งกลุ่มความสัมพันธ์ของบิตด้วยขั้นตอนวิธีแบ่งส่วน (PAR Algorithm) ดังแสดงในรูปที่ 9 การแบ่งส่วนจะทำการรวมบิตที่มีความสัมพันธ์กันสูง (มีค่าไคกำลังสองสูง) ให้อยู่ในกลุ่มเดียวกันและเรียกแต่ละกลุ่มว่า **"หน่วยการสร้าง"** ผลแบ่งส่วน (Partition) ที่จะยอมรับได้ต้องมีคุณสมบัติครบทั้ง 5 ข้อดังต่อไปนี้

1. P จะต้องเป็นการแบ่งส่วน กล่าวคือ
   สมาชิกของ P แต่ละตัวต้องไม่มีสมาชิกร่วม (Disjoint Set)
   ยูเนียนทุกสมาชิกของ P แล้วจะได้เซตของ 0 ถึง l-1 ($\{0, \ldots, l\text{-}1\}$)
2. P จะต้องถูกแบ่งส่วน สมาชิกของ P จะต้องไม่เป็นเซตใหญ่สุดเพียงเซตเดียว
   นั่นคือ $P \neq \{\{0, \ldots, l-1\}\}$
3. ให้ B เป็นสมาชิกของ P ที่มีขนาดมากกว่า 1
   สำหรับทุก $i \in B$ ค่ามากที่สุดของแถวที่ i จะปรากฏอยู่ในสดมภ์ $j \in B \setminus \{i\}$
4. สำหรับทุก $B \in P$ และ $|B| > 1$
   $H_{\max} - H_{\min} < \alpha(H_{\max} - L_{\min})$ where $0 \leq \alpha \leq 1$,
   $H_{\max} = max(\{m_{ij} \mid (i, j) \in B \times B, i \neq j\})$,
   $H_{\min} = min(\{m_{ij} \mid (i, j) \in B \times B, i \neq j\})$, and
   $L_{\min} = min(\{m_{ij} \mid i \in B, j \in \{0, \ldots. l-1\} \setminus B\})$.
5. ไม่มีส่วนแบ่ง P' อื่นใดที่สอดคล้องกับเงื่อนไข 1-4 และมีสมาชิกของ P เป็นเซตย่อยของสมาชิก P'

ในข้อ 4 ค่าของ α เป็นตัวกำหนดความแตกต่างของค่าไคกำลังสองที่จะสามารถรวมอยู่ในส่วนแบ่งเดียวกันได้ คิดเป็นเปอร์เซ็นต์เทียบกับค่าต่ำสุดและค่าสูงสุดภายในกลุ่มบิตที่สนใจ ค่า α ยิ่งน้อยค่าของไคกำลังสองที่จะรวมกันได้จะต้องใกล้เคียงกันกันมาก

ในข้อ 5 อาจกล่าวได้ว่าถ้ามีการแบ่งส่วน 2 แบบที่ตรงตามเงื่อนไข 1-4 จะเลือกการแบ่งส่วนที่มีสมาชิกขนาดใหญ่กว่า

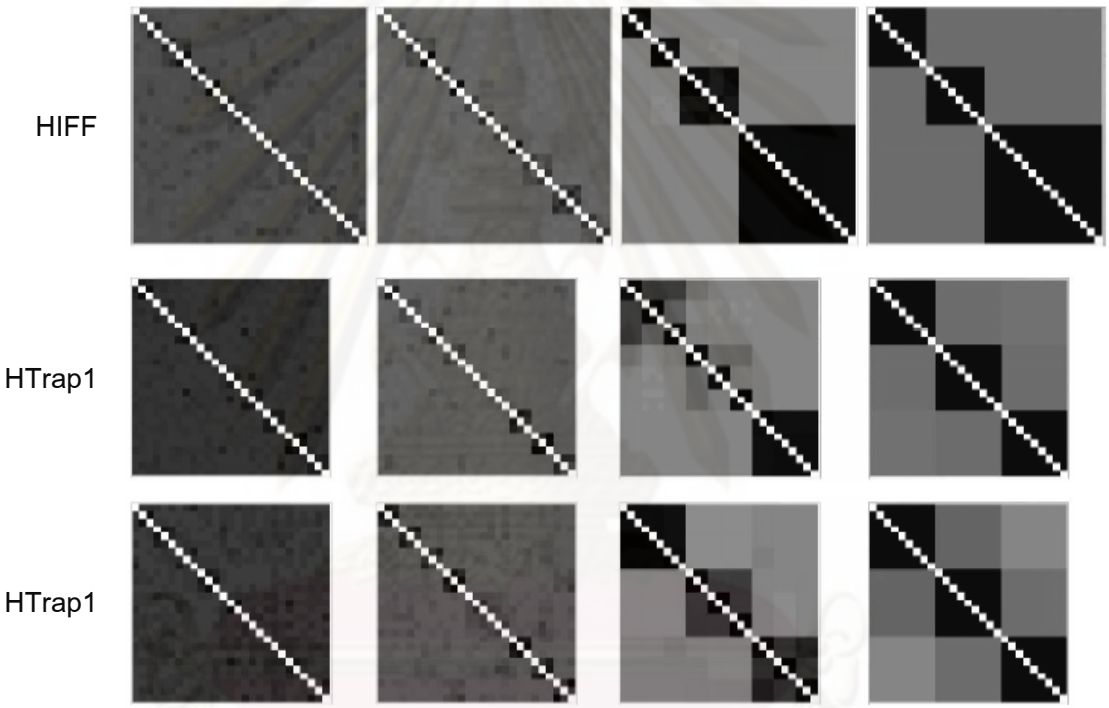
ขั้นตอนวิธีการแบ่งส่วนจะพยายามรวมกลุ่มของบิตที่มีค่าไคกำลังสองสูงเข้าด้วยกันก่อน เริ่มจากแถวแรกไล่ลงไปถึงแถวสุดท้าย ในแต่ละแถวจะจัดเรียงค่าไคกำลังสองจากมากไปหาน้อยและทดสอบว่าส่วนแบ่งที่ได้จากการเพิ่มบิตที่มีค่าไคกำลังสองสูงเข้าไปสอดคล้องกับเงื่อนไขในข้อ 3 และข้อ 4 หรือไม่ ถ้าสอดคล้องกับเงื่อนไขก็จะเพิ่มบิตนั้นเข้ามาอยู่ในส่วนแบ่งเดียวกัน ทำเช่นนี้กับบิตที่มีค่าไคกำลังสองสูงตามลำดับที่จัดเรียงไว้ ถ้าการเพิ่มบิตใหม่เข้ามาไม่สอดคล้องกับเงื่อนไขก็จะให้อยู่ในส่วนแบ่งใหม่และพิจารณาแถวถัดไปที่ยังไม่อยู่ในส่วนแบ่งต่อไปจนครบทุกแถว

$\boldsymbol{M}$= (m_{ij}) denotes $l \times l$ Chi-Square matrix, where $0 \le i, j \le l-1$.
T_i and $R_{i,j}$ denote arrays of numbers indexed by $0 \le i, j \le l-1$.
A and B are partition subsets. P denotes a partition.

Algorithm PAR(M, α)
$P \leftarrow \varnothing$;
for $i = 0$ **to** $l - 1$ **do**
  if $i \notin B$ **for all** $B \in P$ **then**
    $T \leftarrow$ {matrix elements in row i sorted in descending order};
    for $j = 0$ **to** $l - 1$ **do**
      $R_{i,j} = x$ where $m_{ix} = T_j$
    endfor
    $A \leftarrow \{i\}$;
    $B \leftarrow \{i\}$;
    for $j = 0$ **to** $l - 3$ **do**
      $A \leftarrow A \cup \{R_{i,j}\}$;
      if A satisfies the third and the fourth conditions **then**
        $B \leftarrow A$;
      endif
    endfor
    $P \leftarrow P \cup \{B\}$;
  endif
endfor
return P;

รูปที่ 9 ขั้นตอนวิธีแบ่งส่วน (PAR Algorithm)

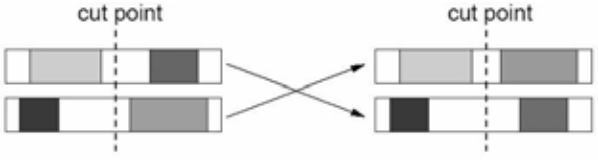
รูปที่ 10 แสดงค่าเมทริกซ์ไคกำลังสองที่ได้ในระหว่างการทำงานของขั้นตอนวิธีเชิงพันธุกรรมจาก 3 ปัญหา แต่ละแถวเป็นผลจากแต่ละปัญหา บริเวณที่มีสีเข้มคือมีค่ามากบริเวณสีจางมีค่าน้อย เมื่อพิจารณารูปในคอลัมน์แรกซึ่งเป็นค่าที่ได้จากประชากรรุ่นแรก ในตอนเริ่มต้นจะมีลักษณะเหมือน ๆ กันเนื่องจากเป็นค่าที่ได้จากการสุ่ม และจะค่อย ๆ เห็นความสัมพันธ์ชัดเจนขึ้นเรื่อย ๆ ใน ประชากรรุ่นถัด ๆ ไป ขั้นตอนวิธีแบ่งส่วนทำหน้าที่รวมกลุ่มบิตที่มีความสัมพันธ์กันมากเข้าด้วยกันเพื่อใช้ในระหว่างการไขว้เปลี่ยน โดยต้องการให้บิตที่อยู่ในกลุ่มเดียวกันไปด้วยกัน



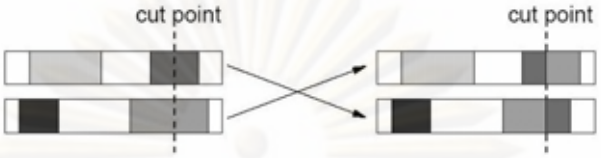
รูปที่ 10 เมทริกซ์ไคกำลังสองระหว่างการทำงานของขั้นตอนวิธีเชิงพันธุกรรม

3.2 การไขว้เปลี่ยน

ตัวดำเนินการไขว้เปลี่ยนสามารถใช้ประโยชน์จากความรู้เกี่ยวกับหน่วยการสร้างด้วยการเลือกจุดตัดที่เหมาะสม โดยจุดตัดไม่ควรแบ่งแยกบิตที่อยู่ในหน่วยการสร้างเดียวกัน ดังแสดงในรูปที่ 11 บริเวณสีเทาและสีดำเป็นหน่วยการสร้างที่หาได้ ภายในรูปแสดงการไขว้เปลี่ยนที่ยังคงรักษาหน่วยการสร้างของแต่ละคำตอบไว้ได้ ส่วนรูปที่ 12 แสดงจุดตัดที่ไม่เหมาะสมของการไขว้เปลี่ยนซึ่งมีผลให้มีการทำลายหน่วยการสร้าง



รูปที่ 11 การไขว้เปลี่ยนที่รักษาหน่วยการสร้าง



รูปที่ 12 การไขว้เปลี่ยนที่ทำลายหน่วยการสร้าง

เพื่อให้การไขว้เปลี่ยนสามารถทำงานได้ตามที่ต้องการ จึงต้องสร้างหน้ากากสำหรับไขว้เปลี่ยน (crossover mask) ของแต่ละหน่วยการสร้าง ก่อนทำการแลกเปลี่ยนบิตเพื่อสร้างคำตอบลูก (offspring) ทุกบิตในพาร์ทิชันเดียวกันจะถูกย้ายไปด้วยกัน ดูตัวอย่างต่อไปนี้

Partition <1 2 3 4 1 3 2 4 5 6 7 8 5 4 6>
Mask Bits <0 1 1 1 0 1 1 1 0 1 1 1 0 1 1>

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Parent 1

y	y	y	y	y	y	y	y	y	y	y	y	y	y	y

Parent 2

0	1	1	1	0	1	1	1	0	1	1	1	0	1	1

Mask Bits

หลังจากการไขว้เปลี่ยนจะได้คำตอบลูกสองคำตอบ

y	x	x	x	y	x	x	x	y	x	x	x	y	x	x

Child 1

x	y	y	y	x	y	y	y	x	y	y	y	x	y	y

Child 2

เมื่อแบ่งกลุ่มแต่ละบิตได้เป็นดังนี้ P= <1 2 3 4 1 3 2 4 5 6 7 8 5 4 6> มีความหมายเป็นการแบ่งส่วนบิตคือ { {1,5}, {2,7}, {3,6}, {4,8,14}, {9,13}, {10,15}, {11}, {12} } ในตัวอย่างบิตที่ 1 และบิตที่ 5 อยู่ในกลุ่มเดียวกัน เป็นกลุ่มหมายเลข 1 สมมติว่าเราเลือกที่จะแลกเปลี่ยนกลุ่มบิตหมายเลข 1 และกลุ่มบิตหมายเลข 5 นั่นคือเราต้องการให้บิตที่ 1 และบิตที่ 5 (กลุ่มบิตหมายเลข 1) และบิตที่ 9 กับบิตที่ 13 (กลุ่มบิตหมายเลข 5) ทั้งหมดนี้ไปด้วยกัน โดยเราจะสร้างหน้ากากสำหรับไขว้เปลี่ยนให้บิตที่ต้องการคงไว้เป็น 0 และบิตที่ต้องการไขว้เปลี่ยนให้มีค่าเป็น 1 ดังนั้นเราจะทำการสุ่มว่ากลุ่มบิตใดๆ ควรจะมีหน้ากากมีค่าเป็น 0 หรือเป็น 1 ทั้งกลุ่ม ซึ่งหมายถึงเลือกที่จะคงไว้หรือจะทำการแลกเปลี่ยนทั้งกลุ่มนั่นเอง ขั้นตอนวิธีในการสร้าง

หน้ากากสำหรับไขว้เปลี่ยนแสดงไว้ในรูปที่ 13 ซึ่งจะทำการสุ่มสำหรับแต่ละกลุ่มว่าจะเก็บบิตในกลุ่มนั้นไว้หรือจะแลกเปลี่ยนกับคำตอบอื่น ถ้าเป็น 1 จะคงไว้ แต่ถ้าเป็น 0 จะแลกเปลี่ยน

```
Flag[0..l-1] ← 0
for i=0 to l-1 do
      if notsetFlag(i) then
            maskbits[i] ← random_int(0,1)
            Flag[i] ← 1
            for j=i+1 to l-1 do
                  if (partition[i] equal partition[j]) then
                        maskbits[j] ← maskbits[i]
                        Flag[j] ← 1
                  end if
            end for
      end if
end for
```

รูปที่ 13 ขั้นตอนวิธีในการสร้างหน้ากากสำหรับไขว้เปลี่ยน

partition เป็นตัวแปรสำหรับเก็บพาร์ทิชัน มีโครงสร้างเป็นแถวลำดับ (array) หนึ่งมิติ เก็บค่าหมายเลขกลุ่มของแต่ละบิต บิตที่มีหมายเลขเดียวกันจะอยู่ในหน่วยการสร้างเดียวกัน หมายเลขใดที่ปรากฏเพียงครั้งเดียว แสดงว่าบิตนั้นเป็นอิสระไม่ขึ้นกับบิตอื่นใด

รหัสเทียมขั้นตอนวิธีเชิงพันธุกรรมที่ใช้หน่วยการสร้างได้แสดงไว้ในรูปที่ 14 ในขั้นตอนการไขว้เปลี่ยนจะทำการสร้างหน้ากากสำหรับไขว้เปลี่ยนทุกครั้ง เพื่อให้เกิดความหลากหลายในการผสมหน่วยการสร้าง

```
gen = 0
Initialize n individuals
Evaluate the individuals
While (terminate condition is not met) Do
  Calculate Chi-square Matrix
  Perform PAR algorithm
  Select n/2 mates
  For each mate
        Generate masked-bit
        BB-wise crossover
  End For
  Perform mutation
  Evaluate new population
  gen = gen + 1
End While
```

รูปที่ 14 ขั้นตอนวิธีเชิงพันธุกรรมที่ใช้หน่วยการสร้าง

3.3 การทดลองใช้การระบุหน่วยการสร้างเพื่อแก้ปัญหาจุดประสงค์เดียว

การทดลองใช้การระบุหน่วยการสร้างด้วยเมทริกซ์ไคกำลังสองในกระบวนการไขว้เปลี่ยนของขั้นตอนวิธีเชิงพันธุกรรมทำให้สามารถแก้ปัญหาปัญหากับดัก ซึ่งเป็นปัญหาที่จะหลอกการทำงานของขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายทำให้ไม่สามารถหาคำตอบเหมาะที่สุด (Optimal Solution) ได้ จากการทดลองปรากฏว่าการระบุหน่วยการสร้างด้วยเมทริกซ์ไคกำลังสองช่วยแก้ปัญหากับดักได้ดีกว่าขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย โดยใช้จำนวนครั้งของการประเมินค่าคำตอบ (Number of Function Evaluation) น้อยกว่า

ในการทดลองเบื้องต้นได้ทดสอบกับปัญหากับดักและปัญหากับดักแบบสลับตำแหน่ง (Shuffle Trap) ซึ่งถูกออกแบบให้กับดักแต่ละชุดไม่ได้อยู่ต่อเนื่องกันถ้าให้ปัญหากับดักมีเค้าร่างเป็นดังนี้

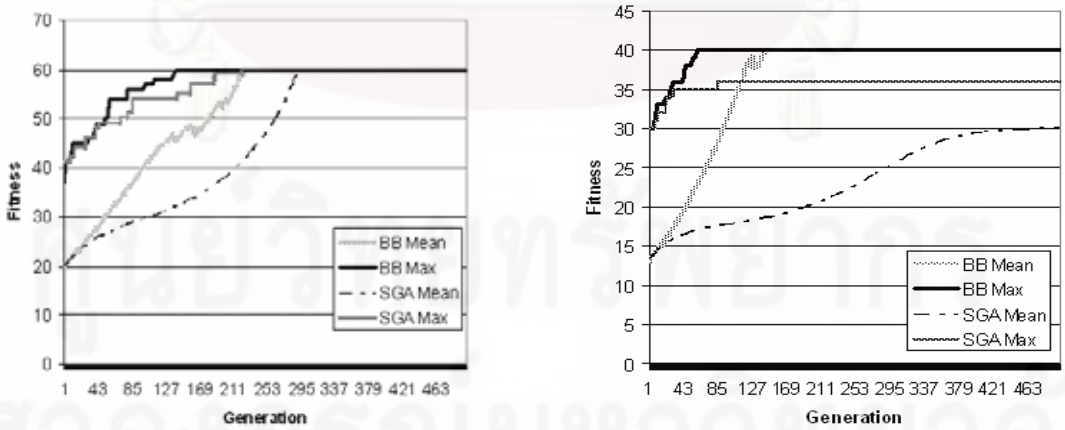
11111 ***** ***** *****

ปัญหากับดักสลับตำแหน่งจะมีเค้าร่างเป็นดังนี้

1*** 1*** 1*** 1*** 1***

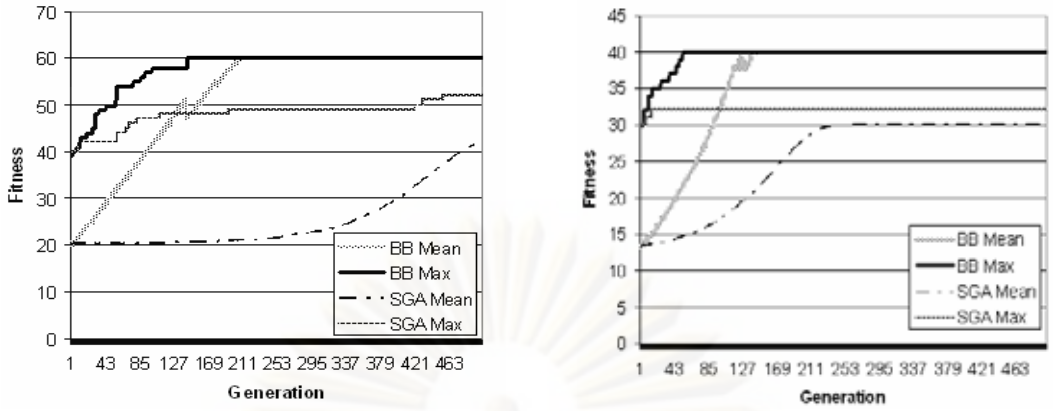
การทำเช่นนี้มีผลกระทบต่อขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายอย่างแน่นอนเนื่องจากกระบวนการไขว้เปลี่ยนเกิดขึ้นแบบสุ่มและไม่ได้สนใจหน่วยการสร้างเลย

ผลการทดลองปรากฏดังแสดงในรูปที่ 15 และรูปที่ 16 เป็นผลการทดลองเฉลี่ยจากการทำงาน 30 ครั้งไม่ขึ้นต่อกัน พบว่าการไขว้เปลี่ยนที่อาศัยหน่วยการสร้าง (ในรูปคือ BB Max) จะหาคำตอบเจอที่จำนวนรุ่นของประชากรน้อยกว่า



(ก) 20x3-trap function (ข) 10x4-trap function

รูปที่ 15 ผลการทดลองใช้การระบุหน่วยการสร้างในปัญหากับดัก



(ก) 20x3-shuffle trap function (ข) 10x4-shuffle trap function

รูปที่ 16 ผลการทดลองใช้การระบุหน่วยการสร้างในปัญหากับดักสลับตำแหน่ง

จากผลการทดลองในทุกๆ ปัญหา พบว่า การเพิ่มขึ้นของค่าความเหมาะสมเฉลี่ยของขั้นตอนวิธีที่ใช้หน่วยการสร้างเพิ่มขึ้นได้เร็วกว่าขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย ดูได้จากค่าเฉลี่ย BB-mean เทียบกับ SGA-mean จะเห็นว่า BB-mean มีค่าใต่สูงขึ้นอย่างรวดเร็วและลู่เข้าภายในจำนวนรุ่นประชากรที่น้อยกว่า ในขณะที่ขั้นตอนวิธีเชิงพันธุกรรมแก้ปัญหาได้เฉพาะปัญหากับดักขนาด 3 บิต ดังแสดงในรูปที่ 15 (ก) ส่วนปัญหาอื่นๆ ไม่สามารถหาคำตอบเหมาะที่สุดได้ ยิ่งปัญหาหลอกมีขนาดใหญ่ขึ้น ขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายจะยิ่งมีผลการทำงานแย่ลง ดูได้จากรูปที่ 15 (ก) จะเห็นว่าขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายสามารถแก้ปัญหากับดักที่มีอันดับ 3 (3-bit Trap) ได้ แต่ในรูปที่ 15 (ข) ไม่สามารถแก้ปัญหากับดักที่มีอันดับ 4 (4-bit Trap) ได้

จากผลการทดลองที่แสดงไว้ในรูปที่ 16 ทั้ง (ก) และ (ข) จะพบว่าในปัญหากับดักสลับตำแหน่งนั้นขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย (ในรูปคือ SGA-MAX) ไม่สามารถหาคำตอบได้ทั้ง 2 กรณี ส่วนขั้นตอนวิธีเชิงพันธุกรรมที่ใช้หน่วยการสร้างสามารถหาคำตอบได้ทุกปัญหาที่ทำการทดลอง ในขั้นตอนการวิจัยนี้ได้ร่วมตีพิมพ์ผลงานวิชาการ 1 บทความ ในงานการประชุมวิชาการระดับนานาชาติ [65]

บทที่ 4

การแก้ปัญหาหลายจุดประสงค์

ขั้นตอนวิธีหนึ่งที่เป็นพื้นฐานสำคัญอย่างหนึ่งของการแก้ปัญหาด้วยคอมพิวเตอร์คือการแบ่งแยกและเอาชนะ (divide and conquer) ที่เป็นเช่นนี้เนื่องจากว่า ถ้าเราสามารถแบ่งแยกย่อยปัญหาลงได้แล้วการแก้ปัญหาขนาดเล็กย่อมทำได้ง่ายและรวดเร็วกว่าการแก้ไขปัญหาขนาดใหญ่ แนวคิดเดียวกันนี้ก่อให้เกิดความพยายามในการระบุหน่วยการสร้างในอดีตที่ชี้ให้เห็นว่า ถ้าสามารถหากลุ่มเชื่อมโยงได้ถูกต้องแล้ว ก็จะเป็นการง่ายที่จะประกอบหน่วยการสร้างด้วยขั้นตอนวิธีเชิงพันธุกรรมโดยอาศัยข้อมูลการเชื่อมโยง [31] และในงานวิจัย [10] [64] [11] [18] ของชัชวิทย์และประภาส เป็นอีกความพยายามหนึ่งในการระบุหน่วยการสร้างสำหรับช่วยการทำงานของขั้นตอนวิธีเชิงพันธุกรรมให้มีประสิทธิภาพมากขึ้นแต่ยังจำกัดอยู่ในปัญหาจุดประสงค์เดียว ดังนั้นในงานวิจัยนี้จึงพยายามปรับปรุงให้เหมาะสมกับปัญหาหลายจุดประสงค์ และในบทนี้จะกล่าวถึงการนำเทคนิคการระบุหน่วยการสร้างมาใช้ในขั้นตอนวิธีเชิงพันธุกรรมสำหรับแก้ปัญหาหลายจุดประสงค์

4.1 การนำเทคนิคการระบุหน่วยการสร้างมาใช้แก้ปัญหาหลายจุดประสงค์

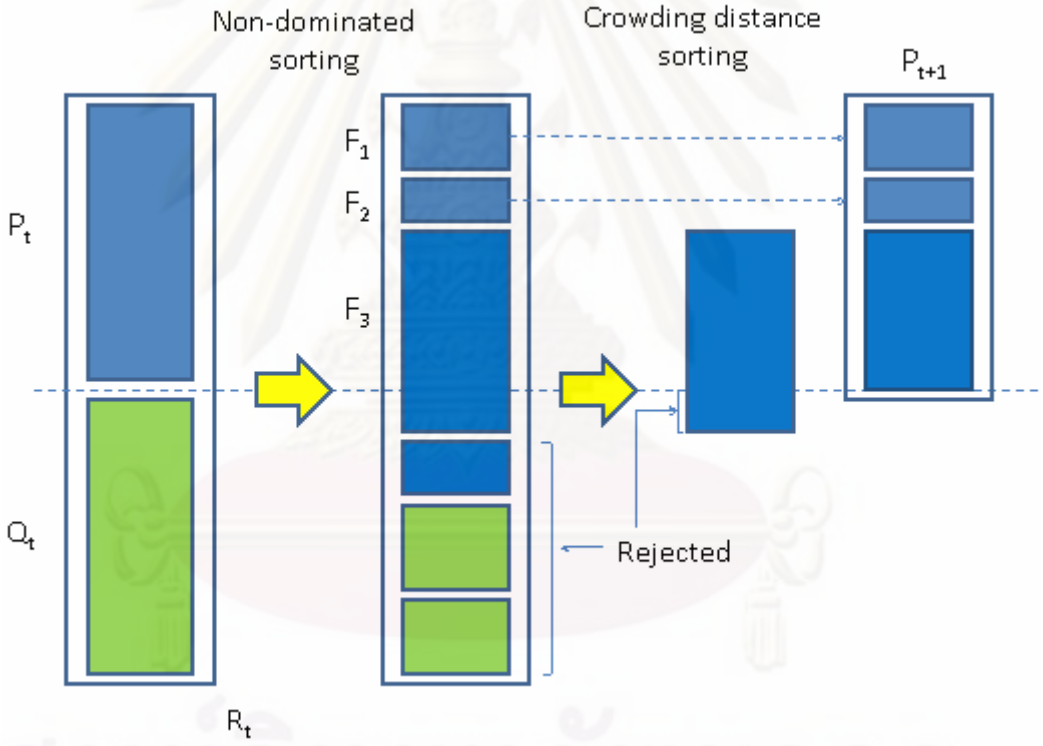
ขั้นตอนวิธีเชิงพันธุกรรมสำหรับแก้ปัญหาหลายจุดประสงค์ที่เลือกนำมาทดสอบคือ NSGA-II [5] เนื่องจากได้รับความนิยมแพร่หลาย มีขั้นตอนการทำงานง่ายรวดเร็ว และไม่จำเป็นต้องมีหน่วยความจำเพิ่มเติมภายนอกที่เรียกว่า หน่วยเก็บถาวร (archive) นอกจากนี้ในงานวิจัยนี้มุ่งเน้นศึกษาผลกระทบจากการปรับปรุงตัวดำเนินการไขว้เปลี่ยน ซึ่งจะสามารถนำไปใช้กับขั้นตอนวิธีเชิงพันธุกรรมอื่นที่มีตัวดำเนินการไขว้เปลี่ยนได้เช่นกัน

ในหัวข้อนี้จะประกอบไปด้วยขั้นตอนวิธี NSGA-II โดยละเอียดและการเพิ่มเติมเทคนิคระบุหน่วยการสร้างให้กับขั้นตอนวิธี NSGA-II ซึ่งจะมีเทคนิคการเพิ่มที่แตกต่างกัน

4.1.1 ขั้นตอนวิธี NSGA-II

ขั้นตอนวิธี NSGA-II ถูกสรุปไว้ในหัวข้อ 2.3.2 ค. ภายในหัวข้อนี้จะเพิ่มเติมรายละเอียดของขั้นตอนวิธีเพื่อใช้ประกอบการอธิบายในหัวข้อถัดไป ในรูปที่ 17 ได้แสดงกระบวนการทำงานของขั้นตอนวิธี NSGA-II และมีรหัสเทียมดังแสดงไว้ในรูปที่ 18

การทำงานของขั้นตอนวิธี NSGA-II มีขั้นตอนง่ายๆ และตรงไปตรงมา ขั้นแรกประชากรรุ่นพ่อแม่ P_t จะถูกรวมเข้ากับประชากรที่ถูกสร้างขึ้นมาใหม่ Q_t เรียกประชากรรวมนี้ว่า R_t ซึ่งมีขนาดเป็น $2N$ เมื่อ N เป็นขนาดประชากร จากนั้นจัดเรียงประชากร R_t ตามลำดับการครอบงำ ประชากรที่ไม่ถูกครอบงำเลยจะอยู่ในลำดับดีสุดคือ F_1 และถ้าขนาดของ F_1 น้อยกว่าขนาดประชากรที่ต้องการ คำตอบใน F_1 จะถูกนำไปเป็นประชากรรุ่นถัดไปทั้งหมด แน่นอนว่าเป็นการเก็บประชากรดีสุดไว้ (elitism) ประชากรที่ยังไม่เต็มของ P_{t+1} จะได้จากการรวมคำตอบในลำดับชั้นถัดไปคือ F_2 ตามด้วย F_3 ไปเรื่อยๆ จนกระทั่งถึงลำดับชั้น F_l ซึ่งประชากรที่เพิ่มเข้ามาเกินขนาดประชากรที่ต้องการ ในกรณีนี้จะทำการจัดเรียงคำตอบในลำดับชั้น F_l ด้วยตัวดำเนินการเปรียบเทียบความหนาแน่นและเลือกคำตอบที่หนาแน่นน้อยสุดไปเรื่อยๆ จนได้จำนวนประชากรตามต้องการเป็น P_{t+1} มีขนาด N และนำไปสู่กระบวนการคัดเลือก ไขว้เปลี่ยนและกลายพันธุ์ตามลำดับต่อไป



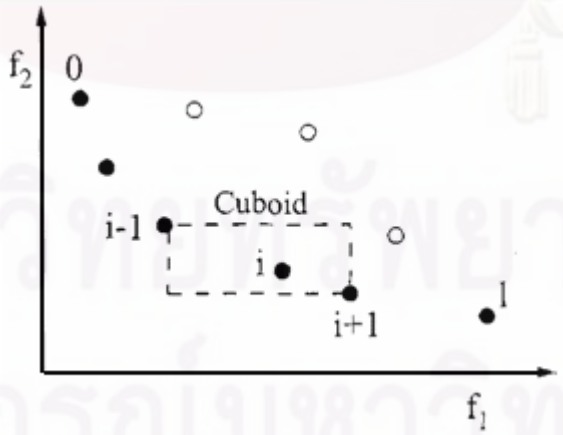
รูปที่ 17 ขั้นตอนการทำงานของ NSGA-II

การทำงานในแต่ละรุ่นจะมีขั้นตอนต่างๆ เขียนเป็นรหัสเทียมได้ดังรูปที่ 18

1:	$\boldsymbol{R}_t = \boldsymbol{P}_t \cup \boldsymbol{Q}_t$	combine parent and offspring population
2:	$\boldsymbol{F}$ = fast-nondominated-sort($\boldsymbol{R}_t$)	$\boldsymbol{F} = (\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots)$, all nondominated fronts of $\boldsymbol{R}_t$
3:	$\boldsymbol{P}_{t+1} = \varnothing$ and $i = 1$	
4:	**until** $\boldsymbol{P}_{t+1} + \boldsymbol{F}_i \leq \boldsymbol{N}$	until the parent population is filled
5:	crowding-distance-assignment($\boldsymbol{F}_i$)	calculate crowding-distance in F_i
6:	$\boldsymbol{P}_{t+1} = \boldsymbol{P}_{t+1} \cup \boldsymbol{F}_i$	include ith nondominated front in the parent pop
7:	$i = i + 1$	check the next front for inclusion
8:	Sort($\boldsymbol{F}_i, \prec_n$)	sort in descending order using $\prec_n$
9:	$\boldsymbol{P}_{t+1} = \boldsymbol{P}_{t+1} \cup \boldsymbol{F}_i[1 : (\boldsymbol{N} - \boldsymbol{P}_{t+1})]$	choose the first ($\boldsymbol{N}$ - $\lvert\boldsymbol{P}_{t+1}\rvert$) elements of F_i
10:	$\boldsymbol{Q}_{t+1}$ = make-new-pop($\boldsymbol{P}_{t+1}$)	use selection crossover and mutation to create a new population Q_{t+1}
11:	$t = t + 1$	increment the generation counter

รูปที่ 18 รหัสเทียมรอบการทำงานหลักของ NSGA-II

ในขั้นตอนวิธี NSGA-II การกระจายตัวของค่าจุดประสงค์ของคำตอบในพาเรโตฟรอนต์ได้มาจากการใช้วิธีการเปรียบเทียบความหนาแน่น $\prec_n$ (crowding comparison procedure) ซึ่งถูกใช้ในการคัดเลือกแบบแข่งขัน ในการประมาณความหนาแน่นของคำตอบรอบ ๆ คำตอบหนึ่ง หาได้จากค่าเฉลี่ยระยะห่างของสองจุดใกล้สุดจากทั้งสองข้างในแต่ละจุดประสงค์ ระยะห่างที่ได้จะเป็นการประมาณขอบเขตของรูปทรงลูกบาศก์ที่มีจุดรอบข้างที่ใกล้ที่สุดเป็นจุดมุม เรียกระยะห่างนี้ว่า ระยะเบียด (crowding distance) ในรูปที่ 19 แสดงการประมาณระยะเบียดของคำตอบที่ *i* ภายในลำดับชั้นเดียวกัน (เป็นรูปวงกลมทึบ) เท่ากับค่าเฉลี่ยของความยาวแต่ละด้านของรูปสี่เหลี่ยม จะเห็นว่าการหาระยะเบียดจะต้องเรียงลำดับคำตอบในลำดับชั้นเดียวกันของแต่ละจุดประสงค์



รูปที่ 19 การคำนวณระยะเบียด

4.1.2 ขั้นตอนวิธี NSGA-II ร่วมกับ BB-wise

ขั้นตอนวิธี NSGA-II ซึ่งได้อธิบายไว้ในหัวข้อก่อนหน้า ได้ถูกปรับปรุงการไขว้เปลี่ยนจากการไขว้เปลี่ยนแบบสองจุด (two-point crossover) มาเป็นการไขว้เปลี่ยนแบบใช้หน่วยการสร้าง (BB-wise crossover)

จากรหัสเทียมของขั้นตอนวิธี NSGA-II ดังแสดงในรูปที่ 18 บรรทัดที่ 10 เป็นขั้นตอนการสร้างประชากรใหม่ (make-new-pop(P_{t+1})) โดยคัดเลือกจากประชากรรุ่นปัจจุบัน (P_{t+1}) แล้วนำไปทำการสร้างประชากรรุ่นถัดไป (Q_{t+1}) ซึ่งจะประกอบไปด้วยการทำงานย่อย ๆ ดังนี้

การคัดเลือก (selection) ใช้วิธีการคัดเลือกแบบแข่งขัน
การไขว้เปลี่ยน (crossover) ใช้วิธีการไขว้เปลี่ยนแบบ 2 จุด (Two-point Crossover)
การกลายพันธุ์ (mutation) ใช้วิธีการสุ่มเพื่อเปลี่ยนค่าบิต
รหัสเทียมของขั้นตอน make-new-pop ในขั้นตอนวิธี NSGA-II ดังแสดงไว้ในรูปที่ 20

```
    Procedure make-new-pop(P_{t+1})
1:  n=0, Q_{t+1} = ∅
2:  while n < N              ; N ≡ 0 (mod 4)
3:      (p1, p2) = selection (P_{t+1})
4:      (c1, c2) = two-point-crossover(p1, p2)
5:      c1' = mutation(c1)
6:      c2' = mutation(c2)
7:      Q_{t+1} = Q_{t+1} ∪ { c1', c2' }
8:      n = n + 2
9:  end while
```

รูปที่ 20 รหัสเทียมส่วนการสร้างประชากรใหม่ (make-new-pop)

เนื่องจากภายใน make-new-pop จะมีการเรียกใช้ two-point-crossover (รูปที่ 20 บรรทัดที่ 4) ซึ่งจะถูกเปลี่ยนเป็น BB-wise-crossover รวมแล้วเรียกว่า make-new-popBB ดังนั้นขั้นตอนวิธี NSGA-II แบบร่วมกับ BB-wise-crossover จึงต้องมีการระบุหน่วยการสร้างก่อนที่จะสร้างประชากรรุ่นใหม่ และจะได้ภาพรวมดังแสดงไว้ในรูปที่ 21 โดยที่บรรทัดที่ 10 จะเลือกประชากรครึ่งดีกว่า (better half) เพื่อนำมาหาหน่วยการสร้างในบรรทัดที่ 11 ซึ่งจะทำการระบุหน่วยการสร้างก่อน (Building-Block-Identification) แล้วนำหน่วยการสร้างที่ได้ไปใช้ในขั้นตอน make-new-popBB ในบรรทัดที่ 12 ต่อไป

1: $\boldsymbol{R}_t = \boldsymbol{P}_t \cup \boldsymbol{Q}_t$
2: $\boldsymbol{F}$ = fast-nondominated-sort($\boldsymbol{R}_t$)
3: $\boldsymbol{P}_{t+1} = \varnothing$ and $i = 1$
4: **until** $\boldsymbol{P}_{t+1} + \boldsymbol{F}_i \leq \boldsymbol{N}$
5: crowding-distance-assignment($\boldsymbol{F}_i$)
6: $\boldsymbol{P}_{t+1} = \boldsymbol{P}_{t+1} \cup \boldsymbol{F}_i$
7: $i = i + 1$
8: Sort($\boldsymbol{F}_i, \prec_n$)
9: $\boldsymbol{P}_{t+1} = \boldsymbol{P}_{t+1} \cup \boldsymbol{F}_i[1 : (\boldsymbol{N} - \boldsymbol{P}_{t+1})]$
10: $\boldsymbol{H}$ = **SelectBetterHalf**($\boldsymbol{P}_{t+1}$)
11: **Building-Block-Identification**($\boldsymbol{H}$)
12: $\boldsymbol{Q}_{t+1}$ = **make-new-popBB**($\boldsymbol{P}_{t+1}$)
13: $t = t + 1$

รูปที่ 21 ขั้นตอนวิธี NSGA-II แบบใช้หน่วยการสร้าง

การระบุหน่วยการสร้างของปัญหาหลายจุดประสงค์เหมือนกับปัญหาจุดประสงค์เดียว แตกต่างกันตรงที่มาของประชากรที่ใช้ในการหาหน่วยการสร้าง ในปัญหาจุดประสงค์เดียวมีค่าจุดประสงค์เป็นตัวเปรียบเทียบคุณภาพคำตอบ ดังนั้นการแบ่งแยกประชากรครึ่งหนึ่งที่ดีกว่าทำได้ง่ายโดยการเปรียบเทียบค่าจุดประสงค์เดียวนั้น การเลือกประชากรมาหาหน่วยการสร้างในปัญหาจุดประสงค์เดียวจึงเลือกใช้ประชากรครึ่งที่ดีกว่า แต่สำหรับปัญหาหลายจุดประสงค์ประชากรครึ่งดีกว่าจะกลายเป็นประชากรครึ่งหนึ่งที่ไม่ถูกครอบงำโดยประชากรอีกครึ่งที่เหลือ นอกจากนี้ยังมีการเลือกประชากรในแบบอื่นอีก คือ เลือกประชากรที่ดีของแต่ละจุดประสงค์มาครึ่งหนึ่งของประชากรทั้งหมดเพื่อใช้หาหน่วยการสร้างของจุดประสงค์นั้นๆ

จากข้อมูลเกี่ยวกับหน่วยการสร้างที่มีในปัญหาจุดประสงค์เดียว จึงคาดว่าในปัญหาหลายจุดประสงค์ย่อมจะมีหน่วยการสร้างของคำตอบในแต่ละจุดประสงค์ ทำให้เราคาดว่าจะสามารถหาคำตอบสุดโต่ง (Extremely Solution) ของแต่ละจุดประสงค์ได้ (ในความเป็นจริงปัญหาหลายจุดประสงค์ไม่ได้สนใจแค่เพียงคำตอบสุดโต่งในแต่ละด้าน แต่สนใจคำตอบที่อยู่ในพาเรโตฟรอนต์ซึ่งจะเป็นการถ่วงดุลของแต่ละจุดประสงค์)

ถ้ากำหนดให้แต่ละจุดประสงค์ปรากฏหน่วยการสร้างเด่นชัดแล้ว ก็ยังมีข้อคำถามที่น่าสนใจว่า

1. การรวมกันของหลาย ๆ จุดประสงค์ จะทำให้คำตอบที่อยู่ในพาเรโตฟรอนต์มีหน่วยการสร้างด้วยหรือไม่ ถ้ามีหน่วยการสร้างก็มีความเป็นไปได้ว่าเราสามารถใช้การระบุหน่วยการสร้างในการแก้ปัญหาได้
2. แต่ถ้าในพาเรโตฟรอนต์ไม่มีหน่วยการสร้างปรากฏให้เห็นเด่นชัด เราจะปรับใช้การระบุหน่วยการสร้างอย่างไรให้เหมาะสมกับปัญหา

จากปัญหาในข้อ 2. ทำให้เกิดคำถามที่สำคัญคือ ถ้าเรารู้หน่วยการสร้างของแต่ละจุดประสงค์แล้วเราจะประกอบหน่วยการสร้างนั้นอย่างไรให้กลายเป็นคำตอบ จากข้อสงสัยที่กล่าวมาทำให้เกิดแนวคิดในการแก้ปัญหาดังต่อไปนี้

แนวทางที่ 1 ใช้หน่วยการสร้างร่วมจากคำตอบที่ไม่ถูกครอบงำโดยตรง

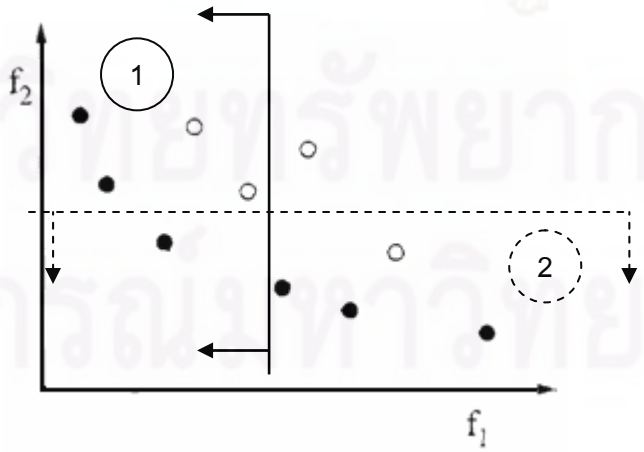
แนวทางที่ 2 รวมหน่วยการสร้างของหลายจุดประสงค์เข้าด้วยกัน ก่อนสร้างคำตอบใหม่

แนวทางที่ 3 หาคำตอบด้วยหน่วยการสร้างของแต่ละจุดประสงค์แยกกัน

วิธีการรวมหน่วยการสร้างของแนวทางที่ 2 ยังแบ่งได้เป็น 2 วิธี คือรวมแบบเชื่อมั่น (mix confident) และรวมแบบขยาย (mix growth) ทำให้ได้กลวิธีการระบุหน่วยการสร้างต่างๆ กันเป็น 4 วิธี คือ

1) NSGA-IIa ใช้หน่วยการสร้างร่วมจากคำตอบในพาเรโตฟรอนต์
2) NSGA-IIb การผสมหน่วยการสร้างแบบเชื่อมั่น
3) NSGA-IIc การผสมหน่วยการสร้างแบบขยาย
4) NSGA-IId ใช้หน่วยการสร้างจากแต่ละจุดประสงค์แยกกัน

ในข้อ 1) คำตอบที่ได้ในแต่ละรุ่นจะถูกจัดเรียงตามลำดับการครอบงำ จากนั้นจะเลือกเอาคำตอบที่ไม่ถูกครอบงำและคำตอบในลำดับชั้น (rank) ต้นๆ มากึ่งหนึ่งของจำนวนประชากร เพื่อนำมาหาหน่วยการสร้าง ส่วนในข้อ 2) 3) และ 4) คำตอบจะถูกจัดเรียงตามคุณภาพคำตอบในแต่ละจุดประสงค์ ในปัญหาที่มี 2 จุดประสงค์ก็จะมีการจัดเรียง 2 รอบ คำตอบที่ดีในจุดประสงค์แรกจำนวนครึ่งหนึ่งของประชากรจะถูกเลือกมาเพื่อใช้หาหน่วยการสร้าง และจะได้หน่วยการสร้างของคำตอบที่ดีสำหรับจุดประสงค์แรก ในจุดประสงค์ถัดไปก็เช่นกัน จะเลือกเอาคำตอบที่ดีครึ่งหนึ่งของประชากรที่เรียงลำดับตามค่าในจุดประสงค์นั้นๆ มาใช้สำหรับหาหน่วยการสร้างของจุดประสงค์นั้น ในรูปที่ 22 บริเวณ (1) เป็นคำตอบที่ดีที่ถูกเลือกตามจุดประสงค์ f_1 และบริเวณ (2) เป็นบริเวณคำตอบที่ดีตามจุดประสงค์ f_2 ในปัญหาที่ต้องการค่าน้อยสุด



รูปที่ 22 บริเวณคำตอบที่ถูกเลือกในแต่ละจุดประสงค์

วิธีในข้อ 2) และ ข้อ 3) จะมีการรวมหน่วยการสร้างของแต่ละจุดประสงค์เป็นคนละแบบก่อนนำไปใช้งาน ส่วนข้อ 4) จะใช้หน่วยการสร้างของแต่ละจุดประสงค์สร้างประชากรรุ่นลูกในปริมาณเท่าๆ กัน รายละเอียดของแต่ละข้อจะกล่าวถึงโดยละเอียดในหัวข้อถัดๆ ไปตามลำดับ เพื่อให้ง่ายต่อความเข้าใจขออธิบายในกรณีปัญหามี 2 จุดประสงค์ การรวมหน่วยการสร้างในแต่ละแนวทางมีรายละเอียดดังนี้

4.1.2.1 ขั้นตอน NSGA-II ร่วมกับ BB-wise แบบที่ 1 (NSGA-IIa)

วิธีการนี้ไม่สนใจหน่วยการสร้างในแต่ละจุดประสงค์ แต่จะสนใจหน่วยการสร้างร่วมที่ได้จากคำตอบที่ไม่ถูกครอบงำในแต่ละรุ่นประชากร หากปริมาณคำตอบไม่มากพอก็จะรวมคำตอบในอันดับ (rank) ต้น ๆ เข้ามาด้วย จากนั้นหาหน่วยการสร้างจากคำตอบทั้งหมดที่เลือกมา ทั้งนี้เนื่องมาจากบางปัญหา เช่น $Trap_k$-$invtrap_k$ [6] ในเซตคำตอบแบบพาเรโตมีหน่วยการสร้างที่เด่นชัดอยู่แล้ว ดังนั้น จึงไม่จำเป็นต้องแบ่งแยกหน่วยการสร้างว่าเป็นของจุดประสงค์ใด จึงทำให้เกิดวิธีการนี้ คือ การระบุหน่วยการสร้างโดยหาจากคำตอบที่เลือกมาโดยไม่แบ่งแยกจุดประสงค์ นอกจากนี้การระบุหน่วยการสร้างในกระบวนการค้นหาไม่จำเป็นต้องถูกต้องกับปัญหาจริงเสมอไป เราสนใจผลลัพธ์จากกระบวนการทั้งหมดของขั้นตอนวิธีเชิงพันธุกรรมว่าได้ปรับปรุงคุณภาพให้ดีกว่าขั้นตอนวิธีเดิมหรือไม่ สิ่งสำคัญคือ จะทำการไขว้เปลี่ยนอย่างไรจึงจะส่งเสริมการปรับปรุงคุณภาพคำตอบ

ขั้นตอนวิธีนี้จะต้องใช้ประชากรครึ่งดีกว่าแบบหลายจุดประสงค์ แต่เนื่องจากขั้นตอนวิธี NSGA-II ได้จัดเรียงการครอบงำด้วยขั้นตอน fast-nondominated-sort เรียบร้อยแล้วจึงไม่จำเป็นต้องจัดเรียงอีกรอบ และสามารถหยิบประชากรครึ่งดีมาใช้ได้เลย รหัสเทียมของขั้นตอนวิธีนี้จึงเหมือนกับที่แสดงไว้ในรูปที่ 21

4.1.2.2 ขั้นตอนวิธี NSGA-II ร่วมกับ BB-wise แบบที่ 2 (NSGA-IIb)

ในที่นี้คาดหวังว่าแต่ละจุดประสงค์เป็นปัญหาที่มีหน่วยการสร้างในตัวเอง แต่เมื่อรวมเป็นปัญหาหลายจุดประสงค์ไม่ปรากฏหน่วยการสร้างในคำตอบที่อยู่ในพาเรโตฟรอนต์ แต่อาจจะสามารถหาหน่วยการสร้างของแต่ละจุดประสงค์ได้โดยการจัดเรียงคำตอบด้วยค่าของจุดประสงค์นั้นๆ ก่อน แล้วเลือกเอาคำตอบครึ่งที่ดีกว่ามาหาหน่วยการสร้างของจุดประสงค์นั้น คำตอบที่เลือกมาจึงไม่จำเป็นต้องเป็นคำตอบที่อยู่ในพาเรโตฟรอนต์และอาจจะไม่ใช่อันดับต้นๆ แต่ก็เป็นคำตอบที่ดีของจุดประสงค์นั้นๆ โดยเฉพาะคำตอบสุดโต่งของแต่ละจุดประสงค์ก็จะอยู่ในกลุ่มที่เลือกมา ดังแสดงในรูปที่ 22 จะเห็นได้ว่าบริเวณ (1) มีคำตอบที่ดีสำหรับจุดประสงค์ f_1 แต่อาจจะไม่ได้เป็นคำตอบที่ดีในจุดประสงค์ f_2 และอาจจะไม่ได้อยู่ในพาเรโตฟรอนต์ด้วย ดังนั้นในการหาหน่วยการสร้างของแต่ละจุดประสงค์จึงต้องมีการจัดเรียงคำตอบตามค่าของ

จุดประสงค์และเลือกคำตอบที่ดีของจุดประสงค์นั้นๆ ก่อนที่จะใช้สำหรับหาหน่วยการสร้างของแต่ละจุดประสงค์ดังที่ปรากฏในรหัสเทียมของกลวิธีนี้ในรูปที่ 23

```
1:  R_t = P_t U Q_t
2:  F = fast-nondominated-sort(R_t)
3:  P_{t+1} = ∅ and i = 1
4:  until P_{t+1} + F_i ≤ N
5:    crowding-distance-assignment(F_i)
6:    P_{t+1} = P_{t+1} ∪ F_i
7:    i = i + 1
8:  Sort(F_i, ≺_n)
9:  P_{t+1} = P_{t+1} ∪ F_i[1 : (N – P_{t+1})]
10: for each objective j
11:     S_j = SortByCurrentObjective(P_{t+1})
12:     H_j = selectBetterHalf(S_j)
13:     BB_j = Building-Block-Identification(H_j)
14: end for
15: MergeConfidentBBs
16: Q_{t+1} = make-new-popBB(P_{t+1})
17: t = t + 1
```

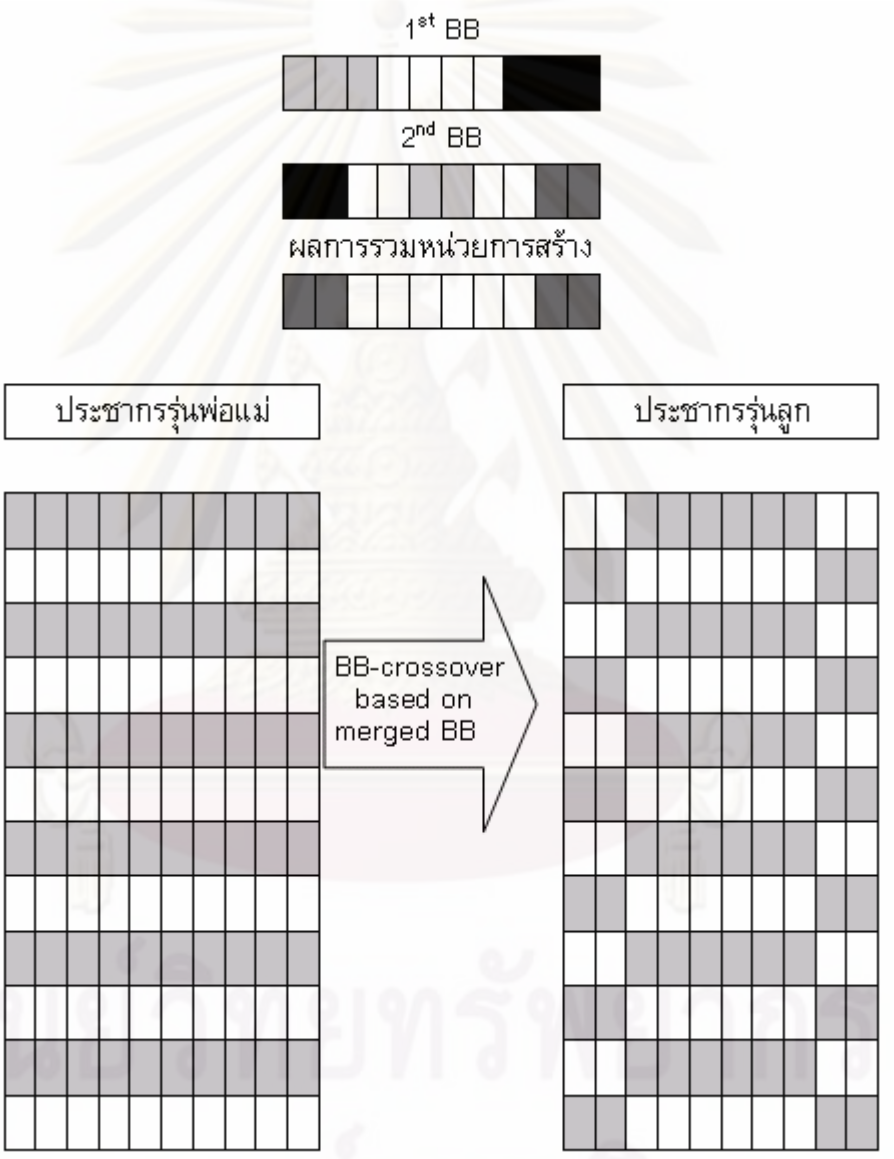
รูปที่ 23 รหัสเทียมขั้นตอนวิธี NSGA-II ร่วมกับ BB-wise แบบที่ 2

จากที่กล่าวข้างต้นแล้วหน่วยการสร้างของแต่ละจุดประสงค์ อาจจะอยู่คนละตำแหน่งไม่ตรงกันหรือเหลื่อมกัน (overlap) หลังจากได้หน่วยการสร้างของแต่ละจุดประสงค์ควรจะมีการรวมหน่วยการสร้างแต่ละแบบเข้าด้วยกันก่อนนำไปใช้งาน ดังปรากฏในรหัสเทียมบรรทัดที่ 15 (MergeConfidentBBs) การรวมหน่วยการสร้างด้วยกลวิธีนี้มีขั้นตอนวิธีรวมหน่วยการสร้างแบบเชื่อมั่นดังรหัสเทียมที่แสดงไว้ในรูปที่ 24

```
    Procedure MergeConfidentBBs
1:  GroupedBB = BBofFirstObjective
2:  for i = 0 to l-2 do
3:      for j = i+1 to l-1 do
4:          forall Objectives except 1st Objective
5:              if not InSamePart(i,j)
6:                  RemoveLinkage(i,j) from GroupedBB
7:              end if
8:          end forall
9:      end for
10: end for
```

รูปที่ 24 รหัสเทียมขั้นตอนวิธีรวมหน่วยการสร้างแบบเชื่อมั่น

การรวมแบบนี้เราต้องการคงความสัมพันธ์ที่น่าเชื่อถือไว้ นั่นคือคู่บิตที่จะถือว่าอยู่ในหน่วยการสร้างเดียวกัน จะต้องปรากฏในทุกๆ จุดประสงค์ว่าอยู่ในหน่วยการสร้างเดียวกันดัง รูปที่ 25 แสดงให้เห็นตัวอย่างการรวมแบบเชื่อมั่น จะเห็นได้ว่าผลการรวมหน่วยการสร้างจะมีขนาดหน่วยการสร้างเล็กลงและจะมีขนาดไม่ใหญ่ไปกว่าหน่วยการสร้างก่อนที่จะถูกรวม จากความเชื่อที่ว่าหน่วยการสร้างเป็นโครงสร้างร่วม (common structure) ของคำตอบที่ดีๆ ดังนั้นในวิธีนี้จึงสนใจเฉพาะหน่วยการสร้างที่ปรากฏตรงกัน หรือเป็นลักษณะร่วมของหน่วยการสร้างจากแต่ละจุดประสงค์นั่นเอง



รูปที่ 25 การรวมหน่วยการสร้างแต่ละจุดประสงค์แบบเชื่อมั่น

4.1.2.3 ขั้นตอน NSGA-II ร่วมกับ BB-wise แบบที่ 3 (NSGA-IIc)

การไขว้เปลี่ยนด้วยหน่วยการสร้างในแต่ละจุดประสงค์ก็อาจจะทำให้หน่วยการสร้างของอีกจุดประสงค์เสียหายไปด้วย ดังนั้นจึงเห็นว่า การรวมหน่วยการสร้างที่เหลื่อมกัน (Overlapping BBs) น่าจะช่วยให้รักษาหน่วยการสร้างของทั้งสองจุดประสงค์ไว้ได้พร้อม ๆ กัน การรวมแบบนี้เพื่อไม่ให้เกิดการทำลายหน่วยการสร้างของจุดประสงค์ใดๆ เลยโดยหน่วยการสร้างที่เหลื่อมกันอยู่จะถูกมองเป็นก้อนเดียวกันและมีขนาดใหญ่ขึ้น

อีกเหตุผลหนึ่งที่สำคัญสำหรับการรวมแบบนี้คือ ธรรมชาติของการสุ่มตัวอย่างเพื่อนำมาหาหน่วยการสร้าง มักจะระบุหน่วยการสร้างที่ไม่สมบูรณ์แบบ อาจจะเนื่องมาจากจำนวนประชากรที่นำมาใช้น้อยเกินไปที่จะแสดงความสัมพันธ์แต่ละคู่บิตได้ครบทั้งหมด จากการทดลองพบว่าบางครั้งการระบุหน่วยการสร้างด้วยคำตอบที่ดีของจุดประสงค์หนึ่งทำให้ได้กลุ่มบิตบางส่วนของหน่วยการสร้างและคำตอบในอีกจุดประสงค์หนึ่งระบุหน่วยการสร้างได้เพียงบางส่วนของหน่วยการสร้างในปัญหาจริง หน่วยการสร้างทั้งสองแบบที่หาได้ก็มักจะเหลื่อมหรือซ้อนกันอยู่ การรวมหน่วยการสร้างด้วยวิธีนี้จะทำให้ได้หน่วยการสร้างที่สมบูรณ์ยิ่งขึ้น เรียกว่าขั้นตอนวิธีรวมหน่วยการสร้างแบบขยาย

รหัสเทียมของขั้นตอนวิธีนี้คล้ายกับวิธีก่อนหน้า ต่างกันที่ขั้นตอนวิธีการรวมหน่วยการสร้าง เปลี่ยนจาก MergeConfidentBBs เป็น MergeGrowthBBs ในบรรทัดที่ 15 ดังแสดงไว้ในรูปที่ 26

```
1:  R_t = P_t ∪ Q_t
2:  F = fast-nondominated-sort(R_t)
3:  P_{t+1} = ∅ and i = 1
4:  until P_{t+1} + F_i ≤ N
5:    crowding-distance-assignment(F_i)
6:    P_{t+1} = P_{t+1} ∪ F_i
7:    i = i + 1
8:  Sort(F_i, ≺_n)
9:  P_{t+1} = P_{t+1} ∪ F_i[1 : (N – P_{t+1})]
10: for each objective j
11:    S_j = SortByCurrentObjective(P_{t+1})
12:    H_j = selectBetterHalf(S_j)
13:    BB_j = Building-Block-Identification(H_j)
14: end for
15: MergeGrowthBBs
16: Q_{t+1} = make-new-popBB(P_{t+1})
17: t = t + 1
```

รูปที่ 26 รหัสเทียมขั้นตอนวิธี NSGA-II ร่วมกับ BB-wise แบบที่ 3

ขั้นตอนวิธีการรวมหน่วยการสร้างแบบขยายถูกแสดงไว้ในรูปที่ 27

แสดงตัวอย่างการรวมได้ดังต่อไปนี้

P1 = < 111 222 345 666> หรือ { {1,2,3}, {4,5,6}, {7}, {8}, {9}, {10,11,12} }

P2 = < 112 233 445 566> หรือ { {1,2}, {3,4}, {5,6}, {7,8}, {9,10}, {11,12} }

G = < 111 111 446 666> หรือ { {1,2,3,4,5,6}, {7,8}, {9,10,11,12} }

กำหนดให้ P1 เป็นหน่วยการสร้างของจุดประสงค์ที่ 1
P2 เป็นหน่วยการสร้างของจุดประสงค์ที่ 2
G เป็นผลลัพธ์จากการรวมหน่วยการสร้างของทั้งสองจุดประสงค์
หมายเหตุ *ผลลัพธ์ที่ได้จากการรวมจะมีกลุ่มหมายเลข 0 เกิดขึ้น มีความหมายเป็นบิตอิสระ ไม่ขึ้นกับบิตอื่น ไม่ได้หมายถึงกลุ่มบิตหน่วยการสร้างกลุ่มใหม่แต่อย่างใด*

ในจุดประสงค์ที่ 1 (P1) มองบิตที่ 1 ถึง 6 เป็นสองหน่วยการสร้าง คือ 111 และ 222 ในขณะที่จุดประสงค์ที่ 2 (P2) มองเป็นสามหน่วยการสร้าง แต่เนื่องจากทั้ง 6 บิตนี้มีหน่วยการสร้างเหลื่อมกันอยู่จึงถูกรวมเป็นก้อนเดียวกัน และในทำนองเดียวกันกลุ่มบิตหมายเลข 6 ใน P1 เหลื่อมกับกลุ่มบิตหมายเลข 5 ใน P2 กลุ่มบิตหมายเลข 5 ใน P2 จึงถูกนำไปรวมเป็นกลุ่มบิตเดียวกับกลุ่มบิตหมายเลข 6 ใน P1 ทำให้กลุ่มบิตหมายเลข 6 มีขนาดโตขึ้นเนื่องจากบิตอิสระหมายเลข 5 (บิตที่ 9) ถูกรวมเข้าไปด้วย โดยมีเงื่อนไขของข้อมูลขาเข้าแสดงอยู่ในหน้าถัดไป

```
     Algorithm MergeGrowthBBs(P1,P2,G)
1:   for i = 0 to |P1| - 1 do
2:       for j = 0 to |P2| - 1 do
3:           if (A_i ∩ B_j) ≠ ∅ and |A_i| > 1 and |B_j| > 1
4:                   A_i ← A_i ∪ B_j
5:           end if
6:       end for
7:       for k = 0 to |G| - 1 do
8:           if (C_k ∩ A_i) ≠ ∅
9:                   G←G \ C_k
10:                  A_i ← A_i ∪ C_k
11:          end if
12:      end for
13:      G ← G ∪ A_i
14:  end for
15:  forall C_i ∈ G and B_j ∈ P2: if C_i ∩ B_j = ∅
16:      G ← G ∪ B_j
17:  return G
```

รูปที่ 27 รหัสเทียมขั้นตอนวิธีรวมหน่วยการสร้างแบบขยาย

กำหนดให้ ข้อมูลขาเข้า

$$\mathbf{P1} = \{ A_0 , \ldots , A_{|\mathbf{P1}|-1} \}, \quad \bigcup_{i=0}^{|P1|-1} A_i = \{0, \ldots , l-1\}, \quad A_i \cap A_j = \varnothing \text{ for all } i \neq j$$

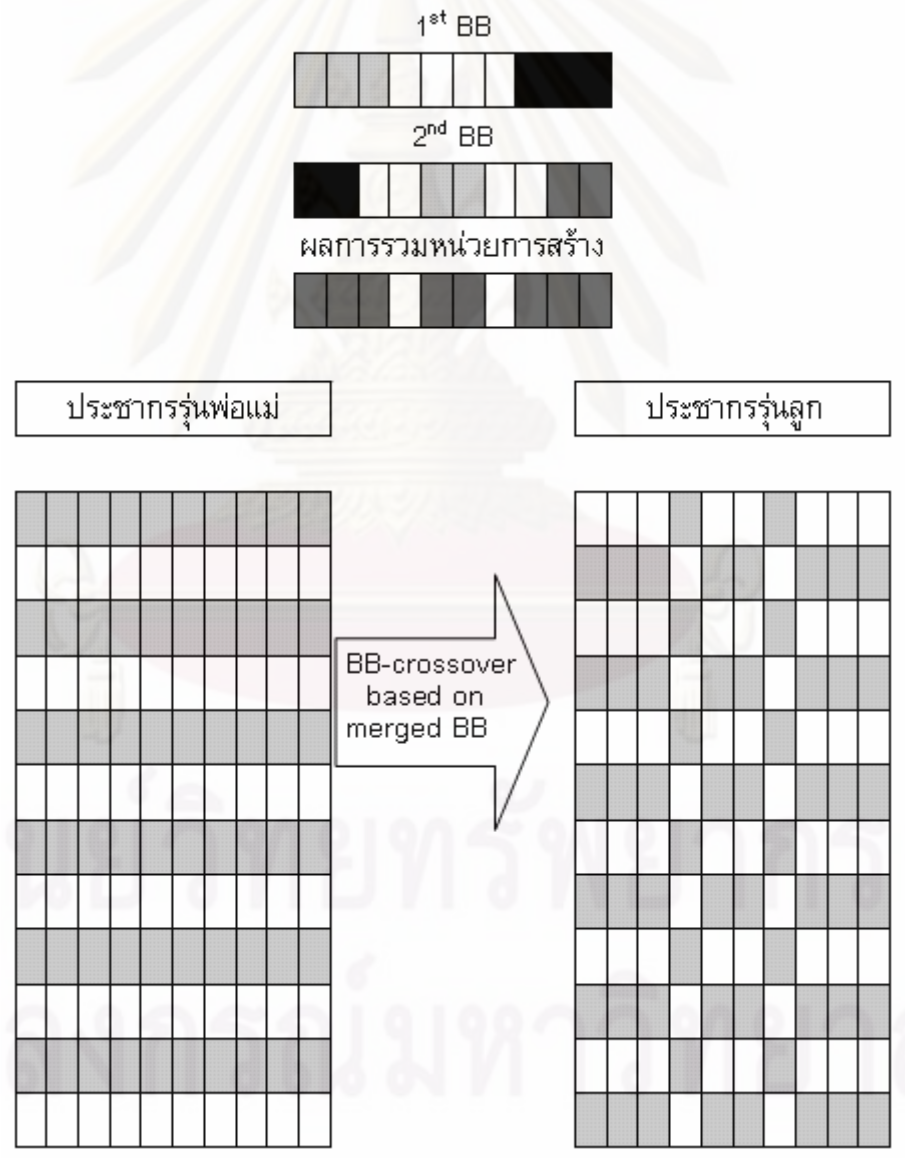
$$\mathbf{P2} = \{ B_0 , \ldots , B_{|\mathbf{P2}|-1} \}, \quad \bigcup_{i=0}^{|P2|-1} B_i = \{0, \ldots , l-1\}, \quad B_i \cap B_j = \varnothing \text{ for all } i \neq j$$

$$\mathbf{G} = \varnothing$$

ข้อมูลขาออก

$$\mathbf{G} = \{C_0, \ldots , C_{|\mathbf{G}|-1}\}, \quad \bigcup_{i=0}^{|G|-1} C_i = \{0, \ldots , l-1\}, \quad C_i \cap C_j = \varnothing \text{ for all } i \neq j$$

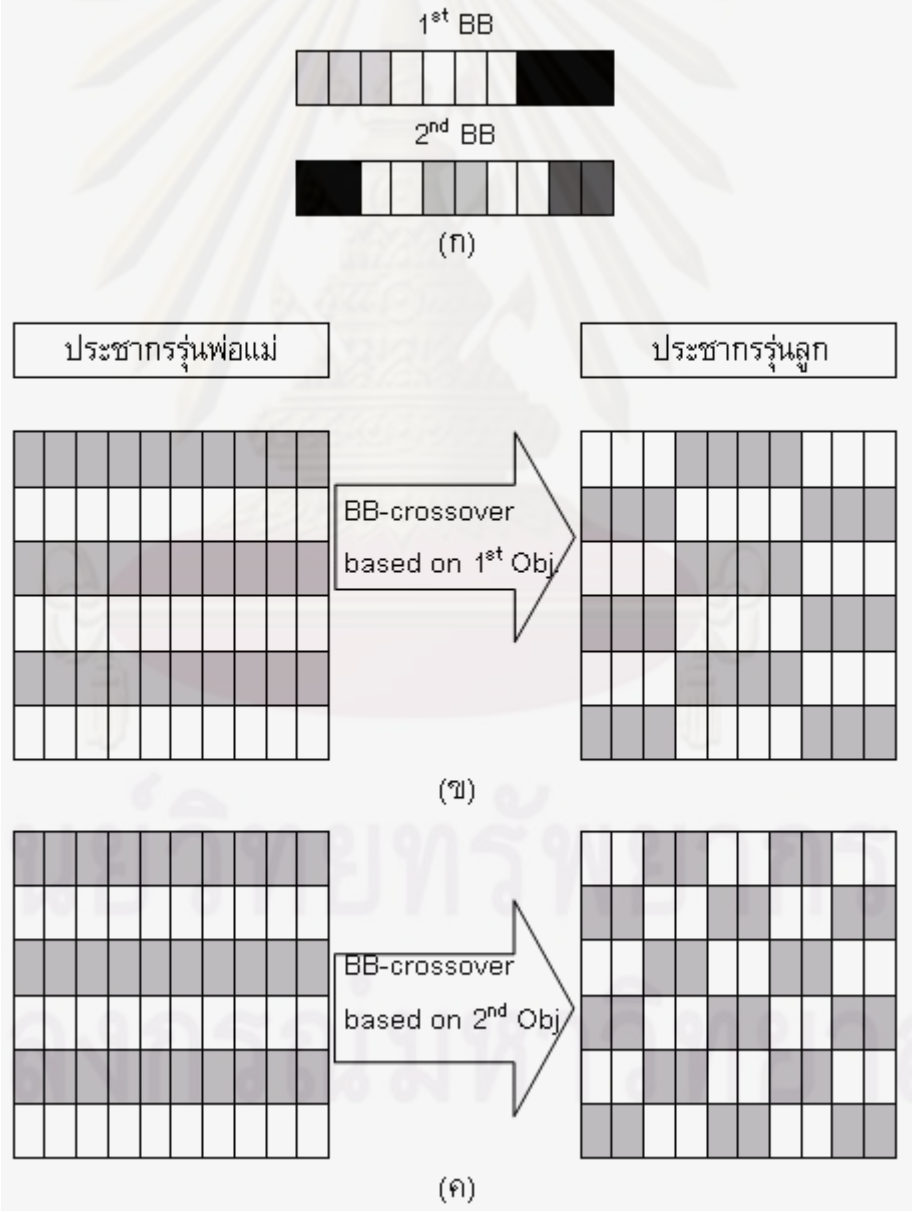
ตัวอย่างการรวมหน่วยการสร้างแต่ละจุดประสงค์แบบขยายได้แสดงไว้ในรูปที่ 28



รูปที่ 28 การรวมหน่วยการสร้างแต่ละจุดประสงค์แบบขยาย

4.1.2.4 ขั้นตอน NSGA-II ร่วมกับ BB-wise แบบที่ 4 (NSGA-IId)

การสร้างคำตอบในประชากรรุ่นถัดไปครึ่งหนึ่งจะสร้างมาจากการไขว้เปลี่ยน (Crossover) ตามแนวทางของหน่วยการสร้างในจุดประสงค์แรก และอีกครึ่งหนึ่งได้จากการไขว้เปลี่ยนยึดตามหน่วยการสร้างของจุดประสงค์ที่สอง แต่กระบวนการเลือกคำตอบก็ยังคงได้จากการสุ่มโดยไม่พิจารณาแยกจุดประสงค์ คือสุ่มเลือกจากประชากรทั้งหมด รูปที่ 29 แสดงลักษณะการทำงาน ส่วนที่เป็นสีดำและสีเทาใน (ก) คือหน่วยการสร้างที่ระบุโดยขั้นตอนวิธีที่เสนอ ผลการใช้งานปรากฏใน (ข) แสดงการสร้างประชากรรุ่นใหม่ที่ใช้หน่วยการสร้างของคำตอบที่ดีในจุดประสงค์แรก และ (ค) เป็นภาพแสดงการสร้างประชากรรุ่นใหม่ที่ใช้หน่วยการสร้างของคำตอบที่ดีในจุดประสงค์ที่สอง



รูปที่ 29 การรวมโดยอาศัยหน่วยการสร้างแต่ละจุดประสงค์แยกกัน

ขั้นตอนวิธีนี้มีรหัสเทียมการทำงานดังแสดงในรูปที่ 30 โดยจะไม่มีขั้นตอนการรวมหน่วยการสร้างแต่จะสร้างประชากรใหม่เท่าๆ กันจากหน่วยการสร้างของแต่ละจุดประสงค์

```
1:  R_t = P_t U Q_t
2:  F = fast-nondominated-sort(R_t)
3:  P_{t+1} = ∅ and i = 1
4:  until P_{t+1} + F_i ≤ N
5:    crowding-distance-assignment(F_i)
6:    P_{t+1} = P_{t+1} ∪ F_i
7:    i = i + 1
8:  Sort(F_i, ≺_n)
9:  P_{t+1} = P_{t+1} ∪ F_i[1 : (N – P_{t+1})]
10: Q_{t+1} = ∅
11: for each objective j
12:    S_j = SortByCurrentObjective(P_{t+1})
13:    H_j = selectBetterHalf(S_j)
14:    BB_j = Building-Block-Identification(H_j)
15:    Q_{t+1} = Q_{t+1} ∪ make-new-popBB(P_{t+1})
16: end for
17: t = t + 1
```

รูปที่ 30 รหัสเทียมขั้นตอนวิธี NSGA-II ร่วมกับ BB-wise แบบที่ 4

ปัญหาที่ใช้ในการทดลองและผลการทดลองเปรียบเทียบขั้นตอนวิธีที่นำเสนอมีรายละเอียดอยู่ในบทถัดไป

บทที่ 5

การทดลอง

ในบทนี้มีรายละเอียดเกี่ยวกับขั้นตอนวิธีที่ใช้เปรียบเทียบ ปัญหาที่ใช้ในการทดลอง และผลการทดลองพร้อมทั้งวิเคราะห์ผล ปัญหาที่ใช้ในการทดลองเป็นปัญหาในกลุ่มปัญหาหลอก (Deceptive Problems) ที่ประกอบกันเป็นปัญหาหลายจุดประสงค์ ได้กล่าวถึงขั้นตอนวิธีที่ใช้เปรียบเทียบอย่างย่อไว้ดังนี้

5.1 ขั้นตอนวิธีที่ใช้เปรียบเทียบ

จากการศึกษาขั้นตอนวิธีที่ใช้แก้ปัญหาหลายจุดประสงค์ พบว่ามีขั้นตอนวิธีที่ใช้หน่วยการสร้างอยู่ไม่มากนัก ขั้นตอนวิธีที่เคยมีรายงานผลการทดลองแก้ปัญหาที่สนใจได้แก่ ขั้นตอนวิธี mBOA, MOMGA-II และ MOMGA-IIa ขั้นตอนวิธีเหล่านี้ได้มีผู้ทดลองเปรียบเทียบผลไว้ในงานวิจัย [4] จึงนำมาเปรียบเทียบผลกับขั้นตอนวิธีที่นำเสนอ

ปัญหาที่ใช้ในการทดลองอยู่ในหัวข้อ 5.2 เป็นปัญหาที่มีหน่วยการสร้างในพาเรโตเด่นชัด โดยคำตอบจะอยู่ในลักษณะก้อนของหน่วยการสร้าง (เช่น 00000 และ 11111 ในกรณีหน่วยการสร้างมีขนาด 5 บิต) ปัญหาเหล่านี้มีการวิเคราะห์คำตอบเหมาะสุดแบบพาเรโตและปริมาณคำตอบที่แตกต่างกันในพาเรโตแล้ว ดังแสดงในตารางที่ 2 ผลการวิเคราะห์เหล่านี้ มีความสำคัญมาก เนื่องจากคำตอบในพาเรโตฟรอนต์ของปัญหาหลายจุดประสงค์ ไม่ได้มีความสัมพันธ์อย่างตรงไปตรงมากับคำตอบของปัญหาในแต่ละจุดประสงค์ มีความเป็นไปได้ที่จะมีคำตอบเหมาะสุดเพียงคำตอบเดียวหรือทุกรูปแบบที่เป็นไปได้ของคำตอบทั้งหมดสามารถอยู่ในเซตคำตอบแบบพาเรโตได้ เช่นปัญหา one-max_zero-max [6] ทุกคำตอบที่เป็นไปได้อยู่ในเซตคำตอบแบบพาเรโต ในขณะที่แต่ละจุดประสงค์มีคำตอบเหมาะสุดเพียงคำตอบเดียว ดังนั้นการออกแบบปัญหาที่ใช้ทดสอบ ส่วนหนึ่งที่สำคัญคือ รู้คำตอบเหมาะสุดของปัญหานั้นหรือไม่ ปัญหา MOP4 ที่ออกแบบไว้สามารถวิเคราะห์หาฟังก์ชันที่ใช้ทดสอบจุดที่เหมาะสุดแบบพาเรโตได้ การขยับเพียงแค่ 1 ตำแหน่งของบิตในจุดประสงค์ใดจุดประสงค์หนึ่ง อาจจะทำให้หน้าตาคำตอบแบบพาเรโตเปลี่ยนไปทันที

ขั้นตอนวิธีที่ใช้ในการทดลองซึ่งยังไม่มีการระบุหน่วยการสร้าง คือ NSGA-II ในการทดลองได้นำขั้นตอนวิธี NSGA-II มาปรับปรุงเพิ่มเติมขั้นตอนสำหรับระบุหน่วยการสร้าง ซึ่งจะมีการระบุหน่วยการสร้างทั้งหมด 4 แบบ (ดังแสดงรายละเอียดไว้ในบทที่ 4) และได้ทดลองเปรียบเทียบขั้นตอนวิธี NSGA-II กับขั้นตอนวิธี NSGA-II ที่มีการระบุหน่วยการสร้างทั้ง 4 แบบ

5.1.1 ขั้นตอนวิธี mBOA

ขั้นตอนวิธี mBOA [48] มีรหัสเทียมดังแสดงในรูปที่ 31 และมีการทำงานเหมือนกันกับ BOA [12] ยกเว้นส่วนของการคัดเลือกซึ่งจะใช้กระบวนการคัดเลือกและการจัดเรียงเพื่อหาลำดับของการครอบงำแบบเดียวกับ NSGA-II คือ mBOA จะทำการรวมประชากรใหม่และประชากรรุ่นพ่อแม่เข้าด้วยกันแล้วจัดเรียงหาลำดับแบบพาเรโต ทำการเลือกโดยอาศัยลำดับแบบพาเรโตและระยะเบียดเพื่อนำกลับมาสร้างเป็นแบบจำลอง (model) ของคำตอบเพื่อเป็นตัวแทนของคำตอบและใช้แบบจำลองนี้สุ่มสร้างคำตอบใหม่ จากนั้นคัดเลือกคำตอบดี ๆ เพื่อนำกลับมาสร้างเป็นแบบจำลองอีก ทำเช่นนี้วนไปเรื่อย ๆ จนได้จำนวนรุ่นประชากรตามที่กำหนด ขั้นตอนวิธีนี้จะสร้างแบบจำลองของแต่ละจุดประสงค์แยกกัน

```
mBOA algorithm
1:      procedure mBOA(N, g, f_k(x))
2:          initialize Population P' where |P'| = N
3:          Begin
4:              Generate random population – size N'
5:              Evaluate Objective Values
6:              Assign Rank (level) based on Pareto dominance - sort
7:              Generate Child Population
8:                  Binary Tournament Selection
9:                  Recombination and Mutation
10:         End
11:         for t = 1 to g do
12:             for each Parent and Child in Population do
13:                 Assign Rank (level) based on Pareto – sort
14:                 Generate layers of sets vectors that are non-dominated
15:                 Loop (inside) by adding solutions to next generation starting from
                        the first front until N' individuals found determine crowding
                        distance between points on each front
16:             end for
17:             Select points (elitist) on the lower front (with lower rank) and are
                    outside a crowding distance
18:             Create next generation
19:             Binary Tournament Selection
20:             Recombination and Mutation
21:         end for
22:     end procedure
```

รูปที่ 31 รหัสเทียมขั้นตอนวิธี mBOA

5.1.2 ขั้นตอนวิธี MOMGA-II และ MOMGA-IIa

รูปที่ 32 แสดงรหัสเทียมของขั้นตอนวิธี MOMGA-II [58] ซึ่งปรับปรุงจาก MOMGA และ fast-messy GA [28] การทำงานแบ่งออกเป็น 3 ขั้นตอนคือ 1.การกำหนดค่าเริ่มต้น 2. การกรองหน่วยการสร้าง (Building Block Filtering) 3. ขั้นการวางเคียง ข้อแตกต่างจาก MOMGA ที่สำคัญอยู่ใน 2 ส่วนแรก คือปรับจากที่ใช้กรรมวิธีเชิงกำหนดเป็นการกำหนดค่าเริ่มต้นบริบูรณ์เชิงสุ่ม (Probabilistically Complete Initialization: PCI) และขั้นการกรองหน่วยการสร้างจะลดจำนวนหน่วยการสร้างด้วยกระบวนการกรอง (Filtering Process) และเก็บหน่วยการสร้างที่ดีที่สุดที่เคยเจอ สำหรับขั้นการวางเคียงเหมือนกันกับ MOMGA

```
MOMGA-II algorithm
1: procedure MOMGA-II(f_k( x⃗ ))
2:     for h = 1 to epoch do
3:         ;  PCI Phase
4:         Perform Probabilistically Complete initialization
5:         Evaluate each pop member's fitness w.r.t.k templates
6:         ;  BB Filtering (BBF) Phase
7:         for i = 1 to Max BBF generations do
8:             if BBF schedule requires cutting at this generation then
9:                 Perform BBF
10:            else
11:                Perform Tournament Thresholding Selection
12:            end if
13:        end for
14:        ;  Juxtapositional Phase
15:        for i = 1 Max Juxtapositional generations do
16:            Cut-and-Splice
17:            Evaluate each population member's fitness w.r.t.k templates
18:            Perform Tournament Thresholding Selection and fitness Sharing
19:            PKnown(t) = Pcurrent(t) ∪ Pknown(t - 1)
20:        end for
21:        Update k templates ; Using best known value in each objective
22:    end for
23: end procedure
```

รูปที่ 32 รหัสเทียมขั้นตอนวิธี MOMGA-II

MOMGA-IIa [4] [59] เป็นงานถัดจาก MOMGA-II อย่างไรก็ดีทั้งสองขั้นตอนวิธีต้องการค่าพารามิเตอร์สำคัญคือ ค่า k ซึ่งจะเป็นขนาดหน่วยการสร้างที่ต้องกำหนดโดยผู้ใช้เพื่อนำไปใช้ในขั้นตอนกำหนดค่าเริ่มต้น คือ การสร้างคำตอบที่มียีนในทุกรูปแบบที่เป็นไปได้ตามขนาด k ที่กำหนด ทำให้ขนาดประชากรเริ่มต้นมีขนาดใหญ่มาก จึงต้องมีขั้นตอนการกรองคำตอบเหล่านี้อีกทีหนึ่ง รหัสเทียมของ MOMGA-IIa แสดงไว้ในรูปที่ 33

```
1.procedure MOMGA-IIa(f_k(x⃗))
2:      for h = 1 to epoch do
3:          ;   PCI Phase
4:          Perform Probabilistically Complete initialization
5:          Evaluate each pop member's fitness w.r.t.(k * r + i + o) templates
6:          ;   BB Filtering (BBF) Phase
7:          for i = 1 to Max BBF generations do
8:          if BBF schedule requires cutting at this generation then
9:              Perform BBF
10:         else
11:             Perform Tournament Thresholding Selection
12:         end if
13:     end for
14:         ;   Juxtapositional Phase
15:         for i = 1 Max Juxtapositional generations do
16:             Cut-and-Splice
17:             Evaluate each pop member's fitness w.r.t.(k * r +i + o) templates
18:             Perform Tournament Thresholding Selection and fitness Sharing
19:             PKnown(t) = Pcurrent(t) ∪ Pknown(t-1)
20:         end for
21:         Update k * r templates  ;Using the Competitive Template Management
                System
22:         Filter i and o templates based on k * r updated templates
23:     end for
24: end procedure
```

รูปที่ 33 รหัสเทียมขั้นตอนวิธี MOMGA-IIa

ขั้นตอนวิธี MOMGA-II และ MOMGA-IIa มีการระบุหน่วยการสร้างแบบ messyGA คือ เข้ารหัสคำตอบเป็นคู่ลำดับ (*p*,*v*) โดยที่ *p* คือตำแหน่งหรือชื่อของบิต *v* คือค่าของบิต ทำให้สามารถย้ายตำแหน่งของบิตไปได้เรื่อย ๆ โดยไม่เสียความหมาย เวลาผสมคำตอบจะทำให้มีกรณีตำแหน่งเกินกำหนด (over-specified) คือตำแหน่งที่ค่าบิตมากกว่า 1 ค่า และมีตำแหน่งต่ำกว่ากำหนด (under-specified) คือไม่มีค่าของบิตในตำแหน่งนั้น มีหลายแนวทางในการตีความตำแหน่งเหล่านี้ ตัวอย่างเช่น ตำแหน่งเกินกำหนดจะใช้ค่าของบิตที่มีจำนวนมากกว่า (Majority Voting) หรือจะเลือกค่าที่เจอก่อน (First Come, First Serve) ส่วนบิตที่ต่ำกว่ากำหนดจะเลือกจากค่าเฉลี่ยของแผ่นแบบแข่งขัน (competitive templates)

เพื่อให้สามารถแก้ปัญหาหลายจุดประสงค์ ทั้งสองขั้นตอนวิธีนี้จะทำการเพิ่มจำนวนแผ่นแบบแข่งขันตามจำนวนจุดประสงค์ ส่วนสำคัญที่เพิ่มเข้ามาของขั้นตอนวิธี MOMGA-IIa คือ ระบบจัดการแผ่นแบบแข่งขัน เนื่องจากมีแผ่นแบบแข่งขันแบ่งเป็น 3 ประเภทในแต่ละจุดประสงค์ ได้แก่ *r* (regular) แบบปกติ, *i* (inverse) ค่าผกผันของแบบปกติ และ *o* (orthogonal) เป็นผลกรองการสุ่มแผ่นแบบปกติแบบตั้งฉาก ทั้งหมดนี้เพื่อให้มีคำตอบการกระจายตัวในรูปแบบยีน (genotype) ด้วย

5.1.3 ขั้นตอนวิธี NSGA-II

ขั้นตอนวิธี NSGA-II [5] มีรายละเอียดของขั้นตอนวิธีอยู่ในหัวข้อ 4.1.1 ขั้นตอนวิธีนี้ถูกเลือกมาปรับปรุงการไขว้เปลี่ยนให้เป็นแบบอาศัยหน่วยการสร้าง เนื่องจากได้รับความนิยมแพร่หลาย มีขั้นตอนการทำงานง่ายรวดเร็ว และไม่จำเป็นต้องมีหน่วยความจำเพิ่มเติมภายนอกที่เรียกว่า หน่วยเก็บถาวร (archive) นอกจากนี้ในงานวิจัยนี้มุ่งเน้นศึกษาผลกระทบจากการปรับปรุงตัวดำเนินการไขว้เปลี่ยน ซึ่งจะสามารถนำไปใช้กับขั้นตอนวิธีเชิงพันธุกรรมอื่นที่มีตัวดำเนินการไขว้เปลี่ยนได้เช่นกัน

5.2 ปัญหาที่ใช้ในการทดลอง

Coello Coello และคณะ [3] ได้แบ่งแยกปัญหาสำหรับทดสอบขั้นตอนวิธีแก้ปัญหาหลายจุดประสงค์ออกเป็น 6 กลุ่มด้วยกัน

1 ฟังก์ชันเพิ่มขยายจากจุดประสงค์เดียว (Extension of Single Objective Functions) [66] [67]

2 ฟังก์ชันหลายจุดประสงค์แบบไม่มีเงื่อนไขบังคับ (Unconstrained Multiobjective Functions) [68] [69]

3 ฟังก์ชันหลายจุดประสงค์แบบมีเงื่อนไขบังคับ (Constrained Mutiobjective Functions) [5] [70]

4 ตัวสร้างฟังก์ชันหลายจุดประสงค์ (Multiobjective Function Generators) [71]

5 ฟังก์ชันทดสอบปัญหาจัดหมู่ (Combinatorial Multiobjective Test Functions) [72]

6 ฟังก์ชันปัญหาในโลกจริง (Real World Multiobjective Test Funtions) [73]

ฟังก์ชันที่ใช้ทดลองในวิทยานิพนธ์นี้เป็นแบบไม่มีเงื่อนไข (Unconstrained), เป็นปัญหาจัดหมู่ (Combinatorial) และเป็นส่วนขยายของปัญหาจุดประสงค์เดียว (Extension of Single Objective) ปัญหาที่ทดลองถูกออกแบบให้เป็น 2 ขั้วตรงข้ามของปัญหาแบ่งแยกย่อยได้แบบมีความยากจำกัด (boundedly difficult problems) คือให้จุดประสงค์หนึ่งมีค่าเหมาะสุดเป็น 1 หนึ่งหมดทุกบิต และอีกจุดประสงค์หนึ่งมีค่าเหมาะสุดเป็น 0 หมดทุกบิต ปัญหาที่ได้ทำการทดลองได้แก่

(1) T1 – Interleaved minimal deception problem

(2) T2 – Complement of T1

(3) T3 – Interleaved 5-bit Trap function

(4) T4 – Complement of T3

(5) T5 – Interleaved symmetric 5-bit Trap

(6) T6 – Complement of T5

(7) T7 – Shift of T6

ปัญหา T1 ถึง T7 ได้ถูกนำมารวมกันเป็นคู่ๆ เพื่อเป็นปัญหาหลายจุดประสงค์ดังแสดงในตารางที่ 2

ตารางที่ 2 ปัญหาหลายจุดประสงค์ที่ได้ทำการทดลอง

ปัญหา	คำอธิบาย	ขนาดของปัญหา	จำนวนจุดในพาเรโต	จำนวนคำตอบที่แตกต่างกันในพาเรโต
MOP1	T1 & T2 (MDP)	30	16	2^{15}
		60	31	2^{30}
		90	46	2^{45}
MOP2	T3 & T4 (Trap-5)	30	7	2^{6}
		60	13	2^{12}
		90	19	2^{18}
MOP3	T5 & T6 (Nonoverlap-Trap5)	30	7	2^{6}
		50	11	2^{10}
		60	13	2^{12}
MOP4	T5 & T7 (Overlapping-Trap5)	30	2	
		50	5	
		60	7	

ตารางที่ 3 แสดงจำนวนคำตอบที่แตกต่างกันในพาเรโตฟรอนต์ของปัญหา MOP2 และ MOP3 เมื่อ m เป็นจำนวนหน่วยการสร้าง ตัวอย่างเช่น ในปัญหาขนาด 30 บิต จะมีจำนวนหน่วยการสร้างทั้งหมด 6 หน่วย แต่ละหน่วยมีขนาด 5 บิต

ตารางที่ 3 จำนวนคำตอบในแต่ละจุดของปัญหา MOP2 และ MOP3

#BB จุดประสงค์ที่ 1	0	1	…	i	…	m
#BB จุดประสงค์ที่ 2	m	m -1	…	m-i	…	0
จำนวนคำตอบ	$\binom{m}{0}$	$\binom{m}{1}$	…	$\binom{m}{i}$	…	$\binom{m}{m}$

รายละเอียดของแต่ละปัญหาแสดงอยู่ในหัวข้อถัดไป

5.2.1 MOP1 - Interleaved minimal deception problem (T1 & T2)

ฟังก์ชันหลอกน้อยสุด (Minimal Deceptive Function) เป็นฟังก์ชันขนาด 2 บิต ปัญหา T1 มีสมการดังนี้

$$g_{MDP}(x) = \begin{cases} 1.0 \text{ if } u = 2 \\ 0.9 \text{ if } u = 0 \\ 0.0 \text{ if } u = 1 \end{cases}$$

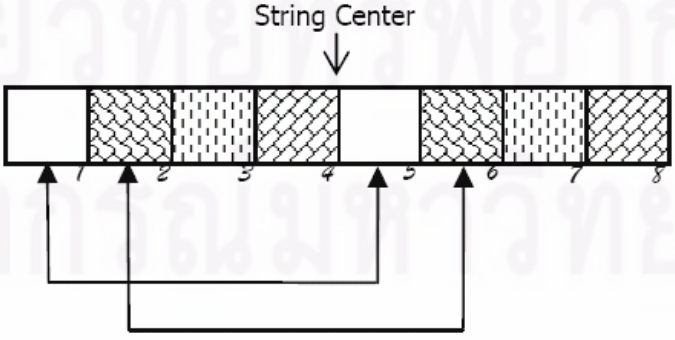
เมื่อ u เป็นผลรวมของสองบิต

ฟังก์ชันนี้ประกอบด้วยความสัมพันธ์ของ 2 ตัวแปร T2 เป็นฟังก์ชันผกผันของฟังก์ชันหลอกน้อยสุด มีสมการเป็นดังนี้

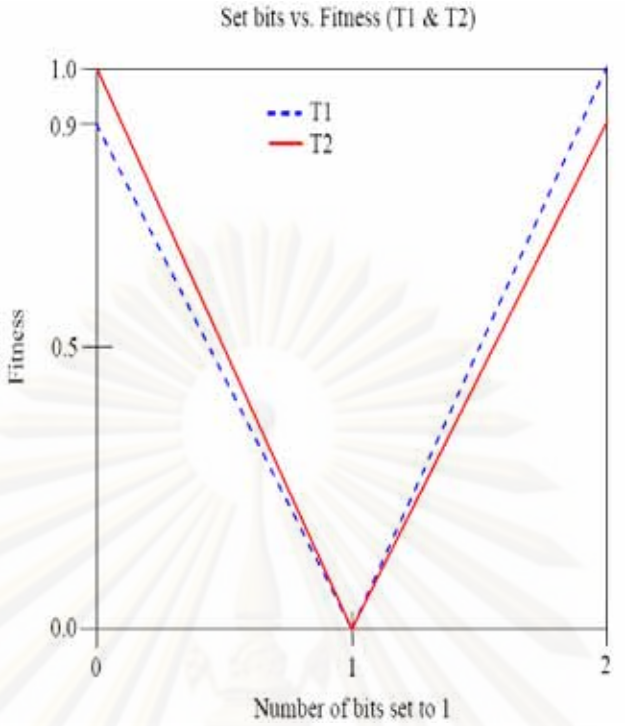
$$g'_{MDP}\, x = g_{MDP}(\bar{x})$$

ปัญหานี้ใช้สำหรับทดสอบการเชื่อมโยงแบบหลวม (loosely linkage) ซึ่งได้มาจากการแบ่งสายอักขระของปัญหาออกเป็น 2 ฝั่ง แต่ละบิตในฝั่งหนึ่งจะถูกจับคู่กับบิตในฝั่งตรงข้าม ดูรูปที่ 34 ช่องสี่เหลี่ยมจตุรัสแทนแต่ละบิต ช่องที่มีลวดลายเหมือนกันถูกเชื่อมโยงเข้าด้วยกัน กำหนดให้ l เป็นขนาดของปัญหา (ความยาวของโครโมโซม) บิตที่ i จะถูกเชื่อมโยงกับบิตที่ $(l/2 +1)^{th}$ โดยที่ $i \leq l/2$ และกำหนดให้บิตแรกเป็นบิตที่ 1

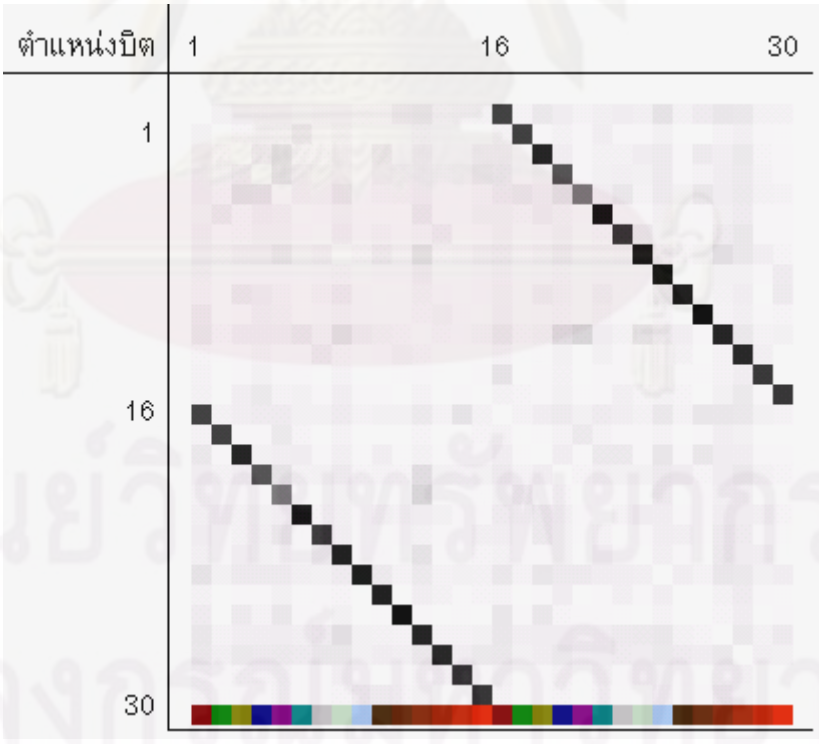
รูปที่ 35 แสดงค่าความเหมาะสมของปัญหา T1 และ T2 ของ 1 คู่บิตที่เชื่อมโยงกัน ในปัญหา T1 ถ้าค่าของบิตเป็น 1 เหมือนกันจะได้ค่าความเหมาะสมเป็น 1.0 แต่ถ้าเป็น 0 เหมือนกัน จะได้ค่าความเหมาะสมเป็น 0.9 ถ้ามีค่าของบิตต่างกันจะมีค่าความเหมาะสมเป็น 0.0 นั่นคือ ถ้าบิตมีค่าเหมือนกันจะได้รับรางวัลแต่ถ้าต่างกันจะไม่ได้อะไร ในปัญหา T2 คล้ายกับ T1 แต่จะสลับค่าความเหมาะสมกรณีที่บิตมีค่าเหมือนกัน



รูปที่ 34 ความเชื่อมโยงในปัญหาหลอกน้อยสุดที่เชื่อมโยงแบบหลวม



รูปที่ 35 ค่าความเหมาะสมของฟังก์ชัน T1 และ T2



รูปที่ 36 ตัวอย่างปริมาณไคกำลังสองและการแบ่งกลุ่มของปัญหา T1 & T2

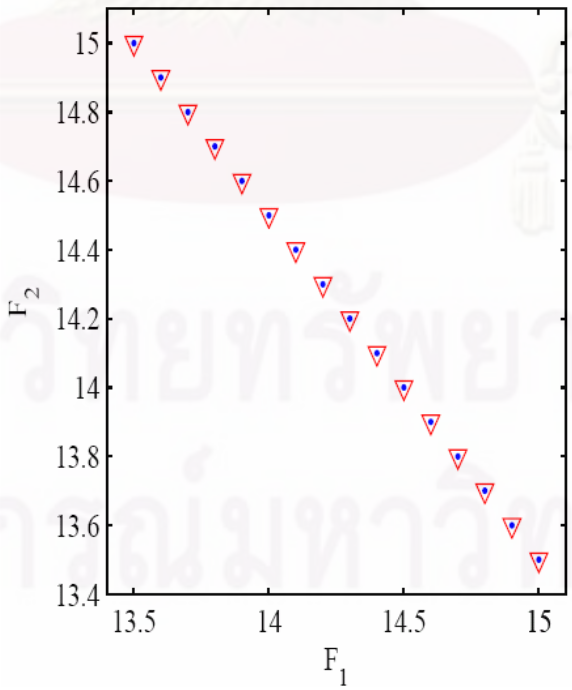
ช่องสี่เหลี่ยมสีเทาและดำด้านบนแต่ละช่องแสดงปริมาณไคกำลังสองของแต่ละบิต แถบสีด้านล่างแสดงถึงกลุ่มความสัมพันธ์หรือหน่วยการสร้าง บิตที่อยู่ในกลุ่มเดียวกันจะมีสีเหมือนกัน ในรูปจะเห็นได้ว่าแถบสีช่องที่ 1 มีความเข้มเท่ากันกับแถบสีช่องที่ 16 แถบสีช่องที่ 2 มีความเข้มเท่ากันกับช่องที่ 17 เรียงลำดับถัดกันไปเรื่อยๆ จะได้การแบ่งกลุ่มแต่ละบิตเป็นดังนี้

Partition = {1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 }

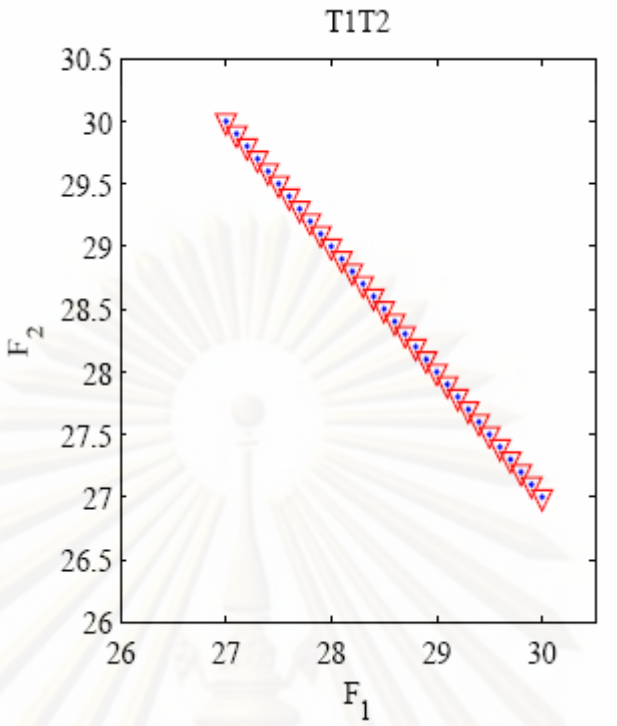
บิตในตำแหน่งที่ 1 กับบิตในตำแหน่งที่ 16 อยู่ในกลุ่มเดียวกัน มีหมายเลขเดียวกันคือหมายเลข 1 และเช่นเดียวกันหมายเลข 2 อยู่ในตำแหน่งที่ 2 และ 17 แสดงว่าบิตที่ 2 และบิตที่ 17 อยู่ในกลุ่มเดียวกัน จากการแบ่งส่วนด้านบนแปลความหมายเป็นหน่วยการสร้างได้ดังนี้

BB = { {1,16} {2,17} {3,18} {4,19} {5,20} {6,21} {7,22} {8,23} {9,24}
{10,25} {11,26} {12,27} {13,28} {14,29} {15,30} }

จุดที่เป็นคำตอบในพาเรโตฟรอนต์ของปัญหา MOP1 ขนาด 30 บิต และ 60 บิต แสดงไว้ในรูปที่ 37 และรูปที่ 38 ตามลำดับ



รูปที่ 37 จุดในพาเรโตฟรอนต์ของปัญหา MOP1 ขนาด 30 บิต



รูปที่ 38 จุดในพาเรโตฟรอนต์ของปัญหา MOP1 ขนาด 60 บิต

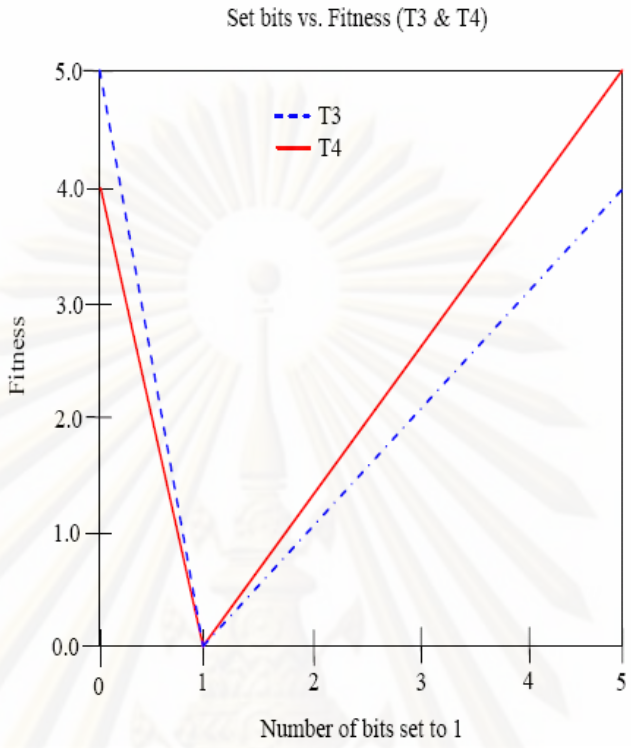
5.2.2 MOP2 - Interleaved 5-bit trap function (T3 & T4)

ปัญหานี้เป็นปัญหาหลอก [74] ใช้สำหรับทดสอบความสามารถในการหาหน่วยการสร้างหลวมที่บิตไม่อยู่ต่อเนื่องกัน บิตที่อยู่ในหน่วยการสร้างเดียวกันจะห่างกันไปทีละ $l/5$ บิต เมื่อ l เป็นขนาดของปัญหา รูปที่ 39 แสดงการเรียงตำแหน่งของบิตในกลุ่ม 5 บิต ช่องที่มีลวดลายเหมือนกันเป็นบิตที่อยู่ในหน่วยการสร้างเดียวกัน นั่นคือบิตที่ i กับ $i+ l/5$, $i+2l/5$, $i+ 3l/5$, และ $i+4l/5$ อยู่ในกลุ่มเดียวกันโดยที่ $i< l/5$ กราฟในรูปที่ 40 แสดงค่าความเหมาะสมของปัญหานี้ตามจำนวนบิตที่มีค่าเป็น 1 จากจำนวน 5 บิต

รูปที่ 39 การเรียงตำแหน่งของบิตในปัญหา T3 และ T4

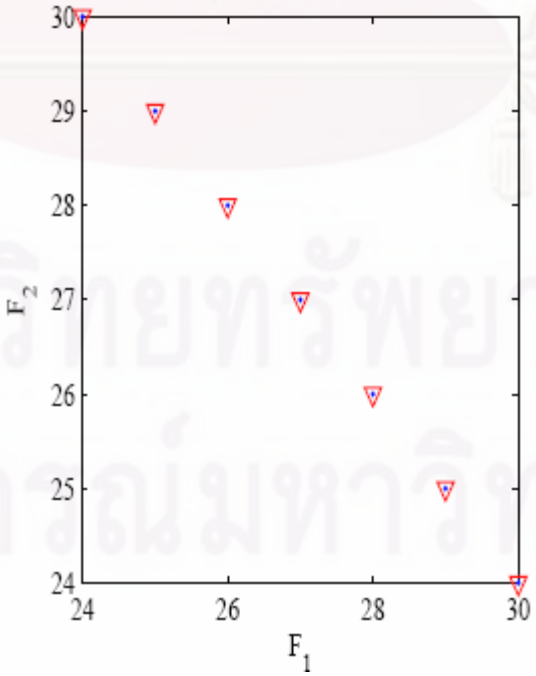
ปัญหานี้มีค่าในแต่ละจุดประสงค์ไม่สมมาตรกัน เมื่อนำไปออกแบบให้เป็นปัญหาที่มีหน่วยการสร้างเหลื่อมกันจะได้ปัญหาที่มีพาเรโตที่กระจัดกระจายและไม่สามารถทำการวิเคราะห์หาจุดที่อยู่ในพาเรโตฟรอนท์ได้สะดวกในปัญหาขนาดใหญ่ เนื่องจากต้องหาทุกกรณีที่

เป็นไปได้ และไม่สามารถทำได้ทันในกำหนดเวลา ดังนั้นจึงได้หันไปใช้ปัญหาที่มีค่าจุดประสงค์สมมาตรกันที่จะเสนอในหัวข้อถัดไป

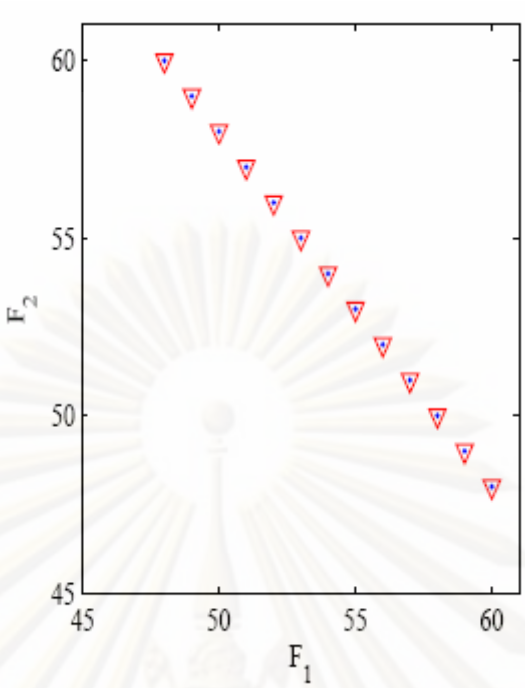


รูปที่ 40 ค่าความเหมาะสมของฟังก์ชัน T3 และ T4

จุดที่เป็นคำตอบในพาเรโตฟรอนต์ของปัญหา MOP2 ขนาด 30 บิต และ 60 บิต แสดงไว้ในรูปที่ 41และรูปที่ 42 ตามลำดับ จะเห็นว่ามีจำนวนจุดน้อยลงจากปัญหา MOP1



รูปที่ 41 จุดในพาเรโตฟรอนต์ของปัญหา MOP2 ขนาด 30 บิต



รูปที่ 42 จุดในพาเรโตฟรอนต์ของปัญหา MOP2 ขนาด 60 บิต

5.2.3 MOP3 - Interleaved symmetric 5-bit Trap (T5 & T6)

ปัญหานี้ได้ถูกนำมาทดสอบประสิทธิภาพของขั้นตอนวิธีเชิงวิวัฒน์ในงานวิจัย [6] และเป็นกลุ่มของฟังก์ชันที่เรียกว่า ฟังก์ชันค่าความเหมาะสมหลอก (Deceptive Fitness Function) [62] [8] ที่หลอกให้อัลกอริทึมค้นหาคำตอบผิดทาง ปัญหากับดักและปัญหากับดักผกผัน (ปัญหากับดักหนึ่งและกับดักศูนย์) มีสมการดังนี้

$$\text{Trapk(u)} = \begin{cases} \text{k} & \text{; if u = k ,} \\ (k-d)\left[1-\dfrac{u}{k-1}\right] & \text{; otherwise} \end{cases}$$

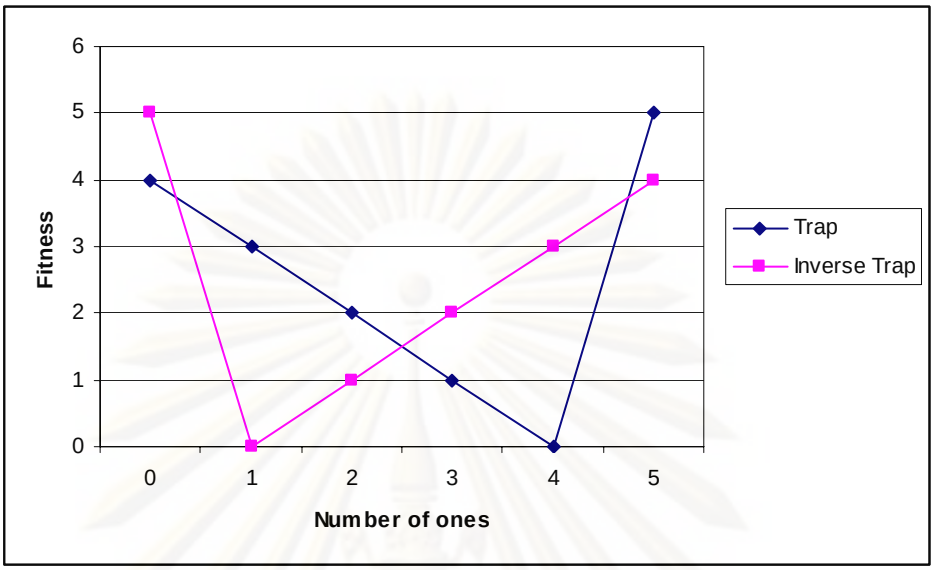
$$\text{Invtrapk(u)} = \begin{cases} \text{k} & \text{; if u = 0 ,} \\ (k-d)\left[\dfrac{u-1}{k-1}\right] & \text{; otherwise} \end{cases}$$

เมื่อ k เป็นขนาดของกับดัก

u เป็นจำนวนของหนึ่ง

d เป็นความแตกต่างของสัญญาณ

กราฟในรูปที่ 43 แสดงค่าความเหมาะสมเมื่อ *k* เท่ากับ 5 และ *d* เท่ากับ 1



รูปที่ 43 ค่าความเหมาะสมของฟังก์ชัน T5 และ T6

ปัญหากับดัก (Trap) สามารถนำมาเรียงต่อกันหลายชุดให้เป็นปัญหาที่ขนาดใหญ่ขึ้น รูปที่ 44 แสดงตัวอย่างการคำนวณค่าความเหมาะสมของคำตอบเมื่อ *k* มีค่าเป็น 5

คำตอบ	11111	11110	11100	00001	00000
f_{Trap}	5	0	1	3	4
$f_{Inverse\ Trap}$	4	3	2	0	5

$f_{Trap} = 5 + 0 + 1 + 3 + 4$ $f_{InverseTrap} = 4 + 3 + 2 + 0 + 5$

รูปที่ 44 ตัวอย่างการคำนวณค่าความเหมาะสมของปัญหา Trap และ InverseTrap

หน่วยการสร้างของปัญหานี้ไม่มีการเหลื่อมกันในแต่ละจุดประสงค์ดังนี้

จุดประสงค์ที่ 1 11111 22222 3333 44444 55555 66666

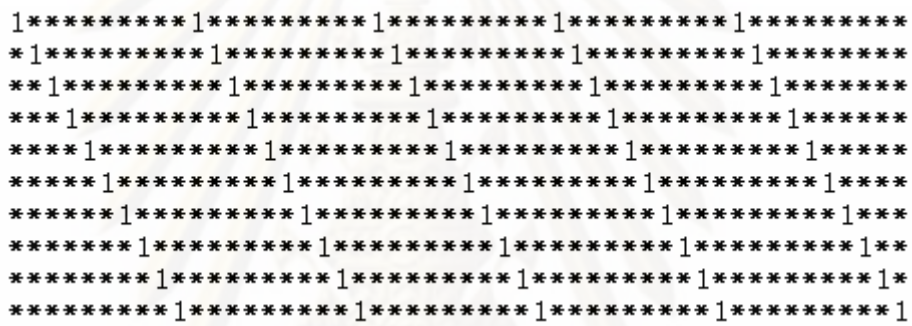
จุดประสงค์ที่ 2 11111 22222 3333 44444 55555 66666

แต่ปัญหาที่ทำการทดลองจะมีหน่วยการสร้างแบบหลวมซึ่งแต่ละบิตของหน่วยการสร้างเดียวกันจะห่างจากบิตถัดกันเท่ากับ l/m เมื่อ l เป็นความยาวของคำตอบ และ m เป็นจำนวนหน่วยการสร้าง การเรียงตำแหน่งจึงเปลี่ยนเป็นดังนี้

จุดประสงค์ที่ 1 123456 123456 123456 123456 123456

จุดประสงค์ที่ 2 123456 123456 123456 123456 123456

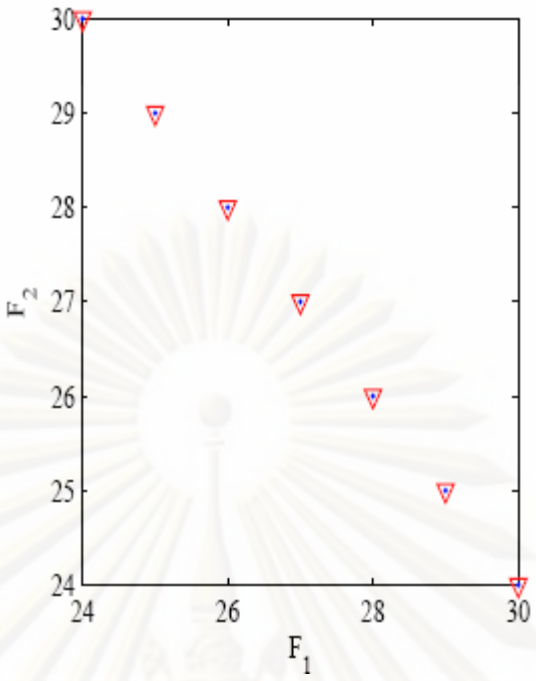
รูปที่ 45 แสดงการเรียงตำแหน่งบิตของแต่ละหน่วยการสร้างของปัญหา MOP3 ขนาด 50 บิต 1 แถวคือ 1 หน่วยการสร้าง จะเห็นว่าแต่ละบิตจะห่างจากบิตข้างๆ เท่าๆ กันคือ 50/5 = 10 บิต ทั้งสองจุดประสงค์มีตำแหน่งหน่วยการสร้างตรงกัน



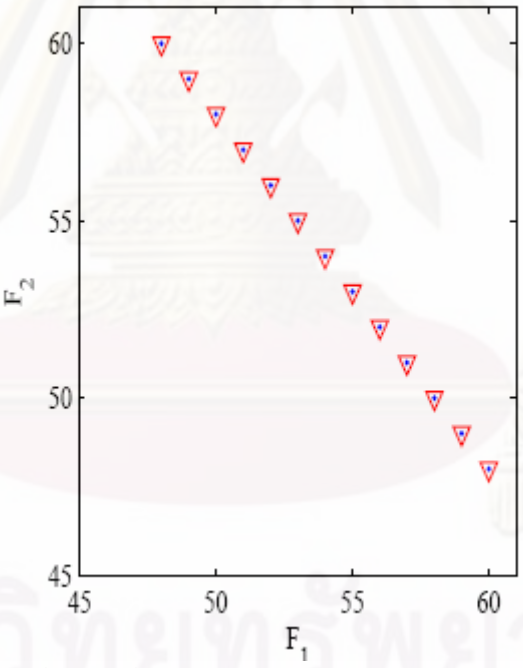
รูปที่ 45 การเรียงตำแหน่งบิตของแต่ละหน่วยการสร้างของปัญหา MOP3

ปัญหานี้มีค่าในแต่ละจุดประสงค์สมมาตรกันทำให้ได้จุดในพาเรโตฟรอนท์ที่มีรูปแบบชัดเจน แม้ว่าจะมีจุดในพาเรโตฟรอนท์ตรงกันกับปัญหาที่ผ่านมา แต่การกระจัดกระจายของจุดน้อยกว่า เมื่อนำไปออกแบบเป็นปัญหาที่มีหน่วยการสร้างของแต่ละจุดประสงค์เหลื่อมกันจะสามารถบอกได้ว่าจุดใดบ้างที่อยู่ในพาเรโตฟรอนท์ เนื่องจากมีรูปแบบการกระจายตัวแน่นอนกว่าปัญหาในหัวข้อก่อนหน้า

จุดที่เป็นคำตอบในพาเรโตฟรอนต์ของปัญหา MOP3 ขนาด 30 บิต และ 60 บิต แสดงไว้ในรูปที่ 46 และรูปที่ 47 ตามลำดับ จะเห็นว่าจำนวนจุดที่อยู่ในพาเรโตฟรอนต์ของปัญหา MOP3 เหมือนกันกับของปัญหา MOP2



รูปที่ 46 จุดในพาเรโตฟรอนต์ของปัญหา MOP3 ขนาด 30 บิต



รูปที่ 47 จุดในพาเรโตฟรอนต์ของปัญหา MOP3 ขนาด 60 บิต

5.2.4 MOP4 - Overlapping MO-Trap (T5 & T7)

ก่อนอื่นขอทำความเข้าใจไว้ในที่นี้ก่อนว่า ปัญหาที่มีการเหลื่อมซ้อนกันของหน่วยการสร้าง (Overlapping BBs) หมายถึงหน่วยการสร้างในแต่ละจุดประสงค์เหลื่อมกันอยู่ ไม่ได้ทับกันพอดี แต่ถ้าเป็นปัญหาที่ไม่มีการเหลื่อมซ้อนกันของหน่วยการสร้าง (Nonoverlapping

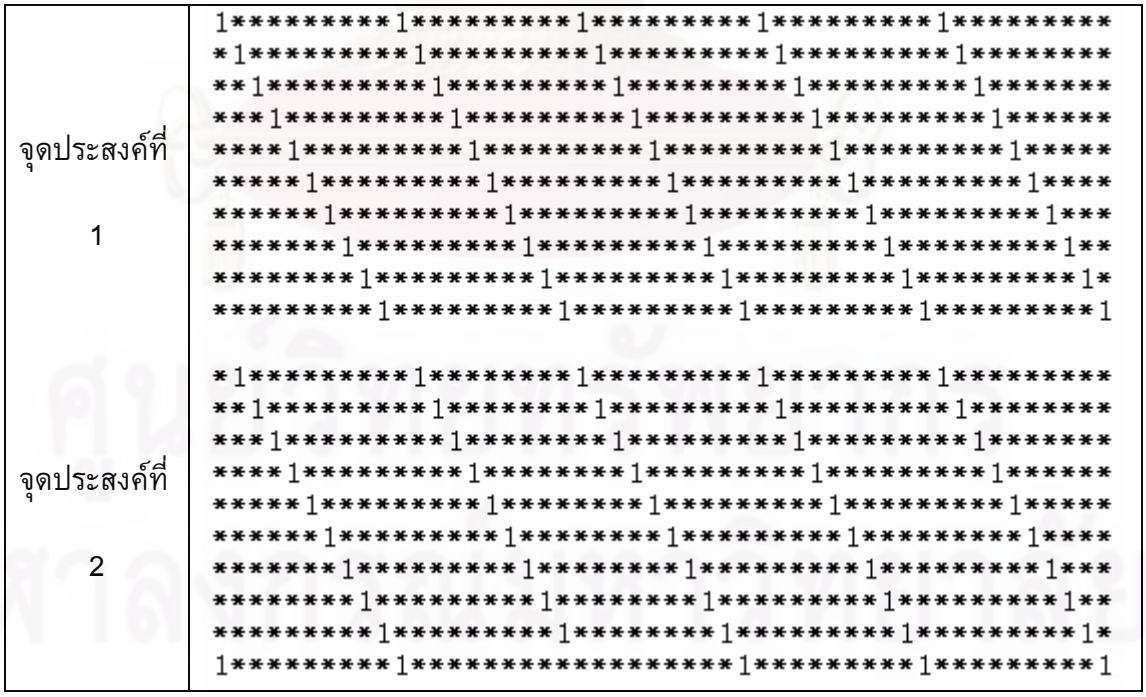
BBs) แสดงว่าหน่วยการสร้างของแต่ละจุดประสงค์ซ้อนทับกันได้พอดีหรือจะเรียกในทางตรงกันข้ามว่าเป็นการซ้อนทับกันแบบสมบูรณ์ (Completely Overlapping BB)

ปัญหานี้ถูกออกแบบสำหรับการทดสอบขั้นตอนวิธีที่นำเสนอซึ่งได้ออกแบบขึ้นมาใหม่ โดยจะมีหน่วยการสร้างของแต่ละจุดประสงค์เหลื่อมกันอยู่ดังรูปที่ 48 จะเห็นได้ว่าแต่ละหน่วยการสร้างมีการเหลื่อมกันอยู่ 2-3 บิต

จุดประสงค์ที่ 1	11111	22222	3333	44444	55555	66666
จุดประสงค์ที่ 2	66111	11222	22333	33444	44555	55666

รูปที่ 48 หน่วยการสร้างของปัญหา Overlapping MO-Trap

นอกจากนี้แล้วปัญหานี้ยังถูกปรับให้มีหน่วยการสร้างไม่ต่อเนื่องหรือหน่วยการสร้างหลวม โดยแต่ละบิตในหน่วยการสร้างเดียวกันจะห่างกันประมาณ l/m เมื่อ m เป็นจำนวนหน่วยการสร้าง ดังแสดงในรูปที่ 49 จะเห็นว่าหน่วยการสร้างแรก ของจุดประสงค์ที่ 1 จะเหลื่อมกับหน่วยการสร้างแรก ของจุดประสงค์ที่ 2 อยู่ 3 บิต (ตรงกัน 3 บิตหลัง ส่วน 2 บิตแรกไม่ตรงกัน แต่จะไปตรงกันกับหน่วยการสร้างสุดท้ายของจุดประสงค์ที่ 2)



รูปที่ 49 การเรียงตำแหน่งบิตของแต่ละหน่วยการสร้างของปัญหา MOP4

5.3 การกำหนดค่าพารามิเตอร์การทดลอง

ขนาดประชากรและจำนวนรุ่นที่ใช้ในการทดลองสำหรับแต่ละปัญหามีค่าต่างกันดังแสดงไว้ในตารางที่ 4 ซึ่งจะขึ้นอยู่กับขนาดของปัญหาและความยากง่ายของปัญหา

ตารางที่ 4 พารามิเตอร์ที่ใช้ในการทดลอง

ปัญหา	ขนาดของปัญหา	**mBOA**		**NSGA-II**		#evaluations
		popsize	ngen	popsize	ngen	
MOP1	30	300	40	300	40	12000
	60	-	-	500	60	30000
	90	700	80	700	80	56000
MOP2	30	1300	40	1300	40	52000
	60	3300	60	3300	120	198000
	90	5200	80	5200	80	416000
MOP3	30	-	-	1300	40	52000
	50	-	-	3300	120	198000
	60			6000	120	720000
MOP4	30	-	-	1300	40	52000
	50	-	-	3300	120	198000
	60	-	-	6600	120	792000

หมายเหตุ: ช่องที่มีเครื่องหมาย - หมายถึง ไม่มีรายงานผลการทดลองในงานวิจัยเดิม [4]

ในผลการทดลองกำหนดให้ขั้นตอนวิธีที่นำเสนอมีชื่อกำกับดังนี้

NSGA-II	ขั้นตอนวิธี NSGA-II ใช้การไขว้เปลี่ยนแบบ 2 จุด
NSGA-IIa	ใช้หน่วยการสร้างร่วมจากคำตอบในพาเรโตฟรอนต์
NSGA-IIb	การผสมหน่วยการสร้างแบบเชื่อมั่น
NSGA-IIc	การผสมหน่วยการสร้างแบบขยาย
NSGA-IId	ใช้หน่วยการสร้างจากแต่ละจุดประสงค์แยกกัน

5.4 ผลการทดลองและวิเคราะห์ผล

ผลการทดลองจะถูกนำเสนอเรียงลำดับไปทีละปัญหา ในปัญหา MOP1 และ MOP2 มีผลการทดลองเปรียบเทียบขั้นตอนวิธีต่างๆ ในงานวิจัยก่อนหน้า [4] เพิ่มเติมเข้ามา ในทุกการ

ทดลองผลการทดลองจะประกอบไปด้วยการไขว้เปลี่ยน 2 รูปแบบ คือ UniformBB- Crossover และ minimal Disruptive โดยที่ UniformBBCrossover จะเลือกที่จะไขว้เปลี่ยนทุกๆ หน่วยการสร้างด้วยโอกาสเท่าๆ กัน แต่ minimal Disruptive จะเลือกเพียงหน่วยการสร้างเดียวเพื่อแลกเปลี่ยนหน่วยการสร้างนั้น เนื่องจากว่าถ้าหน่วยการสร้างมีการเหลื่อมกันอยู่จะเกิดการทำลายหน่วยการสร้างทุกครั้งที่มีการแลกเปลี่ยนหน่วยการสร้าง คือ จะรักษาหน่วยการสร้างของจุดประสงค์หนึ่งไว้ และเกิดการทำลายหน่วยการสร้างของอีกจุดประสงค์หนึ่ง

ผลการทำงานเปรียบเทียบ NSGA-II กับ NSGA-IIa ถึง NSGA-IId ได้มาจากค่าเฉลี่ยของการทดลองที่ไม่ขึ้นต่อกันปัญหาละ 30 ครั้งการทำงาน

5.4.1 ผลการทดลองปัญหา MOP1

ผลการทดลองในตารางที่ 5 เป็นผลที่รายงานในงานวิจัย [4] จะเห็นว่าประสิทธิภาพในการหาคำตอบของ NSGA-II ด้อยกว่าขั้นตอนวิธีอื่นๆ เยอะมาก แต่การทำงานของขั้นตอนวิธี NSGA-II ก็ง่ายและเร็วกว่า

ตารางที่ 5 ตารางเปรียบเทียบแต่ละขั้นตอนวิธีสำหรับปัญหา MOP1

	MOP1							
	Unique Strings Found				Pareto Front pts Found			
	30	**60**	**90**	**120**	**30**	**60**	**90**	**120**
Algorithm	*2^15*	*2^30*	*2^45*	*2^60*	*16*	*31*	*46*	*61*
MOMGA-IIa	32768	300776	57661	32876	16	31	46	61
MOMGA-II	596	21364	28	138	16	31	16	56
mBOA	224		591		16		46	
NSGA-II	7		5		6		3	

ผลการทดลองในตารางที่ 6 และ ตารางที่ 7 เป็นผลที่ได้จากการทดลองขึ้นมาใหม่โดยใช้พารามิเตอร์ตามที่กำหนดไว้ในตารางที่ 4

ตารางที่ 6 ผลของวิธีการไขว้เปลี่ยนแบบ UniformBBcrossover ในปัญหา MOP1

	จำนวนคำตอบในพาเรโตฟรอนต์				จำนวนจุดที่แตกต่างกันในพาเรโต		
	30	60	90		30	60	90
NSGA-II	10.20	12.53	15.93		7.00	6.17	6.40
NSGA-IIa	9.00	17.57	19.97		6.57	8.60	10.03
NSGA-IIb	*9.27	17.47	22.40		*6.83	8.87	*11.37
NSGA-IIc	8.97	*18.13	20.77		6.47	8.70	11.20
NSGA-IId	8.93	17.90	*22.60		6.40	*8.90	10.73
รวม (a-d)	36.17	71.07	85.74		26.27	35.07	43.33

* หมายถึง ค่ามากสุดเปรียบเทียบขั้นตอนวิธี NSGA-IIa – NSGA-IId

ตารางที่ 7 ผลของวิธีการไขว้เปลี่ยนแบบ Minimal-Disruptive ในปัญหา MOP1

	จำนวนคำตอบในพาเรโตฟรอนต์				จำนวนจุดที่แตกต่างกันในพาเรโต		
	30	60	90		30	60	90
NSGA-II	10.20	12.53	15.93		7.00	6.17	6.40
NSGA-IIa	8.77	17.63	20.33		*6.53	8.60	10.23
NSGA-IIb	*9.20	*17.83	21.90		6.37	*8.97	*10.60
NSGA-IIc	9.17	17.67	21.97		6.30	8.60	10.37
NSGA-IId	*9.20	17.37	*22.50		6.33	8.63	10.37
รวม (a-d)	36.34	70.5	86.7		25.53	34.8	41.57

* หมายถึง ค่ามากสุดเปรียบเทียบขั้นตอนวิธี NSGA-IIa – NSGA-IId

จากผลการทดลองในตารางที่ 6 และ ตารางที่ 7 จะเห็นได้ว่าในปัญหาขนาด 30 บิต ขั้นตอนวิธี NSGA-II สามารถทำงานได้ดีกว่าขั้นตอนวิธีอื่น แต่เมื่อปัญหามีขนาดใหญ่ขึ้น ความยาวของหน่วยการสร้างยิ่งยาวขึ้น การระบุหน่วยการสร้างก็จะยิ่งมีความสำคัญมากขึ้น และจะพบว่าในปัญหาขนาดใหญ่คือขนาด 60 และ 90 บิต เมื่อใช้การไขว้เปลี่ยนแบบ 2 จุดของขั้นตอนวิธี NSGA-II จะทำงานได้แย่กว่าการไขว้เปลี่ยนแบบใช้หน่วยการสร้าง

เนื่องจากปัญหานี้เป็นปัญหาหลอกน้อยที่สุด หรือง่ายที่สุดในบรรดาปัญหาหลอกที่ทำการทดลอง ทำให้การไขว้เปลี่ยนแบบสองจุดยังทำงานได้ดีในปัญหาขนาดเล็ก แต่ปัญหานี้ก็ยากขึ้นได้ ถ้ากำหนดให้บิตที่อยู่ในหน่วยการสร้างเดียวกันอยู่ห่างกัน การไขว้เปลี่ยนแบบใช้หน่วยการสร้างจึงทำงานได้ดีกว่าการไขว้เปลี่ยนแบบ 2 จุดในปัญหาขนาดใหญ่ขึ้น เพราะหน่วยการสร้างจะมีความยาวมากขึ้น

เมื่อดูที่จำนวนจุดที่แตกต่างกันในพาราโตทางฝั่งขวาของตาราง จะพบว่า ยิ่งปัญหามีขนาดใหญ่ขึ้น การกระจายตัวของคำตอบ ในพาราโตที่ได้จากการไขว้เปลี่ยนแบบสองจุดกลับลดลง ซึ่งตรงกันข้ามกับการไขว้เปลี่ยนแบบใช้หน่วยการสร้าง จะมีจำนวนจุดเพิ่มขึ้นเรื่อยๆ ตามขนาดของปัญหา

ผลรวมแถวล่างสุดเป็นผลรวมเฉพาะขั้นตอนวิธี NSGA-IIa ถึงขั้นตอนวิธี NSAG-IId เพื่อเปรียบเทียบประสิทธิภาพของวิธีการไขว้เปลี่ยนทั้งสองแบบ เทียบกันทีละปัญหาจะได้ว่าจำนวนคำตอบที่อยู่ในพาเรโตฟรอนท์ของวิธีการไขว้เปลี่ยนแบบ minimal-Disruptive ดีกว่า แต่เมื่อดูจำนวนจุดที่แตกต่างกันในพาเรโตฟรอนต์พบว่าวิธีการไขว้เปลี่ยนแบบ UniformBBCrossover มีการกระจายตัวมากกว่า

เครื่องหมาย * ในตารางแสดงถึงผลการทำงานดีสุดในระหว่างขั้นตอนวิธีที่นำเสนอทั้ง 4 วิธี ในปัญหานี้สรุปได้ว่า ขั้นตอนวิธี NSGA-IIb ให้ผลการทำงานโดยรวมดีสุด ทั้งในการไขว้เปลี่ยนแบบ UniformBBCrossover และการไขว้เปลี่ยนแบบ minimal-Disruptive

5.4.2 ผลการทดลองปัญหา MOP2

ตารางที่ 8 เป็นผลเปรียบเทียบจากงานวิจัย [4] ผลการทำงานของขั้นตอนวิธี NSGA-II หาคำตอบของปัญหานี้ได้น้อยมาก

ตารางที่ 8 ตารางเปรียบเทียบแต่ละขั้นตอนวิธีสำหรับปัญหา MOP2

	MOP2							
	Unique Strings Found				Pareto Front pts Found			
	30	**60**	**90**	**120**	**30**	**60**	**90**	**120**
Algorithm	*2^6*	*2^12*	*2^18*	*2^24*	*7*	*13*	*19*	*25*
MOMGA-IIa	64	565	1280	3594	7	13	19	25
MOMGA-II	32	98	54	31	7	12	6	14
mBOA	30	102	327		7	13	19	
NSGA-II	0	1	0		0	1	0	

ผลการทำงานของขั้นตอนวิธี NSGA-II ทั้งแบบที่ใช้และไม่ใช้หน่วยการสร้างแสดงไว้ในตารางที่ 9 และ ตารางที่ 10

ตารางที่ 9 ผลของวิธีการไขว้เปลี่ยนแบบ UniformBBcrossover ในปัญหา MOP2

	จำนวนคำตอบในพาเรโตฟรอนต์				จำนวนจุดที่แตกต่างกันในพาเรโต		
	30	60	90		30	60	90
NSGA-II	1.80	1.03	1.00		1.77	1.03	1.00
NSGA-IIa	2.80	*3.20	2.80		2.50	2.50	2.43
NSGA-IIb	2.87	*3.20	3.00		2.50	*2.67	2.63
NSGA-IIc	*3.17	2.60	3.50		*2.67	2.33	2.80
NSGA-IId	2.73	3.10	*3.93		2.30	2.63	*3.17
รวม (a-d)	11.57	12.10	13.23		9.97	10.13	11.03

* หมายถึง ค่ามากสุดเปรียบเทียบขั้นตอนวิธี NSGA-IIa – NSGA-IId

ตารางที่ 10 ผลของวิธีการไขว้เปลี่ยนแบบ Minimal-Disruptive ในปัญหา MOP2

	จำนวนคำตอบในพาเรโตฟรอนต์				จำนวนจุดที่แตกต่างกันในพาเรโต		
	30	60	90		30	60	90
NSGA-II	1.80	1.03	1.00		1.77	1.03	1.00
NSGA-IIa	*3.13	2.87	*3.47		2.50	2.40	*2.90
NSGA-IIb	3.10	2.97	3.37		*2.57	2.57	2.80
NSGA-IIc	2.67	3.20	2.93		2.27	2.60	2.53
NSGA-IId	2.90	*3.33	3.10		2.33	*2.67	2.57
รวม (a-d)	11.80	12.37	12.87		9.67	10.24	10.80

* หมายถึง ค่ามากสุดเปรียบเทียบขั้นตอนวิธี NSGA-IIa – NSGA-IId

ปัญหานี้เป็นปัญหายากกว่าปัญหา MOP1 ทำให้ NSGA-II ซึ่งใช้การไขว้เปลี่ยนแบบสองจุดทำงานได้แย่กว่าการไขว้เปลี่ยนแบบอาศัยหน่วยการสร้างทั้งในปัญหาขนาดเล็กและปัญหาขนาดใหญ่

จากผลรวมของจำนวนคำตอบจะได้ว่าจำนวนคำตอบที่อยู่ในพาเรโตฟรอนท์ของการไขว้เปลี่ยนแบบ minimal-Disruptive ดีกว่า สำหรับจำนวนจุดที่แตกต่างกันในพาเรโตฟรอนต์พบว่าวิธีการไขว้เปลี่ยนแบบ UniformBBCrossover มีการกระจายตัวของจุดมากกว่า

ผลการนับ * ในปัญหา MOP2 ขั้นตอนวิธี NSGA-IIa ให้ผลการทำงานดีที่สุดในการไขว้เปลี่ยนแบบ Minimal-Disruptive แต่การไขว้เปลี่ยนแบบ UniformBBcrossover แต่ละวิธีจะดีพอๆกันยกเว้น NSGA-IIa จะแย่สุด

5.4.3 ผลการทดลองปัญหา MOP3

ปัญหานี้มีหน่วยการสร้างเหมือนกันกับปัญหา MOP2 แต่ต่างกันที่ค่าของแต่ละจุดประสงค์มีความสมดุลกัน ทำให้การกระจายตัวของคำตอบในพาเรโตไม่เหมือนกันกับปัญหา MOP2 ได้ทำการทดลองเปรียบเทียบขั้นตอนวิธี NSGA-II เดิมกับขั้นตอนวิธีที่ได้ปรับปรุง มีผลเปรียบเทียบดังแสดงในตารางที่ 11 และตารางที่ 12

ผลการทดลองปัญหานี้มีแนวโน้มคล้ายกับปัญหา MOP1 คือในปัญหาขนาดเล็ก NSGA-II ให้ผลการทำงานดีกว่าขั้นตอนวิธีที่นำเสนอ แต่ในปัญหาขนาดใหญ่ ขั้นตอนวิธีที่นำเสนอจะให้ผลการทำงานดีกว่า

ตารางที่ 11 ผลของวิธีการไขว้เปลี่ยนแบบ UniformBBcrossover ในปัญหา MOP3

	จำนวนคำตอบในพาเรโตฟรอนต์				จำนวนจุดที่แตกต่างกันในพาเรโต		
	30	50	60		30	50	60
NSGA-II	4.83	2.90	2.93		3.67	2.77	2.93
NSGA-IIa	2.70	3.17	4.50		*2.37	2.73	3.67
NSGA-IIb	*2.87	3.00	4.67		*2.37	2.60	3.57
NSGA-IIc	2.80	3.30	*5.03		*2.37	*2.87	*3.90
NSGA-IId	2.63	*3.37	4.03		2.20	*2.87	3.17
รวม (a-d)	11.00	12.84	18.23		9.31	11.07	14.31

* หมายถึง ค่ามากสุดเปรียบเทียบขั้นตอนวิธี NSGA-IIa – NSGA-IId

ตารางที่ 12 ผลของวิธีการไขว้เปลี่ยนแบบ Minimal-Disruptive ในปัญหา MOP3

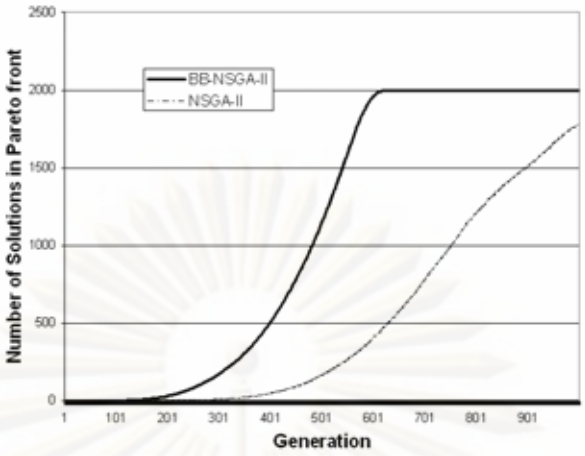
	จำนวนคำตอบในพาเรโตฟรอนต์				จำนวนจุดที่แตกต่างกันในพาเรโต		
	30	50	60		30	50	60
NSGA-II	4.83	2.90	2.93		3.67	2.77	2.93
NSGA-IIa	*3.23	3.10	*4.73		*2.67	2.60	3.53
NSGA-IIb	2.90	3.17	4.37		2.47	*2.77	3.50
NSGA-IIc	2.93	2.90	4.33		2.53	2.50	3.53
NSGA-IId	2.90	*3.53	4.70		2.37	*2.77	*3.67
รวม (a-d)	11.96	12.70	18.13		10.04	10.64	14.23

* หมายถึง ค่ามากสุดเปรียบเทียบขั้นตอนวิธี NSGA-IIa – NSGA-IId

จากผลรวมของจำนวนคำตอบจะได้ว่า จำนวนคำตอบที่ต่างกันในพาเรโตฟรอนต์ของวิธี UniformBBCrossover มีจำนวนมากกว่า รวมทั้งจำนวนจุดที่แตกต่างกันในพาเรโตฟรอนต์จะพบว่าวิธีการไขว้เปลี่ยนแบบ UniformBBCrossover มีการกระจายตัวดีกว่าด้วย

ในการไขว้เปลี่ยนแบบ UniformBBcrossover ขั้นตอนวิธี NSGA-IIa ให้ผลแย่สุด และขั้นตอนวิธี NSGA-IIc ให้ผลดีสุด ส่วนการไขว้เปลี่ยนแบบ minimal-Disruptive ขั้นตอนวิธี NSGA-IIa และ NSGA-IId ดีเท่าๆ กัน แต่ NSGA-IIc กลับมีผลการทำงานแย่สุด

เทคนิคที่นำเสนอสามารถระบุหน่วยการสร้างเป็นพาร์ทิชันที่สอดคล้องกับปัญหา แต่การผสมหน่วยการสร้างก็ยังทำให้คำตอบลู่เข้าเร็วกว่าการไม่ใช้หน่วยการสร้าง จึงมีผลทำให้การกระจายตัวของคำตอบน้อยกว่า [75] ดูรูปที่ 50 จะเห็นได้ว่าปริมาณคำตอบที่ลู่เข้าสู่พาราโตฟอนท์ของขั้นตอนวิธีที่อาศัยหน่วยการสร้างยังไม่ดีพอ ในประชากรรุ่นแรกๆ เนื่องจากคำตอบส่วนใหญ่มีลักษณะไม่ต่างจากการสุ่ม แต่เมื่อผ่านไปสักระยะการระบุหน่วยการสร้างทำได้ถูกต้อง ทำให้คำตอบที่นำมาไขว้เปลี่ยนกันสามารถปรับปรุงคุณภาพได้อย่างรวดเร็ว และเข้าไปอยู่ในพาเรโตฟรอนท์ได้เร็วและมากกว่าการไขว้เปลี่ยนแบบสองจุด



รูปที่ 50 จำนวนคำตอบในพาเรโตฟรอนต์ที่หาได้ในปัญหา MOP3 ขนาด 50 บิต

5.4.4 ผลการทดลองปัญหา MOP4

ผลการทดลองแสดงไว้ในตารางที่ 13 และ ตารางที่ 14

ตารางที่ 13 ผลของวิธีการไขว้เปลี่ยนแบบ UniformBBcrossover ในปัญหา MOP4

	จำนวนคำตอบในพาเรโตฟรอนต์				จำนวนจุดที่แตกต่างกันในพาเรโต		
	30	50	60		30	50	60
NSGA-II	0.67	12.33	9.53		0.67	2.10	2.33
NSGA-IIa	0.47	*9.63	17.13		0.47	*2.20	*2.93
NSGA-IIb	0.40	7.23	16.63		0.40	1.73	2.67
NSGA-IIc	*0.50	9.53	15.23		*0.50	2.03	2.83
NSGA-IId	0.33	6.77	*17.60		0.33	1.90	2.80
รวม (a-d)	1.70	33.16	66.59		1.70	7.86	11.23

* หมายถึง ค่ามากสุดเปรียบเทียบขั้นตอนวิธี NSGA-IIa – NSGA-IId

ตารางที่ 14 ผลของวิธีการไขว้เปลี่ยนแบบ Minimal-Disruptive ในปัญหา MOP4

	จำนวนคำตอบในพาเรโตฟรอนต์				จำนวนจุดที่แตกต่างกันในพาเรโต		
	30	50	60		30	50	60
NSGA-II	0.67	12.33	9.53		0.67	2.10	2.33
NSGA-IIa	*0.50	9.87	14.60		*0.50	2.23	2.57
NSGA-IIb	0.40	11.57	12.67		0.40	2.23	2.13
NSGA-IIc	0.40	*11.73	14.03		0.40	*2.30	2.60
NSGA-IId	*0.50	9.10	*16.07		*0.50	2.27	*2.67
รวม (a-d)	1.80	42.27	57.37		1.80	9.03	9.97

* หมายถึง ค่ามากสุดเปรียบเทียบขั้นตอนวิธี NSGA-IIa – NSGA-IId

ฟังก์ชันของปัญหานี้เหมือนกันกับปัญหาก่อนหน้า แต่ปัญหานี้เป็นปัญหาที่ได้ออกแบบขึ้นมาใหม่ โดยให้แต่ละหน่วยการสร้างของแต่ละจุดประสงค์เหลื่อมกันอยู่ ไม่ได้ซ้อนทับกันพอดี

ผลการทำงานยังอยู่ในลักษณะเดียวกันกับปัญหา MOP1 และ MOP3 คือ ในปัญหาขนาดเล็กขั้นตอนวิธี NSGA-II ที่ใช้การไขว้เปลี่ยนแบบ 2 จุด ให้ผลการทำงานดีกว่า แต่เมื่อปัญหามีขนาดใหญ่ขึ้น ผลการทำงานของขั้นตอนวิธีที่นำเสนอจะกลับได้ผลดีกว่า ยกเว้นขั้นตอนวิธี NSGA-IIb

จากผลรวมของจำนวนคำตอบจะได้ว่าในปัญหาขนาด 30 บิตและ 50 บิต ทั้งจำนวนคำตอบและจำนวนจุดที่แตกต่างกันในพาเรโตฟรอนท์ของวิธี minimal-Disruptive ดีกว่า แต่ในปัญหา 60 บิต UniformBBCrossover กลับดีกว่าทั้งในด้านจำนวนจุดที่แตกต่างกันในพาเรโตฟรอนต์และในด้านจำนวนคำตอบที่หาเจอในพาเรโตฟรอนต์

เนื่องจากพาเรโตฟรอนต์ของปัญหา MOP4 ในขนาดเล็ก (30 บิต) และขนาดกลาง (50 บิต) มีจำนวนจุดน้อยคือ 2 และ 5 จุดตามลำดับ ทำให้การระบุหน่วยการสร้างของปัญหาทำได้ยาก ในปัญหาขนาด 30 บิตจึงได้ผลการทำงานของขั้นตอนวิธีที่นำเสนอแย่กว่าขั้นตอนวิธี NSGA-II ปกติ อย่างไรก็ดี ในปัญหาขนาด 50 บิต ที่ใช้การไขว้เปลี่ยนแบบ minimal Disruptive พบว่าคำตอบของขั้นตอนวิธีที่นำเสนอ มีการกระจายตัวดีกว่าคำตอบของขั้นตอนวิธี NSGA-II ในทุกๆวิธีที่นำเสนอ ในที่นี้การกระจายตัวดีกว่าไม่ได้แสดงว่าจำนวนคำตอบที่อยู่ในพาเรโตฟรอนท์จะมากกว่า แต่ถ้าดูจำนวนคำตอบในพาเรโตฟรอนท์ของ NSGA-II ในปัญหาขนาด 50 บิต เฉลี่ยแล้วสูงกว่าขั้นตอนวิธีที่นำเสนอทุกกรณีเช่นกัน

ในปัญหา MOP4 ขั้นตอนวิธี NSGA-IIa ให้ผลดีสุดในวิธีการไขว้เปลี่ยนแบบ UniformBBcrossover แต่ในการไขว้เปลี่ยนแบบ Minimal-Disruptive ขั้นตอนวิธี NSGA-IIc ให้ผลการทำงานดีที่สุด

5.5 สรุปผลการทดลอง

ขั้นตอนวิธี NSGA-II จะให้ผลการทดลองดีกว่าขั้นตอนวิธีที่นำเสนอในปัญหาขนาดเล็ก แต่ในปัญหาขนาดใหญ่ ขั้นตอนวิธี NSGA-II จะกลับให้ผลการทำงานที่แย่กว่า ที่เป็นเช่นนี้เนื่องจากในปัญหาขนาดใหญ่จะมีความยาวของหน่วยการสร้างมากขึ้น ทำให้การระบุหน่วยการสร้างทำงานได้ผลดีกว่าการไขว้เปลี่ยนแบบ 2 จุด เนื่องจากการไขว้เปลี่ยนแบบ 2 จุด จะแบ่งแยกหน่วยการสร้างเสมอ ไม่ว่าจะทำการไขว้เปลี่ยน ณ จุดใด ทำให้มีโอกาสทำลายหน่วยการสร้างมากกว่าเป็นการประกอบหน่วยการสร้าง เนื่องจากปัญหาที่ทำการทดลองเป็นปัญหาแบบ 2 ขั้ว คือ ในจุดประสงค์หนึ่งมีค่าเหมาะสุดเป็น ‘1’ หมดทุกบิต ส่วนอีกจุดประสงค์หนึ่งมีค่า

เหมาะสุดเป็น ‘0’ หมดทุกบิต ดังนั้นการไขว้เปลี่ยนแบบ 2 จุด จึงมีโอกาสแลกเปลี่ยนหน่วยการสร้างที่เป็น 0 กับหน่วยการสร้างที่เป็น 1 ส่วนการไขว้เปลี่ยนแบบใช้หน่วยการสร้างจะแลกเปลี่ยนกันทั้งหน่วยการสร้าง ทำให้ลดโอกาสทำลายหน่วยการสร้างลง หรือไม่ทำลายหน่วยการสร้างเลยในกรณีที่ระบุหน่วยการสร้างได้สมบูรณ์ (ยกเว้นปัญหา MOP4 ที่หน่วยการสร้างของแต่ละจุดประสงค์เหลื่อมกันอยู่)

อย่างไรก็ดีการไม่ทำลายหน่วยการสร้างเลยก็เท่ากับว่าจะเหลือเพียงการประกอบหน่วยการสร้างเท่านั้น จึงจำเป็นต้องมีหน่วยการสร้างนั้นปรากฏอยู่ในคำตอบและต้องมีมากเพียงพอ [9] ดังนั้นในการทดลองกับปัญหาขนาดเล็กที่ใช้ขนาดประชากรเล็กด้วย จึงมีโอกาสที่จำนวนหน่วยการสร้างที่เป็นคำตอบไม่เพียงพอสำหรับการประกอบให้ได้คำตอบ

อีกเหตุผลหนึ่งคือ ในปัญหาขนาดเล็ก มีจำนวนจุดในพาเรโตน้อย ส่งผลให้ขั้นตอนวิธีที่ใช้หน่วยการสร้างเกิดการลู่เข้าเร็วเกินไป ทำให้ประสิทธิภาพการระบุหน่วยการสร้างลดลง และมีการกระจายตัวน้อยกว่าวิธีการไขว้เปลี่ยนแบบสองจุด

บทที่ 6

สรุปผลการวิจัย

บทนี้ได้สรุปผลการวิจัยที่ได้ศึกษาวิจัย และกล่าวถึงสมมติฐานหน่วยการสร้างสำหรับปัญหาหลายฐาน (multimodal problems) พร้อมทั้งเขียนข้อเสนอแนะสำหรับงานวิจัยที่น่าจะศึกษาต่อ

6.1 บทสรุป

วิทยานิพนธ์ฉบับนี้ได้ศึกษาการใช้การระบุหน่วยการสร้างโดยใช้เมทริกซ์ไคกำลังสอง (Building Block Identification by Chi-square Matrix: BICM) ร่วมกับขั้นตอนวิธี NSGA-II เพื่อแก้ปัญหาหลายจุดประสงค์

จากการทดลองพบว่าการใช้หน่วยการสร้างร่วมกับขั้นตอนวิธี NSGA-II มีประโยชน์ในการแก้ปัญหาหลายจุดประสงค์ ที่มีหน่วยการสร้างที่เด่นชัด โดยเฉพาะปัญหาที่มีขนาดใหญ่หรือหน่วยการสร้างที่มีความยาวมาก

จากการศึกษาเกี่ยวกับหน่วยการสร้างในบทที่ 2 และขั้นตอนวิธีสำหรับแก้ปัญหาหลายจุดประสงค์ ได้เลือกใช้ขั้นตอนวิธีระบุหน่วยการสร้างด้วยไคกำลังสอง เนื่องจากทำงานได้รวดเร็วและใช้ทรัพยากรน้อย [55] เมื่อเทียบกับการหาหน่วยการสร้างวิธีอื่นๆอย่างเช่น การหาหน่วยการสร้างด้วยขั้นตอนวิธี BOA [12] ซึ่งใช้ทรัพยากรต่างกันกว่า 10 เท่า ทั้งด้านเวลาการทำงานและหน่วยความจำ

ในบทที่ 3 ได้นำเทคนิคการระบุหน่วยการสร้าง ดังกล่าวมาใช้แก้ปัญหาจุดประสงค์เดียวที่มีหน่วยการสร้างแบบหลวม เปรียบเทียบผลการทำงานกับขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย พบว่าขั้นตอนวิธีเชิงพันธุกรรมที่ใช้การระบุหน่วยการสร้างแก้ปัญหาได้ดีกว่าขั้นตอนวิธีเชิงพันธุกรรมอย่างง่าย เมื่อขนาดของหน่วยการสร้างยาวมากขึ้นขั้นตอนวิธีเชิงพันธุกรรมอย่างง่ายจะยิ่งไม่สามารถหาคำตอบได้

ในบทที่ 4 ได้ปรับใช้ขั้นตอนวิธีระบุหน่วยการสร้างกับปัญหาหลายจุดประสงค์โดยเสนอแนวทางในการผสมหน่วยการสร้างที่ได้จากแต่ละจุดประสงค์ ซึ่งหามาจากคำตอบที่ดีของจุดประสงค์นั้นๆ ในแต่ละรุ่นประชากร การผสมหน่วยการสร้างมี 4 วิธีได้แก่ 1)หาหน่วยการสร้างร่วมของทุกจุดประสงค์ จากคำตอบที่มีอยู่ในพาเรโตฟรอนต์ วิธีการนี้ไม่ได้แยกหน่วยการสร้างว่าเป็นของจุดประสงค์หนึ่งจุดประสงค์ใด 2) หาหน่วยการสร้างของแต่ละจุดประสงค์ แล้วรวมแบบเชื่อมั่น คือ ต้องปรากฏหน่วยการสร้างในทุกๆ จุดประสงค์ ถึงจะเชื่อได้ว่าตำแหน่งนั้นมีหน่วยการสร้างจริง 3) รวมหน่วยการสร้างแบบขยาย คือ ถ้ามีบางส่วนของหน่วยการสร้างใน

แต่ละจุดประสงค์เหลื่อมกันอยู่ก็จะรวมทุกๆ บิตของแต่ละหน่วยการสร้างนั้นเข้าด้วยกัน เป็นหน่วยการสร้างขนาดใหญ่ขึ้น 4) ไม่ผสมหน่วยการสร้างของแต่ละจุดประสงค์ แต่จะใช้หน่วยการสร้างเหล่านี้สร้างประชากรรุ่นใหม่ในปริมาณเท่าๆกัน

รหัสเทียมของการผสมแต่ละวิธีแสดงไว้ในบทที่ 4 โดยมีเหตุผลในการผสมแต่ละแบบคือ แบบที่ 1) ใช้หน่วยการสร้างร่วมจากคำตอบในพาเรโตฟรอนต์ วิธีนี้ต้องการแสดงให้เห็นว่าปัญหาที่มีหน่วยการสร้างเด่นชัดในคำตอบที่อยู่ในพาเรโตฟรอนท์อยู่แล้วไม่จำเป็นต้องแยกหน่วยการสร้างของแต่ละจุดประสงค์ ตัวอย่างเช่น MOP3 มีหน่วยการสร้างเด่นชัดอยู่แล้วในพาเรโตฟรอนต์ ทำให้การผสมหรือไม่ผสมหน่วยการสร้างได้ผลลัพธ์ไม่แตกต่างกันอย่างมีนัยสำคัญ

การผสมแบบที่ 2) การผสมหน่วยการสร้างแบบเชื่อมั่น มีจุดเด่นคือ ถ้าการระบุหน่วยการสร้างได้ขนาดของหน่วยการสร้างขนาดใหญ่เกินไป (ใหญ่กว่าขนาดหน่วยการสร้างจริงของปัญหาจริง) การผสมแบบนี้จะชี้ให้เห็นบริเวณที่ขึ้นต่อกันจริงๆของทุกจุดประสงค์ และจะช่วยแบ่งย่อยหน่วยการสร้างนั้นให้เล็กลง สอดคล้องกับปัญหาจริง และช่วยให้เกิดการแลกเปลี่ยนหน่วยการสร้างได้มากขึ้น การระบุหน่วยการสร้างที่ทำให้ขนาดหน่วยการสร้างใหญ่เกินไปเกิดจากหลายสาเหตุ เช่นใช้จำนวนประชากรน้อยไปไม่เหมาะกับปัญหานั้นๆ หรืออีกเหตุผลหนึ่งคือการใช้จำนวนประชากรปริมาณมากๆ กับปัญหาที่มีความสัมพันธ์ระหว่างบิตสูงจะทำให้ไม่สามารถแยกแยะความสัมพันธ์นั้นออกเป็นกลุ่มย่อยๆ ได้

การผสมแบบที่ 3) การผสมหน่วยการสร้างแบบขยาย จะมีประโยชน์ในกรณีที่หน่วยการสร้างของแต่ละจุดประสงค์ไม่อยู่ในตำแหน่งเดียวกัน การไขว้เปลี่ยนตามหน่วยการสร้างของจุดประสงค์ใดจุดประสงค์หนึ่ง ย่อมจะไปทำลายหน่วยการสร้างของ จุดประสงค์อื่นด้วย ดังนั้นความพยายามของการผสมแบบนี้ คือการรักษาหน่วยการสร้างของแต่ละจุดประสงค์ไปพร้อมๆกัน

การผสมแบบที่ 4) ใช้หน่วยการสร้างจากแต่ละจุดประสงค์แยกกัน เป็นการใช้งานหน่วยการสร้างของแต่ละจุดประสงค์แยกกัน โดยไม่มีการผสม การใช้หน่วยการสร้างของจุดประสงค์ใดก็เพื่อรักษาหน่วยการสร้างของจุดประสงค์นั้น วิธีนี้จึงสร้างประชากรรุ่นใหม่จากแต่ละหน่วยการสร้างที่ได้จากแต่ละจุดประสงค์ในปริมาณเท่าๆ กัน

หลังจากได้วิธีการใช้งานหน่วยการสร้างสำหรับปัญหาหลายจุดประสงค์แล้ว ได้ออกแบบการทดลองเพื่อวัดประสิทธิภาพของการรวมแต่ละแบบที่นำเสนอโดยได้ใช้ปัญหาหลายจุดประสงค์ที่ได้จากการนำปัญหาจุดประสงค์เดียวซึ่งมีหน่วยการสร้างมารวมกันเป็นปัญหาหลายจุดประสงค์ และได้ปัญหาทั้งหมด 4 ปัญหาคือ MOP1-MOP4 ปัญหา MOP1-MOP2 เคยมีผู้ทำวิจัยเปรียบเทียบผลไว้แล้วในงานวิจัย [4] สำหรับปัญหา MOP3 เคยมีผู้วิเคราะห์และใช้งานใน [6] ซึ่งจะมีจุดที่อยู่ในพาราโตฟรอนต์เหมือนกันกับ MOP2 และสุดท้ายเป็นปัญหา

MOP4 ที่ออกแบบขึ้นมาใหม่โดยใช้ฟังก์ชันใน MOP3 และปรับให้หน่วยการสร้างมีการเหลื่อมกัน เคยมีความพยายามที่จะใช้ MOP2 ปรับให้เป็นแบบมีหน่วยการสร้างเหลื่อมกัน แต่เนื่องจากไม่สามารถวิเคราะห์หาจุดในพาเรโตฟรอนท์ได้ในปัญหาขนาดใหญ่ เนื่องจากฟังก์ชันของแต่ละจุดประสงค์ไม่สมดุลกัน จึงไม่มีรูปแบบของจุดในแกนจุดประสงค์ที่จะช่วยการวิเคราะห์ได้ง่าย ถ้าจะใช้งานก็ต้องหาทุกรูปแบบที่เป็นไปได้ ซึ่งจะกินเวลามาก

ทุกปัญหาที่ใช้ในการทดลองมีหน่วยการสร้างแบบหลวม คือ แต่ละบิตของหน่วยการสร้างไม่ได้อยู่เรียงชิดติดต่อกัน แต่จะกระจายอยู่ห่างกันเท่าๆ กัน รายละเอียด ของแต่ละปัญหาอยู่ในหัวข้อ 5.2 ในบทที่ 5

ผลการทดสอบแต่ละขั้นตอนวิธีได้แสดงผลเปรียบเทียบไว้ในหัวข้อ 5.4 ในบทที่ 5 พบว่าการปรับใช้การระบุหน่วยการสร้างโดยแทนที่การไขว้เปลี่ยนแบบ 2 จุดของขั้นตอนวิธี NSGA-II โดยใช้หน่วยการสร้างที่ได้จากการผสมทั้ง 4 แบบซึ่งได้นำเสนอในบทที่ 4 ขั้นตอนวิธีที่นำเสนอให้ผลการทำงานดีกว่าขั้นตอนวิธี NSGA-II แบบเดิมในปัญหาที่มีขนาดใหญ่ในทุกๆ ปัญหา แต่ในปัญหาขนาดเล็กพบว่าส่วนใหญ่แล้ว NSGA-II จะให้ผลการทำงานที่ดีกว่าขั้นตอนวิธีที่นำเสนอ ยกเว้นในปัญหา MOP2 ที่ขั้นตอนวิธีที่นำเสนอมีผลการทำงานดีกว่า NSGA-II ในทุกกรณี

การไขว้เปลี่ยนที่ใช้หน่วยการสร้างนอกจากจะมีการผสมแยกเป็น 4 วิธีแล้ว ยังทำการทดลองทั้งแบบ UniformBBcrossover และ Minimal-Disruptive ทำให้มีการทดลองทั้งหมด 8 การทดลองจาก 4 ปัญหา (MOP1-MOP4) โดย UniformBBcrossover จะสุ่มแลกเปลี่ยนทุกๆ หน่วยการสร้างด้วยโอกาสเท่าๆ กัน แต่ใน Minimal-Disruptive จะสุ่มมา 1 หน่วยการสร้างแล้วแลกเปลี่ยนหน่วยการสร้างนั้น ที่เหลือคงไว้เหมือนเดิม เนื่องจากต้องการลดการทำลายหน่วยการสร้างที่เหลื่อมกันอยู่ ถ้ายิ่งมีการไขว้เปลี่ยนมากจะยิ่งทำลายหน่วยการสร้างที่เหลื่อมกันอยู่ทั้งสองข้าง

ผลการทำงานเปรียบเทียบขั้นตอนวิธีนำเสนอคือ NSGA-II ที่ใช้หน่วยการสร้างซึ่งมีการผสมหน่วยการสร้างต่างๆกัน 4 วิธี (NSGA-IIa ถึง NSGA-IId)

ในปัญหา MOP1 การผสมแบบเชื่อมั่น NSGA-IIb ให้ผลการทำงานดีที่สุด ทั้งในการไขว้เปลี่ยนแบบ UniformBBcrossover และการไขว้เปลี่ยนแบบ Minimal-Disruptive

ในปัญหา MOP2 ขั้นตอนวิธี NSGA-IIa ให้ผลการทำงานดีที่สุดในการไขว้เปลี่ยนแบบ Minimal-Disruptive แต่การไขว้เปลี่ยนแบบ UniformBBcrossover แต่ละวิธีจะดีพอๆกัน ยกเว้น NSGA-IIa จะแย่สุด

ในปัญหา MOP3 การไขว้เปลี่ยนแบบ UniformBBcrossover ขั้นตอนวิธี NSGA-IIa แย่สุดแต่ขั้นตอนวิธี NSGA-IIc ให้ผลดีสุด ส่วนการไขว้เปลี่ยนแบบ minimal-Disruptive ขั้นตอนวิธี NSGA-IIa และ NSGA-IId ดีเท่าๆ กัน แต่ NSGA-IIc กลับมีผลการทำงานแย่สุด

ส่วนปัญหา MOP4 ขั้นตอนวิธี NSGA-IIa ให้ผลการทำงานดีสุดในการไขว้เปลี่ยนแบบ UniformBBcrossover แต่สำหรับการไขว้เปลี่ยนแบบ Minimal-Disruptive ขั้นตอนวิธี NSGA-IIc ให้ผลการทำงานดีสุด

จากการทดลองการไขว้เปลี่ยนทั้งแบบ Minimal-Disruptive และแบบ UniformBB-crossover โดยรวมจากทุกๆ ปัญหาพบว่าการใช้การไขว้เปลี่ยนแบบ Minimal-Disruptive ให้จำนวนคำตอบที่แตกต่างกันในพาเรโตฟรอนต์มากกว่า แต่การไขว้เปลี่ยนแบบ UniformBB-crossover มีการกระจายตัวของจุดในพาเรโตฟรอนต์มากกว่า

สรุปผลที่ได้จากงานวิจัยนี้ประกอบไปด้วย

1. ได้เสนอการใช้งานการระบุหน่วยการสร้างในขั้นตอนวิธีเชิงพันธุกรรมเพื่อนำไปใช้แก้ปัญหาหลายจุดประสงค์
2. นำเสนอขั้นตอนวิธีการประกอบหน่วยการสร้างจากแต่ละจุดประสงค์ทั้งหมด 4 วิธี และ ทดลองเปรียบเทียบผลการทำงานแต่ละวิธี
3. ออกแบบปัญหาที่ใช้ทดสอบเป็นแบบมีหน่วยการสร้างที่เหลื่อมกันระหว่างจุดประสงค์และรวบรวมปัญหาที่ใช้ทดสอบประสิทธิภาพที่มีหน่วยการสร้างเด่นชัด

6.2 สมมติฐานของหน่วยการสร้างกับการแก้ปัญหาหลายฐาน (multimodal problems)

ในปัญหาจุดประสงค์เดียวกรณีที่มีค่าความเหมาะสมหลายฐาน (multimodal) จะมีหน่วยการสร้างหลายรูปแบบ โดยนิยามของหน่วยการสร้างคือ ลักษณะร่วมของคำตอบที่มีค่าความเหมาะสมสูง ดังนั้นมีความเป็นได้ที่แต่ละฐานมีหน่วยการสร้างเฉพาะตัว แต่เนื่องจากวิธีการระบุหน่วยการสร้างที่ใช้ในงานวิจัยนี้มองหน่วยการสร้างเป็นการแบ่งพาร์ทิชันของตำแหน่งบิต จึงอาจจะมองว่าหน่วยการสร้างของแต่ละฐานเป็นหน่วยการสร้างเดียวกัน คือมีตำแหน่งร่วมกัน มีผลให้ไม่สามารถจัดการกับปัญหาหลายฐานได้ดีเท่าที่ควร

วิธีการหนึ่งที่จะช่วยแก้ปัญหานี้ได้คือ การสร้างประชากรย่อย (sub-population) และพยายามควบคุมให้แต่ละกลุ่มหาคำตอบในบริเวณที่แตกต่างกัน คือ ให้หาคำตอบของแต่ละ

ฐานแยกกัน เช่น การควบคุมความหลายหลายของแต่ละกลุ่มประชากรย่อยให้แต่ละกลุ่มมีหน้าตาแตกต่างกัน [45] [46]

การแก้ปัญหาหลายจุดประสงค์ซึ่งมีความเชื่อว่าแต่ละจุดประสงค์มีลักษณะเป็นฐานแต่ละฐานเหมือนปัญหาจุดประสงค์เดียว ดังนั้นการแก้ปัญหาหลายจุดประสงค์ที่มองหน่วยการสร้างเป็นพาร์ทิชัน จึงน่าจะไม่เพียงพอสำหรับการแก้ปัญหา แต่หน่วยการสร้างของปัญหาหลายจุดประสงค์ไม่เหมือนกับปัญหาจุดประสงค์เดียว เนื่องจากคำตอบที่ต้องการมีหลายค่าซึ่งเกิดจากการถ่วงดุลของแต่ละฐาน ไม่ใช่คำตอบเดียวของแต่ละฐาน ดังนั้นการหาหน่วยการสร้างของปัญหาหลายจุดประสงค์จะเป็นการหาหน่วยการสร้างอีกรูปแบบหนึ่งโดยเฉพาะและแตกต่างจากจุดประสงค์เดียว

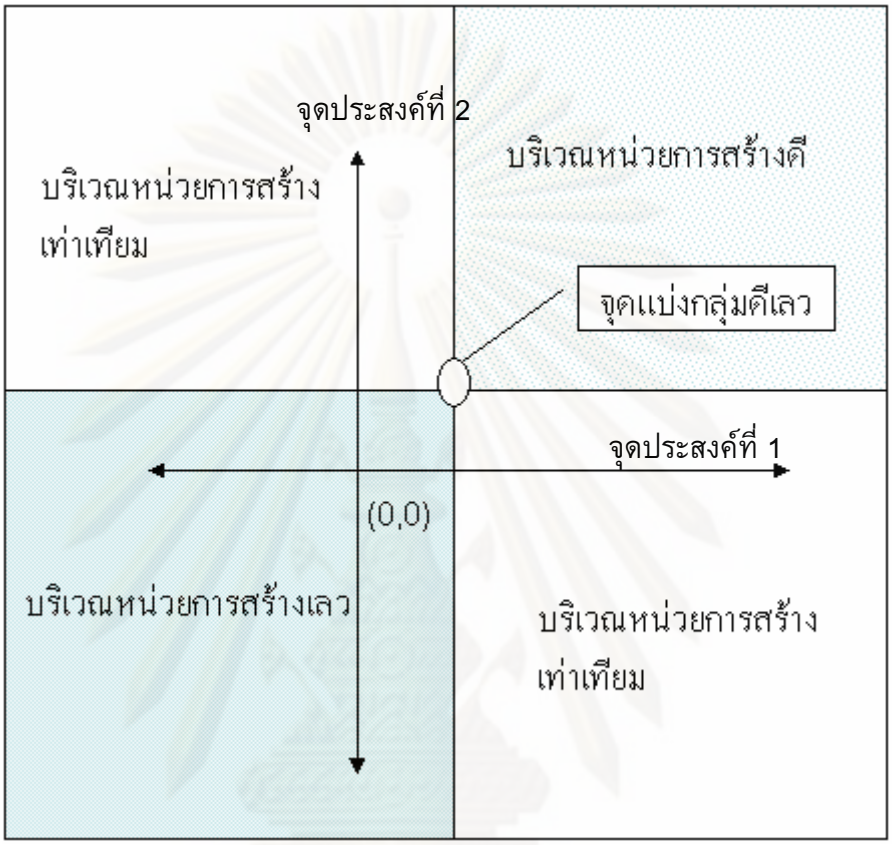
การสร้างประชากรย่อยสำหรับแต่ละคำตอบย่อมทำได้ยากเนื่องจากจำนวนคำตอบในพาเรโต ฟรอนต์มีปริมาณไม่แน่นอนและมีความเป็นไปได้ว่าคำตอบทุกคำตอบอาจอยู่ในพาเรโตฟรอนต์ทั้งหมด การแก้ปัญหาการกระจายตัวในพาเรโตฟรอนต์เพื่อให้ได้คำตอบหลายคำตอบจึงเป็นปัญหาหนึ่งที่ได้รับความสนใจ วิธีการหนึ่งที่ช่วยแก้ปัญหาคือ การควบคุมความหลากหลายของคำตอบ เพื่อให้มีการกระจายตัวของคำตอบในพาเรโตฟรอนต์ ขั้นตอนวิธีสำหรับแก้ปัญหาหลายจุดประสงค์จึงมักจะเป็นการประมาณจุดในพาเรโตฟรอนต์และพยายามหาเส้นทางเชื่อมของแต่ละจุดในกรณีที่คำตอบอยู่ในโดเมนต่อเนื่อง แต่ถ้าคำตอบเป็นโดเมนไม่ต่อเนื่องก็อาจจะประมาณได้ไม่ถูกต้องสมบูรณ์

การเพิ่มความหลากหลายของประชากรทำได้หลายวิธี [20] ตัวอย่างเช่น

1. การใช้วิธีการทางเทอร์โมไดนามิกส์ สำหรับตรวจสอบพลังงานอิสระ (free energy)ของคำตอบในประชากร
2. การปรับปรุงขั้นตอนการคัดเลือกประชากร (modify selection operation)
3. การจำกัดการจับคู่คำตอบ (mating restriction)
4. การใช้ค่าความน่าจะเป็นการกลายพันธุ์ที่เปลี่ยนค่าได้ (variable-rate mutation)
5. การปรับค่าความกดดันการคัดเลือก (modification of selective pressure)

การคัดเลือกประชากรสำหรับสร้างประชากรรุ่นใหม่สามารถปรับปรุงให้มีความหลากหลายโดยการเลือกเฉพาะคำตอบที่แตกต่างจากคำตอบที่มีอยู่ คำตอบที่ซ้ำกันจะถูกลบทิ้งไป

ในงานวิจัยของ Day [4] ให้มุมมองหน่วยการสร้างสำหรับปัญหาหลายจุดประสงค์ที่แตกต่างจากปัญหาจุดประสงค์เดียวไว้ ในกรณีสองจุดประสงค์และต้องการคำตอบมีค่ามากสุดเป็นดังนี้



รูปที่ 51การแบ่งกลุ่มหน่วยการสร้างในปัญหาหลายจุดประสงค์ตามแนวคิดของ Day

ในปัญหาจุดประสงค์เดียวจะแบ่งพื้นที่จุดประสงค์ออกเป็นสองฝั่งคือฝั่งดีและเลว ในขณะที่ปัญหาหลายจุดประสงค์จะแบ่งออกเป็น 4 พื้นที่ (หรือ 3 กลุ่มที่แตกต่างกัน) คือกลุ่มที่ดีกว่า กลุ่มที่เท่าเทียม และกลุ่มที่ด้อยกว่า ดังแสดงในรูป การหาจุดแบ่งของปัญหาหลายจุดประสงค์ทำได้ยากกว่าปัญหาจุดประสงค์เดียว แตกต่างจากแนวคิดในงานวิจัยนี้ที่สนใจหน่วยการสร้างของแต่ละจุดประสงค์ นั่นคือแบ่งความดีเลวตามค่าของแต่ละจุดประสงค์คล้ายกับปัญหาจุดประสงค์เดียว

6.3 ข้อเสนอแนะ

ขอเสนอแนะแง่มุมน่าสนใจที่ควรจะทำการวิจัยและศึกษาต่อไปได้แก่

1. เนื่องจากปัญหาที่ทดลองมีจำนวนจุดในพาเรโตฟรอนต์น้อยเกินไปไม่เพียงพอสำหรับการระบุหน่วยการสร้าง จึงน่าจะเพิ่มเติมหน่วยเก็บภายนอก (Archive) เพื่อใช้เก็บคำตอบดีๆ ของแต่ละจุดประสงค์ แยกกันสำหรับแต่ละจุดประสงค์ โดยไม่สนใจการครอบงำ เพื่อให้การระบุหน่วยการสร้างถูกต้องมากยิ่งขึ้นและยังช่วยรักษาความหลากหลายของคำตอบ และช่วยให้มีการกระจายตัวในพาเรโตฟรอนต์ได้ดี
2. ปรับใช้เทคนิคการระบุหน่วยการสร้างที่สามารถตรวจสอบการเหลื่อมกันของหน่วยการสร้างและยังรักษาหน่วยการสร้างในปัญหาที่มีการเหลื่อมกันอย่างซับซ้อนได้เช่น ขั้นตอนวิธีที่เสนอโดย Tsuji [76] หรือของ Tian-Li Yu [33] [34]
3. ทดลองกับปัญหาที่มีพาเรโตฟรอนต์กระจายเป็นจำนวนมาก หรือเพียงพอสำหรับระบุหน่วยการสร้าง อาจจะทดลองกับปัญหาที่มีขนาดใหญ่ขึ้น ให้เห็นประสิทธิภาพอย่างแท้จริง
4. การระบุหน่วยการสร้างต้องการค่า Threshold ที่เหมาะกับปัญหา ถ้าเป็นไปได้ควรจะวิเคราะห์ค่า Threshold นี้ว่าสัมพันธ์กับตัวแปรใดบ้าง เช่น ขนาดของปัญหา ความแตกต่างระหว่างค่ามากสุดและน้อยสุดของไคกำลังสองก่อนนำมาทดลองกับปัญหาอื่นๆ
5. ปัญหาที่ใช้ในการทดลอง มีจำนวนจุดในพาเรโตฟรอนท์น้อยเกินไป และมีการลู่เข้าเร็วเกินไป ทำให้การกระจายตัวของคำตอบไม่ดี และการระบุหน่วยการสร้างทำได้ไม่ดี แนวทางแก้ไขที่น่าจะช่วยได้
    - การรักษาความหลากหลายของประชากรไว้ (diversity control)
    - การใช้หน่วยการเก็บภายนอก (Archive) เพื่อให้มีปริมาณคำตอบที่เหมาะสมและเพียงพอสำหรับใช้หาหน่วยการสร้าง
6. นำไปทดลองใช้กับปัญหาในโลกจริง เช่น ปัญหาการปรับพารามิเตอร์การเลือกลักษณะ (feature selection) ในปัญหาการเรียนรู้ของเครื่องที่มีมิติใหญ่ คาดว่าปัญหานี้น่าจะมีหน่วยการสร้างเนื่องจากพารามิเตอร์ที่จะช่วยกันทำงานได้ดีมักจะขึ้นต่อกัน
7. ใช้วิธีการผสมทั้ง 4 แบบร่วมกัน ไม่แยกวิธี โดยจะใช้วิธีการเลือกเอาคำตอบที่ไม่ถูกครอบงำจากทั้ง 4 คำตอบที่ได้

รายการอ้างอิง

[1] Arslan, T., Horrocks, D. H. and Ozdemir, E. Structural synthesis of cell-based VLSI circuits using a multi-objective genetic algorithm. Electronic Letters 32 (1996): 651-652.

[2] Sal, D. and Graña, M. A multiobjective evolutionary algorithm for hyperspectral image watermarking. In M. Grana and R. J. Duro (eds), Computational Intelligence for Remote Sensing, pp. 63-78. Heidelberg: Springer Berlin, 2008.

[3] Coello Coello, C. A., Van Veldhuizen, D. A. and Lamont, G. B. Evolutionary algorithms for solving multi-objective problems. New York: Kluwer Academic Publishers, 2002.

[4] Day, R. O. Explicit building block multiobjective evolutionary computation: Methods and application. Doctoral dissertation, The Graduate School of Engineering and Management, Air Force Institute of Technology. 2005.

[5] Deb, K., Pratap, A., Agrawal, S. and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2002): 182-197.

[6] Sastry, K., Pelikan, M. and Goldberg, D. E. Limits of scalability of multiobjective estimation of distribution algorithms. In Proceedings of the Congress on Evolutionary Computation, 2005.

[7] Zitzler, E. Evolutionary algorithms for multiobjective optimization: Methods and applications. Doctoral dissertation, Swiss Federal Institute of Technology Zurich. 1999.

[8] Goldberg, D. E. Genetic algorithms in search optimization and machine learning. MA: Addison Wesley, Reading, 1989.

[9] Goldberg, D. E. The Design of innovation: Lessons from and for competent genetic algorithms. Boston, MA: Kluwer Academic Publishers, 2002.

[10] Aporntewan, C. Building-block identification by simultaneity matrix. Doctoral dissertation, Computer Engineering, Chulalongkorn university. 2004.

[11] Aporntewan, C. and Chongstitvatana, P. Building-block identification by simultaneity matrix. Soft Computing 11 (April 2007): 541-548.

[12] Pelikan, M., Goldberg, D. E. and Cantú-Paz, E. BOA: The bayesian optimization algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, 1999.

[13] Pelikan, M. and Goldberg, D. E. Escaping hierarchical traps with competent genetic algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), 2001.

[14] Larrañaga, P. and Lozano, J. A., eds. Estimation of distribution algorithms: A new tool for evolutionary computation. New York: Springer, 2001.

[15] Pelikan, M., Sastry, K. and Cantú-Paz, E., eds. Scalable optimization via probabilistic modeling: From algorithms to applications. New York: Springer, 2006.

[16] Heckerman, D., Geiger, D. and Chickering, M. Learning bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Redmond, MA: Microsoft Research, 1994.

[17] Mitchell, T. M. Machine Learning. Singapore: McGraw-Hill, 1997.

[18] Aporntewan, C. and Chongstitvatana, P. Chi-square matrix: An approach for building-block identification. In Proceedings of 9th Asian Computing Science Conference, 2004.

[19] Lima, C. F., et al. Substructural neighborhoods for local search in the bayesian optimization algorithm. In Parallel Problem Solving from Nature. PPSN IX, 2006.

[20] Chaiyaratana, N., Piroonratana, T. and Sangkawelert, N. Effects of diversity control in single-objective and multi-objective genetic algorithms. Journal of Heuristics 13 (2007): 1-34.

[21] Grefenstette, J. J. Deception considered harmful. In Foundations of Genetic Algorithms 2, 1993.

[22] Menon, A., ed. Frontiers of evolutionary computation. New York: Kluwer Academic Publishers, 2004.

[23] Harik, G. R. Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA). Technical Report 99010, IlliGAL Technical Report, 1999.

[24] Thierens, D. and Goldberg, D. E. Mixing in genetic algorithms. In Proceedings of the Fifth International Conference on Genetic algorithms, 1993.

[25] De Bonet, J. S., Isbell, C. L. and Viola, P. MIMIC: Finding optima by estimating probability densities. In Advance in Neural Information Processing Systems, 1996.

[26] Pelikan, M. and Mühlenbein, H. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. K. Chawdhry (eds.), Advance in Soft Computing-Engineering Design and Manufacturing, pp. 521-535. London: Springer, 1999.

[27] Goldberg, D. E., Korb, B. and Deb, K. Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems 3 (1990): 493-530.

[28] Goldberg, D. E., Deb, K., Kargupta, H. and Harik, G. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In Proceedings of the 5th International Conference on Genetic Algorithm, 1993.

[29] Kargupta, H. The gene expression messy genetic algorithm. In Proceedings of Congress on Evolutionary Computation, 1996.

[30] Munetomo, M. and Goldberg, D. E. Identifying linkage by nonlinearity check. IlliGAL Report No. 98012. Urbana, IL: University of Illinois at Urbana-Champaign, 1998.

[31] Munetomo, M. and Goldberg, D. E. Identifying linkage groups by nonlinearity/non-monotonicity detection. In Proceedings of the Genetic and Evolutionary Computation Conference, 1999.

[32] Pelikan, M. Bayesian optimization algorithm: From single level to hierarchy. Doctoral dissertation, University of Illinois at Urbana-Champaign. 2002.

[33] Yu, T.-L., Goldberg, D. E., Yassine, A. and Chen, Y.-P. Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. In Proceedings of Artificial Neural Networks in Engineering (ANNIE 2003), 2003.

[34] Yu, T.-L. A matrix approach for finding extrema: Problems with modularity, hierarchy, and overlap. Doctoral dissertation, University of Illinois at Urbana-Champaign. 2006.

[35] Smith, J. and Fogarty, T. C. Recombination strategy adaptation via evolution of gene linkage. In Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, 1996.

[36] Harik, G. R. and Goldberg, D. E. Learning linkage. In Foundation of Genetic Algorithms 4, 1997.

[37] Fernandez, C. Integration analysis of product architecture to support effective team co-location. Master's Thesis, Massachusetts Institute of Technology. 1998.

[38] Whitfield, R., Smith, J. and Duffy, A. Identifying component modules. In Proceedings of the Seventh International Conference on Aritificial Intelligence in Design AID02, 2002.

[39] Rosenberg, R. S. Simulation of genetic populations with biochemical properties. Doctoral dissertation, University of Michigan. 1967.

[40] Schaffer, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, 1985.

[41] Jin, Y., Olhofer, M. and Sendhoff, B. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how?. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO, 2001.

[42] Parsopoulos, K. E. and Vrahatis, M. N. Particle swarm optimization method in multiobjective problems. In Proceedings of The 2002 ACM Symposium on Applied Computing, 2002.

[43] Fonseca, C. M. and Fleming, P. J. Multiobjective genetic algorithms. In IEE Colloquium on Genetic Algorithms for Control Systems Engineering, 1993.

[44] Horn, J., Nafpliotis, N. and Goldberg, D. E. A niched Pareto genetic algorithm for multiobjective optimization. In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, 1994.

[45] Goldberg, D. E. and Richardson, J. Genetic algorithms with sharing for multimodal function optimization. In Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA'87), 1987.

[46] Oei, C., Goldberg, D. E. and Chang, S.-J. Tournament selection, niching and the preservation of diversity. Technical Report 91011, Illinois Genetic Algorithms Laboratory (IlliGAL), 1991.

[47] Srinivas, N. and Deb, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2 (1994): 221-248.

[48] Khan, N., Goldberg, D. E. and Pelikan, M. Multi-objective bayesian optimization algorithm. Technical Report 2002009, Urbana, Illinois: Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2002.

[49] Zitzler, E. and Thiele, L. An evolutionary approach for multiobjective optimization: The strength Pareto approach. Technical Report 43, Computer Engineering and Networks Laboratory, ETH Zurich, 1998.

[50] Zitzler, E., Laumanns, M. and Thiele, L. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In Evolutionary methods for design, optimisation and control with application to industrial problems (EUROGEN 2001), 2002.

[51] Knowles, J. D. and Corne, D. W. The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimization. In Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99), 1999.

[52] Corne, D. W., Knowles, J. D. and Oates, M. J. The Pareto envelope-based selection algorithm for multiobjective optimization. In Proceedings of the Parallel Problem Solving from Nature VI Conference, 2000.

[53] Corne, D. W., Jerram, N. R., Knowles J. D. and Oates, M. J. PESA-II: Regionbased selection in evolutionary multiobjective optimization. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), 2001.

[54] Andersson, J. and Krus, P. Metamodel representations for robustness assessment in multiobjective optimization. In International Conference on Engineering Design Iced 01 GLASGOW, 2001.

[55] Andersson, J. Multiobjective optimization in engineering design. Doctoral dissertation, Division of Fluid and Mechanical Engineering Systems, Linköpings University. 2001.

[56] Gruneninger, T. and Wallace, D. Multi-modal optimization using genetic algorithms. Technical Report 96.02,CADlab, MIT, Cambridge, 1996.

[57] Van Veldhuizen, D. A. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Doctoral dissertation, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology. 1999.

[58] Zydallis, J. B., Van Veldhuizen, D. A. and Lamont, G. B. A statistical comparison of multiobjective evolutionary algorithms including the MOMGA-II. In First International Conference on Evolutionary Multi-Criterion Optimization, 2001.

[59] Day, R. O. and Lamont, G. B. An effective explicit building block MOEA, the MOMGA-IIa. In 2005 IEEE Congress on Evolutionary Computation (CEC'2005), 2005.

[60] Ho, S.-Y., Shu, L.-S. and Chen, J.-H. Intelligent evolutionary algorithms for large parameter optimization problems. IEEE Transactions on Evolutionary Computation 8 (2004): 522–541.

[61] Lobo, F. G. and Lima, C. F. Revisiting evolutionary algorithms with on-the-fly population size adjustment. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO'06, 2006.

[62] Clergue, M. and Collard, P. GA-hard functions built by combination of trap functions. In Proceedings of the 2002 Congress on Evolutionary Computation, 2002 (CEC '02), 2002.

[63] Nijssen, S. and Back, T. An analysis of the behavior of simplified evolutionary algorithms on trap functions. IEEE Transactions on Evolutionary Computation 7 (2003): 11-12.

[64] Aporntewan, C. and Chongstitvatana, P. Simultaneity matrix for solving hierarchically decomposable functions. In Proceedings of the Genetic and Evolutionary Computation (GECCO 2004), 2004.

[65] Punyaporn, W., Ponsawat, J. and Chongstitvatana, P. Solving additively decomposable functions by building blocks identification. In The 4th International Joint Conference on Computer Science and Software Engineering (JCSSE2007), 2007.

[66] Kursawe, F. A variant of evolution strategies for vector optimization. In Parallel Problem Solving from Nature. 1st Workshop, PPSN I, 1991.

[67] Deb, K. Construction of test problems for multi-objective optimization. In Proceedings of the Genetic and Evolutionary Computation Conference 99, 1999.

[68] Van Veldhuizen, D. A. and Lamont, G. B. On measuring multiobjective evolutionary algorithm performance. In 2000 Congress on Evolutionary Computation, 2000.

[69] Zitzler, E., Deb, K. and Thiele, L. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8 (2000): 173-195.

[70] Binh, T. T. and Korn, U. Multiobjective evolution strategy for constrained optimization problems. In Proceedings of the 15th IMACS World Congress on Scientific Computation, 1997.

[71] Deb, K. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evolutionary Computation 7 (1999): 205-230.

[72] Horn, J. and Nafpliotis, N. Multiobjective optimization using the niched Pareto genetic algorithm. IlliGAL Report No. 93005, Urbana, IL: University of Illinois at Urbana-Champaign, 1993.

[73] Knowles, J. D. and Corne, D. W. Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computation 8 (2000): 149-172.

[74] Deb, K. and Goldberg, D. E. Analyzing deception in trap functions. In Foundation of Genetic Algorithms 2, 1993.

[75] Ponsawat, J., Punyaporn, W., Nachol, N. and Chongstitvatana, P. Solving multi-objective problems with building blocks identification. In 7th International Symposium on Communications and Information Technologies (ISCIT2007), 2007.

[76] Tsuji, M., Munetomo, M. and Akama, K. A crossover for complex building blocks overlapping. In Proceedings of the Genetic and Evolutionary Computation, 2006.

ประวัติผู้เขียนวิทยานิพนธ์

นายจิระเดช พลสวัสดิ์ เกิดวันที่ 8 พฤศจิกายน พ.ศ. 2520 ที่จังหวัดอุบลราชธานี สำเร็จการศึกษาปริญญาวิศกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จากมหาวิทยาลัยขอนแก่น ในปีการศึกษา 2542 และสำเร็จการศึกษาในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2546 หลังจากนั้นได้เข้าศึกษาในหลักสูตรวิศวกรรมศาสตรดุษฎีบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2547 โดยได้รับทุนอุดหนุนการศึกษา โครงการพัฒนาอาจารย์สาขาขาดแคลนเพื่อศึกษาในประเทศ ตามความต้องการของมหาวิทยาลัยขอนแก่น