การปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยกลไกการเรียนรู้ด้วยตนเอง สำหรับปัญหาออฟติไมเซชัน

A MODIFIED MARRIAGE IN HONEY-BEES OPTIMIZATION WITH A SELF-ORGANIZING ABILITY FOR OPTIMIZATION PROBLEMS

พัชรวดี พูลสำราญ

PATCHARAWADEE POOLSAMRAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2555

KMITL-2012-IT-D-001-002

A MODIFIED MARRIAGE IN HONEY-BEES OPTIMIZATION WITH A SELF-ORGANIZING ABILITY FOR OPTIMIZATION PROBLEMS

PATCHARAWADEE POOLSAMRAN

A THESIS SUBMITTED IN PATIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2012

KMITL-2012-IT-D-001-002

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยกลไกการเรียนรู้ด้วยตนเองสำหรับปัญหาออพติไมเซชัน

A Modified Marriage in Honey-bees Optimization with a Self-Organizing Ability for Optimization Problems

นักศึกษา นางสาวพัชรวดี พูลสำราญ

รหัสประจำตัว 51066302

ปริญญา ปรัชญาดุษฎีบัณฑิต

สาขาวิชา เทคโนโลยีสารสนเทศ

อาจารย์ที่ปรึกษาวิทยานิพนธ์ รองศาสตราจารย์ ดร.อาริต ธรรมโน

คณะกรรมการสอบวิทยานิพนธ์	ลายมือชื่อ
ศาสตราจารย์ ดร.ชิดชนก เหลือสินทรัพย์	
รองศาสตราจารย์ ดร.วรพจน์ กรีสุระเดช	
รองศาสตราจารย์ ดร.อาริต ธรรมโน	
รองศาสตราจารย์ ดร.พรฤดี เนติโสภากุล	
ผู้ช่วยศาสตราจารย์ ดร.กันต์พงษ์ วรรัตน์ปัญญา	

วัน / เดือน/ ปี ที่สอบ วันอังคารที่ 31 กรกฎาคม 2555 เวลา 10.00 น.

สถานที่สอบ ณ ห้อง M04 คณะเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศรับรองแล้ว

(รองศาสตราจารย์ ดร.จันทร์บูรณ์ สถิตวิริยวงศ์)

คณบดีคณะเทคโนโลยีสารสนเทศ

วันที่ 28 **เดือน** สิงหาคม **พ.ศ.** 2555

หัวข้อวิทยานิพนธ์	การปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยกลไกการเรียนรู้ด้วยตนเองสำหรับปัญหาออฟติไมเซชัน
นักศึกษา	นางสาวพัชรวดี พูลสำราญ
รหัสนักศึกษา	51066302
ปริญญา	ปรัชญาดุษฎีบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
พ.ศ.	2555
อาจารย์ที่ปรึกษา	รศ. ดร. อาริต ธรรมโน

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอการปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยกลไกการเรียนรู้ด้วยตนเองเพื่อใช้ในการแก้ปัญหาออฟติไมเซชัน โดยอัลกอริทึมที่นำเสนอแบ่งเป็น 2 โมเดลคือ อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด ซึ่งทั้งสองโมเดลได้มีการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม ดังนี้ 1) ผึ้งนางพญาสามารถเลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นได้ โดยทำการแบ่งพื้นที่ในการค้นหาคำตอบเป็นพื้นที่รังของผึ้งนางพญาและทำการจัดกลุ่มผึ้งตัวผู้ให้เป็นสมาชิกของรังใดรังหนึ่งด้วยวิธีการแบ่งกลุ่มแบบฟัซซี และ 2) การคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง โดยได้มีการกำหนดช่วงชีวิตของผึ้งตัวผู้ไว้เป็นค่าคงที่ ถ้าผึ้งตัวผู้ได้รับการผสมพันธุ์จะตายลง ส่วนผึ้งตัวผู้ที่ไม่ได้รับการผสมพันธุ์สามารถดำรงชีวิตต่อไปจนกว่าอายุครบตามที่กำหนดไว้ และเมื่อมีผึ้งตัวผู้ตายลงจึงทำการสุ่มเลือกตัวอ่อนผึ้งขึ้นมาแทนที่

การทดลองในงานวิจัยนี้ได้วัดประสิทธิภาพของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขด้วยฟังก์ชันมาตรฐานจำนวน 6 ฟังก์ชันและได้เปรียบเทียบผลการทดลองกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม ส่วนอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดได้ทดสอบกับปัญหาการเดินทางของพนักงานขายจำนวน 2 ข้อมูลและได้เปรียบเทียบผลการทดลองกับอัลกอริทึม PGACS ผลการทดลองของทั้งสองโมเดลสรุปได้ว่า อัลกอริทึมที่นำเสนอมีประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดสูงกว่าอัลกอริทึมที่นำมาเปรียบเทียบด้วย

Thesis Title	A modified marriage in honey-bees optimization with a self-organizing ability for optimization problems
Student	Miss Patcharawadee Poolsamran
Student ID.	51066302
Degree	Doctor of Philosophy
Programme	Information Technology
Year	2012
Thesis Advisor	Assoc. Prof. Dr. Arit Thammano

ABSTRACT

This thesis proposes a modified marriage in honey-bees optimization with a self-organizing ability, which is an adaptation of Abbass's marriage in honey-bee optimization (MBO), for solving the optimization problems. The proposed algorithm is divided into 2 models based on the data type of problem; the numerical optimization problem and the combinatorial optimization problem. The proposed algorithm is an improvement over the original MBO algorithm in two respects. Firstly, the proposed algorithm divides the problem space into several colonies, each of which has its own queen. The drones are assigned to the queens' colonies by using the fuzzy c-means algorithm. The queen is to mate with some drones from other colonies. Secondly, all drones have their own life span. The drones that mate with the queen immediately die whereas the unmated ones stay alive until their life span is reached. The replacements for the dead drones are randomly selected one by one from broods.

The numerical proposed algorithm has been evaluated by six benchmark functions and compared with the original MBO algorithm whereas the combinatorial proposed algorithm has been evaluated by two datasets of the travelling salesman problem and compared with the Parallelized genetic ant colony systems (PGACS) algorithm. The experimental results show that both proposed model are very effective in solving the optimization problems as compared with the other algorithms.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ. ดร. อาริต ธรรมโน ที่ให้ความรู้ ให้คำชี้แนะในการคิดและการแก้ไขปัญหา ให้ความช่วยเหลือตลอดจนให้กำลังใจที่ดีแก่ข้าพเจ้า

ขอกราบขอบพระคุณคณาจารย์คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่านที่ได้ประสิทธิ์ประสาทวิชาให้แก่ข้าพเจ้า

ขอขอบพระคุณกระทรวงวิทยาศาสตร์และเทคโนโลยีที่ได้มอบทุนรัฐบาลกระทรวงวิทยาศาสตร์และเทคโนโลยีให้แก่ข้าพเจ้าและขอขอบคุณเจ้าหน้าที่งานนักเรียนทุน สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ ที่ให้การสนับสนุนและดูแลเอาใจใส่เป็นอย่างดีแก่ข้าพเจ้า

ขอขอบคุณเพื่อน พี่ น้องทุกคนที่คอยให้กำลังใจแก่ข้าพเจ้า รวมทั้งเจ้าหน้าที่คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังและเจ้าหน้าที่มหาวิทยาลัยบูรพาทุกท่านที่คอยประสานงานให้งานสำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณบิดา มารดาและขอขอบคุณครอบครัวที่คอยให้ความรัก ให้คำปลอบโยน ให้กำลังใจที่อบอุ่นแก่ข้าพเจ้า

คุณงามความดีอันใดซึ่งเกิดจากวิทยานิพนธ์นี้ ข้าพเจ้าขอมอบให้แก่บิดา มารดา อันเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่านด้วย

พัชรวดี พูลสำราญ

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VIII

สารบัญรูป IX

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญของปัญหา 1

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา 2

1.3 ขอบเขตของการวิจัย 2

1.4 ประโยชน์ที่คาดว่าจะได้รับ 3

1.5 ขั้นตอนของการดำเนินงานวิจัย 3

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 4

2.1 ปัญหาออฟติไมเซชัน 4

2.1.1 ประเภทของปัญหาออฟติไมเซชันที่แบ่งตามจำนวนข้อจำกัด 5

2.1.2 ประเภทของปัญหาออฟติไมเซชันที่แบ่งตามจำนวนของฟังก์ชันเป้าหมาย 6

2.1.3 ประเภทของปัญหาออฟติไมเซชันที่แบ่งตามชนิดข้อมูลของพารามิเตอร์ 7

2.2 ทฤษฎีที่เกี่ยวข้อง 9

2.2.1 ปัญหาการเดินทางของพนักงานขาย 9

2.2.2 เทคนิคที่ใช้ในการแก้ปัญหาออฟติไมเซชัน 12

2.2.3 ชีววิทยาของผึ้ง 12

2.3 งานวิจัยที่เกี่ยวข้อง 13

2.3.1 MBO: Marriage in honey-bees optimization (a haplometrosis polygynous swarming approach) 13

2.3.2 Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization 17

2.3.3 Optimal water distribution network design with honey-bee mating optimization 19

สารบัญ (ต่อ)

หน้า

2.3.4 An intelligent genetic algorithm designed for global optimization of multi-minima functions 24
2.3.5 Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation 32
2.3.6 A hybrid genetic algorithm and particle swarm optimization for multimodel functions 37
2.3.7 Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems 40
2.3.8 Minimizing the multimodal functions with ant colony optimization approach 42
2.3.9 Parallelized genetic ant colony systems for solving the traveling salesman problem 45
บทที่ 3 การปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยกลไกการเรียนรู้ด้วยตนเอง สำหรับปัญหาออฟติไมเซชัน 50
3.1 การวิเคราะห์ปัญหาของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 50
3.1.1 จุดเด่นของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 51
3.1.2 จุดด้อยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 52
3.2 แนวทางการปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 52
3.3 โครงสร้างการทำงานของอัลกอริทึมที่นำเสนอ 54
3.4 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข 56
3.4.1 การแทนค่าจีโนไทป์ของผึ้ง 56
3.4.2 การคำนวณค่าฟิตเนส 57
3.4.3 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข 58
3.4.4 ขั้นตอนการทำงานของโอเปอร์เรเตอร์ 64
3.5 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด 66
3.5.1 การแทนค่าจีโนไทป์ของผึ้ง 66
3.5.2 การคำนวณค่าฟิตเนส 66
3.5.3 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด 67

สารบัญ (ต่อ)

หน้า

3.5.4 ขั้นตอนการทำงานของโอเปอร์เรเตอร์ 73

บทที่ 4 การทดสอบการปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยกลไกการเรียนรู้
ด้วยตนเองสำหรับปัญหาออฟติไมเซชัน 78

4.1 ข้อมูลที่ใช้ในการทดสอบ 79

4.1.1 ฟังก์ชันมาตรฐานสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข 79

4.1.2 ชุดข้อมูลมาตรฐานสำหรับปัญหาออฟติไมเซชันเชิงการจัด 78

4.2 เครื่องมือที่ใช้ในการวัดประสิทธิภาพ 84

4.3 การทดสอบเพื่อศึกษาปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง 85

4.3.1 การกำหนดค่าพารามิเตอร์ 87

4.3.2 ผลการทดลอง 87

4.3.3 สรุปผลการทดลอง 91

4.4 การทดสอบเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรัง
(แบบกำหนดจำนวนผึ้งนางพญา) 96

4.4.1 การกำหนดค่าพารามิเตอร์ 96

4.4.2 ผลการทดลอง 98

4.4.3 สรุปผลการทดลอง 100

4.5 การทดสอบเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรัง
(แบบไม่มีการกำหนดจำนวนผึ้งนางพญา) 105

4.5.1 การกำหนดค่าพารามิเตอร์ 105

4.5.2 ผลการทดลอง 105

4.5.3 สรุปผลการทดลอง 108

4.6 การทดสอบอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข 114

4.6.1 การกำหนดค่าพารามิเตอร์ 114

4.6.2 ผลการทดลอง 116

4.6.3 สรุปผลการทดลอง 117

4.7 การทดสอบอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด 123

4.7.1 การกำหนดค่าพารามิเตอร์ 123

4.7.2 ผลการทดลอง 123

สารบัญ (ต่อ)

หน้า

4.7.3 สรุปผลการทดลอง 125
4.8 สรุปผลการทดสอบอัลกอริทึมที่นำเสนอ 127
บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ 131
5.1 สรุปงานวิจัย 131
5.2 สรุปผลการทดสอบ 132
5.3 จุดเด่นของงานวิจัย 133
5.4 ปัญหาที่พบในการงานวิจัย 134
5.5 แนวทางในการพัฒนาต่อในอนาคต 134
เอกสารอ้างอิง 135
ภาคผนวก 138
ภาคผนวก ก ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่ 139
ประวัติผู้เขียน 165

สารบัญตาราง

ตารางที่ หน้า

2.1 จำนวนคำตอบที่เป็นไปได้ของปัญหาการเดินทางของพนักงานขายแบบสมมาตร 10

2.2 การกำหนดค่าพารามิเตอร์ในโอเปอร์เรชันการหมุน 34

4.1 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม MBO และอัลกอริทึม MBO-newDrone 87

4.2 ผลการทดลองของอัลกอริทึม MBO 88

4.3 ผลการทดลองของอัลกอริทึม MBO-newDrone 90

4.4 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม MBO และอัลกอริทึม MBO-fixSwarming 96

4.5 ผลการทดลองของอัลกอริทึม MBO-fixSwarming 99

4.6 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม MBO และอัลกอริทึม SMBO-oldDrone 105

4.7 ผลการทดลองของอัลกอริทึม SMBO-oldDrone 108

4.8 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม MBO และอัลกอริทึม SMBO-numerical 114

4.9 ผลการทดลองของอัลกอริทึม SMBO-numerical 117

4.10 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม SMBO-combinatorial 123

4.11 การเปรียบเทียบผลการทดลองของอัลกอริทึม PGACS และอัลกอริทึมที่นำเสนอ SMBO-combinatorial สำหรับข้อมูล ST70 124

4.12 การเปรียบเทียบผลการทดลองของอัลกอริทึม PGACS และอัลกอริทึมที่นำเสนอ SMBO-combinatorial สำหรับข้อมูล EIL101 125

สารบัญรูป

รูปที่ หน้า

2.1 การหาคำตอบที่เหมาะสมที่สุดด้วยวิธีการเขียนกราฟ 8
2.2 ตัวอย่างปัญหาการเดินทางของพนักงานขายแบบสมมาตร 11
2.3 การแทนค่าจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้แบบแฮพพลอยด์ 13
2.4 โอเปอร์เรชันครอสโอเวอร์แบบแฮพพลอยด์ 15
2.5 โครงสร้างการทำงานของโมเดล IGA 30
2.6 โครงสร้างการทำงานของโมเดล RQGA 35
2.7 โครงสร้างการทำงานของโมเดล GA-PSO 37
2.8 โครงสร้างการทำงานของโมเดล PSO-GA 41
2.9 การจำลองการเคลื่อนที่แบบกระโดดของมด 44
2.10 ขั้นตอนการเลือกโครโมโซมมดของอัลกอริทึม PGACS 47
3.1 โครงสร้างรังแบบมีผึ้งนางพญาหลายตัวภายในรังเดียวกันของอัลกอริทึม การผสมพันธุ์ของผึ้งแบบดั้งเดิม 51
3.2 โครงสร้างรังแบบมีหลายรังและมีผึ้งนางพญาหลายตัวของอัลกอริทึมที่นำเสนอ 53
3.3 โครงสร้างการทำงานของอัลกอริทึมที่นำเสนอ 55
3.4 การแทนค่าจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้สำหรับปัญหาออฟติไมเซชันเชิงตัวเลข 56
3.5 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข 63
3.6 โอเปอร์เรเตอร์ครอสโอเวอร์แบบสองจุด 64
3.7 การแทนค่าจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้สำหรับปัญหาออฟติไมเซชันเชิงการจัด 66
3.8 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด 72
3.9 การทำงานของโอเปอร์เรเตอร์ครอสโอเวอร์แบบวงกลม 73
3.10 การทำงานของโอเปอร์เรเตอร์มิวเตชันแบบแลกเปลี่ยนค่าในตำแหน่งติดกัน 74
3.11 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบเคลื่อนตำแหน่ง 75
3.12 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบแลกเปลี่ยนตำแหน่ง 75
3.13 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบแทรกตำแหน่ง 76
3.14 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบกลับค่าอย่างง่าย 76
3.15 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบกลับค่า 77
3.16 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบเรียงสลับ 77
4.1 กราฟของฟังก์ชัน Michalewicz 79

สารบัญรูป (ต่อ)

รูปที่ หน้า

4.2 กราฟของฟังก์ชัน Rastrigin 80

4.3 กราฟของฟังก์ชัน Rosenbrocks' valley 80

4.4 กราฟของฟังก์ชัน Sphere 81

4.5 กราฟของฟังก์ชัน Weighted sphere 82

4.6 กราฟของฟังก์ชัน Goldstein-Price 82

4.7 ตำแหน่งเมืองและข้อมูลเส้นทางที่สั้นที่สุดของข้อมูล ST70 83

4.8 ตำแหน่งเมืองและข้อมูลเส้นทางที่สั้นที่สุดของข้อมูล EIL101 84

4.9 เปรียบเทียบโครงสร้างการทำงานระหว่างอัลกอริทึม MBO และ
อัลกอริทึม MBO-newDrone 86

4.10 กราฟแสดงการเปรียบเทียบผลการทดลองที่ 1 ระหว่างอัลกอริทึม MBO และ
อัลกอริทึม MBO-newDrone 95

4.11 เปรียบเทียบโครงสร้างการทำงานระหว่างอัลกอริทึม MBO และ
อัลกอริทึม MBO-fixSwarming 97

4.12 กราฟแสดงการเปรียบเทียบผลการทดลองระหว่างอัลกอริทึม MBO และ
อัลกอริทึม MBO-fixSwarming 104

4.13 เปรียบเทียบโครงสร้างการทำงานของอัลกอริทึม MBO และ
อัลกอริทึม SMBO-oldDrone 106

4.14 จำนวนของผึ้งนางพญาจากการรันอัลกอริทึม SMBO-oldDrone 112

4.15 กราฟแสดงการเปรียบเทียบผลการทดลองระหว่างอัลกอริทึม MBO และ
อัลกอริทึม SMBO-oldDrone 113

4.16 เปรียบเทียบโครงสร้างการทำงานของอัลกอริทึม MBO และ
อัลกอริทึม SMBO-numerical 115

4.17 จำนวนของผึ้งนางพญาจากการรันอัลกอริทึม SMBO-numerical 121

4.18 กราฟแสดงการเปรียบเทียบผลการทดลองระหว่างอัลกอริทึม MBO และ
อัลกอริทึม SMBO-numerical 122

4.19 กราฟแสดงการเปรียบเทียบผลการทดลองของข้อมูล ST70 126

4.20 กราฟแสดงการเปรียบเทียบผลการทดลองของข้อมูล EIL101 127

4.21 กราฟแสดงการเปรียบเทียบประสิทธิภาพของแต่ละอัลกอริทึมโดยแยกตามฟังก์ชัน 129

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ศาสตร์ด้านปัญญาประดิษฐ์ได้รับการพัฒนามาอย่างต่อเนื่องและได้รับการยอมรับว่ามีประสิทธิภาพสูงในการแก้ปัญหาที่มีความซับซ้อนทั้งในงานด้านการพยากรณ์ (prediction) การจัดกลุ่มข้อมูล (classification) การแบ่งกลุ่มข้อมูล (clustering) ตลอดจนงานทางด้านออฟติไมเซชัน (optimization) ในช่วงหลายทศวรรษที่ผ่านมานี้ได้มีนักวิจัยจำนวนมากหันมาให้ความสนใจในการพัฒนาและปรับปรุงอัลกอริทึมทางด้านปัญญาประดิษฐ์ไว้มากมาย โดยเฉพาะอย่างยิ่งในกลุ่มอัลกอริทึมที่เกิดจากการเลียนแบบความชาญฉลาดทางชีววิทยา (biologically inspired algorithms) ตัวอย่างอัลกอริทึมในกลุ่มนี้คือ โครงข่ายประสาทเทียม (neural network) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบการทำงานของสมองมนุษย์ จีเนติกอัลกอริทึม (genetic algorithm) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบวิวัฒนาการทางพันธุกรรมของสิ่งมีชีวิต อัลกอริทึมอาณานิคมมด (ant colony optimization) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการหาอาหารของมด และอัลกอริทึมพาร์ทิเคิลสวอร์ม (particle swarm optimization) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการหาอาหารของฝูงนกหรือฝูงปลา อัลกอริทึมที่กล่าวมาทั้งหมดนี้ล้วนเป็นอัลกอริทึมที่มีชื่อเสียงทั้งสิ้น และเมื่อปี พ.ศ. 2535 Hussein A. Abbass [1] ได้เสนออัลกอริทึมใหม่ คือ อัลกอริทึมการผสมพันธุ์ของผึ้ง (marriage in honey-bee optimization) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการผสมพันธุ์ระหว่างผึ้งนางพญากับผึ้งตัวผู้เพื่อใช้ในการแก้ปัญหาความสอดคล้องของนิพจน์ (propositional satisfiability problem: SAT) ต่อมาได้มีนักวิจัยหลายท่านได้นำอัลกอริทึมนี้ไปประยุกต์ใช้แก้ปัญหาต่างๆ ตัวอย่างเช่น Haddad Bozorg [2] ได้นำอัลกอริทึมการผสมพันธุ์ของผึ้งไปประยุกต์ใช้ในการประมาณค่าความต้องการกักเก็บน้ำ ซึ่งเป็นส่วนหนึ่งของการบริหารและการจัดการทรัพยากรน้ำ โดยได้ทำการเปรียบเทียบประสิทธิภาพกับจีเนติกอัลกอริทึม ผลการทดลองพบว่า อัลกอริทึมการผสมพันธุ์ของผึ้งมีประสิทธิภาพในการประมาณค่าสูงกว่าจีเนติกอัลกอริทึม นอกจากนี้ยังได้นำอัลกอริทึมดังกล่าวไปทดสอบกับฟังก์ชันออฟติไมเซชันทั้งแบบมีข้อจำกัด (constraint) และไม่มีข้อจำกัด ผลการทดสอบแสดงให้เห็นว่า อัลกอริทึมการผสมพันธุ์ของผึ้งให้ค่าความผิดพลาดในการประมาณค่าฟังก์ชันเพียงเล็กน้อย ดังนั้น จากการศึกษาการประยุกต์ใช้อัลกอริทึมการผสมพันธุ์ของผึ้งในช่วงที่ผ่านมาสามารถกล่าวได้ว่า อัลกอริทึมการผสมพันธุ์ของผึ้งเป็นอัลกอริทึมหนึ่งที่มีประสิทธิภาพในการประยุกต์ใช้กับปัญหาออฟติไมเซชัน

งานวิจัยนี้มีจุดมุ่งหมายเพื่อนำเสนออัลกอริทึมที่เกิดจากการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่นำเสนอโดย Hussein A. Abbass ให้มีกลไกการ

เรียนรู้ด้วยตนเองเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันให้มีประสิทธิภาพดียิ่งขึ้น โดยได้ทำการปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 2 แง่มุม คือ 1) กำหนดให้ผึ้งนางพญามีการเลือกคู่เพื่อผสมพันธุ์กับผึ้งตัวผู้จากรังอื่น ยกเว้นกรณีที่มีรังผึ้งเพียงรังเดียวที่อนุญาตให้ผึ้งนางพญาสามารถเลือกผสมพันธุ์กับผึ้งตัวผู้ของรังตนเองได้ โดยได้ทำการแบ่งพื้นที่ในการค้นหาคำตอบเป็นพื้นที่รังของผึ้งนางพญาและมีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังจึงทำให้เกิดการแข่งขันในการสร้างรังระหว่างผึ้งนางพญาและตัวอ่อนผึ้ง โดยผึ้งที่มีค่าฟิตเนสสูงมีโอกาสในการสร้างรังที่มีขนาดใหญ่กว่าผึ้งที่มีค่าฟิตเนสต่ำกว่า และ 2) การคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง โดยได้มีการกำหนดช่วงชีวิตของผึ้งตัวผู้ไว้ เมื่อผึ้งตัวผู้ได้รับการผสมพันธุ์จะตายลง ส่วนผึ้งตัวผู้ที่ไม่ได้รับการผสมพันธุ์สามารถดำรงชีวิตต่อไปจนกว่าอายุครบตามที่กำหนดไว้ แต่เมื่อมีผึ้งตัวผู้ตายลงจะทำการสุ่มเลือกตัวอ่อนผึ้งขึ้นมาแทนที่ จากนั้นผึ้งตัวผู้ทั้งหมดจะถูกจัดกลุ่มให้เป็นสมาชิกในรังของผึ้งนางพญาตามวิธีการแบ่งกลุ่มแบบฟัซซี การทดสอบประสิทธิภาพของอัลกอริทึมที่นำเสนอได้แบ่งเป็น 2 ส่วนคือ ส่วนแรกเป็นการทดสอบอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข (numerical optimization problem) โดยได้ทำการทดสอบกับฟังก์ชันมาตรฐานและเปรียบเทียบผลการทดลองกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและส่วนที่สองเป็นการทดสอบอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด (combinatorial optimization problem) โดยได้ทำการทดสอบกับปัญหาการเดินทางของพนักงานขาย (travelling salesman problem) และเปรียบเทียบผลการทดลองกับอัลกอริทึม PGACS (Parallelized genetic ant colony systems) ซึ่งนำเสนอโดย Chen และ Chien [3]

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1. เพื่อวิเคราะห์หาจุดเด่นและจุดด้อยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

2. เพื่อวิเคราะห์ปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งและปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

3. เพื่อปรับปรุงและพัฒนาอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมให้มีประสิทธิภาพในแก้ปัญหาออฟติไมเซชันทั้งแบบเชิงตัวเลขและเชิงการจัดได้มากขึ้น

1.3 ขอบเขตการวิจัย

1. งานวิจัยนี้นำเสนออัลกอริทึมที่เกิดจากการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันทั้งแบบเชิงตัวเลขและเชิงการจัดที่มีฟังก์ชันเป้าหมายหนึ่งฟังก์ชัน

2. งานวิจัยนี้ทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอด้วยฟังก์ชันออฟติ-ไมเซชันแบบไม่มีข้อจำกัด

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้ความเข้าใจอัลกอริทึมที่เกิดจากการเลียนแบบความชาญฉลาดทางชีววิทยา

2. มีความรู้เกี่ยวกับเครื่องมือที่ใช้ในการวัดประสิทธิภาพการทำงานของอัลกอริทึมที่ใช้ในการแก้ปัญหาออฟติไมเซชัน

3. ได้อัลกอริทึมที่มีประสิทธิภาพสำหรับการแก้ปัญหาออฟติไมเซชัน

4. ได้ต้นแบบงานวิจัยสำหรับการศึกษาและพัฒนางานทางด้านปัญญาประดิษฐ์

1.5 ขั้นตอนการศึกษา

1. ศึกษางานวิจัยที่นำเสนออัลกอริทึมที่เลียนแบบความชาญฉลาดทางชีววิทยาเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันและปัญหาอื่นๆ ที่เกี่ยวข้อง

2. ศึกษาทฤษฎีที่เกี่ยวข้องกับการพัฒนาอัลกอริทึมสำหรับการแก้ปัญหาออฟติไมเซชัน เช่น เครื่องมือที่ใช้การวัดประสิทธิภาพของการทำงานของอัลกอริทึม ลักษณะของข้อมูลที่ใช้ในการทดสอบ เป็นต้น

3. วางแผนและออกแบบโมเดลใหม่

4. พัฒนาโปรแกรมตามรูปแบบของโมเดลที่ได้ออกแบบไว้โดยใช้โปรแกรม MATLAB เวอร์ชัน 7.6.0.324

5. ออกแบบการทดลองและจัดเตรียมข้อมูลทดสอบเพื่อใช้ในการวัดประสิทธิภาพโมเดลที่นำเสนอ จากนั้นจึงทำการทดสอบตามแผนการทดลองที่กำหนดไว้

6. สรุปผลการทดลอง

7. จัดทำรายงานการวิจัย

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้ได้แบ่งเนื้อหาออกเป็น 3 ส่วน ดังนี้ ส่วนแรกกล่าวถึงปัญหาออฟติไมเซชันและประเภทของปัญหาออฟติไมเซชัน ส่วนที่ 2 อธิบายถึงทฤษฎีพื้นฐานที่เกี่ยวข้องกับงานวิจัยนี้ และส่วนที่ 3 เป็นรายละเอียดของงานวิจัยที่เกี่ยวข้อง

2.1 ปัญหาออฟติไมเซชัน

ปัญหาออฟติไมเซชัน (optimization problem) คือ การหาค่าพารามิเตอร์ที่เหมาะสมที่สุดที่ทำให้ค่าฟังก์ชันเป้าหมาย (objective function) มีค่าดีที่สุดภายใต้ข้อจำกัด (constraint) ของปัญหานั้นๆ ซึ่งค่าฟังก์ชันเป้าหมายที่ดีที่สุดอาจเป็นค่าต่ำสุดหรือค่าสูงสุดขึ้นอยู่กับปัญหาที่ศึกษา [4][5] ปัญหาออฟติไมเซชันมีองค์ประกอบ 3 ส่วน [4] ดังนี้

- $\bar{x}$ แทนเซตของตัวแปรตัดสินใจ (decision variables) หรือพารามิเตอร์ ซึ่งเป็นได้ทั้งเลขจำนวนเต็ม ค่าไบนารี (0,1) เลขจำนวนจริงหรือค่าข้อมูลไม่ต่อเนื่อง
- $f(\cdot)$ แทนฟังก์ชันเป้าหมายที่มีวัตถุประสงค์เพื่อหาค่าต่ำสุดหรือค่าสูงสุด โดยฟังก์ชันเป้าหมายที่มีค่าสูงสุดมีค่าเท่ากับส่วนกลับของฟังก์ชันเป้าหมายที่มีค่าต่ำสุดเสมอ (maximize $f(\cdot)$ = minimize $-f(\cdot)$)
- g_i, h_j แทนข้อจำกัดแบบเท่ากัน (equality constraint) และแบบไม่เท่ากัน (inequality constraint) ตามลำดับ

โดยพารามิเตอร์และฟังก์ชันเป้าหมายเป็นองค์ประกอบหลักของปัญหาออฟติไมเซชันทุกประเภท ในขณะที่ข้อจำกัดของปัญหาเป็นเงื่อนไขที่อาจมีหรือไม่มีก็ได้ขึ้นอยู่กับปัญหานั้นๆ จากนิยามของปัญหาออฟติไมเซชันนี้ เป้าหมายของการแก้ปัญหาออฟติไมเซชันจึงหมายถึงการค้นหาคำตอบที่เหมาะสมที่สุด (optimal solution) ซึ่งเขียนแทนด้วย $\bar{x}^*$ โดยที่ $f(\bar{x}^*) \le f(\bar{x})$ สำหรับทุกๆ ค่าของ $\bar{x}$ ที่เป็นไปได้ทั้งหมด

ปัญหาออฟติไมเซชันเป็นปัญหาที่ได้รับความสนใจเป็นอย่างมากทั้งในทางทฤษฎีและทางปฏิบัติ เนื่องจากความต้องการในการค้นหาคำตอบที่เหมาะสมที่สุดเป็นปัญหาที่พบอยู่ในหลากหลายสาขาวิชา เช่น งานด้านวิศวกรรม งานด้านอุตสาหกรรม งานด้านเศรษฐศาสตร์และการลงทุน งานด้านการขนส่งและโลจิสติกส์ เป็นต้น สำหรับตัวอย่างปัญหาที่น่าสนใจ เช่น การคำนวณหาขนาดที่เหมาะสมที่สุดของชิ้นส่วนในการออกแบบอุปกรณ์ การค้นหาเส้นทางการกระจายสินค้าที่เหมาะสมที่สุดระหว่างศูนย์กระจายสินค้าและลูกค้า การจัดตารางโครงการที่เหมาะสมที่สุด เป็นต้น ทั้งนี้เพื่อทำให้ต้นทุนในการผลิตมีค่าต่ำสุดหรือได้ผลตอบแทนสูงที่สุด

2.1.1 ประเภทของปัญหาออฟติไมเซชันที่แบ่งตามจำนวนข้อจำกัด

ปัญหาออฟติไมเซชันสามารถแบ่งได้เป็น 2 ประเภทตามจำนวนข้อจำกัด [4] คือ ปัญหาออฟติไมเซชันแบบไม่มีข้อจำกัด (unconstrainted optimization problem) และปัญหาออฟติไมเซชันแบบมีข้อจำกัด (constrainted optimization problem) โดยปัญหาออฟติไมเซชันแบบไม่มีข้อจำกัด คือ การหาค่าพารามิเตอร์ที่เหมาะสมที่สุดที่ทำให้ฟังก์ชันเป้าหมายมีค่าต่ำสุดหรือค่าสูงสุด โดยไม่มีการกำหนดขอบเขตของค่าพารามิเตอร์ รูปแบบทั่วไปของปัญหาออฟติไมเซชันแบบไม่มีข้อจำกัด ดังสมการที่ 2.1

$$\text{minimize} \quad f(\bar{x}) = f(x_1, x_2, ..., x_n) \tag{2.1}$$

เมื่อ $\bar{x}$ คือเซตของพารามิเตอร์ที่มีสมาชิกจำนวน n ตัว

$f(\cdot)$ คือฟังก์ชันเป้าหมาย

สำหรับปัญหาออฟติไมเซชันแบบมีข้อจำกัด คือ การหาค่าพารามิเตอร์ที่เหมาะสมที่สุดที่ทำให้ฟังก์ชันเป้าหมายมีค่าต่ำสุดหรือค่าสูงสุด โดยมีการกำหนดขอบเขตของค่าพารามิเตอร์บางตัวหรือทั้งหมด เงื่อนไขที่ใช้ในการกำหนดขอบเขต เรียกว่า "ข้อจำกัด" ซึ่งข้อจำกัดของปัญหาออฟติ-ไมเซชันนี้ แบ่งได้เป็น 2 กลุ่มคือ ข้อจำกัดแบบเท่ากัน ซึ่งเป็นข้อจำกัดที่กำหนดความสัมพันธ์ของพารามิเตอร์บางตัวให้เท่ากับเงื่อนไขของปัญหา ส่วนข้อจำกัดแบบไม่เท่ากันเป็นข้อจำกัดที่กำหนดความสัมพันธ์ของพารามิเตอร์บางตัวให้น้อยกว่าหรือมากกว่าเงื่อนไขของปัญหา รูปแบบทั่วไปของปัญหาออฟติไมเซชันแบบมีข้อจำกัด ดังสมการที่ 2.2

$$\begin{aligned} \text{minimize} \quad & f(\bar{x}) = f(x_1, x_2, ..., x_n) \\ \text{subject to} \quad & g_i(\bar{x}) = 0, \quad i = 1, ..., k \\ & h_j(\bar{x}) \le 0, \quad j = 1, ...p \end{aligned} \tag{2.2}$$

เมื่อ $\bar{x}$ คือเซตของพารามิเตอร์ที่มีสมาชิกจำนวน n ตัว

$f(\cdot)$ คือฟังก์ชันเป้าหมาย

g_i คือข้อจำกัดแบบเท่ากัน ซึ่งมีจำนวน k สมการ

h_j คือข้อจำกัดแบบไม่เท่ากัน ซึ่งมีจำนวน p สมการ

2.1.2 ประเภทของปัญหาออฟติไมเซชันที่แบ่งตามจำนวนของฟังก์ชันเป้าหมาย

ปัญหาออฟติไมเซชันสามารถแบ่งได้เป็น 2 ประเภทตามจำนวนฟังก์ชันเป้าหมายคือ ปัญหาออฟติไมเซชันแบบหนึ่งเป้าหมาย (single-objective optimization problem) และปัญหาออฟติไมเซชันแบบหลายเป้าหมาย (multiobjective optimization problem) ปัญหาออฟติไมเซชันแบบหนึ่งเป้าหมายคือ การหาค่าพารามิเตอร์ที่เหมาะสมที่สุดที่ทำให้ฟังก์ชันเป้าหมายมีค่าต่ำสุดหรือค่าสูงสุด สำหรับปัญหาออฟติไมเซชันแบบหลายเป้าหมายคือ การหาค่าพารามิเตอร์ที่เหมาะสมที่สุดที่ทำให้ฟังก์ชันเป้าหมายมีค่าดีที่สุดโดยการพิจารณาระดับความสำคัญหรือกำหนดค่าถ่วงน้ำหนักของฟังก์ชันเป้าหมายที่แตกต่างกัน การหาคำตอบที่ดีที่สุดโดยให้ระดับความสำคัญของทุกฟังก์ชันเป้าหมายเท่ากันเป็นสิ่งที่ทำได้ยากหรือเป็นไปไม่ได้เนื่องจากฟังก์ชันเป้าหมายแต่ละฟังก์ชันมีค่าต่ำสุดหรือค่าสูงสุด ณ จุดที่แตกต่างกัน ดังนั้น ความแตกต่างที่สำคัญระหว่างปัญหาออฟติไมเซชันทั้งสองประเภทคือ ปัญหาออฟติไมเซชันแบบหนึ่งเป้าหมายมีคำตอบที่ดีที่สุดเพียงคำตอบเดียว ในขณะที่ปัญหาออฟติไมเซชันแบบหลายเป้าหมายมีคำตอบที่ดีที่สุดมากกว่าหนึ่งคำตอบ ซึ่งเป็นคำตอบที่เกิดจากการกำหนดระดับความสำคัญของฟังก์ชันเป้าหมายที่แตกต่างกัน เซตของคำตอบที่ดีที่สุดนี้ เรียกว่า "ชุดคำตอบพาเรโต" (Pareto-optimal solutions) โดยสมาชิกในชุดคำตอบนี้มีคุณลักษณะที่สำคัญคือ ไม่สามารถเพิ่มค่าฟังก์ชันเป้าหมายฟังก์ชันใดฟังก์ชันหนึ่งให้ดีขึ้นโดยไม่มีการลดค่าฟังก์ชันเป้าหมายของฟังก์ชันอื่นๆ ให้น้อยลง [6]

นอกจากนี้ปัญหาออฟติไมเซชันทั้งสองประเภทยังสามารถแบ่งเป็นกลุ่มย่อยได้ 2 กลุ่มตามข้อจำกัด คือ แบบไม่มีข้อจำกัดและแบบมีข้อจำกัด ปัญหาออฟติไมเซชันแบบหนึ่งเป้าหมายและไม่มีข้อจำกัด มีองค์ประกอบ 2 ส่วนคือ ฟังก์ชันเป้าหมาย ซึ่งมีหนึ่งฟังก์ชันและมีพารามิเตอร์ตั้งแต่หนึ่งตัวขึ้นไป ส่วนปัญหาออฟติไมเซชันแบบหนึ่งเป้าหมายและมีข้อจำกัด ได้มีองค์ประกอบเพิ่มขึ้นอีก 1 ส่วนคือ ข้อจำกัดของปัญหา ดังสมการที่ 2.2 ส่วนปัญหาออฟติไมเซชันแบบหลายเป้าหมายและไม่มีข้อจำกัด มีองค์ประกอบ 2 ส่วนคือ ฟังก์ชันเป้าหมาย ซึ่งมีมากกว่าหนึ่งฟังก์ชันและมีพารามิเตอร์ตั้งแต่หนึ่งตัวขึ้นไป ส่วนปัญหาออฟติไมเซชันแบบหลายเป้าหมายและมีข้อจำกัด ได้มีองค์ประกอบเพิ่มขึ้นอีก 1 ส่วนคือ ข้อจำกัดของปัญหา ดังสมการที่ 2.3

$$\begin{aligned} \text{minimize} \quad & F(\bar{x}) = F\{f_1(\bar{x}), f_2(\bar{x}), \ldots, f_m(\bar{x})\} \\ & f(\bar{x}) = f(x_1, x_2, \ldots, x_n) \\ \text{subject to} \quad & g_i(\bar{x}) = 0, \quad i = 1, \ldots, k \\ & h_j(\bar{x}) \le 0, \quad j = 1, \ldots, p \end{aligned} \tag{2.3}$$

เมื่อ $\bar{x}$ คือเซตของพารามิเตอร์ที่มีจำนวน n ตัว

$F(\cdot)$ คือฟังก์ชันเป้าหมายรวม

$f_a(\cdot)$ คือฟังก์ชันเป้าหมายที่ a เมื่อ a = 1, 2, …, m

m คือจำนวนของฟังก์ชันเป้าหมายทั้งหมด

g_i คือข้อจำกัดแบบเท่ากัน ซึ่งมีจำนวน k สมการ

h_j คือข้อจำกัดแบบไม่เท่ากัน ซึ่งมีจำนวน p สมการ

2.1.3 ประเภทของปัญหาออฟติไมเซชันที่แบ่งตามชนิดข้อมูลของพารามิเตอร์

ปัญหาออฟติไมเซชันสามารถแบ่งได้เป็น 2 ประเภทตามชนิดข้อมูลของพารามิเตอร์ [4][7] คือ ปัญหาออฟติไมเซชันเชิงตัวเลข (numerical optimization problem) และปัญหาออฟติไมเซชันเชิงการจัด (combinatorial optimization problem) ปัญหาออฟติไมเซชันเชิงตัวเลขคือ ปัญหาออฟติไมเซชันที่มีค่าพารามิเตอร์แบบต่อเนื่อง (continuous optimization problem) ซึ่งมีค่าเป็นเลขจำนวนจริง รูปแบบทั่วไปของปัญหาออฟติไมเซชันเชิงตัวเลข ดังสมการที่ 2.4

$$\begin{aligned} \underset{x\in R^n}{\text{minimize}} \quad & f(\bar{x}) = f(x_1, x_2, ..., x_n) \\ \text{subject to} \quad & g_i(\bar{x}) = 0, \quad i = 1, ..., k \\ & h_j(\bar{x}) \le 0, \quad j = 1, ..., p \end{aligned} \tag{2.4}$$

เมื่อ $\bar{x}$ คือเซตของพารามิเตอร์ที่มีค่าเป็นเลขจำนวนจริง จำนวน n ตัว

$f(\cdot)$ คือฟังก์ชันเป้าหมาย

g_i คือข้อจำกัดแบบเท่ากัน ซึ่งมีจำนวน k สมการ

h_j คือข้อจำกัดแบบไม่เท่ากัน ซึ่งมีจำนวน p สมการ

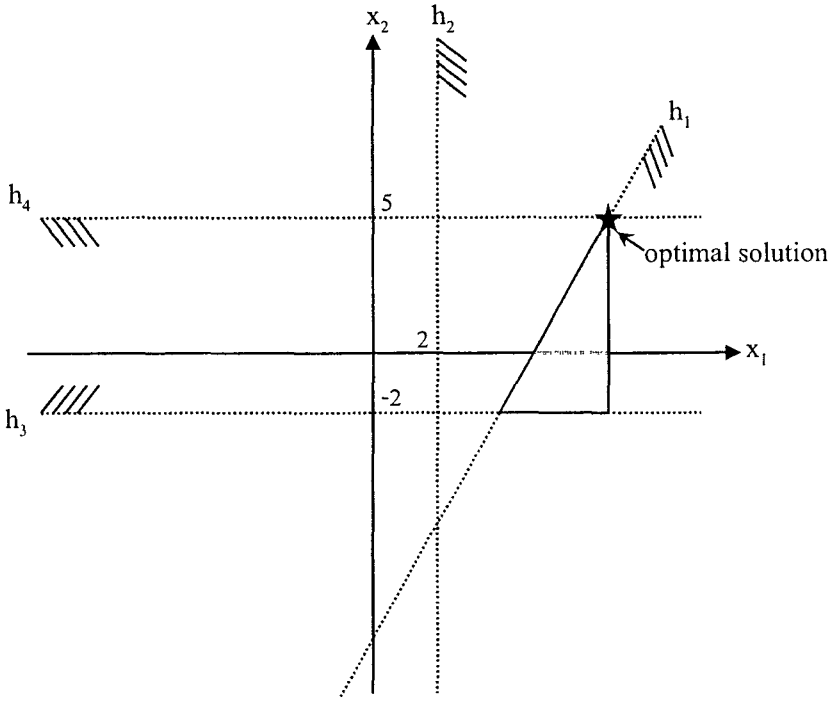
ตัวอย่างปัญหาออฟติไมเซชัน maximize $(x_1 + 2x_2)$ ที่มีเงื่อนไขข้อจำกัด 4 ข้อ คือ

1) $2x_1 - x_2 \le 10$
2) $x_1 \ge 2$
3) $x_2 \ge -2$
4) $x_2 \le 5$

สามารถนำมาเขียนในรูปของปัญหาออฟติไมเซชันได้ ดังนี้

$$\begin{aligned} &\text{minimize} \quad f_{new}(\bar{x}) = -x_1 - 2x_2 \\ &\text{subject to} \\ &\qquad h_1: \quad 2x_1 - x_2 \le 10 \\ &\qquad h_2: \quad -x_1 + 2 \le 0 \\ &\qquad h_3: \quad -x_2 - 2 \le 0 \\ &\qquad h_4: \quad x_2 - 5 \le 0 \end{aligned}$$

จากปัญหาข้างต้นสามารถหาคำตอบที่ทำให้ฟังก์ชันเป้าหมายมีค่าสูงสุดโดยวิธีการเขียนกราฟ ได้ดังรูปที่ 2.1



รูปที่ 2.1 การหาคำตอบที่เหมาะสมที่สุดด้วยวิธีการเขียนกราฟ

จากรูปที่ 2.1 พื้นที่แรเงาแสดงคำตอบที่เป็นไปได้ทั้งหมดที่อยู่ภายใต้ข้อจำกัดของปัญหา และคำตอบที่เหมาะสมที่สุด (optimal solution: $\bar{x}^*$) คือ x_1 และ x_2 เท่ากับ 7.5 และ 5 ตามลำดับ โดยที่ค่าฟังก์ชันเป้าหมายที่ดีที่สุดคือ $x_1 + 2x_2 = 7.5 + 2(5) = 17.5$ จากตัวอย่างข้างต้นนี้ จะเห็นว่าถึงแม้ว่าปัญหาออฟติไมเซชันเชิงตัวเลขที่มีฟังก์ชันเป้าหมายและข้อจำกัดเป็นระบบสมการเชิงเส้น (linear equation system) ซึ่งง่ายต่อการค้นหาคำตอบที่เหมาะสมที่สุด แต่เซตของคำตอบที่เป็นไปได้ (feasible solution) ทั้งหมดเป็นเซตอนันต์ (infinite set) ซึ่งหมายความว่า คำตอบที่เป็นไปได้ทั้งหมดไม่สามารถนับหรือแจกแจงสมาชิก ดังนั้น การค้นหาคำตอบที่เหมาะสมที่สุดจึงเป็นปัญหาที่ท้าทายอย่างยิ่ง นอกจากนี้ถ้าปัญหาออฟติไมเซชันเชิงตัวเลขมีความซับซ้อนมากขึ้น เช่น ฟังก์ชันเป้าหมายหรือข้อจำกัดเป็นสมการไม่เชิงเส้น (nonlinear equation) เป็นต้น ยิ่งทำให้ยากต่อการค้นหาคำตอบที่เหมาะสมที่สุดมากขึ้น ด้วยเหตุนี้นักวิจัยจึงให้ความสนใจเป็นอย่างมากในการพัฒนาอัลกอริทึมเพื่อใช้แก้ปัญหาด้านออฟติไมเซชัน

ปัญหาออฟติไมเซชันอีกประเภทหนึ่งคือ ปัญหาออฟติไมเซชันเชิงการจัด ซึ่งหมายถึงปัญหาออฟติไมเซชันที่มีค่าพารามิเตอร์แบบไม่ต่อเนื่อง (discrete optimization problem) ลักษณะการแก้ปัญหาประเภทนี้เป็นการจัดหมู่ (combination) หรือการเรียงสับเปลี่ยน (permutation) เซตของพารามิเตอร์ทั้งหมดของปัญหา โดยปัญหาออฟติไมเซชันประเภทนี้มีการกำหนดฟังก์ชันเป้าหมาย พารามิเตอร์และข้อจำกัดแตกต่างกันตามปัญหาที่ศึกษา เช่น ปัญหาการเดินทางของ

พนักงานขาย (travelling salesman problem) ซึ่งเป็นปัญหาของการหาเส้นทางที่สั้นที่สุดจากเมืองเริ่มต้นเพื่อเดินทางผ่านไปยังเมืองทุกเมืองและเดินทางกลับมายังเมืองเริ่มต้นอีกครั้ง โดยต้องเดินทางผ่านเมืองใดเมืองหนึ่งได้เพียงหนึ่งครั้งเท่านั้น จากปัญหานี้สามารถกำหนดฟังก์ชันเป้าหมาย พารามิเตอร์และข้อจำกัดได้ดังนี้ ฟังก์ชันเป้าหมายคือ ระยะทางที่สั้นที่สุด พารามิเตอร์คือ เมือง และข้อจำกัดคือ ต้องเลือกเดินทางครบทุกเมืองและสามารถเดินทางผ่านเมืองใดเมืองหนึ่งได้หนึ่งครั้ง ดังนั้น ลักษณะของการแก้ปัญหานี้จึงหมายถึง การเรียงสับเปลี่ยนเมืองทั้งหมด ซึ่งรายละเอียดของปัญหานี้จะกล่าวในหัวข้อถัดไป นอกจากปัญหาการเดินทางของพนักงานขายแล้ว ยังมีปัญหาออฟติไมเซชันเชิงการจัดที่น่าสนใจอีกจำนวนมาก เช่น ปัญหาการจัดตารางโครงการ (project scheduling problem) ปัญหาการบรรจุของลงถุง (knapsack problem) ปัญหาการมอบหมายงาน (job assignment problem) เป็นต้น

2.2 ทฤษฎีที่เกี่ยวข้อง

2.2.1 ปัญหาการเดินทางของพนักงานขาย

ปัญหาการเดินทางของพนักงานขาย เป็นปัญหาที่มีเป้าหมายเพื่อค้นหาเส้นทางที่สั้นที่สุด โดยพนักงานขายเริ่มออกเดินทางจากเมืองที่พักอาศัยเพื่อไปยังเมืองทุกเมืองและเดินทางกลับมายังเมืองของตนเอง โดยที่พนักงานขายสามารถเดินทางผ่านเมืองแต่ละเมืองได้เพียงหนึ่งครั้งเท่านั้น ปัญหาการเดินทางของพนักงานขายนี้สามารถแบ่งได้เป็น 2 ประเภท [8] คือ ปัญหาการเดินทางของพนักงานขายแบบสมมาตร (symmetric travelling salesman problem) และปัญหาการเดินทางของพนักงานขายแบบไม่สมมาตร (asymmetric travelling salesman problem) ซึ่งทั้งสองปัญหามีความแตกต่างกันที่สำคัญคือ ระยะทางที่เชื่อมระหว่างเมือง กรณีปัญหาการเดินทางของพนักงานขายแบบสมมาตร ระยะทางระหว่างเมือง i ไปยังเมือง j เขียนแทนด้วย d_{ij} มีระยะทางเท่ากับระยะทางระหว่างเมือง j มายังเมือง i เขียนแทนด้วย d_{ji} กล่าวคือ $d_{ij} = d_{ji}$ แต่สำหรับกรณีปัญหาการเดินทางของพนักงานขายแบบไม่สมมาตรจะแตกต่างกันคือ ระยะทางระหว่างเมือง i ไปยังเมือง j มีระยะทางไม่เท่ากับระยะทางระหว่างเมือง j มายังเมือง i กล่าวคือ $d_{ij} \neq d_{ji}$

ปัญหาการเดินทางของพนักงานขายนี้ถือว่าเป็นปัญหาที่ได้รับความนิยมเป็นอย่างยิ่งในกลุ่มของปัญหาออฟติไมเซชันเชิงการจัด เนื่องจากความท้าทายของปัญหานี้คือ จำนวนคำตอบที่เป็นไปได้ทั้งหมดของปัญหาแปรผันตามจำนวนเมือง โดยมีวิธีการคำนวณจำนวนคำตอบที่เป็นไปได้ทั้งหมด ดังนี้ กำหนดให้ ปัญหาการเดินทางของพนักงานขายแบบสมมาตรมีจำนวน n เมือง คำตอบที่เป็นไปได้ทั้งหมดคือ (n-1)!/2 [8] ตัวอย่างการคำนวณนี้ เช่น กำหนดให้ ปัญหาการเดินทางของพนักงานขายแบบสมมาตรประกอบด้วยเมืองจำนวน 20 เมือง ดังนั้น คำตอบที่เป็นไปได้ทั้งหมดเท่ากับ (20-1)!/2 = 19!/2 = 60,822,550,204,416,000 คำตอบ เป็นต้น สำหรับปัญหาออฟติ-

ไมเซชันแบบไม่สมมาตร คำตอบที่เป็นไปได้ทั้งหมดคำนวณได้จาก (n-1)! [8] ตัวอย่างการคำนวณนี้ เช่น กำหนดให้ ปัญหาการเดินทางของพนักงานขายแบบไม่สมมาตรประกอบด้วยเมืองจำนวน 20 เมือง ดังนั้น คำตอบที่เป็นไปได้ทั้งหมดเท่ากับ (20-1)! = 19! = 121,645,100,408,832,000 คำตอบ เป็นต้น ตารางที่ 2.1 แสดงจำนวนคำตอบที่เป็นไปได้ของปัญหาการเดินทางของพนักงานขายแบบสมมาตร เมื่อกำหนดจำนวนเมืองตั้งแต่ 5-30 เมือง จากตารางที่ 2.1 จะเห็นว่า เมื่อจำนวนเมืองเพิ่มขึ้น จำนวนคำตอบที่เป็นไปได้ยิ่งเพิ่มขึ้นอย่างทวีคูณ ย่อมส่งผลทำให้ต้องใช้เวลาในการค้นหาเส้นทางที่เหมาะสมที่สุดนานมากยิ่งขึ้น ด้วยเหตุนี้ปัญหาการเดินทางของพนักงานขาย จึงเป็นปัญหาหนึ่งที่ถูกจัดให้อยู่ในกลุ่มปัญหาเอ็นพีบริบูรณ์ (NP-complete problem)

ตารางที่ 2.1 จำนวนคำตอบที่เป็นไปได้ของปัญหาการเดินทางของพนักงานขายแบบสมมาตร

จำนวนเมือง	คำตอบที่เป็นไปได้	
5	(5-1)!/2	12
10	(10-1)!/2	181,440
15	(15-1)!/2	43,589,145,600
20	(20-1)!/2	60,822,550,204,416,000
25	(25-1)!/2	310,224,200,866,619,719,680,000
30	(30-1)!/2	4,420,880,996,869,850,977,271,808,000,000

ปัญหาการเดินทางของพนักงานขายแบบสมมาตรเขียนแทนด้วยกราฟแบบไม่มีทิศทาง (undirected graph) แต่สำหรับปัญหาการเดินทางของพนักงานขายแบบไม่สมมาตรเขียนแทนด้วยกราฟแบบมีทิศทาง (directed graph or digraph) กราฟทั้งสองชนิดประกอบด้วยเซตของจุดยอด (vertex) และเซตของเส้นเชื่อม (edge) โดยปกติแต่ละจุดยอดมีการเชื่อมถึงกันแบบทั่วถึง กำหนดให้ $G = (V, E)$ โดยที่ $V = \{1, 2, \ldots, i, \ldots, n\}$ แทนเซตของจุดยอดหรือเมือง เมื่อ n แทนจำนวนเมือง และ $E = \{(i, j) \mid i, j \in V\}$ แทนเซตของเส้นเชื่อมระหว่างสองเมืองใดๆ

ฟังก์ชันเป้าหมายของปัญหาการเดินทางของพนักงานขาย ดังสมการที่ 2.5

$$\text{minimize } f(t) = \sum_{i=1}^{n} d_{i\pi(i)} \tag{2.5}$$

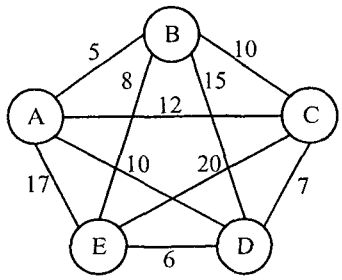
เมื่อ t คือเซตของเมืองที่เกิดจากการเรียงสับเปลี่ยนจำนวน n เมือง ซึ่งเรียกว่า ทัวร์ (tour)

$f(\cdot)$ คือฟังก์ชันเป้าหมาย

$\pi(i)$ คือเมืองลำดับถัดไปที่เลือกเดินทางต่อจากเมือง i

$d_{i\pi(i)}$ คือระยะทางระหว่างเมือง i ไปยังเมืองลำดับถัดไป

ตัวอย่างของปัญหาการเดินทางของพนักงานขายแบบสมมาตรที่ประกอบด้วยเมืองจำนวน 5 เมือง โดยแต่ละเมืองมีเส้นเชื่อมถึงกันแบบทั่วถึง ดังรูปที่ 2.2



รูปที่ 2.2 ตัวอย่างปัญหาการเดินทางของพนักงานขายแบบสมมาตร

จากตัวอย่างข้างต้น มีคำตอบที่เป็นไปได้ทั้งหมด (5-1)!/2 = 12 คำตอบ

1.	A-B-C-D-E	5+10+7+6+17	= 45
2.	A-B-C-E-D	5+10+20+6+10	= 51
3.	A-B-D-C-E	5+15+7+20+17	= 64
4.	A-B-D-E-C	5+15+6+20+12	= 58
5.	A-B-E-C-D	5+8+20+7+10	= 50
6.	A-B-E-D-C	5+8+6+7+12	= 38
7.	A-C-B-D-E	12+10+15+6+17	= 60
8.	A-C-B-E-D	12+10+8+6+10	= 46
9.	A-C-D-B-E	12+7+15+8+17	= 59
10.	A-C-E-B-D	12+20+8+15+10	= 65
11.	A-D-C-B-E	10+7+10+8+17	= 52
12.	A-D-B-C-E	10+15+10+20+17	= 72

ดังนั้นจากคำตอบที่เป็นไปได้ทั้งหมด เส้นทางที่สั้นที่สุด (t^*) คือ A-B-E-D-C และระยะทางที่สั้นที่สุด ($f(t^*)$) คือ 38 จากตัวอย่างข้างต้นนี้แสดงให้เห็นว่า การหาเส้นทางที่สั้นที่สุดของปัญหาการเดินทางของพนักงานขายที่มีจำนวนเมืองน้อยจะไม่ยากเกินไปในการค้นหาคำตอบที่เหมาะสมที่สุด แต่ถ้ามีจำนวนเมืองเพิ่มมากขึ้นย่อมต้องใช้เวลาในการค้นหาคำตอบที่เหมาะสมที่สุดนานมากยิ่งขึ้น ด้วยเหตุนี้จึงมีนักวิจัยจำนวนมากพยายามพัฒนาอัลกอริทึมใหม่เพื่อใช้ในการแก้ปัญหาออฟติไมเซชันประเภทนี้

2.2.2 เทคนิคที่ใช้ในการแก้ปัญหาออฟติไมเซชัน

เทคนิคที่ใช้ในการแก้ปัญหาออฟติไมเซชันเพื่อหาคำตอบที่เหมาะสมที่สุด แบ่งเป็น 2 กลุ่ม คือ เทคนิคการหาคำตอบที่ดีที่สุดด้วยหลักการทางคณิตศาสตร์ (mathematical optimization) และเทคนิคการหาคำตอบที่ดีที่สุดด้วยหลักการการประมาณค่า (approximation optimization) เทคนิคการหาคำตอบที่ดีที่สุดด้วยหลักการทางคณิตศาสตร์ เป็นวิธีการแก้ปัญหาแบบดั้งเดิม ตัวอย่างเทคนิคในด้านนี้ เช่น วิธีการโปรแกรมเชิงเส้น (linear programming) วิธีการหาฟังก์ชันอนุพันธ์ (differential) วิธีการซิมเพล็ก (simplex method) เป็นต้น อย่างไรก็ดี วิธีการหาคำตอบที่ดีที่สุดด้วยหลักการทางคณิตศาสตร์มีกฎเกณฑ์และข้อจำกัดตายตัว เมื่อนำไปใช้ในการแก้ปัญหาที่มีขนาดใหญ่ วิธีการนี้จะใช้เวลาในการแก้ปัญหานานมาก ดังนั้น วิธีนี้จึงเหมาะสมกับปัญหาขนาดเล็กหรือปัญหาที่มีความซับซ้อนน้อย นอกจากนี้ปัญหาออฟติไมเซชันบางปัญหาไม่สามารถหาอนุพันธ์ของฟังก์ชันเป้าหมายได้ ด้วยเหตุผลทางด้านเวลาและข้อจำกัดของปัญหา นักวิจัยจึงได้พยายามหาวิธีการแก้ปัญหาออฟติไมเซชันที่มีความยืดหยุ่นมากขึ้นจนทำให้เกิดการพัฒนาเทคนิคใหม่ในกลุ่มที่สองคือ เทคนิคการหาคำตอบที่ดีที่สุดด้วยหลักการการประมาณค่า ซึ่งเทคนิคนี้อาจได้หรือไม่ได้คำตอบที่เหมาะสมที่สุดเสมอ แต่จะได้คำตอบที่มีค่าใกล้เคียงกับคำตอบที่เหมาะสมที่สุดโดยใช้เวลาในการประมวลผลน้อยลง ดังนั้น เทคนิคนี้จึงเหมาะสำหรับปัญหาออฟติไมเซชันที่มีขนาดใหญ่ อย่างปัญหาแบบ NP-hard (non-deterministic polynomial time problem) ตัวอย่างเทคนิคในด้านนี้ที่ได้รับความนิยมเป็นอย่างมากคือ วิธีการเมตาฮิวริสติก (metaheuristic) ซึ่งตัวอย่างอัลกอริทึมที่นิยมใช้ คือ จีเนติกอัลกอริทึม (genetic algorithm) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบวิวัฒนาการทางพันธุกรรมของสิ่งมีชีวิต อัลกอริทึมพาร์ทิเคิลสวอร์ม (particle swarm optimization) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการหาอาหารของฝูงนกหรือฝูงปลา อัลกอริทึมอาณานิคมมด (ant colony optimization) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการหาอาหารของมด และอัลกอริทึมการผสมพันธุ์ของผึ้ง (marriage in honey-bees optimization) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการผสมพันธุ์ของผึ้งนางพญา

2.2.3 ชีววิทยาของผึ้ง

ผึ้งเป็นแมลงที่มีวิวัฒนาการสูงที่สุดในบรรดาแมลงทุกชนิดที่มีอยู่บนโลกนี้โดยมีหลากหลายสายพันธุ์ ผึ้งสายพันธุ์ที่สามารถสร้างน้ำผึ้งได้ (honey bee) ถือว่ามีวิวัฒนาการสูงที่สุดในบรรดาผึ้งทั้งหมด โดยปกติ ฝูงผึ้งตามธรรมชาติอยู่ร่วมกันภายในรัง โดยแต่ละรังมีทั้งผึ้งที่โตเต็มวัยและตัวอ่อนผึ้ง ผึ้งที่โตเต็มวัยแบ่งเป็น 3 ประเภท [9][10] คือ ผึ้งนางพญา (queen) ผึ้งตัวผู้ (drone) และผึ้งงาน (worker) ซึ่งผึ้งแต่ละประเภทมีบทบาทหน้าที่ที่แตกต่างกัน ผึ้งนางพญาเป็นผึ้งตัวเมียที่มีหน้าที่หลักคือ การผสมพันธุ์และการวางไข่ โดยในช่วงฤดูกาลผสมพันธุ์ ผึ้งนางพญาจะบินออกจากรังเพื่อไปผสมพันธุ์กับผึ้งตัวผู้ ในระหว่างการผสมพันธุ์ สเปิร์มของผึ้งตัวผู้จะถูกเก็บไว้ในถุงเก็บ

สเปิร์ม (spermatheca) ที่อยู่ในช่องท้องของผึ้งนางพญา หลังจากการผสมพันธุ์แล้วผึ้งตัวผู้จะตายทันที ผึ้งนางพญาแต่ละตัวสามารถผสมพันธุ์กับผึ้งตัวผู้ได้หลายตัวโดยเฉลี่ยประมาณ 7-21 ตัว ดังนั้นสเปิร์มที่อยู่ในถุงเก็บสเปิร์มจึงมีปริมาณมากพอที่ผึ้งนางพญาสามารถนำไปใช้ในการปฏิสนธิกับไข่ได้ตลอดช่วงชีวิต สำหรับผึ้งงานซึ่งเป็นผึ้งตัวเมียเช่นเดียวกับผึ้งนางพญาแต่ไม่ได้มีหน้าที่ผสมพันธุ์ หน้าที่หลักของผึ้งงานคือ การสร้างรัง การทำความสะอาดรัง การป้องกันภัย การหาอาหาร การป้อนอาหารให้กับผึ้งนางพญา ผึ้งตัวผู้และตัวอ่อนผึ้ง และในบางครั้งผึ้งงานอาจทำหน้าที่วางไข่ได้ในกรณีที่เกิดความผิดปกติภายในรัง อย่างไรก็ตาม ไข่ที่เกิดจากผึ้งงานเป็นไข่ที่ไม่ได้รับการปฏิสนธิ (unfertilized egg) ดังนั้นไข่เหล่านี้จะฟักและพัฒนาไปเป็นผึ้งตัวผู้ ในขณะที่ไข่ที่เกิดจากผึ้งนางพญามีทั้งไข่ที่ได้รับการปฏิสนธิ (fertilized egg) และไข่ที่ไม่ได้รับการปฏิสนธิ กรณีที่ไข่ได้รับการปฏิสนธิจะฟักและพัฒนาไปเป็นผึ้งตัวเมีย (ผึ้งนางพญาและผึ้งงาน) ด้วยเหตุนี้ ผึ้งตัวผู้และผึ้งตัวเมียจึงมีความแตกต่างกันในด้านพันธุกรรมคือ ผึ้งตัวผู้มีจำนวนโครโมโซม 1 ชุด (haploid chromosome: 1n) ในขณะที่ผึ้งตัวเมียมีโครโมโซม 2 ชุด (diploid chromosome: 2n)

2.3 งานวิจัยที่เกี่ยวข้อง

2.3.1 MBO: Marriage in honey-bees optimization (a haplometrosis polygynous swarming approach)

Hussein A. Abbass [1] นำเสนออัลกอริทึมการผสมพันธุ์ของผึ้ง ซึ่งถือได้ว่าเป็นอัลกอริทึมแรกของกลุ่มอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการผสมพันธุ์ของผึ้งและอัลกอริทึมนี้ยังถูกจัดให้อยู่ในกลุ่มของอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมอันชาญฉลาดของสัตว์ที่อยู่รวมกันเป็นสังคม (swarm intelligence) อีกด้วย จากหลักการทางชีววิทยาของผึ้ง Hussein A. Abbass ได้แทนค่าจีโนไทป์ (genotype) ของผึ้งนางพญาและผึ้งตัวผู้ด้วยอาร์เรย์ไบนารี แต่ละอิลิเมนต์ของอาร์เรย์แทนค่ายีนต์ (gene) ซึ่งมีค่าที่เป็นไปได้ 2 ค่าคือ 0 และ 1 ดังรูปที่ 2.3 สำหรับจีโนไทป์ของผึ้งตัวผู้เป็นจีโนไทป์แบบแฮพพลอยด์ (haploid) ซึ่งมีโครโมโซมเพียงชุดเดียว จึงต้องมีการระบุตำแหน่งยีนต์ว่าง (m) จำนวนครึ่งหนึ่งของความยาวจีโนไทป์ด้วย สำหรับผึ้งงานแทนค่าด้วยฟังก์ชันฮิวริสติก (heuristic function) เพื่อใช้ในการค้นหาแบบโลคอล (local search)

0	1	1	0	1	1	0	0

1	0	1	0	0	1	1	1
	m			m	m		m

(ก) จีโนไทป์ของผึ้งนางพญา (ข) จีโนไทป์ของผึ้งตัวผู้

รูปที่ 2.3 การแทนค่าจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้แบบแฮพพลอยด์

อัลกอริทึมการผสมพันธุ์ของผึ้ง ประกอบด้วยขั้นตอนหลัก ดังนี้

1) ขั้นตอนการกำหนดค่าเริ่มต้น (initialization process)

ค่าพารามิเตอร์เริ่มต้นที่ต้องกำหนดประกอบด้วย จำนวนของผึ้งนางพญา ขนาดของถุงเก็บสเปิร์มของผึ้งนางพญา จำนวนของผึ้งตัวผู้ จำนวนของตัวอ่อนผึ้ง ค่าความน่าจะเป็นในการมิวเตชัน (mutation probability) และจำนวนรอบการทำงาน นอกจากนี้ยังได้มีการกำหนดค่าประชากรเริ่มต้นโดยการสุ่มตามจำนวนของผึ้งนางพญาและผึ้งตัวผู้ จากนั้นให้คำนวณค่าฟิตเนสและทำการจัดเรียงประชากรเริ่มต้นตามค่าฟิตเนสที่คำนวณได้ ประชากรที่มีค่าฟิตเนสสูงจะถูกเลือกให้เป็นผึ้งนางพญาเริ่มต้นและประชากรส่วนที่เหลือกำหนดให้เป็นผึ้งตัวผู้เริ่มต้น

2) ขั้นตอนการผสมพันธุ์ (mating process)

กำหนดค่าเริ่มต้นของค่าพลังงานและค่าความเร็วในการบินให้กับผึ้งนางพญาแต่ละตัว โดยการสุ่มค่าอยู่ในช่วง [0.5, 1] หลังจากนั้นผึ้งนางพญาจะเลือกผสมพันธุ์กับผึ้งตัวผู้โดยการพิจารณาจากค่าความน่าจะเป็นในการผสมพันธุ์ (marriage probability) ดังสมการที่ 2.6

$$\text{prob}(Q,D) = e^{\frac{-\text{difference}}{\text{speed}(t)}} \tag{2.6}$$

เมื่อ prob(Q,D) คือค่าความน่าจะเป็นในการผสมพันธุ์ระหว่างผึ้งนางพญา Q และผึ้งตัวผู้ D

difference คือค่าผลต่างสมบูรณ์ระหว่างค่าฟิตเนสของผึ้งนางพญา Q และผึ้งตัวผู้ D

speed(t) คือค่าความเร็วในการบินของผึ้งนางพญา Q ณ รอบการทำงานที่ t

ผึ้งนางพญาตัดสินใจเลือกผสมพันธุ์กับผึ้งตัวผู้ที่มีค่าความน่าจะเป็นในการผสมพันธุ์สูงที่สุด จากสมการที่ 2.6 จะเห็นว่า ค่าความน่าจะเป็นในการผสมพันธุ์จะมีค่าสูงเมื่อผึ้งนางพญามีค่าความเร็วในการบินสูงหรือค่าฟิตเนสของผึ้งนางพญาและผึ้งตัวผู้มีค่าใกล้เคียงกัน จากนั้นสเปิร์ม (จีโนไทป์) ของผึ้งตัวผู้จะถูกนำมาเก็บไว้ในถุงเก็บสเปิร์มของผึ้งนางพญา ก่อนที่ผึ้งนางพญาจะเริ่มการเลือกคู่ครั้งถัดไป ค่าพลังงานและค่าความเร็วในการบินของผึ้งนางพญาจะถูกปรับลดลง ดังนี้

$$\text{speed}(t+1) = \alpha \times \text{speed}(t) \tag{2.7}$$

$$\text{energy}(t+1) = \text{energy}(t) - \text{step} \tag{2.8}$$

เมื่อ α คืออัตราการลดลงของค่าความเร็วในการบิน ซึ่งกำหนดให้มีค่าอยู่ในช่วง [0, 1]

step คือปริมาณการลดลงของค่าพลังงาน คำนวณได้จาก

$\text{step} = (0.5 \times \text{energy}) / \text{spermatheca-size}$

ขั้นตอนการผสมพันธุ์ของผึ้งนางพญาแต่ละตัวสิ้นสุดลงเมื่อถุงเก็บสเปิร์มเต็มหรือค่าพลังงานน้อยกว่าค่าที่กำหนดไว้ซึ่งปกติกำหนดค่าเท่ากับศูนย์

3) ขั้นตอนการสร้างตัวอ่อนผึ้งและการเลี้ยงดู (breeding and feeding process)

ขั้นตอนการสร้างตัวอ่อนผึ้งเริ่มต้นด้วยการเลือกผึ้งนางพญาเพื่อทำหน้าที่สร้างตัวอ่อนผึ้ง โดยผึ้งนางพญาถูกสุ่มเลือกด้วยวิธีการเลือกแบบวงล้อหมุน (roullete wheel selection) ผึ้งนางพญาที่มีค่าฟิตเนสสูงจะมีโอกาสได้รับเลือกให้ทำหน้าที่สร้างตัวอ่อนผึ้งสูงกว่าผึ้งนางพญาที่มีค่าฟิตเนสต่ำกว่า จากนั้นผึ้งนางพญาที่ได้รับเลือกจะสุ่มเลือกสเปิร์มจากถุงเก็บสเปิร์มของตนเองเพื่อทำการครอสโอเวอร์ (crossover) งานวิจัยนี้ได้นำเสนอโอเปอร์เรชันครอสโอเวอร์แบบแฮพพลอยด์ ดังรูปที่ 2.4 การครอสโอเวอร์นี้ทำการสร้างตัวอ่อนผึ้งด้วยการเติมตำแหน่งยีนต์ว่างของจีโนไทป์ของผึ้งตัวผู้ด้วยยีนต์ในตำแหน่งเดียวกันของจีโนไทป์ของผึ้งนางพญา ต่อจากนั้นตัวอ่อนผึ้งที่ได้จะถูกนำไปมิวเตท (mutate) ถึงแม้ว่าตัวอ่อนผึ้งเกิดจากพ่อแม่เดียวกัน แต่การมิวเตชัน (mutation) จะช่วยให้เกิดความแตกต่างในจีโนไทป์ของตัวอ่อนผึ้งมากขึ้น โอเปอร์เรเตอร์มิวเตชันจะสุ่มบางยีนต์ในจีโนไทป์เพื่อทำการกลับบิตจาก 0 เป็น 1 หรือจาก 1 เป็น 0 กระบวนการสร้างตัวอ่อนผึ้งนี้สิ้นสุดลงเมื่อจำนวนของตัวอ่อนผึ้งครบตามที่กำหนดไว้

จีโนไทป์ของผึ้งนางพญา	0	1	1	0	1	1	0	0
จีโนไทป์ของผึ้งตัวผู้	1		1	0			1	
จีโนไทป์ของตัวอ่อนผึ้ง	1	1	1	0	1	1	1	0

รูปที่ 2.4 โอเปอร์เรชันครอสโอเวอร์แบบแฮพพลอยด์

หลังจากนั้นจะเข้าสู่ขั้นตอนการเลี้ยงดู โดยมีผึ้งงานซึ่งแทนด้วยฟังก์ชันฮิวริสติกทำหน้าที่ดูแลตัวอ่อนผึ้ง งานวิจัยนี้ใช้ผึ้งงานจำนวน 5 ตัวซึ่งประกอบด้วยฟังก์ชันวอร์คแซท (walkSAT) ฟังก์ชันการสุ่มเดิน (random walk) ฟังก์ชันการกลับบิตแบบสุ่ม (random flip) ฟังก์ชันการสุ่มค่าใหม่ (random new) และฟังก์ชันการครอสโอเวอร์แบบหนึ่งจุด (one-point crossover) ตัวอ่อนผึ้งแต่ละตัวได้รับการเลี้ยงดูจากผึ้งงานที่แตกต่างกันด้วยวิธีการเลือกแบบวงล้อหมุน โดยการพิจารณาจากค่าฟิตเนสของผึ้งงาน ซึ่งค่าฟิตเนสของผึ้งงานคำนวณได้จากความสามารถในการปรับค่าฟิตเนสของตัวอ่อนผึ้ง

4) ขั้นตอนการคัดเลือกใหม่ (updating process)

ขั้นตอนนี้ทำหน้าที่คัดเลือกผึ้งนางพญาใหม่ โดยการเปรียบเทียบค่าฟิตเนสระหว่างผึ้งนางพญาในรอบปัจจุบันกับตัวอ่อนผึ้ง ผึ้งนางพญาในรอบปัจจุบันที่มีค่าฟิตเนสต่ำกว่าตัวอ่อนผึ้งจะ

ถูกคัดทิ้งและตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าจะถูกเลือกให้เป็นผึ้งนางพญาใหม่แทน เมื่อกระบวนการการคัดเลือกผึ้งนางพญาใหม่สิ้นสุดลง ตัวอ่อนผึ้งที่ไม่ได้รับเลือกจะถูกคัดทิ้งเช่นกัน

5) ทำซ้ำขั้นตอนที่ 2-4 จนกระทั่งจำนวนรอบการทำงานครบตามที่กำหนดไว้ จึงหยุดการทำงาน

อัลกอริทึมใหม่ที่นำเสนอในงานวิจัยนี้ได้ทำการทดสอบกับปัญหา 3-SAT (3-propositional satisfiability problem) จำนวน 50 ปัญหา โดยแต่ละปัญหามีจำนวนตัวแปร 3 ตัวและมีเงื่อนไขข้อจำกัดจำนวน 215 เงื่อนไข การทดลองได้ศึกษาการกำหนดค่าพารามิเตอร์ที่เหมาะสม ดังนี้ 1) จำนวนของผึ้งนางพญา มีทั้งหมด 5 ค่าคือ 1, 2, 3, 4 และ 5 ตามลำดับ 2) ขนาดของถุงเก็บสเปิร์ม มีทั้งหมด 3 ค่าคือ 7, 14 และ 21 ตามลำดับ 3) จำนวนรอบการทำงาน มีทั้งหมด 5 ค่าคือ 60, 30, 20, 15 และ 12 ตามลำดับและ 4) จำนวนของตัวอ่อนผึ้งที่สร้างในแต่ละรอบการทำงาน มีทั้งหมด 5 ค่าคือ 20, 40, 60, 80 และ 100 ตามลำดับ การกำหนดค่าพารามิเตอร์ที่ความสัมพันธ์กันระหว่างจำนวนรอบการทำงานและจำนวนของตัวอ่อนผึ้งส่งผลให้ แต่ละการทดลองมีโอกาสในการค้นหาคำตอบของปัญหาได้เท่ากัน เช่น การทดลองที่หนึ่ง จำนวนรอบการทำงาน 60 ครั้งคู่กับจำนวนตัวอ่อนผึ้ง 20 ตัวจึงทำให้มีโอกาสในการค้นหาคำตอบได้ 1,200 ครั้ง เป็นต้น จากผลการทดลองทั้งหมด พบว่าอัลกอริทึมการผสมพันธุ์ของผึ้งให้คำตอบที่ดีที่สุด เมื่อกำหนดจำนวนของผึ้งนางพญา 1 ตัว ขนาดถุงเก็บสเปิร์มเท่ากับ 21 และจำนวนของตัวอ่อนผึ้งต่อหนึ่งรอบการทำงานเท่ากับ 60 ตัว นอกจากนี้ยังได้มีการพิจารณาประสิทธิภาพของผึ้งงานทั้ง 5 ตัว พบว่า ฟังก์ชันวอร์คแซท มีค่าฟิตเนสเฉลี่ยสูงที่สุด ซึ่งแสดงให้เห็นว่า ฟังก์ชันนี้มีประสิทธิภาพในการค้นหาแบบโลคอลดีที่สุด ดังนั้นเพื่อวัดประสิทธิภาพของผึ้งงานวอร์คแซท จึงได้ทำการทดสอบซ้ำใหม่อีกหนึ่งครั้ง โดยกำหนดให้มีผึ้งงานเพียง 1 ตัวคือ ฟังก์ชันวอร์คแซท ผลการทดลอง พบว่า คำตอบที่ดีที่สุดโดยเฉลี่ยมีค่าความผิดพลาดสูงกว่าการทดสอบแรก ดังนั้น จากผลการทดลองนี้สรุปได้ว่า อัลกอริทึมการผสมพันธุ์ของผึ้งมีประสิทธิภาพในการทำงานได้ดีเมื่อมีการประยุกต์ใช้ร่วมกับฟังก์ชันฮิวริสติกมากกว่าหนึ่งฟังก์ชัน

จากการศึกษางานวิจัยนี้สามารถวิเคราะห์ให้เห็นจุดเด่นที่สำคัญของอัลกอริทึมที่นำเสนอคือ ขั้นตอนการทำงานหลักของอัลกอริทึมที่นำเสนอมีความคล้ายคลึงกับจีเนติกอัลกอริทึมซึ่งเป็นอัลกอริทึมที่มีประสิทธิภาพในการแก้ปัญหาออฟติไมเซชัน โดยได้มีการใช้โอเปอร์เรเตอร์การเลือก โอเปอร์เรเตอร์ครอสโอเวอร์และโอเปอร์เรเตอร์มิวเตชันในขั้นตอนการสร้างตัวอ่อนผึ้งเช่นเดียวกับในจีเนติกอัลกอริทึม ถึงแม้ว่ามีการนำเสนอโอเปอร์เรเตอร์ครอสโอเวอร์แบบแฮพพลอยด์ใหม่ แต่โอเปอร์เรเตอร์ดังกล่าวมีลักษณะการทำงานเหมือนกับโอเปอร์เรเตอร์ครอสโอเวอร์หลายจุดของจีเนติกอัลกอริทึม นอกจากนี้อัลกอริทึมที่นำเสนอยังมีโอเปอร์เรเตอร์เพิ่มขึ้นอีกหนึ่งโอเปอร์เรเตอร์ คือ โอเปอร์เรเตอร์ผึ้งงาน ซึ่งช่วยให้อัลกอริทึมการผสมพันธุ์ของผึ้งมีประสิทธิภาพสูงในการค้นหาคำตอบยิ่งขึ้น อย่างไรก็ตาม อัลกอริทึมที่นำเสนอยังคงมีจุดด้อยที่สำคัญ 2 ส่วนคือ 1) ลักษณะการทำงานของอัลกอริทึมเป็นแบบมีรังผึ้งหนึ่งรังที่ประกอบด้วยผึ้งนางพญาหลายตัว (single-colony

multiple-queens model) ดังนั้นในขั้นตอนการผสมพันธุ์ผึ้งนางพญาจึงมีการเลือกผสมพันธุ์กับผึ้งตัวผู้ภายในรังเดียวกัน ซึ่งเป็นสาเหตุที่ทำให้ผึ้งนางพญามีโอกาสที่จะเลือกผสมพันธุ์กับผึ้งตัวผู้ชุดเดียวกันหรือใกล้เคียงกัน ซึ่งส่งผลให้ตัวอ่อนผึ้งที่สร้างใหม่มีความหลากหลายน้อยจึงมีโอกาสตกในพื้นที่โลคอลได้ และ 2) ในแต่ละรอบการทำงาน ผึ้งตัวผู้จะถูกสร้างใหม่โดยการสุ่ม จึงทำให้ต้องมีการคำนวณค่าฟิตเนสใหม่ทุกรอบการทำงาน ขั้นตอนนี้จึงทำให้สิ้นเปลืองเวลาในการคำนวณ

2.3.2 Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization

Omid Bozorg Haddad, Abbas Afshar และ Miguel A. Mariño [2] ศึกษาการนำอัลกอริทึมการผสมพันธุ์ของผึ้งที่นำเสนอโดย Hussein A. Abbass มาประยุกต์ใช้กับปัญหาออฟติไมเซชัน โดยได้แบ่งการศึกษาประสิทธิภาพของอัลกอริทึมการผสมพันธุ์ของผึ้งเป็น 2 ส่วนคือ ส่วนแรกเป็นการทดสอบอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยฟังก์ชันออฟติไมเซชัน (function optimization) ทั้งแบบที่มีข้อจำกัดและแบบไม่มีข้อจำกัด และส่วนที่สองคือ การทดสอบอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยปัญหาการจัดการน้ำของเขื่อนที่กั้นแม่น้ำ Dez (Dez reservoir) ของประเทศอิหร่าน

อัลกอริทึมการผสมพันธุ์ของผึ้งที่นำเสนอโดย Hussein A. Abbass เป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการผสมพันธุ์ของผึ้งนางพญา ขั้นตอนการทำงานของอัลกอริทึมนี้สามารถแบ่งเป็น 5 ขั้นตอน โดยสรุป ได้ดังนี้

1) กำหนดค่าพารามิเตอร์และค่าประชากรเริ่มต้น โดยค่าพารามิเตอร์เริ่มต้นประกอบด้วย จำนวนของผึ้งนางพญา ขนาดของถุงเก็บสเปิร์มของผึ้งนางพญา จำนวนของผึ้งตัวผู้ จำนวนของตัวอ่อนผึ้ง ค่าความน่าจะเป็นในการมิวเตชัน และจำนวนรอบการทำงาน และได้กำหนดค่าประชากรเริ่มต้นทั้งผึ้งนางพญาและผึ้งตัวผู้ โดยการสุ่มตามจำนวนที่ได้กำหนดไว้

2) ผึ้งนางพญาทุกตัวมีหน้าที่ในการผสมพันธุ์และสามารถผสมพันธุ์กับผึ้งตัวผู้ได้หลายตัว ผึ้งนางพญาแต่ละตัวจะผสมพันธุ์กับผึ้งตัวผู้จำนวนมากหรือน้อยขึ้นอยู่กับค่าพลังงานเริ่มต้นและขนาดของถุงเก็บสเปิร์ม โดยผึ้งนางพญาแต่ละตัวตัดสินใจเลือกผสมพันธุ์จากค่าความน่าจะเป็นในการผสมพันธุ์ ผึ้งตัวผู้ที่มีค่าความน่าจะเป็นในการผสมพันธุ์สูงที่สุดจะถูกเลือกเพื่อผสมพันธุ์และสเปิร์มของผึ้งตัวผู้จะถูกเก็บไว้ในถุงเก็บสเปิร์มของผึ้งนางพญาเพื่อใช้ในการปฏิสนธิกับไข่ในช่วงการวางไข่ จากนั้นให้ทำการปรับค่าพลังงานและค่าความเร็วในการบินลดลงก่อนเริ่มการเลือกคู่ครั้งถัดไป การผสมพันธุ์ของผึ้งนางพญาแต่ละตัวสิ้นสุดลง เมื่อถุงเก็บสเปิร์มเต็มหรือค่าพลังงานน้อยกว่าค่าที่กำหนดไว้

3) ผึ้งนางพญาถูกสุ่มเลือกด้วยวิธีการเลือกแบบวงล้อหมุนเพื่อทำหน้าที่สร้างตัวอ่อนผึ้ง ผึ้งนางพญาที่ได้รับเลือกจะสร้างตัวอ่อนผึ้งโดยการสุ่มเลือกสเปิร์มจากถุงเก็บสเปิร์มเพื่อทำการครอสโอเวอร์กับจีโนไทป์ของตนเอง จากนั้นตัวอ่อนผึ้งที่ได้จะถูกนำไปมิวเตท กระบวนการ

สร้างตัวอ่อนผึ้งสิ้นสุดลงเมื่อจำนวนของตัวอ่อนผึ้งครบตามที่กำหนดไว้ หลังจากนั้นผึ้งงานซึ่งแทนด้วยฟังก์ชันฮิวริสติกจะทำหน้าที่ดูแลตัวอ่อนผึ้งโดยการปรับค่าจีโนไทป์ของตัวอ่อนผึ้ง ซึ่งประสิทธิภาพในการปรับค่าจีโนไทป์ตัวอ่อนผึ้งของผึ้งงานแต่ละตัวจะถูกนำไปคำนวณเป็นค่าฟิต-เนสของผึ้งงานนั้นด้วย

4) คัดเลือกผึ้งนางพญาใหม่โดยการเปรียบเทียบค่าฟิตเนสระหว่างผึ้งนางพญาในรอบปัจจุบันกับตัวอ่อนผึ้ง ผึ้งนางพญาในรอบปัจจุบันที่มีค่าฟิตเนสต่ำกว่าตัวอ่อนผึ้งจะถูกคัดทิ้งและตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าจะถูกเลือกให้เป็นผึ้งนางพญาใหม่

5) ทำซ้ำขั้นตอนที่ 2-4 จนกระทั่งจำนวนรอบการทำงานครบตามที่กำหนดไว้ จึงหยุดการทำงาน

งานวิจัยนี้ได้แบ่งการทดลองเป็น 2 ส่วนคือ การหาค่าฟังก์ชันออฟติไมเซชันและการหาค่าปริมาณการกักเก็บน้ำที่เหมาะสมที่สุดของระบบการจัดการน้ำ

1) การหาค่าฟังก์ชันออฟติไมเซชัน

ฟังก์ชันออฟติไมเซชันที่นำมาใช้ทดสอบมีจำนวน 3 ฟังก์ชัน ซึ่งประกอบด้วยฟังก์ชันออฟ-ติไมเซชันแบบไม่มีข้อจำกัด จำนวน 2 ฟังก์ชัน แบ่งเป็นการหาค่าต่ำสุดและการหาค่าสูงสุดตัวอย่างละ 1 ฟังก์ชันและฟังก์ชันออฟติไมเซชันแบบมีข้อจำกัด จำนวน 1 ฟังก์ชัน การทดลองได้ทำการเปรียบเทียบประสิทธิภาพการหาค่าฟังก์ชันออฟติไมเซชันของอัลกอริทึมการผสมพันธุ์ของผึ้งเทียบกับจีเนติกอัลกอริทึมที่ศึกษาทดลองโดย Gen และ Cheng [11] จากการรันจำนวน 10 ครั้ง ผลการทดลองแสดงให้เห็นว่า อัลกอริทึมการผสมพันธุ์ของผึ้งให้ค่าความผิดพลาดโดยเฉลี่ยในการหาค่าต่ำสุดหรือค่าสูงสุดน้อยกว่างานวิจัยที่นำมาเปรียบเทียบด้วย

2) การหาค่าปริมาณการกักเก็บน้ำที่เหมาะสมที่สุดของระบบการจัดการน้ำ

ข้อมูลที่ใช้ในการทดสอบเป็นข้อมูลความสัมพันธ์ระหว่างปริมาณน้ำที่ไหลเข้าสู่เขื่อนและปริมาณความต้องการใช้น้ำในช่วงเวลา 5 ปีที่ผ่านมา โดยปริมาณน้ำที่ไหลเข้าสู่เขื่อนต่อปีโดยเฉลี่ยประมาณ $5{,}900\times10^6$ ลูกบาศก์เมตร ความต้องการใช้น้ำต่อปีโดยเฉลี่ยประมาณ $5{,}303\times10^6$ ลูกบาศก์เมตรและปริมาณการกักเก็บต่อปีที่มีประสิทธิภาพคือ $2{,}510\times10^6$ ลูกบาศก์เมตร การทดลองที่สองนี้ได้ทำการเปรียบเทียบประสิทธิภาพจากค่าความคลาดเคลื่อนของปริมาณการปล่อยน้ำจากเขื่อนกับความต้องการใช้น้ำของอัลกอริทึมการผสมพันธุ์ของผึ้งเทียบกับจีเนติกอัลกอริทึม ผลการทดลองแสดงให้เห็นว่า อัลกอริทึมการผสมพันธุ์ของผึ้งให้ค่าความคลาดเคลื่อนน้อยกว่าจีเนติกอัลกอริทึม ดังนั้น จากผลการทดลอง สรุปได้ว่า อัลกอริทึมการผสมพันธุ์ของผึ้งมีประสิทธิภาพในการหาค่าปริมาณการกักเก็บน้ำที่เหมาะสมที่สุดของเขื่อน Dez ได้ดีกว่าอัลกอริทึมที่เปรียบเทียบด้วย

จากการศึกษางานวิจัยนี้ จะเห็นว่า งานวิจัยนี้ได้ประยุกต์ใช้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับปัญหาออฟติไมเซชันเชิงตัวเลขโดยไม่มีการปรับปรุงโครงสร้างการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม ดังนั้น จุดด้อยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบ

ดั้งเดิมจึงยังไม่ได้รับการแก้ไขปรับปรุง แต่อย่างไรก็ตาม งานวิจัยนี้ถือได้ว่าเป็นประโยชน์เป็นอย่างมากแก่นักวิจัยรุ่นต่อไป เนื่องจากงานวิจัยนี้ได้มีการทดสอบกับปัญหาออฟติไมเซชันเชิงตัวเลขทั้งที่เป็นฟังก์ชันมาตรฐานและเป็นปัญหาที่เกิดขึ้นจริงในโลก (real-world problem) ซึ่งผลการทดลองแสดงให้เห็นว่า อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่เสนอโดย Hussein A. Abbass สามารถนำไปใช้ในการแก้ปัญหาออฟติไมเซชันเชิงตัวเลขได้เป็นอย่างดี ดังนั้น งานวิจัยนี้จึงเป็นงานวิจัยที่ช่วยสนับสนุนและเพิ่มน้ำหนักให้กับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมในเรื่องประสิทธิภาพการค้นหาคำตอบที่เหมาะที่สุด

2.3.3 Optimal water distribution network design with honey-bee mating optimization

S. Mohan และ K.S. Jinesh Babu [12] ได้นำเสนอโมเดลใหม่ที่เกิดจากการปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งที่นำเสนอโดย Hussein A. Abbass เพื่อประยุกต์ใช้ในการแก้ปัญหาการออกแบบโครงสร้างระบบการจ่ายน้ำ ซึ่งระบบการจ่ายน้ำเป็นโครงสร้างพื้นฐานที่มีความสำคัญโดยเฉพาะในเขตชุมชนเมืองและมีการใช้เงินลงทุนจำนวนมหาศาล ดังนั้นงานวิจัยนี้จึงมีความพยายามในการศึกษาวิธีการออกแบบโครงสร้างการจ่ายน้ำที่ประหยัดต้นทุนที่สุด ระบบการจ่ายน้ำมีองค์ประกอบหลายส่วนที่สำคัญ ตัวอย่างเช่น แหล่งกักเก็บน้ำ ถังเก็บน้ำ ปั๊มน้ำ วาล์วควบคุมน้ำ ท่อส่งน้ำ จุดจ่ายน้ำ เป็นต้น โดยท่อส่งน้ำทำหน้าที่เป็นอุปกรณ์ที่เชื่อมโยงตั้งแต่แหล่งกักเก็บน้ำจนถึงจุดจ่ายน้ำให้แก่ผู้บริโภคหรือกล่าวได้ว่าเป็นโครงสร้างหลักของระบบการจ่ายน้ำ ดังนั้น ท่อส่งน้ำจึงเป็นอุปกรณ์ที่ใช้เงินเป็นสัดส่วนหลักของเงินลงทุนทั้งหมด ด้วยเหตุนี้ การออกแบบท่อส่งน้ำที่มีความเหมาะสมกับความต้องการจึงช่วยประหยัดต้นทุนได้ ซึ่งถือว่าเป็นประเด็นหลักของปัญหาการออกแบบโครงสร้างการจ่ายน้ำ

อัลกอริทึมการผสมพันธุ์ของผึ้งได้ถูกนำมาปรับใช้เพื่อแก้ปัญหาการออกแบบโครงสร้างระบบการจ่ายน้ำเพื่อหาขนาดเส้นผ่าศูนย์กลางของท่อส่งน้ำที่เหมาะสมที่สุดที่ทำให้ใช้ต้นทุนในการก่อสร้างน้อยที่สุดหรือประหยัดต้นทุนในการก่อสร้างมากที่สุด

$$\text{minimize } Z = \sum_{i=1}^{N} C_i(L, \Phi) \tag{2.9}$$

subject to

$$H_J \geq H_J^{min} \quad j = 1, 2, 3, \ldots, nd$$

$$q_J^{in} - q_J^{out} - q_J = 0 \quad j = 1, 2, 3, \ldots, nd$$

$$\left(\sum_{i=1}^{np_L} HL_i \right)_L = 0 \quad L = 1, 2, 3, \ldots, nL$$

เมื่อ $C_i(L,\Phi)$ คือต้นทุนการใช้ท่อส่งน้ำที่มีความยาว L และมีเส้นผ่าศูนย์กลางขนาด Φ

N คือจำนวนท่อส่งน้ำในโครงสร้างการจ่ายน้ำ

H_j คือหัวไฮโดรลิค ณ จุดจ่ายน้ำ j

H_j^{min} คือหัวไฮโดรลิคที่ขนาดเล็กที่สุดที่ต้องติดตั้ง ณ จุดจ่ายน้ำ j

q_j^{in} คือปริมาณน้ำที่ไหลผ่านเข้ามา ณ จุดจ่ายน้ำ j

q_j^{out} คือปริมาณน้ำที่ไหลออกจากจุดจ่ายน้ำ j

q_j คือปริมาณน้ำที่ต้องการใช้ ณ จุดจ่ายน้ำ j

HL_i คือปริมาณการสูญเสียน้ำภายในท่อส่งน้ำ i

nd คือจำนวนจุดจ่ายน้ำ

np_L คือจำนวนท่อส่งน้ำภายในลูป L

nL คือจำนวนลูปในระบบจ่ายน้ำ

จากสมการที่ 2.9 จะเห็นว่า ปัญหาการออกแบบโครงสร้างระบบการจ่ายน้ำมีข้อจำกัด 3 ข้อ คือ 1) หัวไฮโดรลิค ณ ตำแหน่งจ่ายน้ำใดๆ ควรมีขนาดใหญ่กว่าความต้องการใช้น้ำ ณ จุดนั้น 2) ความแตกต่างของปริมาณน้ำที่ไหลเข้าและที่ไหลผ่านไป ณ จุดจ่ายน้ำใดๆ ควรเท่ากับปริมาณน้ำที่ต้องการใช้ ณ จุดจ่ายน้ำนั้นและ 3) ผลรวมของการสูญเสียน้ำภายในลูปใดๆ ต้องเท่ากับศูนย์

โมเดลที่นำเสนอในงานวิจัยนี้ได้แทนค่าจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้เช่นเดียวกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่เสนอโดย Hussein A. Abbass แต่ได้ปรับการแทนค่าผึ้งงานที่แตกต่างออกไปจากเดิมคือ ในอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม ผึ้งงานแทนค่าด้วยฟังก์ชันฮิวริสติก แต่ในโมเดลที่นำเสนอใหม่นี้แทนค่าจีโนไทป์ของผึ้งงานด้วยอาร์เรย์ไบนารีเหมือนกับจีโนไทป์ของผึ้งนางพญา โดยกำหนดให้แต่ละยีนต์ในจีโนไทป์ของผึ้งนางพญา ผึ้งตัวผู้ และผึ้งงานแทนค่าเส้นผ่าศูนย์กลางของท่อส่งน้ำ ทั้งนี้ขนาดของท่อส่งน้ำต้องมีขนาดตามที่วางขายจริงในท้องตลาดและราคาต้นทุนของท่อส่งน้ำกำหนดตามราคาตลาดด้วยเช่นกัน นอกจากนี้ยังได้ศึกษาเปรียบเทียบการใช้โอเปอร์เรเตอร์ครอสโอเวอร์ จำนวน 3 ตัว ดังนี้

1) การครอสโอเวอร์แบบหนึ่งจุด (single-point crossover) โอเปอร์เรเตอร์นี้ทำการสุ่มเลือกจุดครอส (cross point) จำนวน 1 จุด กำหนดให้เป็นจุด c จากนั้นให้คัดลอกยีนต์ของผึ้งนางพญาจากตำแหน่งแรกจนถึงตำแหน่งที่ c ไปยังจีโนไทป์ของตัวอ่อนผึ้งในตำแหน่งเดียวกันและให้ทำการคัดลอกยีนต์ของผึ้งตัวผู้ตำแหน่งที่ c+1 จนถึงตำแหน่งสุดท้ายไปยังจีโนไทป์ของตัวอ่อนผึ้งในตำแหน่งเดียวกัน อย่างไรก็ตาม ถ้ายีนต์ที่คัดลอกจากผึ้งตัวผู้เป็นยีนต์ว่างให้ทำการเติมด้วยยีนต์ของผึ้งนางพญาให้เต็มทุกตำแหน่ง

2) การครอสโอเวอร์แบบหลายจุด (multipoint crossover) โอเปอร์เรเตอร์นี้มีลักษณะการทำงานเหมือนกับการครอสโอเวอร์แบบหนึ่งจุด แต่การครอสโอเวอร์แบบหลายจุด ต้องสุ่มเลือกจุดครอสจำนวนตั้งแต่ 2 จุดขึ้นไป

3) การครอสโอเวอร์แบบยูนิฟอร์ม (uniform crossover) โอเปอร์เรเตอร์นี้มีลักษณะการทำงานเหมือนกับการครอสโอเวอร์แบบแฮพพลอยด์ในอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม คือ ทำการคัดลอกจีโนไทป์ของผึ้งตัวผู้ไปยังจีโนไทป์ของตัวอ่อนผึ้ง จากนั้นให้เติมตำแหน่งยีนต์ว่างด้วยยีนต์ของผึ้งนางพญาในตำแหน่งเดียวกัน

ขั้นตอนการทำงานของโมเดลที่นำเสนอ มีดังนี้

1) การกำหนดค่าพารามิเตอร์เริ่มต้น

ค่าพารามิเตอร์ที่ต้องกำหนดประกอบด้วย จำนวนของผึ้งนางพญา ขนาดของถุงเก็บสเปิร์มของผึ้งนางพญา จำนวนของผึ้งตัวผู้ จำนวนของตัวอ่อนผึ้ง ค่าความน่าจะเป็นในการมิวเตชันและจำนวนรอบการทำงาน

2) การกำหนดค่าประชากรเริ่มต้น

ทำการสร้างประชากรเริ่มต้นจำนวนเท่ากับผลรวมของจำนวนของผึ้งนางพญา ผึ้งตัวผู้และผึ้งงานด้วยวิธีการสุ่ม จากนั้นให้คำนวณหาค่าฟิตเนสของประชากรทั้งหมดและทำการจัดเรียงลำดับประชากรดังกล่าวตามค่าฟิตเนส เพื่อทำการแบ่งประชากรเริ่มต้นเป็น 3 กลุ่ม โดยประชากรที่มีค่าฟิตเนสสูงจะจัดให้เป็นผึ้งนางพญาเริ่มต้น และประชากรที่มีค่าฟิตเนสต่ำลงมาจะจัดให้เป็นผึ้งตัวผู้เริ่มต้นและผึ้งงานเริ่มต้นตามลำดับ

3) การผสมพันธุ์ระหว่างผึ้งนางพญาและผึ้งตัวผู้

ขั้นตอนนี้ยังคงมีการทำงานเช่นเดียวกันอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม กล่าวคือ ผึ้งนางพญาทุกตัวมีหน้าที่ในการผสมพันธุ์และสามารถผสมพันธุ์กับผึ้งตัวผู้ได้หลายตัว โดยผึ้งนางพญาแต่ละตัวตัดสินใจเลือกผสมพันธุ์จากค่าความน่าจะเป็นในการผสมพันธุ์ ผึ้งตัวผู้ที่มีค่าความน่าจะเป็นในการผสมพันธุ์สูงที่สุดจะถูกเลือกเพื่อผสมพันธุ์และสเปิร์มของผึ้งตัวผู้จะถูกเก็บไว้ในถุงเก็บสเปิร์มของผึ้งนางพญา จากนั้นทำการปรับค่าพลังงานและค่าความเร็วในการบินลดลงก่อนเริ่มการเลือกคู่ครั้งถัดไป การผสมพันธุ์ของผึ้งนางพญาแต่ละตัวสิ้นสุดลง เมื่อถุงเก็บสเปิร์มเต็มหรือค่าพลังงานน้อยกว่าค่าที่กำหนดไว้ ดังนั้น ผึ้งนางพญาแต่ละตัวจะผสมพันธุ์กับผึ้งตัวผู้จำนวนมากหรือน้อยขึ้นอยู่กับค่าพลังงานเริ่มต้นและขนาดของถุงเก็บสเปิร์ม

4) การสร้างตัวอ่อน

ผึ้งนางพญาถูกสุ่มเลือกด้วยวิธีการเลือกแบบวงล้อหมุนเพื่อทำหน้าที่สร้างตัวอ่อนผึ้ง ผึ้งนางพญาที่ได้รับเลือกจะสร้างตัวอ่อนผึ้งโดยการสุ่มเลือกสเปิร์มจากถุงเก็บสเปิร์มเพื่อทำการครอส-โอเวอร์กับจีโนไทป์ของตนเอง งานวิจัยนี้มีการใช้โอเปอร์เรเตอร์ครอสโอเวอร์ 3 ตัวคือ การครอส-โอเวอร์แบบหนึ่งจุด การครอสโอเวอร์แบบหลายจุดและการครอสโอเวอร์แบบยูนิฟอร์ม จากนั้นตัว

อ่อนผึ้งที่ได้จะถูกมิวเตท กระบวนการสร้างตัวอ่อนผึ้งสิ้นสุดลงเมื่อจำนวนของตัวอ่อนผึ้งครบตามที่กำหนดไว้

5) การเลี้ยงดูตัวอ่อน

ผึ้งงานทำหน้าที่ดูแลตัวอ่อนผึ้ง โดยการสลับค่ายีนต์ระหว่างตัวอ่อนผึ้งและผึ้งงานในบางตำแหน่ง ซึ่งการพิจารณาสลับยีนต์ได้มีการกำหนดค่าความน่าจะเป็นในการสลับยีนต์ ด้วยค่าคงที่ค่าน้อยๆ ค่าหนึ่ง

6) การคัดเลือกผึ้งนางพญาใหม่

การคัดเลือกผึ้งนางพญาใหม่ทำได้โดยการเปรียบเทียบค่าฟิตเนสระหว่างผึ้งนางพญาในรอบปัจจุบันกับตัวอ่อนผึ้ง ผึ้งนางพญาในรอบปัจจุบันที่มีค่าฟิตเนสต่ำกว่าตัวอ่อนผึ้งจะถูกคัดทิ้ง และตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าจะถูกเลือกให้เป็นผึ้งนางพญาใหม่

7) ทำซ้ำขั้นตอนที่ 3-6 จนกระทั่งจำนวนรอบการทำงานครบตามที่กำหนดไว้ จึงหยุดการทำงาน

งานวิจัยนี้ได้ทดสอบโมเดลที่นำเสนอด้วยปัญหาการออกแบบโครงสร้างระบบการจ่ายน้ำ จำนวน 2 ตัวอย่างคือ ปัญหา WDN-I ที่เสนอโดย Alperovits และ Shamir [13] และปัญหา WDN-II ที่เสนอโดย Fujiwara และ Khang [14] และทำการเปรียบเทียบผลการทดลองกับโมเดลที่ปรับปรุงมาจากจีเนติกอัลกอริทึม จำนวน 3 โมเดล อัลกอริทึมซิมูเลทเทดแอนนิลลิง (simulated annealing) อัลกอริทึมการกระโดดของกบ (shuffled frog leaping algorithm) และอัลกอริทึม complete enumeration

การทดลองที่ 1 เป็นการทดสอบกับปัญหา WDN-I โดยได้กำหนดจำนวนรอบการทำงานเท่ากับ 100 ขนาดของถุงเก็บสเปิร์มเท่ากับ 20 ค่าความเร็วในการบินเริ่มต้นเท่ากับ 0.6 และจำนวนของตัวอ่อนผึ้งต่อผึ้งนางพญา 1 ตัวเท่ากับ 20 สำหรับค่าพารามิเตอร์อื่นๆ ได้มีการศึกษาเปรียบเทียบการกำหนดค่าที่เหมาะสม ดังนี้ 1) จำนวนของผึ้งนางพญา มีทั้งหมด 4 ค่าคือ 1, 2, 3 และ 4 ตามลำดับ 2) จำนวนของผึ้งตัวผู้ มีทั้งหมด 5 ค่าคือ 100, 150, 200, 250 และ 300 ตามลำดับและ 3) จำนวนของผึ้งงาน มีทั้งหมด 6 ค่าคือ 25, 50, 75, 100, 125 และ 150 ตามลำดับ นอกจากนี้ยังได้ศึกษาเปรียบเทียบการใช้โอเปอร์เรเตอร์ครอสโอเวอร์ที่เหมาะสมจำนวน 3 ตัวคือ การครอสโอเวอร์แบบหนึ่งจุด การครอสโอเวอร์แบบหลายจุดและการครอสโอเวอร์แบบยูนิฟอร์ม และศึกษาการเปรียบเทียบการใช้โอเปอร์เรเตอร์ผึ้งงานและแบบที่ไม่มีการใช้โอเปอร์เรเตอร์ผึ้งงาน จากผลการทดลอง สรุปได้ว่า

- การกำหนดค่าพารามิเตอร์ที่ทำให้โมเดลที่นำเสนอนี้สามารถค้นหาคำตอบที่เหมาะสมที่สุดของปัญหา WDN-I ได้คือ กำหนดค่าจำนวนของผึ้งนางพญาเท่ากับ 3 จำนวนของผึ้งตัวผู้เท่ากับ 200 และจำนวนของผึ้งงานเท่ากับ 100

- การใช้การครอสโอเวอร์แบบยูนิฟอร์มทำให้โมเดลที่นำเสนอสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ดีกว่าการครอสโอเวอร์แบบหนึ่งจุดหรือหลายจุด
- การใช้โอเปอร์เรเตอร์ผึ้งงานเพื่อดูแลตัวอ่อนผึ้งทำให้โมเดลที่นำเสนอสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ดีกว่าโมเดลที่ไม่มีการใช้โอเปอร์เรเตอร์ผึ้งงาน

การทดลองที่ 2 เป็นการทดสอบกับปัญหา WDN-II โดยได้กำหนดจำนวนรอบการทำงานเท่ากับ 150 ขนาดของถุงเก็บสเปิร์มเท่ากับ 20 ค่าความเร็วในการบินเริ่มต้นเท่ากับ 0.6 และจำนวนของตัวอ่อนผึ้งต่อผึ้งนางพญา 1 ตัวเท่ากับ 20 สำหรับค่าพารามิเตอร์อื่นๆ ได้มีการศึกษาเปรียบเทียบการกำหนดค่าที่เหมาะสม ดังนี้ 1) จำนวนของผึ้งนางพญา มีทั้งหมด 4 ค่าคือ 3, 4, 5 และ 6 ตามลำดับ 2) จำนวนของผึ้งตัวผู้ มีทั้งหมด 5 ค่าคือ 100, 150, 200, 250 และ 300 ตามลำดับและ 3) จำนวนของผึ้งงาน มีทั้งหมด 6 ค่าคือ 25, 50, 75, 100, 125 และ 150 ตามลำดับ จากผลการทดลองสรุปได้ว่า

- การกำหนดค่าพารามิเตอร์ที่ทำให้โมเดลที่นำเสนอนี้ให้คำตอบที่ดีที่สุดคือ กำหนดค่าจำนวนของผึ้งนางพญาเท่ากับ 5 จำนวนของผึ้งตัวผู้เท่ากับ 100 และจำนวนของผึ้งงานเท่ากับ 100
- การใช้การครอสโอเวอร์แบบยูนิฟอร์มทำให้โมเดลที่นำเสนอสามารถค้นหาคำตอบได้ดีกว่าการครอสโอเวอร์แบบหนึ่งจุดหรือหลายจุด
- การใช้โอเปอร์เรเตอร์ผึ้งงานเพื่อดูแลตัวอ่อนผึ้งทำให้โมเดลที่นำเสนอสามารถค้นหาคำตอบได้ดีกว่าโมเดลที่ไม่มีการใช้โอเปอร์เรเตอร์ผึ้งงาน

จากการเปรียบเทียบผลการทดลองระหว่างโมเดลที่นำเสนอกับอัลกอริทึมที่นำมาเปรียบเทียบ พบว่า โมเดลที่นำเสนอสามารถหาคำตอบที่เหมาะสมที่สุดของปัญหา WDN-I ได้เช่นเดียวกับอัลกอริทึมอื่นที่นำมาเปรียบเทียบและได้ใช้จำนวนรอบในการประมวลผลเพื่อหาค้นหาคำตอบน้อยกว่า สำหรับปัญหา WDN-II โมเดลที่นำเสนอยังไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ แต่มีค่าความผิดพลาดน้อยกว่าเมื่อเปรียบเทียบกับอัลกอริทึมอื่นๆ

จากการศึกษางานวิจัยนี้สามารถวิเคราะห์จุดเด่นที่สำคัญได้ว่า อัลกอริทึมที่นำเสนอเป็นความพยายามในการปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันเชิงการจัดที่มีการแทนค่าจีโนไทป์แบบข้อความ (text representation) ซึ่งแตกต่างจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่ใช้ในการแก้ปัญหาออฟติไมเซชันเชิงตัวเลขที่มีการแทนค่าจีโนไทป์แบบตัวเลข ซึ่งอาจเป็นการแทนค่าแบบไบนารี (binary representation) หรือการแทนค่าแบบเลขจำนวนจริง (real representation) ดังนั้นถ้ามีการแทนค่าปัญหาที่แตกต่างกันย่อมต้องมีการปรับเปลี่ยนอัลกอริทึมให้เหมาะสมกับวิธีการแทนค่าด้วย งานวิจัยนี้จึงเป็นประโยชน์แก่นักวิจัยที่มีความสนใจอัลกอริทึมใหม่ที่เหมาะสมกับปัญหาออฟติไมเซชันเชิงการจัด อย่างไรก็ตามถึงแม้ว่าอัลกอริทึมที่นำเสนอได้มีการปรับเปลี่ยนการแทนค่ารวมทั้งโอเปอร์เรเตอร์ที่ใช้งาน แต่

ไม่ได้มีการปรับปรุงโครงสร้างการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม จึงทำให้จุดด้อยของอัลกอริทึมที่นำเสนอยังคงมีเหมือนกับในอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

2.3.4 An intelligent genetic algorithm designed for global optimization of multi-minima functions

Li-ning, Ying-wu และ Huai-ping [15] นำเสนอโมเดล IGA (intelligent genetic algorithm) ซึ่งเป็นโมเดลที่เกิดจากการปรับปรุงจีเนติกอัลกอริทึมให้มีความชาญฉลาดมากยิ่งขึ้น โดยมีแนวคิดว่า การใช้โอเปอร์เรเตอร์ครอสโอเวอร์หรือมิวเตชันเพียงโอเปอร์เรเตอร์ละ 1 ตัวอาจลดประสิทธิภาพและประสิทธิผลของจีเนติกอัลกอริทึมในการแก้ปัญหาด้านออฟติไมเซชัน ดังนั้นงานวิจัยนี้จึงได้นำเสนอโมเดลใหม่ที่สามารถเลือกทำงานร่วมกับโอเปอร์เรเตอร์ครอสโอเวอร์และโอเปอร์เรเตอร์มิวเตชันได้โดยแต่ละโอเปอร์เรเตอร์มีจำนวนมากกว่าหนึ่งตัว ทั้งนี้ได้มีวิธีการเลือกใช้งานแต่ละโอเปอร์เรเตอร์อย่างอัตโนมัติ นอกจากนี้ยังได้มีการนำเสนอโอเปอร์เรเตอร์ใหม่เพื่อทำให้ประสิทธิภาพในการทำงานของจีเนติกอัลกอริทึมดียิ่งขึ้น ซึ่งเรียกว่า "โอเปอร์เรเตอร์แบบรีแอทเทม" (reattempt operator) โดยโอเปอร์เรเตอร์นี้ทำหน้าที่ในการปรับเปลี่ยนประชากรเริ่มต้นใหม่ก่อนการทำงานในรอบการทำงานถัดไป ทั้งนี้โอเปอร์เรเตอร์นี้ทำงานเมื่อในรอบการทำงานก่อนหน้ามีการค้นหาคำตอบได้ซ้ำกันครบจำนวนครั้งที่กำหนดไว้ เช่น เมื่อรอบการทำงานที่ 1 ถึงรอบการทำงานที่ 10 ค้นหาคำตอบได้ซ้ำกันครบ 10 ครั้ง โอเปอร์เรเตอร์แบบรีแอทเทมจึงทำการปรับเปลี่ยนประชากรเริ่มต้นใหม่ก่อนเริ่มการทำงานในรอบการทำงานที่ 11 เป็นต้น

ในงานวิจัยนี้ได้ใช้โอเปอร์เรเตอร์ครอสโอเวอร์จำนวน 5 ตัว ประกอบด้วย Cr = {Cr_1, Cr_2, ..., Cr_5} โอเปอร์เรเตอร์มิวเตชันจำนวน 8 ตัว ประกอบด้วย Mu = {Mu_1, Mu_2, ..., Mu_8} และโอเปอร์เรเตอร์แบบรีแอทเทมจำนวน 5 ตัว ประกอบด้วย Zb = {Zb_1, Zb_2, ..., Zb_5} ซึ่งมีรายละเอียดดังนี้

1) โอเปอร์เรเตอร์ครอสโอเวอร์ ทำหน้าที่สร้างโครโมโซมรุ่นลูกจากโครโมโซมพ่อและโครโมโซมแม่

- Cr_1 : choose big and choose small

โอเปอร์เรเตอร์นี้ทำการครอสโอเวอร์เพื่อสร้างโครโมโซมลูกจำนวน 2 โครโมโซมโดยโครโมโซมแรกเลือกจากยีนต์ที่มีค่ามากและโครโมโซมที่สองเลือกจากยีนต์ที่มีค่าน้อย โดยพิจารณาแต่ละตำแหน่งของยีนต์

ตัวอย่างเช่น โครโมโซมพ่อ (1, 6, 8, 10) และโครโมโซมแม่ (2, 7, 3, 9) ดังนั้นโครโมโซมลูกที่เลือกจากยีนต์ที่มีค่ามากในแต่ละตำแหน่ง จะได้ (2, 7, 8, 10) และโครโมโซมลูกที่เลือกจากยีนต์ที่มีค่าน้อยในแต่ละตำแหน่ง จะได้ (1, 6, 3, 9)

- Cr_2 : orthogonal crossover with quantization

โอเปอร์เรเตอร์นี้ทำการครอสโอเวอร์โดยการกำหนดพื้นที่การค้นหาจากเวคเตอร์ค่าต่ำสุดและค่าสูงสุดที่ได้จากโครโมโซมพ่อแม่ แล้วทำการสร้างเซตของจุด (β) ในแต่ละมิติ โดยกำหนดให้ β_i แทนเซตของจุดในมิติที่ i และจำนวนจุดในแต่ละมิติเท่ากับ Q_2 จากนั้นให้เลือกโครโมโซมลูกจากคอมไบเนชัน (combination) ที่สร้างจากจุดใน β_i

ตัวอย่างเช่น โครโมโซมพ่อ (1, 6, 8, 10) และโครโมโซมแม่ (2, 7, 3, 9) ดังนั้นขอบเขตล่างและขอบเขตบนของพื้นที่การค้นหาคือ (1, 6, 3, 9) และ (2, 7, 8, 10) ตามลำดับ สมมติว่า กำหนดค่า Q_2 เท่ากับ 3 จะได้เซตของจุดในแต่ละมิติ ดังนี้ $\beta_1 = \{1, 1.5, 2\}$ $\beta_2 = \{6, 6.5, 7\}$ $\beta_3 = \{3, 5.5, 8\}$ และ $\beta_4 = \{9, 9.5, 10\}$ จากนั้นให้ทำการสร้างคอมไบเนชันจาก β_i เช่น (1, 6, 5.5, 9.5) เป็นต้น เพื่อคัดเลือกโครโมโซมลูกจำนวน 2 โครโมโซมจากโครโมโซมลูกที่ดีที่สุด

- Cr_3 : arithmetical crossover

โอเปอร์เรเตอร์นี้ทำการครอสโอเวอร์โดยการสร้างบอร์ด (board) เพื่อระบุตำแหน่งการครอสโอเวอร์ด้วยการสุ่มตัวเลข 0 หรือ 1 จำนวนเท่ากับความยาวของโครโมโซม ซึ่งค่า 0 แทนตำแหน่งที่ไม่มีการครอสโอเวอร์และค่า 1 แทนตำแหน่งครอสโอเวอร์ จากนั้นให้สร้างอาร์เรย์สัมประสิทธิ์ t โดยการสุ่มตัวเลขจำนวนจริงที่อยู่ในช่วง [0, 1] แล้วจึงทำการครอสโอเวอร์เพื่อสร้างโครโมโซมลูกลำดับที่ 1 ได้จาก $c_i = t_i \times g_{i1} + (1 - t_i) \times g_{i2}$ และโครโมโซมลูกลำดับที่ 2 ได้จาก $c_i = t_i \times g_{i2} + (1 - t_i) \times g_{i1}$ เมื่อ c_i, g_{i1} และ g_{i2} คือยีนต์ตำแหน่งที่ i ของโครโมโซมลูก โครโมโซมพ่อและโครโมโซมแม่ ตามลำดับ

ตัวอย่างเช่น โครโมโซมพ่อ (1, 6, 8, 10) และโครโมโซมแม่ (2, 7, 3, 9) ดังนั้นความยาวของโครโมโซมเท่ากับ 4 ให้ทำการสร้างบอร์ดโดยการสุ่มได้ (1, 0, 1, 0) แสดงว่ามีการครอสโอเวอร์ 2 ตำแหน่ง และทำการสร้างอาร์เรย์สัมประสิทธิ์โดยการสุ่มได้ t = (0.2, 0.5, 0.4, 0.3) จากนั้นสร้างโครโมโซมลูกโดยการครอสโอเวอร์เฉพาะยีนต์ตำแหน่งที่ 1 และ 3 จะได้โครโมโซมลูก (1.8, 6, 5, 10) และ (1.2, 7, 6, 9)

- Cr_4 : one point cross operator

โอเปอร์เรเตอร์นี้ทำการครอสโอเวอร์เพื่อสร้างโครโมโซมลูกโดยการปรับค่ายีนต์ของโครโมโซมพ่อแม่ด้วยตัวแปรสุ่ม 1 ตัว กำหนดให้ r แทนตัวแปรสุ่ม โครโมโซมลูกลำดับที่ 1 ได้จาก $c_i = g_{i1} + (1 - r) \times g_{i2}$ และโครโมโซมลูกลำดับที่ 2 ได้จาก $c_i = g_{i2} + (1 - r) \times g_{i1}$ เมื่อ c_i, g_{i1} และ g_{i2} คือยีนต์ตำแหน่งที่ i ของโครโมโซมลูก โครโมโซมพ่อและโครโมโซมแม่ ตามลำดับ

ตัวอย่างเช่น โครโมโซมพ่อ (1, 6, 8, 10) และโครโมโซมแม่ (2, 7, 3, 9) ให้ทำการสุ่มค่า r ได้ 0.8 จากนั้นทำการครอสโอเวอร์เพื่อสร้างโครโมโซมลูกจะได้ (1.4, 7.4, 8.6, 11.8) และ (2.2, 8.2, 4.6, 11)

- Cr_5 : double point cross operator

โอเปอร์เรเตอร์นี้ทำการครอสโอเวอร์เพื่อสร้างโครโมโซมลูกโดยการปรับค่ายีนต์ของโครโมโซมพ่อแม่ด้วยตัวแปรสุ่ม 2 ตัว กำหนดให้ r_1 และ r_2 แทนตัวแปรสุ่ม โครโมโซมลูกลำดับที่ 1 ได้จาก $c_1 = r_1 \times g_{11} + (1 - r_2) \times g_{12}$ และโครโมโซมลูกลำดับที่ 2 ได้จาก $c_1 = r_1 \times g_{12} + (1 - r_2) \times g_{11}$ เมื่อ c_i, g_{i1} และ g_{i2} คือยีนต์ตำแหน่งที่ i ของโครโมโซมลูก โครโมโซมพ่อและโครโมโซมแม่ ตามลำดับ

ตัวอย่างเช่น โครโมโซมพ่อ (1, 6, 8, 10) และโครโมโซมแม่ (2, 7, 3, 9) ให้ทำการสุ่มค่า r_1 และ r_2 ได้ 0.2 และ 0.5 ตามลำดับ จากนั้นทำการครอสโอเวอร์เพื่อสร้างโครโมโซมลูกจะได้ (1.2, 4.7, 3.1, 6.5) และ (0.9, 4.4, 4.6, 6.8)

2) โอเปอร์เรเตอร์มิวเตชัน ทำหน้าที่มิวเตทโครโมโซมลูกที่ได้จากขั้นตอนการครอสโอเวอร์ โดยการปรับเปลี่ยนยีนต์บางตำแหน่งของโครโมโซมลูก

- Mu_1 : random number mutation

โอเปอร์เรเตอร์นี้ทำการมิวเตทโดยการสุ่มค่า ε เพื่อแทนค่ายีนต์ในตำแหน่งสุ่มใด ๆ ในโครโมโซม เช่น โครโมโซม $(x_1, x_2, \ldots, x_i, \ldots, x_n)$ ถูกมิวเตทตำแหน่งที่ i ได้เป็นโครโมโซม $(x_1, x_2, \ldots, \varepsilon, \ldots, x_n)$

- Mu_2 : random one-dimension mutation

โอเปอร์เรเตอร์นี้ทำการมิวเตทโดยการสุ่มค่า ε เพื่อนำไปบวกกับค่าของยีนต์ในตำแหน่งสุ่มใดๆ ในโครโมโซม เช่น โครโมโซม $(x_1, x_2, \ldots, x_i, \ldots, x_n)$ ถูกมิวเตทตำแหน่งที่ i ได้เป็นโครโมโซม $(x_1, x_2, \ldots, x_i + \varepsilon, \ldots, x_n)$

- Mu_3 : additional perturbations mutation

โอเปอร์เรเตอร์นี้ทำการมิวเตทโดยการสุ่มค่า $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ เมื่อ n คือจำนวนยีนต์ในโครโมโซม เพื่อนำไปบวกกับค่าของยีนต์ในโครโมโซม เช่น โครโมโซม $(x_1, x_2, \ldots, x_i, \ldots, x_n)$ ถูกมิวเตทได้เป็นโครโมโซม $(x_1 + \varepsilon_1, x_2 + \varepsilon_2, \ldots, x_i + \varepsilon_i, \ldots, x_n + \varepsilon_n)$

- Mu_4 : multi-place interchange mutation

โอเปอร์เรเตอร์นี้ทำการมิวเตทโดยการสลับค่าของยีนต์ตำแหน่งที่ k แทนที่ยีนต์ตำแหน่งที่ k+j เมื่อ k = 1, 2, …, i หรือ k = 1, 2, …, n-j และ n คือจำนวนยีนต์ในโครโมโซม เช่น โครโมโซม $(x_1, \ldots, x_i, x_{i+1}, \ldots, x_j, \ldots, x_n)$ ถูกมิวเตทได้เป็นโครโมโซม $(x_j, \ldots, x_n, x_{i+1}, \ldots, x_1, \ldots, x_i)$

- Mu_5 : backformation mutation

โอเปอร์เรเตอร์นี้ทำการมิวเตทโดยการสลับค่าของยีนต์ ช่วงระหว่างตำแหน่งที่ i ถึงยีนต์ตำแหน่งที่ j ในโครโมโซม เช่น โครโมโซม $(x_1, \ldots, x_i, x_{i+1}, \ldots, x_{j-1}, x_j, \ldots, x_n)$ ถูกมิวเตทได้เป็นโครโมโซม $(x_1, \ldots, x_j, x_{j-1}, \ldots, x_{i+1}, x_i, \ldots, x_n)$

- Mu_6 : *i*-step circular replacement mutation

โอเปอร์เรเตอร์นี้ทำการมิวเตทโดยการเลื่อนยีนต์เป็นวงกลมจากซ้ายไปขวาจำนวน i ครั้ง เช่น โครโมโซม $(x_1, \ldots, x_i, x_{i+1}, \ldots, x_n)$ ถูกมิวเตทได้เป็นโครโมโซม $(x_{n-i+1}, \ldots, x_n, x_1, \ldots, x_{n-i})$

- Mu_7 : forth sling mutation

โอเปอร์เรเตอร์นี้ทำการมิวเตทโดยการปรับค่าของยีนต์ตำแหน่งที่ *i* ดังนี้ $x_t = x_t + r_1 \times r_2$ เมื่อ r_1 คือตัวแปรสุ่มทิศทางและ r_2 คือตัวแปรสุ่มระยะทาง

- Mu_8 : gauss mutation

โอเปอร์เรเตอร์นี้ทำการมิวเตทโดยการปรับค่าของยีนต์ด้วยค่าตัวเลขจำนวนจริงที่ได้จากการสุ่มที่มีการกระจายแบบปกติ ดังนี้ $x_t = x_t + N(0, \delta)$

3) โอเปอร์เรเตอร์แบบรีแอทเทม ทำหน้าที่ในการปรับเปลี่ยนประชากรเริ่มต้นใหม่ก่อนการทำงานในรอบการทำงานถัดไป ทั้งนี้โอเปอร์เรเตอร์นี้ทำงานเมื่อในรอบการทำงานก่อนหน้ามีการค้นหาคำตอบได้ซ้ำกันครบจำนวนครั้งที่กำหนดไว้

- Zb_1 : dynamic updating reattempt

โอเปอร์เรเตอร์นี้ทำการสร้างโครโมโซมใหม่จำนวนหนึ่งโดยการสุ่มเพื่อนำไปปรับปรุงประชากรในรอบปัจจุบัน

- Zb_2 : immunity reattempt

โอเปอร์เรเตอร์นี้ทำการปรับปรุงประชากรในรอบปัจจุบันโดยการใช้หลักการของอัลกอริทึมระบบภูมิคุ้มกัน (artificial immune system) ซึ่งเป็นอัลกอริทึมที่เกิดจากการเลียนแบบระบบภูมิคุ้มกันภายในร่างกายของมนุษย์

- Zb_3 : simplex reattempt

โอเปอร์เรเตอร์นี้ทำการปรับปรุงประชากรในรอบปัจจุบันโดยการใช้วิธีการซิมเพล็ก (simplex method) มาช่วยในการค้นหาคำตอบ

- Zb_4 : chaos search reattempt

โอเปอร์เรเตอร์นี้ทำการปรับปรุงประชากรในรอบปัจจุบันโดยการใช้หลักการของการค้นหาแบบเคออส (chaos) มาช่วยในการค้นหาแบบโลคอล

- Zb_5 : steepest descent reattempt

โอเปอร์เรเตอร์นี้ทำการปรับปรุงประชากรในรอบปัจจุบันโดยการใช้หลักการของการค้นหาแบบ steepest descent

ขั้นตอนการทำงานของโมเดล IGA ประกอบด้วยขั้นตอนหลัก ดังนี้

1) การแทนค่าโครโมโซมและการกำหนดค่าประชากรเริ่มต้น

กำหนดให้มีประชากรเริ่มต้นจำนวน m ตัว แต่ละตัวแทนค่าด้วยอาร์เรย์ของเลขจำนวนจริงที่มีความยาว n อิลิเมนต์ เมื่อ n คือขนาดของปัญหา ดังนั้น อาร์เรย์จึงมีขนาดเท่ากับ $m \times n$ ประชากรเริ่มต้นทั้งหมดนี้ถูกสร้างโดยการสุ่มค่าให้อยู่ในช่วง [0, 1] จากนั้นจึงทำการปรับค่าแต่ละคอลัมน์ให้อยู่ในช่วงค่าต่ำสุดและค่าสูงสุดของแต่ละตัวแปรในปัญหาที่กำลังทดสอบ

2) การคำนวณค่าฟิตเนสของประชากรและโอเปอร์เรเตอร์

ประชากรและโอเปอร์เรเตอร์ทุกตัวถูกวัดประสิทธิภาพด้วยค่าฟิตเนส ค่าฟิตเนสของประชากรคำนวณได้ ดังสมการที่ 2.10

$$\text{fitness}(x) = f(x) + \sum_{i=1}^{r} \{M \times [\max(g_i(x), 0)]\} + \sum_{i=1}^{q} \{M \times [h_j(x)]\} \qquad (2.10)$$

สำหรับค่าฟิตเนสของโอเปอร์เรเตอร์แต่ละตัวคำนวณได้จากการเปรียบเทียบประสิทธิภาพของประชากรรุ่นลูกเทียบกับประชากรรุ่นพ่อแม่ ถ้าค่าฟิตเนสเฉลี่ยของประชากรในรุ่นลูกมากกว่าในรุ่นพ่อแม่ โอเปอร์เรเตอร์ใดก็ตามที่ถูกใช้ในรอบปัจจุบันจะได้รับการเพิ่มค่าฟิตเนส งานวิจัยนี้ได้เพิ่มค่าฟิตเนสของโอเปอร์เรเตอร์โดยการบวกเพิ่มครั้งละ 1 คะแนน

3) การคัดเลือกประชากรใหม่

ประชากรใหม่ในรอบถัดไปคัดเลือกจากประชากรรุ่นพ่อแม่และรุ่นลูกในรอบปัจจุบันด้วยวิธีการสุ่มที่มีการกระจายแบบปกติ ทั้งนี้โครโมโซมที่มีค่าฟิตเนสสูงที่สุด 5 อันดับแรกจะถูกเลือกให้เป็นประชากรใหม่โดยอัตโนมัติเพื่อรักษาคำตอบที่ดีไว้

4) การเลือกใช้โอเปอร์เรเตอร์ครอสโอเวอร์

โอเปอร์เรชันครอสโอเวอร์ ทำหน้าที่ในการสร้างประชากรรุ่นลูก ในงานวิจัยนี้ได้ใช้โอเปอร์เรเตอร์ครอสโอเวอร์จำนวน 5 ตัว ดังนั้น ขั้นตอนการครอสโอเวอร์จึงต้องมีกระบวนการในการเลือกใช้โอเปอร์เรเตอร์ครอสโอเวอร์ ซึ่งกระบวนการดังกล่าว มีขั้นตอนดังนี้

4.1 กำหนดให้ Mc[i] แทนค่าฟิตเนสของโอเปอร์เรเตอร์ครอสโอเวอร์ตัวที่ i เมื่อ i = 1, 2, …, 5 และกำหนดค่าฟิตเนสเริ่มต้นเท่ากับ 10

4.2 คำนวณหาค่าความน่าจะเป็นในการเลือกโอเปอร์เรเตอร์แต่ละตัว ได้ดังนี้

$$PMc[i] = \frac{Mc[i]}{\sum_{i=1}^{5} Mc[i]} \qquad (2.11)$$

4.3 เลือกโอเปอร์เรเตอร์ครอสโอเวอร์ด้วยวิธีการเลือกแบบวงล้อหมุนโดยคำนวณสัดส่วนในวงล้อจากค่าความน่าจะเป็นในการเลือกที่คำนวณจากสมการที่ 2.11

4.4 นำโอเปอร์เรเตอร์ที่ได้รับเลือกไปทำการครอสโอเวอร์ระหว่างโครโมโซมพ่อและโครโมโซมแม่ ทั้งนี้ต้องมีการพิจารณาค่าความน่าจะเป็นในการครอสโอเวอร์ด้วย งานวิจัยนี้ได้กำหนดค่าความน่าจะเป็นในการครอสโอเวอร์ของแต่ละโครโมโซมแตกต่างกันตามค่าฟิตเนสของโครโมโซมนั้น ซึ่งคำนวณได้ดังนี้

$$Pc[i] = \begin{cases} 0 & f_i = f_{max} \\ \dfrac{(f_{max} - f_{avg})}{f_i} & f_i > f_{max} - f_{avg} \\ 1 & f_i \le f_{max} - f_{avg} \end{cases} \qquad (2.12)$$

เมื่อ Pc[i] คือค่าความน่าจะเป็นในการครอสโอเวอร์ของโครโมโซมตัวที่ i

f_{max} คือค่าฟิตเนสที่สูงที่สุดของประชากรทั้งหมด

f_{avg} คือค่าฟิตเนสเฉลี่ยของประชากรทั้งหมด

f_i คือค่าฟิตเนสของโครโมโซมตัวที่ i

5) การเลือกใช้โอเปอร์เรเตอร์มิวเตชัน

โอเปอร์เรชันมิวเตชัน ทำหน้าที่ปรับเปลี่ยนยีนต์บางตำแหน่งในโครโมโซม งานวิจัยนี้ได้ใช้โอเปอร์เรเตอร์มิวเตชันจำนวน 8 ตัว ดังนั้น ในขั้นตอนการมิวเตชันจึงต้องมีกระบวนการในการเลือกใช้โอเปอร์เรเตอร์มิวเตชัน ซึ่งกระบวนการดังกล่าว มีขั้นตอนดังนี้

5.1 กำหนดให้ Mm[i] แทนค่าฟิตเนสของโอเปอร์เรเตอร์มิวเตชันตัวที่ i เมื่อ i = 1, 2, …, 8 และกำหนดค่าฟิตเนสเริ่มต้นเท่ากับ 10

5.2 คำนวณหาค่าความน่าจะเป็นในการเลือกโอเปอร์เรเตอร์แต่ละตัว ได้ดังนี้

$$PMm[i] = \frac{Mm[i]}{\sum_{i=1}^{8} Mm[i]} \qquad (2.13)$$

5.3 เลือกโอเปอร์เรเตอร์มิวเตชันด้วยวิธีการเลือกแบบวงล้อหมุนโดยคำนวณสัดส่วนในวงล้อจากค่าความน่าจะเป็นในการเลือกที่คำนวณจากสมการที่ 2.13

5.4 นำโอเปอร์เรเตอร์ที่ได้รับเลือกไปทำการมิวเตชัน ทั้งนี้ต้องมีการพิจารณาค่าความน่าจะเป็นในการมิวเตชันด้วย งานวิจัยนี้ได้กำหนดค่าความน่าจะเป็นในการมิวเตชันของแต่ละโครโมโซมแตกต่างกันตามค่าฟิตเนสของโครโมโซมนั้น ซึ่งคำนวณได้ดังนี้

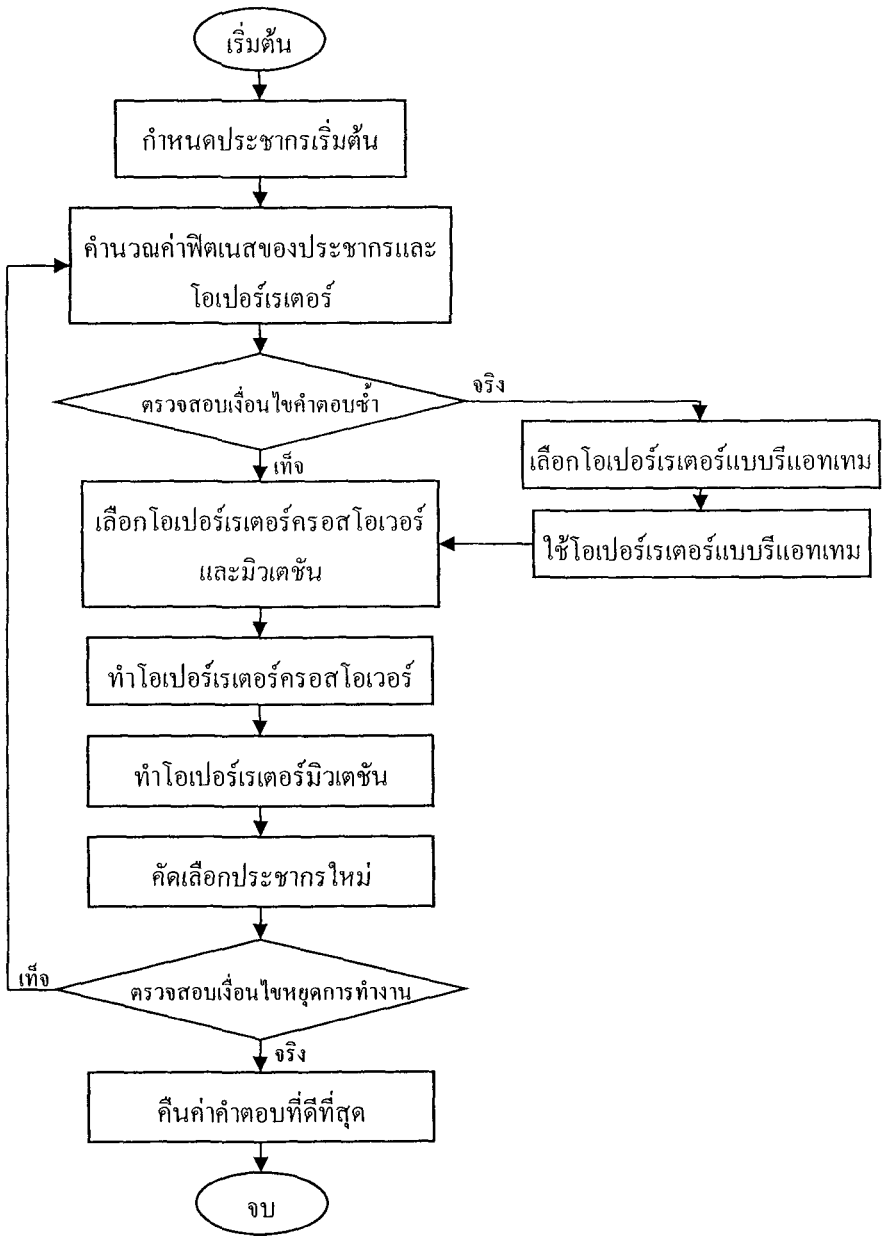
$$Pm[i] = \begin{cases} 0.5 \cdot \dfrac{(f_{max} - f_i)}{(f_{max} - f_{avg})} & f_i > f_{avg} \\ 0.5 & f_i \le f_{avg} \end{cases} \qquad (2.14)$$

เมื่อ Pm[i] คือค่าความน่าจะเป็นในการมิวเตชันของโครโมโซมตัวที่ i

f_{max} คือค่าฟิตเนสที่สูงที่สุดของประชากรทั้งหมด

f_{avg} คือค่าฟิตเนสเฉลี่ยของประชากรทั้งหมด

f_i คือค่าฟิตเนสของโครโมโซมตัวที่ i



รูปที่ 2.5 โครงสร้างการทำงานของโมเดล IGA

6) การเลือกใช้โอเปอร์เรเตอร์แบบรีแอทเทม

โอเปอร์เรชันแบบรีแอทเทม ทำหน้าที่ปรับเปลี่ยนประชากรบางส่วนใหม่ก่อนเริ่มต้นการทำงานในรอบการทำงานถัดไป โดยโอเปอร์เรชันนี้เริ่มทำงานเมื่อค่าคำตอบที่ดีที่สุดที่ได้จากโมเดล IGA ไม่มีการเปลี่ยนแปลงติดต่อกันครบ 10 รอบการทำงาน งานวิจัยนี้ได้ใช้โอเปอร์เรเตอร์แบบรีแอทเทมจำนวน 5 ตัว ดังนั้น ในขั้นตอนการรีแอทเทมจึงต้องมีกระบวนการในการเลือกใช้โอเปอร์เรเตอร์แบบรีแอทเทม ซึ่งกระบวนการดังกล่าว มีขั้นตอนดังนี้

6.1 กำหนดให้ Mr[i] แทนค่าฟิตเนสของโอเปอร์เรเตอร์แบบรีแอทเทมตัวที่ i เมื่อ i = 1, 2, ..., 5 และกำหนดค่าฟิตเนสเริ่มต้นเท่ากับ 10

6.2 คำนวณหาค่าความน่าจะเป็นในการเลือกโอเปอร์เรเตอร์แต่ละตัว ได้ดังนี้

$$PMr[i] = \frac{Mr[i]}{\sum_{i=1}^{8} Mr[i]} \qquad (2.15)$$

6.3 เลือกโอเปอร์เรเตอร์แบบรีแอทเทมด้วยวิธีการเลือกแบบวงล้อหมุนโดยคำนวณสัดส่วนในวงล้อจากค่าความน่าจะเป็นในการเลือกที่คำนวณจากสมการที่ 2.15

6.4 นำโอเปอร์เรเตอร์ที่ได้รับเลือกไปทำการสร้างประชากรรุ่นลูกจากประชากรในรอบปัจจุบัน ถ้าโครโมโซมใดในประชากรรุ่นลูกมีค่าฟิตเนสดีกว่าประชากรรุ่นพ่อแม่ ให้ทำการแทนโครโมโซมลูกที่มีค่าฟิตเนสดีกว่าลงในประชากรรุ่นพ่อแม่ จากนั้นให้ทำการเพิ่มค่าฟิตเนสของโอเปอร์เรเตอร์แบบรีแอทเทมที่ได้รับเลือกแล้วจึงเริ่มต้นรอบการทำงานใหม่

การทดลองได้ทดสอบประสิทธิภาพของโมเดล IGA กับฟังก์ชันออฟติไมเซชันจำนวน 10 ฟังก์ชันซึ่งประกอบด้วยฟังก์ชันที่มีพารามิเตอร์ตั้งแต่ 2–100 ตัว โดยแต่ละฟังก์ชันมีการรันทั้งหมด 100 ครั้งและได้มีการเปรียบเทียบผลการทดลองของโมเดลที่นำเสนอกับโมเดลในงานวิจัยอื่นๆ จำนวน 9 โมเดลซึ่งเป็นโมเดลที่พัฒนามาจากจีเนติกอัลกอริทึม อัลกอริทึมซิมูเลทเทดแอนนิลลิง และอัลกอริทึมการค้นหาแบบตาบู (tabu search) การทดสอบในครั้งนี้ได้กำหนดเกณฑ์การวัดประสิทธิภาพ 2 วิธีคือ การวัดประสิทธิภาพจากอัตราความสำเร็จในการค้นหาคำตอบ (success rate) และการวัดประสิทธิภาพจากค่าความผิดพลาด ผลการทดลอง พบว่า โมเดลที่นำเสนอมีอัตราความสำเร็จในการค้นหาคำตอบเท่ากับ 100% ซึ่งเป็นอัตราที่สูงที่สุดเมื่อเทียบกับโมเดลอื่นและยังมีค่าความผิดพลาดน้อยที่สุดอีกด้วย ดังนั้น จากผลการทดลองสามารถสรุปได้ว่า โมเดลที่นำเสนอมีประสิทธิภาพและประสิทธิผลมากกว่าโมเดลอื่นๆ ที่นำมาเปรียบเทียบ

จากการศึกษางานวิจัยนี้สามารถวิเคราะห์จุดเด่นที่สำคัญได้ดังนี้ 1) โมเดลที่นำเสนอมีประสิทธิภาพสูงในการค้นหาคำตอบ เนื่องจากมีการใช้โอเปอร์เรเตอร์ครอสโอเวอร์และโอเปอร์เรเตอร์มิวเตชันมากกว่าหนึ่งตัว เช่น โครโมโซมพ่อและแม่เดียวกัน แต่เมื่อเลือกใช้โอเปอร์-

เรเตอร์ครอสโอเวอร์ที่แตกต่างกันย่อมได้โครโมโซมลูกที่แตกต่างกันด้วย ดังนั้นการใช้โอเปอร์เรเตอร์ครอสโอเวอร์และโอเปอร์เรเตอร์มิวเตชันมากกว่าหนึ่งตัวจึงทำให้เกิดความหลากหลายในการสร้างประชากรรุ่นลูกได้มากยิ่งขึ้น และ 2) มีการนำเสนอโอเปอร์เรเตอร์แบบรีแอทเทม เพื่อแก้ปัญหาการตกในพื้นที่โลคอล ซึ่งเป็นจุดด้อยที่สำคัญของจีเนติกอัลกอริทึมที่บางครั้งให้คำตอบที่ดีที่สุดที่เป็นคำตอบแบบโลคอล (local solution) อย่างไรก็ตาม อัลกอริทึมที่นำเสนอได้มีการใช้โอเปอร์เรเตอร์จำนวนมากขึ้นกว่าจีเนติกอัลกอริทึมแบบดั้งเดิมย่อมทำให้สิ้นเปลืองเวลาในการคำนวณเพื่อเลือกใช้โอเปอร์เรเตอร์ที่เหมาะสมในแต่ละรอบการทำงาน

2.3.5 Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation

Ling Wang, Fang Tang และ Hao Wu [16] ได้นำเสนอโมเดล RQGA ซึ่งเกิดจากการผสมผสานระหว่างโมเดล RGA (real-coded genetic algorithm) และโมเดล QGA (quantum genetic algorithm) เข้าด้วยกัน โมเดล RGA เป็นโมเดลที่มีขั้นตอนการทำงานตามหลักการของจีเนติกอัลกอริทึมโดยได้มีการแทนค่าโครโมโซมด้วยค่าจำนวนจริง ในขณะที่โมเดล QGA เป็นโมเดลที่มีขั้นตอนการทำงานตามหลักการของจีเนติกอัลกอริทึมเช่นกัน แต่มีการแทนค่าโครโมโซมแบบคิว-บิต (Q-bit)

การทำงานของโมเดล RGA มีขั้นตอนหลัก ดังนี้

1) กำหนดค่าประชากรเริ่มต้น $P_R(t) = \{ x_1^t, x_2^t, ..., x_N^t \}$ เมื่อ x_i^t คือโครโมโซมลำดับที่ i ของรอบการทำงานที่ t โดยแต่ละยีนต์ในโครโมโซมแทนค่าด้วยจำนวนจริง

2) คำนวณค่าฟิตเนสของแต่ละโครโมโซมโดยการแทนค่าพารามิเตอร์ของปัญหาด้วยค่าของยีนต์ จากนั้นให้ทำการจัดเรียงโครโมโซมตามค่าฟิตเนส เพื่อหาโครโมโซมที่ดีที่สุด

3) ตรวจสอบเงื่อนไขการหยุดทำงาน ถ้าเงื่อนไขเป็นจริงให้ระบุว่าโครโมโซมที่ดีที่สุดเป็นคำตอบที่ดีที่สุดของโมเดลและหยุดการทำงาน แต่ถ้าเงื่อนไขเป็นเท็จให้ทำขั้นตอนถัดไป

4) สร้างประชากรรุ่นลูก $P_R(t+1)$ ด้วยโอเปอร์เรชันการครอสโอเวอร์และมิวเตชัน ตามสมการที่ 2.16 และ 2.17

$$\begin{aligned} c_1 &= \alpha x + (1-\alpha) \times y \\ c_2 &= \alpha y + (1-\alpha) \times x \end{aligned} \tag{2.16}$$

และ

$$c_{ij}' = c_{ij} + \xi_{ij} \tag{2.17}$$

เมื่อ x, y และ c คือโครโมโซมของแม่ พ่อและลูกตามลำดับ

α คืออัตราการเปลี่ยนแปลงค่ายีนต์ ซึ่งมีค่าอยู่ในช่วง [0, 1]

ξ คือตัวแปรสุ่มที่มีการกระจายแบบปกติ

i คือลำดับของโครโมโซมลูก

j คือลำดับของยีนต์ในโครโมโซมลูก

5) คำนวณค่าฟิตเนสของประชากรรุ่นลูกและหาโครโมโซมที่ดีที่สุดในประชากรรุ่นลูก ถ้าโครโมโซมที่ดีที่สุดที่คำนวณได้ดีกว่าโครโมโซมที่ดีที่สุดในรอบปัจจุบัน ให้ทำการปรับค่าโครโมโซมที่ดีที่สุดใหม่

6) คัดเลือกประชากรใหม่ด้วยวิธีการเลือกแบบจัดเรียง (rank-based selection) โดยการเรียงลำดับประชากรทั้งหมดตามค่าฟิตเนส จากนั้นให้คัดเลือกประชากรที่มีค่าฟิตเนสสูงที่สุด N/5 ลำดับแรกมาทำการคัดลอกแล้วเพิ่มเข้าไปในประชากรและทำการลบประชากรที่มีค่าฟิตเนสต่ำที่สุด N/5 ลำดับออก เพื่อรักษาจำนวนประชากรให้คงที่

7) ทำซ้ำขั้นตอนที่ 3-6

การทำงานของโมเดล QGA มีขั้นตอนหลัก ดังนี้

1) กำหนดค่าประชากรเริ่มต้น $P_Q(t) = \{ p_1^t, p_2^t, ..., p_N^t \}$ เมื่อ p_i^t คือโครโมโซมลำดับที่ i ของรอบการทำงานที่ t โดยแต่ละยีนต์ในโครโมโซมแทนค่าด้วยคิวบิต ดังนี้

$$p_j^t = \begin{bmatrix} \alpha_1^t & \alpha_2^t & \cdots & \alpha_m^t \\ \beta_1^t & \beta_2^t & \cdots & \beta_m^t \end{bmatrix} \qquad (2.18)$$

เมื่อ $|\alpha_i|^2 + |\beta_i|^2 = 1$

2) คำนวณค่าฟิตเนสของแต่ละโครโมโซม โดยต้องทำการถอดรหัสโครโมโซมให้เป็นค่าไบนารีสตริง (binary string) ก่อนนำไปแทนค่าพารามิเตอร์ในฟังก์ชันฟิตเนส วิธีการถอดรหัสโครโมโซมว่าแต่ละยีนต์ควรมีค่าเป็น 0 หรือ 1 ทำได้โดยการสุ่มค่า η ซึ่งอยู่ในช่วง [0, 1] ถ้าค่า $|\alpha_i|^2$ มีค่ามากกว่าค่า η ให้กำหนดค่ายีนต์ลำดับที่ i เท่ากับ 1 แต่ถ้าค่า $|\alpha_i|^2$ น้อยกว่าหรือเท่ากับค่า η ให้กำหนดค่ายีนต์ลำดับที่ i เท่ากับ 0 เมื่อได้ถอดรหัสครบทุกยีนต์แล้วจึงทำการแปลงไบนารีสตริงเป็นจำนวนจริงเพื่อนำไปแทนค่าพารามิเตอร์ในฟังก์ชันฟิตเนสเพื่อคำนวณค่าฟิตเนส จากนั้นให้ทำการจัดเรียงโครโมโซมตามค่าฟิตเนสเพื่อหาโครโมโซมที่ดีที่สุด

3) สร้างประชากร $P'_Q(t)$ โดยการใช้โอเปอร์เรเตอร์การหมุน (rotation operator) เพื่อปรับค่าโครโมโซมของประชากร $P_Q(t)$ ดังสมการที่ 2.19

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \times \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \quad (2.19)$$

เมื่อ θ_i คือมุมที่ต้องการหมุน คำนวณได้จาก $\theta_i = s(\alpha_i, \beta_i) \times \Delta\theta_i$ ซึ่งค่า $s(\alpha_i, \beta_i)$ และค่า $\Delta\theta_i$ ดูได้จากตารางที่ 2.2 โดย b และ r แทนโครโมโซมที่ดีที่สุดและโครโมโซมที่กำลังทำโอเปอร์เรชันการหมุนตามลำดับ ค่า b_i และ r_i แทนค่าบิตลำดับที่ i ในไบนารีสตริงของ b และ r ตามลำดับ และ $f(\cdot)$ คือค่าฟิตเนส

ตารางที่ 2.2 การกำหนดค่าพารามิเตอร์ในโอเปอร์เรชันการหมุน [16]

r_i	b_i	$f(r) < f(b)$	$\Delta\theta_i$	$s(\alpha_i, \beta_i)$			
				$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	False	0	0	0	0	0
0	0	True	0	0	0	0	0
0	1	False	0	0	0	0	0
0	1	True	0.05π	-1	+1	±1	0
1	0	False	0.01π	-1	+1	±1	0
1	0	True	0.025π	+1	-1	0	±1
1	1	False	0.005π	+1	-1	0	±1
1	1	True	0.025π	+1	-1	0	±1

4) สร้างประชากรรุ่นลูก $P_Q(t+1)$ จากประชากร $P'_Q(t)$ ด้วยวิธีการครอสโอเวอร์แบบคิวบิต ซึ่งมีลักษณะการทำงานเหมือนกับวิธีการครอสโอเวอร์แบบหนึ่งจุด เมื่อกำหนดให้จุดครอสคือตำแหน่ง i จะได้

โครโมโซมพ่อ

$$\left[\begin{array}{c|c|c|c|c|c|c} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,i} & \alpha_{1,i+1} & \cdots & \alpha_{1,m} \\ \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,i} & \beta_{1,i+1} & \cdots & \beta_{1,m} \end{array}\right]$$

X

โครโมโซมแม่

$$\left[\begin{array}{c|c|c|c|c|c|c} \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,i} & \alpha_{2,i+1} & \cdots & \alpha_{2,m} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,i} & \beta_{2,i+1} & \cdots & \beta_{2,m} \end{array}\right]$$

$\Rightarrow$

โครโมโซมลูก #1

$$\left[\begin{array}{c|c|c|c|c|c|c} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,i} & \alpha_{2,i+1} & \cdots & \alpha_{2,m} \\ \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,i} & \beta_{2,i+1} & \cdots & \beta_{2,m} \end{array}\right]$$

โครโมโซมลูก #2

$$\left[\begin{array}{c|c|c|c|c|c|c} \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,i} & \alpha_{1,i+1} & \cdots & \alpha_{1,m} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,i} & \beta_{1,i+1} & \cdots & \beta_{1,m} \end{array}\right]$$

และวิธีการมิวเตชันแบบคิวบิต โดยทำการสลับค่า α และ β เมื่อกำหนดให้ตำแหน่งที่มิวเตทคือตำแหน่งที่ i จะได้

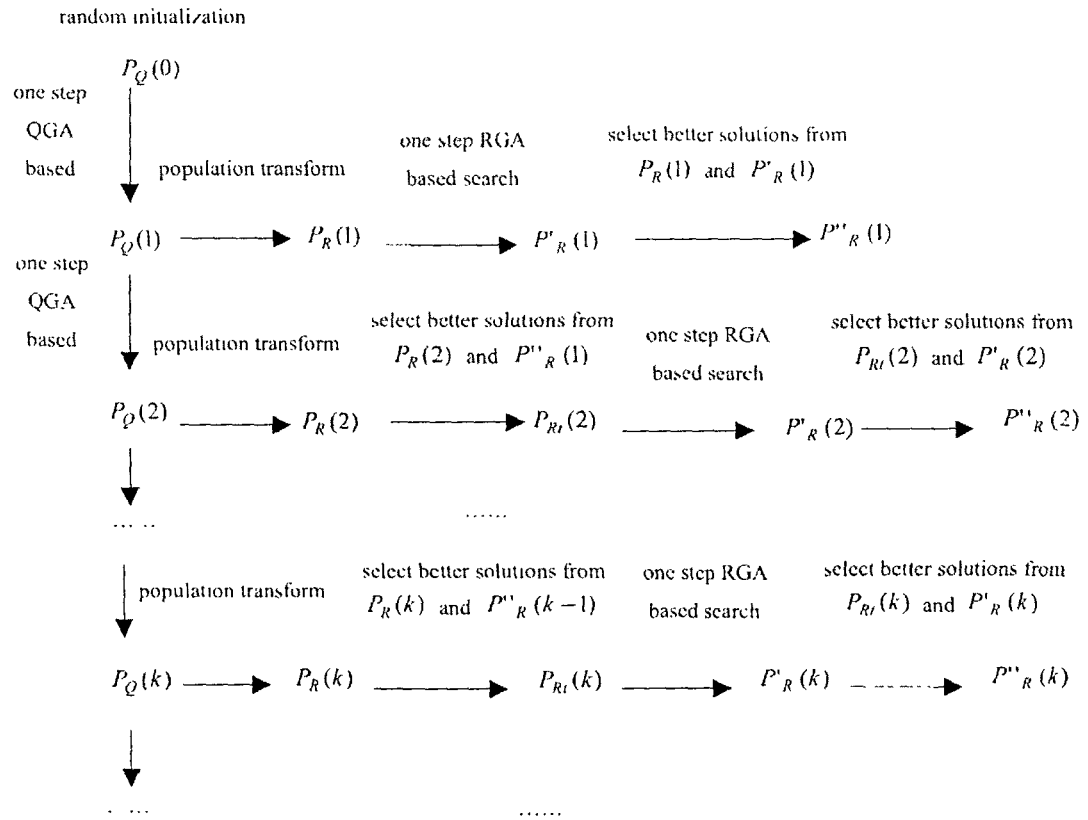
$$\left[\begin{array}{c|c|c|c|c|c} \alpha_1 & \alpha_2 & \cdots & \alpha_i & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_i & \cdots & \beta_m \end{array}\right] \Rightarrow \left[\begin{array}{c|c|c|c|c|c} \alpha_1 & \alpha_2 & \cdots & \beta_i & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \alpha_i & \cdots & \beta_m \end{array}\right]$$

5) คำนวณค่าฟิตเนสของประชากรรุ่นลูกและหาโครโมโซมที่ดีที่สุดในประชากรรุ่นลูก ถ้าโครโมโซมที่ดีที่สุดที่คำนวณได้ดีกว่าโครโมโซมที่ดีที่สุดในประชากรรุ่นพ่อแม่ ให้ทำการปรับค่าโครโมโซมที่ดีที่สุดใหม่

6) คัดเลือกประชากรใหม่ด้วยวิธีการเลือกแบบจัดเรียง (rank-based selection) โดยการเรียงลำดับประชากรทั้งหมดตามค่าฟิตเนส จากนั้นให้คัดเลือกประชากรที่มีค่าฟิตเนสสูงที่สุดเป็น N/5 ลำดับแรกมาทำการคัดลอกแล้วเพิ่มเข้าไปในประชากรและทำการลบประชากรที่มีค่าฟิตเนสต่ำที่สุด N/5 ลำดับออก เพื่อรักษาจำนวนประชากรให้คงที่

7) ทำซ้ำขั้นตอนที่ 3-6

โมเดลที่นำเสนอได้นำหลักการของโมเดล RGA และโมเดล QGA ที่กล่าวมาข้างต้นมาทำงานร่วมกัน โดยมีโครงสร้างการทำงานดังรูปที่ 2.6



รูปที่ 2.6 โครงสร้างการทำงานของโมเดล RQGA [16]

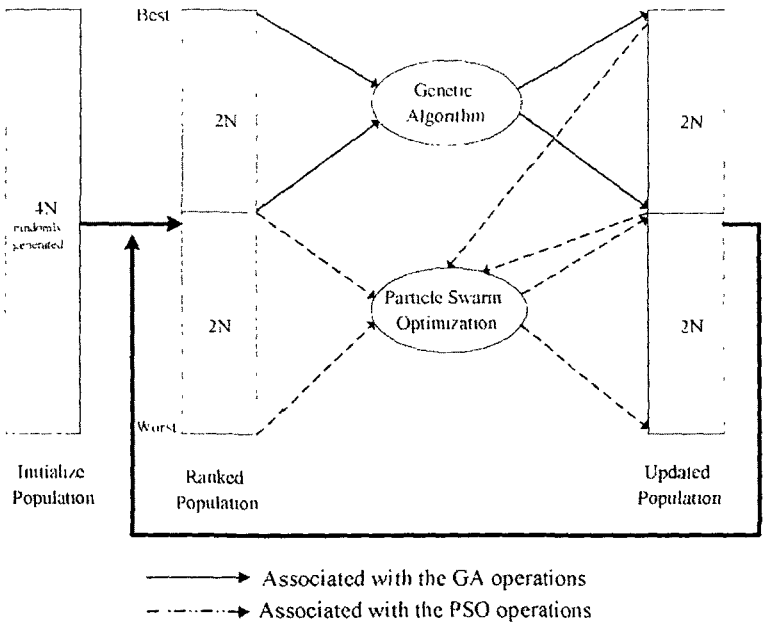
จากรูปที่ 2.6 จะเห็นว่า การทำงานของโมเดลที่นำเสนอเริ่มต้นด้วยการทำงานของโมเดล QGA เมื่อจบการทำงานรอบที่ 1 ของโมเดล QGA จะได้ประชากร $P_Q(1)$ จากนั้นโมเดล RGA จะนำประชากร $P_Q(1)$ มาแปลงเป็นโครโมโซมที่แทนค่าด้วยจำนวนจริงเพื่อให้ได้ประชากรเริ่มต้น $P_R(1)$ ของโมเดล RGA จากนั้นโมเดล RGA จะสร้างประชากรลูก $P'_R(1)$ และทำการคัดเลือกประชากรใหม่ $P''_R(1)$ จากประชากรพ่อแม่ $P_R(1)$ และประชากรลูก $P'_R(1)$ ที่มีค่าฟิตเนสสูง หลังจากนั้นการทำงานจะกลับไปยังโมเดล QGA อีกครั้ง ทั้งนี้ตั้งแต่รอบที่ 2 เป็นต้นไป โมเดล RGA จะนำค่าประชากรที่ดี P''_R ของรอบก่อนหน้ามาคัดเลือกให้เป็นประชากรเริ่มต้นของรอบปัจจุบันด้วย

งานวิจัยนี้ได้วัดประสิทธิภาพของโมเดลที่นำเสนอเปรียบเทียบกับโมเดล QGA โดยแบ่งการทดลองเป็น 2 ส่วนคือ ส่วนแรก เป็นการทดสอบกับปัญหาออฟติไมเซชันแบบไม่มีข้อจำกัดจำนวน 2 ฟังก์ชันซึ่งแต่ละฟังก์ชันมีพารามิเตอร์จำนวน 2 ตัว การทดลองนี้ได้กำหนดจำนวนประชากรให้สัมพันธ์กับจำนวนรอบการทำงาน โดยแบ่งเป็น 5 รูปแบบคือ 30×100, 40×75, 50×60, 60×50 และ 100×30 แต่ละรูปแบบมีการรันจำนวน 50 ครั้ง จากผลการทดลอง พบว่า โมเดล RQGA สามารถค้นหาค่าต่ำสุดของทั้งสองฟังก์ชันได้ถึงแม้ว่าจะมีการกำหนดค่าจำนวนประชากรและจำนวนรอบการทำงานที่แตกต่างกัน ส่วนการทดลองที่สองเป็นการทดสอบการประมาณค่าพารามิเตอร์ของ 2 ฟังก์ชันซึ่งประกอบด้วยฟังก์ชัน Hammerstein และปัญหา state-space ที่มีพารามิเตอร์จำนวน 5 และ 4 ตัวตามลำดับ จากผลการทดลอง พบว่า โมเดล RQGA ให้ค่าความผิดพลาดในการประมาณค่าน้อยกว่าโมเดล QGA ดังนั้น จากผลการทดลองทั้งหมด สรุปได้ว่าโมเดล RQGA มีประสิทธิภาพในการทำงานสูงกว่าโมเดลที่นำมาเปรียบเทียบ

จากการศึกษางานวิจัยนี้สามารถวิเคราะห์จุดเด่นที่สำคัญของโมเดลที่นำเสนอได้ว่า โมเดลที่นำเสนอเกิดจากการผสมผสานกันระหว่างโมเดล RGA และโมเดล QGA ซึ่งโมเดลทั้งสองมีความแตกต่างที่สำคัญในเรื่องของการแทนค่าโครโมโซม โมเดล RGA มีการแทนค่าโครโมโซมด้วยเลขจำนวนจริง ในขณะที่โมเดล QGA มีการแทนค่าโครโมโซมด้วยเลขไบนารี ดังนั้น การแทนค่าโครโมโซมที่แตกต่างนี้ จึงทำให้เกิดการผสมผสานการค้นหาคำตอบทั้งระดับบิตและระดับไบต์ หรือทั้งเชิงลึกและเชิงกว้าง การค้นหาในเชิงกว้างช่วยให้สามารถค้นหาคำตอบในพื้นที่กว้างได้อย่างทั่วถึง ส่วนการค้นหาเชิงลึกช่วยในการค้นหาให้ได้คำตอบในพื้นที่แคบจนได้คำตอบที่ดีที่สุด แต่อย่างไรก็ตาม ความแตกต่างในการแทนค่าโครโมโซมนี้ทำให้สิ้นเปลืองเวลาในการแปลงการแทนค่าสลับไปมาระหว่างทั้งสองโมเดลในทุกรอบการทำงาน

2.3.6 A hybrid genetic algorithm and particle swarm optimization for multimodel functions

Yi-Tung Kao และ Erwie Zahara [17] นำเสนอโมเดล GA-PSO ซึ่งเกิดจากการผสมผสานกันระหว่างจีเนติกอัลกอริทึมและอัลกอริทึมพาร์ทิเคิลสวอร์มเพื่อใช้ในการทดสอบฟังก์ชันออฟติไมเซชันจำนวน 17 ฟังก์ชัน โครงสร้างการทำงานของโมเดลที่นำเสนอ ดังรูปที่ 2.7 จากรูปจะเห็นว่า อัลกอริทึมทั้งสองนี้ใช้ประชากรร่วมกัน โดยกำหนดให้ประชากรเริ่มต้นมีจำนวนเท่ากับ 4N เมื่อ N แทนขนาดของปัญหาที่ใช้ในการทดสอบ จากนั้นให้คำนวณค่าฟิตเนสของประชากรทั้งหมดและทำการจัดเรียงลำดับตามค่าฟิตเนสแล้วจึงทำการแบ่งประชากรเป็น 2 ส่วนขนาดเท่าๆ กัน ประชากรในส่วนแรกกำหนดให้เป็นประชากรเริ่มต้นของจีเนติกอัลกอริทึม ซึ่งประชากรแต่ละตัวเรียกว่า "โครโมโซม" และประชากรในส่วนที่สองกำหนดให้เป็นประชากรเริ่มต้นของอัลกอริทึมพาร์ทิเคิลสวอร์ม ซึ่งประชากรแต่ละตัวเรียกว่า "พาร์ทิเคิล" นอกจากนี้ประชากรใหม่ที่ได้คัดเลือกมาจากประชากรรุ่นพ่อแม่และรุ่นลูกของจีเนติกอัลกอริทึมจะถูกนำมารวมเป็นประชากรเริ่มต้นในรอบการทำงานปัจจุบันของอัลกอริทึมพาร์ทิเคิลสวอร์มด้วย เมื่อจบการทำงานในแต่ละรอบของโมเดล GA-PSO ประชากรใหม่ที่ได้จากอัลกอริทึมทั้งสองจะถูกนำมารวมกันเพื่อทำการจัดเรียงประชากรทั้งหมดตามค่าฟิตเนสก่อนเริ่มการทำงานในรอบถัดไป



รูปที่ 2.7 โครงสร้างการทำงานของโมเดล GA-PSO [17]

ขั้นตอนการทำงานของโมเดล GA-PSO ประกอบด้วยขั้นตอนหลัก ดังนี้

1) สร้างประชากรเริ่มต้นโดยการสุ่มจำนวน 4N เมื่อ N แทนขนาดของปัญหาที่ใช้ทดสอบ

2) คำนวณหาค่าฟิตเนสของประชากรทั้งหมดและทำการจัดเรียงประชากรตามค่าฟิตเนส เพื่อแบ่งประชากรเป็น 2 ส่วนขนาดเท่าๆ กัน

3) การทำงานของจีเนติกอัลกอริทึม

3.1 กำหนดประชากรเริ่มต้นจำนวน 2N ตัวจากประชากรส่วนแรก

3.2 สร้างประชากรรุ่นลูกด้วยโอเปอร์เรชันการครอสโอเวอร์ ตามสมการที่ 2.20 โดยกำหนดค่าความน่าจะเป็นในการครอสโอเวอร์เท่ากับ 1

$$x_i' = \begin{cases} \text{Uniform}(0,1) \times x_i + (1 - \text{Uniform}(0,1)) \times x_{i+1} & i = 1, 2, ..., 2N-1 \\ \text{Uniform}(0,1) \times x_i + (1 - \text{Uniform}(0,1)) \times x_1 & i = 2N \end{cases} \quad (2.20)$$

3.3 ทำการมิวเตทประชากรรุ่นลูกด้วยโอเปอร์เรเตอร์มิวเตชัน ตามสมการที่ 2.21 โดยกำหนดค่าความน่าจะเป็นในการมิวเตชันเท่ากับ 0.2

$$x_k' = x_k + \text{rand} \times N(0, 1) \quad (2.21)$$

3.4 คัดเลือกประชากรใหม่ที่มีค่าฟิตเนสสูงจากประชากรรุ่นพ่อแม่และรุ่นลูก

4) การทำงานของอัลกอริทึมพาร์ทิเคิลสวอร์ม

4.1 คัดเลือกประชากรเริ่มต้นที่มีค่าฟิตเนสสูงจากประชากรส่วนที่สองและประชากรใหม่ที่ได้จากจีเนติกอัลกอริทึมในขั้นตอนที่ 3.4

4.2 ปรับค่าความเร็วในการเคลื่อนที่ของพาร์ทิเคิล ตามสมการที่ 2.22 โดยทำการหาพาร์ทิเคิลที่ดีที่สุด (p_{gd}) ได้จากการจัดเรียงพาร์ทิเคิลทั้งหมดตามค่าฟิตเนส พาร์ทิเคิลที่มีค่าฟิต-เนสสูงที่สุดจะถูกเลือกให้เป็นพาร์ทิเคิลที่ดีที่สุดและทำการหาพาร์ทิเคิลเพื่อนบ้าน (p_{id}) ได้จากการแบ่งประชากรเป็น N คู่ พาร์ทิเคิลที่มีค่าฟิตเนสสูงกว่าจะถูกกำหนดให้เป็นพาร์ทิเคิลเพื่อนบ้านที่ดีที่สุด

$$v_{id}^{new} = w \times v_{id}^{old} + c_1 \times \text{rand} \times (p_{id} - x_{id}^{old}) + c_2 \times \text{rand} \times (p_{gd} - x_{id}^{old}) \quad (2.22)$$

เมื่อ c_1 และ c_2 คือค่าคงที่ค่าบวกค่าหนึ่ง กำหนดให้มีค่าเท่ากับ 2

rand คือค่าถ่วงน้ำหนัก ซึ่งได้จากการสุ่ม

w คือค่าถ่วงน้ำหนักของค่าความเร็ว คำนวณได้จาก w = [0.5 + rand / 2.0]

4.3 ปรับค่าตำแหน่งของพาร์ทิเคิล ตามสมการที่ 2.23 ทั้งนี้ความเร็วในการเคลื่อนที่ต้องไม่เกินค่าความเร็วสูงสุด v_{max} ที่กำหนดไว้

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new} \tag{2.23}$$

5) ตรวจสอบเงื่อนไขการหยุดการทำงานจากค่าเบี่ยงเบนมาตรฐานของค่าฟิตเนสของประชากรที่ดีที่สุด N+1 จำนวน งานวิจัยนี้กำหนดให้ ν มีค่าเท่ากับ $1x10^{-4}$

$$S_f = \left[\sum_{i=1}^{N+1}\left(f(x_i) - \bar{f}\right)^2 / (N+1)\right]^{1/2} < \nu \tag{2.24}$$

ถ้าเงื่อนไขที่ 2.24 เป็นจริงให้หยุดการทำงาน แต่ถ้าเงื่อนไขเป็นเท็จให้ทำซ้ำขั้นตอนที่ 2-4 จนกระทั่งเงื่อนไขเป็นจริง

งานวิจัยนี้ได้ทดสอบประสิทธิภาพของโมเดลที่นำเสนอด้วยฟังก์ชันออฟติไมเซชันจำนวน 17 ฟังก์ชันโดยแต่ละฟังก์ชันมีพารามิเตอร์ตั้งแต่ 2-10 ตัวและได้เปรียบเทียบผลการทดลองกับโมเดลอื่นๆ ที่ปรับมาจากจีเนติกอัลกอริทึม อัลกอริทึมซิมูเลทแอนนิลลิงและอัลกอริทึมการค้นหาแบบตาบู จำนวน 9 โมเดล ผลการทดลอง พบว่า โมเดลที่นำเสนอมีอัตราความสำเร็จในการค้นหาคำตอบของฟังก์ชันทั้งหมดเท่ากับ 100% และจำนวนหนึ่งในสามของฟังก์ชันทั้งหมดที่ใช้ทดสอบให้ค่าความผิดพลาดน้อยที่สุด

จากการศึกษางานวิจัยนี้สามารถวิเคราะห์จุดเด่นและจุดด้อยที่สำคัญได้ว่า จีเนติกอัลกอริทึมเป็นอัลกอริทึมมีประสิทธิภาพในการทำงานเพราะมีโอเปอร์เรเตอร์การเลือก โอเปอร์เรเตอร์การครอสโอเวอร์และโอเปอร์เรเตอร์การมิวเตชัน แต่อุปสรรคหนึ่งของการใช้จีเนติกอัลกอริทึมคือปัญหาการตกในพื้นที่โลคอล แต่โมเดลที่นำเสนอได้พยายามแก้ปัญหานี้โดยการใช้อัลกอริทึมพาร์-ทิเคิลสวอร์ม ซึ่งเป็นอัลกอริทึมที่การจัดเก็บความรู้หรือประสบการณ์ที่ได้จากรอบการทำงานที่ผ่านมาเพื่อนำมาใช้ในการตัดสินใจในรอบปัจจุบัน แต่รูปแบบการผสมผสานของโมเดลที่นำเสนอทำให้อัลกอริทึมพาร์ทิเคิลสวอร์มสูญเสียคุณลักษณะเด่นไป เนื่องจากในแต่ละรอบการทำงานของอัลกอริทึมพาร์ทิเคิลสวอร์มในโมเดลที่นำเสนอไม่มีการจัดเก็บตำแหน่งที่ดีที่สุดไว้ ดังนั้นอัลกอริทึมพาร์ทิเคิลสวอร์มจึงทำหน้าที่เหมือนโอเปอร์เรเตอร์หนึ่งของจีเนติกอัลกอริทึมเท่านั้น แต่อย่างไรก็ตาม การแทนค่าปัญหาของทั้งสองอัลกอริทึมมีความสอดคล้องกัน โครโมโซมของจีเนติกอัลกอริทึมสามารถแปลงเป็นพาร์ทิเคิลของอัลกอริทึมพาร์ทิเคิลสวอร์มได้โดยตรงจึงไม่สิ้นเปลืองเวลาในการแปลงรูปแบบการแทนค่าปัญหา

2.3.7 Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems

W.F. Abd-El-Wahed, A.A. Mousa และ M.A. El-shorbagy [18] นำเสนอโมเดล PSO-GA ที่เกิดจากการผสมผสานกันระหว่างอัลกอริทึมพาร์ทิเคิลสวอร์มและจีเนติกอัลกอริทึม โดยขั้นตอนการทำงานของโมเดลที่นำเสนอ มีรายละเอียดดังนี้

1) กำหนดพาร์ทิเคิลเริ่มต้นโดยการสุ่ม โดยแต่ละพาร์ทิเคิลประกอบด้วยค่าตำแหน่ง $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ และค่าความเร็วในการเคลื่อนที่ $v_i = (v_{i1}, v_{i2}, \ldots, v_{in})$

2) คำนวณค่าฟิตเนสของพาร์ทิเคิลทั้งหมด

3) กำหนดค่า p_i และ p_g เมื่อ p_i แทนตำแหน่งที่ดีที่สุดที่พาร์ทิเคิล i เคยเคลื่อนที่ผ่านมาในรอบก่อนหน้าและ p_g แทนตำแหน่งที่ดีที่สุดที่พาร์ทิเคิลทั้งหมดเคยเคลื่อนที่ผ่านมาในรอบก่อนหน้า

4) ปรับค่าความเร็วในการเคลื่อนที่ของแต่ละพาร์ทิเคิล

$$v_i^{k+1} = w \times v_i^k + c_1 \times r_1 \times (p_i - x_i^k) + c_2 \times r_2 \times (p_g - x_i^k) \tag{2.25}$$

เมื่อ c_1 และ c_2 คือค่าคงที่ค่าบวกค่าหนึ่ง

r_1 และ r_2 คือตัวแปรสุ่มที่มีการกระจายแบบยูนิฟอร์มในช่วง [0, 1]

w คือค่าถ่วงน้ำหนักของความเร็วในการเคลื่อนที่

5) ปรับค่าตำแหน่งของแต่ละพาร์ทิเคิล

$$x_i^{k+1} = x_i^k + \chi v_i^{k+1} \tag{2.26}$$

โดยที่

$$\chi = \frac{2}{\left|-2 - \tau - \sqrt{\tau^2 + \tau}\right|} \tag{2.27}$$

เมื่อ τ คือจำนวนครั้งที่พาร์ทิเคิลเป็นคำตอบที่ไม่มีความเป็นไปได้ (infeasibility solution)

6) คำนวณค่าฟิตเนสใหม่ของพาร์ทิเคิลทั้งหมด

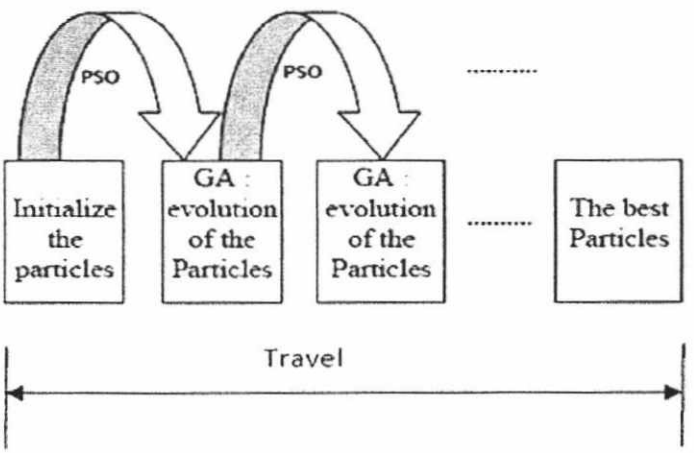
7) ปรับค่า p_i ของแต่ละพาร์ทิเคิลและค่า p_g ของพาร์ทิเคิลทั้งหมด

8) จัดเรียงพาร์ทิเคิลทั้งหมดตามค่าฟิตเนส จากนั้นให้กำหนดพาร์ทิเคิลทั้งหมดให้เป็นประชากรเริ่มต้นของจีเนติกอัลกอริทึม และแต่ละพาร์ทิเคิล เรียกว่า โครโมโซม ตั้งแต่ขั้นตอนนี้เป็นต้นไป

9) เลือกโครโมโซมพ่อแม่เพื่อสร้างโครโมโซมลูก โดยโครโมโซมที่มีค่าฟิตเนสสูงจะได้รับโอกาสในการเลือกให้เป็นโครโมโซมพ่อหรือแม่มากกว่าโครโมโซมที่มีค่าฟิตเนสต่ำกว่า

10) สร้างประชากรรุ่นลูกโดยการครอสโอเวอร์และการมิวเตชัน

11) เลือกลบบางโครโมโซมออกจากประชากรในรอบก่อนหน้าและทำการเพิ่มโครโมโซมลูกที่มีค่าฟิตเนสสูงแทนที่โครโมโซมที่ลบทิ้งออกไป



รูปที่ 2.8 โครงสร้างการทำงานของโมเดล PSO-GA [18]

งานวิจัยนี้ได้แบ่งการทดลองเป็น 2 ส่วนคือ ส่วนแรกเป็นการทดสอบโมเดลที่นำเสนอกับฟังก์ชันออฟติไมเซชันจำนวน 17 ฟังก์ชัน และได้ทำการเปรียบเทียบประสิทธิภาพกับโมเดล GA-PSO [17] ผลการทดลอง พบว่า โมเดลที่นำเสนอให้ค่าความผิดพลาดในการค้นหาคำตอบที่ดีที่สุดน้อยกว่าโมเดลที่นำมาเปรียบเทียบ ส่วนการทดลองที่สองเป็นการทดสอบโมเดลที่นำเสนอกับฟังก์ชันออฟติไมเซชันแบบมีข้อจำกัด และได้เปรียบเทียบประสิทธิภาพกับโมเดล constrained-PSO ผลการทดลอง พบว่า โมเดลทั้งสองมีค่าความผิดพลาดเฉลี่ยใกล้เคียงกันแต่โมเดลที่นำเสนอมีอัตราความสำเร็จในการค้นหาคำตอบเท่ากับ 100% จากผลการทดลอง สรุปได้ว่า โมเดลที่นำเสนอมีประสิทธิภาพในการแก้ปัญหาออฟติไมเซชัน โดยการใช้คุณลักษณะเด่นของอัลกอริทึมพาร์ทิเคิลสวอร์มและจีเนติกอัลกอริทึมร่วมกัน

จากการศึกษางานวิจัยนี้สามารถวิเคราะห์จุดเด่นที่สำคัญได้ว่า 1) โมเดลที่นำเสนอที่เกิดจากการผสมผสานระหว่างจีเนติกอัลกอริทึมและอัลกอริทึมพาร์ทิเคิลสวอร์มยังคงรักษาคุณลักษณะเด่นของอัลกอริทึมทั้งสองไว้ ซึ่งแตกต่างจากงานวิจัยก่อนหน้า [17] คือ อัลกอริทึมพาร์ทิเคิลสวอร์มมีจุดเด่นคือ อัลกอริทึมนี้มีการจัดเก็บความรู้หรือประสบการณ์ที่ได้จากรอบการทำงานที่ผ่านมาเพื่อนำมาใช้ในการตัดสินใจในรอบปัจจุบัน ส่วนจีเนติกอัลกอริทึม ถึงแม้ว่าไม่มีการจัดเก็บความรู้ แต่จีเนติกอัลกอริทึมมีโอเปอร์เรเตอร์การเลือก โอเปอร์เรเตอร์การครอสโอเวอร์และโอเปอร์เรเตอร์การ

มิวเตชันที่มีประสิทธิภาพ ดังนั้นโมเดลที่นำเสนอจึงมีคุณลักษณะเด่นของทั้งสองอัลกอริทึมร่วมกัน และ 2) การแทนค่าปัญหาของทั้งสองอัลกอริทึมมีความสอดคล้องกัน พาร์ทิเคิลในอัลกอริทึมพาร์-ทิเคิลสวอร์มของสามารถแปลงเป็นโครโมโซมในจีเนติกอัลกอริทึมได้โดยตรง

2.3.8 Minimizing the multimodal functions with ant colony optimization approach

M. Duran Toksari [19] ได้นำเสนอโมเดลใหม่ที่เกิดจากการปรับใช้อัลกอริทึมอาณานิคมมดกับปัญหาออฟติไมเซชันเชิงตัวเลข ซึ่งอัลกอริทึมอาณานิคมมดที่นำมาใช้นี้เป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการหาอาหารของมด โดยปกติตามธรรมชาติ มดเป็นสัตว์สังคมที่อยู่ร่วมกันภายในรังและมีการกำหนดหน้าที่ที่แตกต่างกัน มดงานเป็นมดที่ทำหน้าที่ในการหาอาหารในระหว่างที่มดเดินออกจากรังไปหาอาหารหรือเดินกลับมาที่รัง มดแต่ละตัวจะมีการปล่อยสารเคมีชนิดหนึ่งลงบนพื้นเพื่อระบุเส้นทางที่ใช้ ซึ่งสารเคมีดังกล่าวเรียกว่า "ฟีโรโมน" ถือว่ามีประโยชน์อย่างยิ่งในการช่วยตัดสินใจเลือกเส้นทางในการลำเลียงอาหารและถือว่าเป็นกลไกที่สำคัญที่ใช้สื่อสารกันระหว่างมด ซึ่งมดแต่ละตัวจะตัดสินใจเลือกใช้เส้นทางที่มีปริมาณฟีโรโมนสะสมสูงเป็นเส้นทางที่ใช้ในการลำเลียงอาหารและเส้นทางดังกล่าวมักเป็นเส้นทางที่สั้นที่สุดเสมอ

หลักการของอัลกอริทึมอาณานิคมมด มีรายละเอียดดังนี้

- เส้นทางระหว่างรังมดไปจนถึงแหล่งอาหารถูกแทนด้วยกราฟ โดยแต่ละโหนดในกราฟเชื่อมถึงกันด้วยค่าฟีโรโมน
- มดแต่ละตัวตัดสินใจเลือกเส้นทางในการหาอาหารจากระดับปริมาณของฟีโรโมน ดังนั้นเส้นทางที่มีปริมาณฟีโรโมนสูงจะมีโอกาสถูกเลือกใช้ได้มากกว่าเส้นทางที่มีปริมาณฟีโรโมนต่ำกว่า
- เมื่อมดทั้งหมดเดินกลับมาถึงรังเรียบร้อยแล้ว เส้นทางทั้งหมดจะถูกปรับค่าฟีโรโมนตามอัตราส่วนของมดที่ใช้เส้นทางนั้นๆ
- ฟีโรโมนเป็นสารเคมีชนิดหนึ่งจึงได้มีการกำหนดค่าอัตราการระเหยด้วยค่าคงที่ค่าหนึ่ง ดังนั้นเส้นทางใดที่ไม่มีมดเลือกเดินในช่วงเวลาหนึ่งหรือมีมดจำนวนน้อยเลือกเดิน ปริมาณของฟี-โรโมนของเส้นทางนั้นอาจระเหยจนเหลือปริมาณน้อย ในขณะเดียวกันถ้าเส้นทางใดที่มีมดเลือกเดินจำนวนมากจะทำให้เส้นทางนั้นมีปริมาณฟีโรโมนสูง และในที่สุดจะกลายเป็นเส้นทางที่ดึงดูดความสนใจให้มดเลือกเดินมากยิ่งขึ้น
- กระบวนการทำงานจะทำซ้ำตามจำนวนรอบที่กำหนดไว้ เส้นทางที่มีระดับฟีโรโมนสูงที่สุดที่เชื่อมระหว่างรังมดไปยังแหล่งอาหารจะถูกเลือกเป็นเส้นทางที่ดีที่สุด

งานวิจัยนี้ได้ปรับการทำงานของอัลกอริทึมอาณานิคมมดที่กล่าวมาข้างต้นให้สามารถแก้ปัญหาฟังก์ชันออฟติไมเซชันเชิงตัวเลขได้ โดยการแทนค่าตำแหน่งของมดแต่ละตัวด้วยอาร์เรย์จำนวนจริงที่มีความยาวเท่ากับจำนวนพารามิเตอร์ของฟังก์ชันออฟติไมเซชัน สมมติให้ ฟังก์ชัน

ออฟติไมเซชันมีพารามิเตอร์จำนวน 2 ตัว ดังนั้นอาร์เรย์จะประกอบด้วย $(x^k_{initial}, y^k_{initial})$ เมื่อ k = 1, 2, ..., m และ m แทนจำนวนมดที่ใช้ในการทดลอง การกำหนดค่าตำแหน่งเริ่มต้นให้แต่ละอาร์เรย์อาจกำหนดจากการสุ่มค่าหรือกำหนดค่าเริ่มต้นเหมือนกันทั้งหมดก็ได้ ในแต่ละรอบการทำงาน มดแต่ละตัวมีการเคลื่อนที่ไปยังตำแหน่งถัดไป ตามสมการที่ 2.28

$$\begin{aligned} x^k_t &= x^{best}_{t-1} \pm \Delta \quad (t = 1, 2, ..., l) \\ y^k_t &= y^{best}_{t-1} \pm \Delta \quad (t = 1, 2, ..., l) \end{aligned} \tag{2.28}$$

เมื่อ x^k_t, y^k_t คือตำแหน่งของมดตัวที่ k ในรอบการทำงานที่ t

$x^{best}_{t-1}, y^{best}_{t-1}$ คือตำแหน่งที่ดีที่สุดของมด ในรอบการทำงานที่ t-1

Δ คือค่าระยะทางในการเคลื่อนที่ ซึ่งได้จากการสุ่มค่าในช่วง $[-\alpha_t, \alpha_t]$ งานวิจัยนี้ได้มีการปรับค่า $\alpha_{t+1} = 0.1 \times \alpha_t$ เมื่อจบการทำงานในแต่ละรอบการทำงาน

$\pm$ คือทิศทางการเคลื่อนที่ ซึ่งพิจารณาทิศทางการเคลื่อนที่ได้ ตามเงื่อนไข ดังนี้

$$\begin{aligned} \bar{x}^{best}_{initial} &= x^{best}_{initial} + (x^{best}_{initial} \times 0.01) \\ \bar{y}^{best}_{initial} &= y^{best}_{initial} + (y^{best}_{initial} \times 0.01) \end{aligned} \tag{2.29}$$

ถ้า $f(\bar{x}^{best}_{initial}, y^{best}_{initial})$ น้อยกว่า $f(x^{best}_{initial}, y^{best}_{initial})$ ให้กำหนดเครื่องหมายเป็น +

ถ้า $f(\bar{x}^{best}_{initial}, y^{best}_{initial})$ มากกว่าหรือเท่ากับ $f(x^{best}_{initial}, y^{best}_{initial})$ ให้กำหนดเครื่องหมายเป็น –

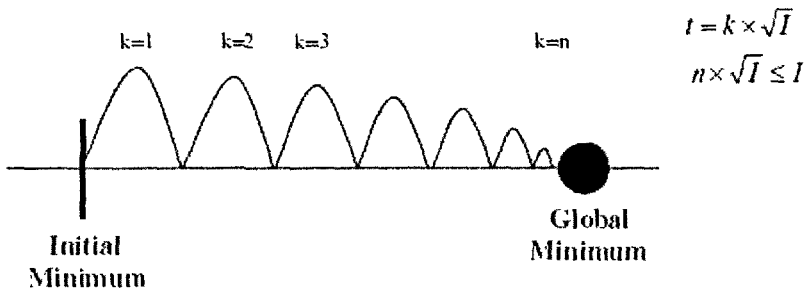
ถ้า $f(x^{best}_{initial}, \bar{y}^{best}_{initial})$ น้อยกว่า $f(x^{best}_{initial}, \bar{y}^{best}_{initial})$ ให้กำหนดเครื่องหมายเป็น +

ถ้า $f(x^{best}_{initial}, \bar{y}^{best}_{initial})$ มากกว่าหรือเท่ากับ $f(x^{best}_{initial}, \bar{y}^{best}_{initial})$ ให้กำหนดเครื่องหมายเป็น –

เมื่อปรับค่าตำแหน่งของมดแต่ละตัวเรียบร้อยแล้ว ขั้นตอนถัดไปคือ การปรับค่าฟีโรโมน

$$\begin{aligned} \tau_t &= 0.1 \times \tau_{t-1} \\ \tau_t &= \tau_{t-1} + (0.01 \times f(x^{best}_{t-1}, y^{best}_{t-1})) \end{aligned} \tag{2.30}$$

กระบวนการทำงานทั้งหมดจะทำซ้ำจนกระทั่งครบจำนวนรอบที่กำหนดไว้ซึ่งการกำหนดจำนวนรอบการทำงานเป็นการกำหนดระยะการค้นหาให้มีค่าลดลงอย่างสม่ำเสมอ โดยกำหนดให้มีค่าเท่ากับ $n \times \sqrt{I} \le I$ โดยที่แต่ละรอบการทำงาน $t = k \times \sqrt{I}$



รูปที่ 2.9 การจำลองการเคลื่อนที่แบบกระโดดของมด [19]

งานวิจัยนี้ได้แบ่งการทดลองเป็น 2 ส่วนคือ การทดลองส่วนแรกเป็นการทดสอบโมเดลที่นำเสนอกับฟังก์ชันออฟติไมเซชันจำนวน 4 ฟังก์ชันคือ ฟังก์ชัน Goldstein and Price ฟังก์ชัน Branin ฟังก์ชัน Hartmann และฟังก์ชัน Shekel ที่มีพารามิเตอร์จำนวน 2, 2, 3 และ 4 ตัวตามลำดับ และได้ทำการเปรียบเทียบกับโมเดลที่นำเสนอโดย Dixon และ Szegö [20] ผลการทดลอง พบว่า โมเดลที่นำเสนอสามารถค้นหาคำตอบที่เหมาะสมที่สุดของฟังก์ชันที่ใช้ทดสอบได้ทั้งหมด ส่วนการทดลองที่สองเป็นการทดสอบกับฟังก์ชัน rosenbrock ซึ่งได้กำหนดพารามิเตอร์จำนวน 2 ค่าคือ 2 และ 4 ตัว และได้ทำการเปรียบเทียบกับอัลกอริทึมซิมเพล็ก อัลกอริทึมซิมูเลทแอนนิลลิง อัลกอริทึมการค้นหาแบบตาบู โมเดล ARS (adaptive random search) และโมเดล VNS (variable neighbourhood search) ผลการทดลอง พบว่า โมเดลที่นำเสนอให้ค่าความผิดพลาดน้อยที่สุดเมื่อเปรียบเทียบกับโมเดลที่นำมาเปรียบเทียบ ดังนั้น จากการทดลองทั้งสองส่วนนี้ สรุปได้ว่า โมเดลที่นำเสนอได้ปรับใช้อัลกอริทึมอาณานิคมมดมาช่วยในการแก้ปัญหาด้านออฟติไมเซชันเชิงตัวเลขได้อย่างมีประสิทธิภาพ

จากการศึกษางานวิจัยนี้สามารถวิเคราะห์จุดเด่นที่สำคัญได้ดังนี้ จากการศึกษางานวิจัยที่เกี่ยวกับอัลกอริทึมอาณานิคมมด พบว่า อัลกอริทึมอาณานิคมมดมีประสิทธิภาพดีในการแก้ปัญหาออฟติไมเซชันเชิงการจัดโดยเฉพาะอย่างยิ่งปัญหาการเดินทางของพนักงานขาย แต่สำหรับการวิจัยนี้ได้พยายามนำอัลกอริทึมอาณานิคมมดมาปรับใช้กับปัญหาออฟติไมเซชันเชิงตัวเลขได้เป็นอย่างดี แต่อย่างไรก็ตาม ถ้ามองถึงกลไกการทำงานของอัลกอริทึมอาณานิคมมดเป็นหลักจะทำให้เห็นสิ่งที่เปลี่ยนแปลงไปคือ เดิมอัลกอริทึมอาณานิคมมดใช้กลไกของปริมาณฟีโรโมนเป็นกลไกในการนำทางเพื่อให้สามารถค้นหาคำตอบที่ดีที่สุดได้ แต่สำหรับโมเดลที่นำเสนอนี้ฟีโรโมนไม่ได้ถูกนำมาใช้เพื่อการค้นหาคำตอบ สามารถที่จะกล่าวได้ว่า โมเดลที่นำเสนอสูญเสียความเป็นอัลกอริทึมที่เกิดจากการเลียนแบบพฤติกรรมการหาอาหารของมดไป

2.3.9 Parallelized genetic ant colony systems for solving the traveling salesman problem

Shyi-Ming Chen และ Chih-Yoa Chien นำเสนออัลกอริทึม PGACS (Parallelized genetic ant colony systems) [3] เพื่อใช้ในการแก้ปัญหาการเดินทางของพนักงานขาย ซึ่งอัลกอริทึมที่นำเสนอนี้เกิดจากการผสมผสานระหว่างจีเนติกอัลกอริทึมและอัลกอริทึมอาณานิคมมด โดยได้ใช้อัลกอริทึมอาณานิคมมดทำหน้าที่สร้างประชากรเริ่มต้นให้กับจีเนติกอัลกอริทึม จากนั้นจีเนติกอัลกอริทึมจะทำหน้าที่ค้นหาคำตอบที่เหมาะสมที่สุดโดยการใช้โอเปอร์เรเตอร์ครอสโอเวอร์จำนวน 2 ตัวคือ การครอสโอเวอร์แบบแกนกระดูก (bone-crossover) และการครอสโอเวอร์แบบสองจุด ทั้งนี้การครอสโอเวอร์แบบแกนกระดูกเป็นโอเปอร์เรเตอร์ที่นำเสนอใหม่ในงานวิจัยนี้ นอกจากนี้ยังได้ใช้โอเปอร์เรเตอร์มิวเตชันจำนวน 2 ตัวคือ การมิวเตชันเส้นทาง (route-mutation) และการมิวเตชันฟีโรโมน (pheromone-mutaion)

ขั้นตอนการทำงานของอัลกอริทึม PGACS มีรายละเอียด ดังนี้

1) ขั้นตอนการกำหนดค่าเริ่มต้น ให้ทำการสุ่มเมืองเริ่มต้นของประชากรมด จากนั้นให้ทำการแบ่งกลุ่มของมดเป็น g กลุ่มโดยแต่ละกลุ่มประกอบด้วยมด N ตัว และให้ทำการกำหนดค่าฟีโรโมนเริ่มต้น (τ_0) ของเส้นทางระหว่างเมืองต่างๆ

2) ขั้นตอนการเลือกเมือง มดตัวที่ k ที่เป็นสมาชิกของกลุ่มที่ i จะเลือกเมืองในลำดับถัดจากเมือง r จากสมการต่อไปนี้ โดยต้องมีการสุ่มค่า q เพื่อนำมาเปรียบเทียบกับค่าคงที่ที่กำหนดไว้

<u>กรณีที่ 1</u> ถ้าค่า q ที่สุ่มได้มีค่าน้อยกว่าค่า q_0 ซึ่งเป็นค่าคงที่ที่กำหนดไว้ในช่วง [0, 1]

$$s = \arg\max_{u \in J_{k,i}(r)} [\tau_i(r,u) \times \eta(r,u)^{\beta}] \tag{2.31}$$

<u>กรณีที่ 2</u> ถ้าค่า q ที่สุ่มได้มีค่ามากกว่าค่า q_0

$$P_{k,i}(r,s) = \begin{cases} \dfrac{\tau_i(r,s) \times \eta(r,s)^{\beta}}{\sum_{u \in J_k(r)} [\tau_i(r,u) \times \eta(r,u)^{\beta}]} & , \text{if } s \in J_{k,g}(r) \\ 0 & , \text{otherwise} \end{cases} \tag{2.32}$$

เมื่อ $P_{k,i}(r, s)$ คือค่าความน่าจะเป็นในการเลือกเมือง s ถัดจากเมือง r ของมดตัวที่ k ในกลุ่มที่ i

$\tau_i(r,s)$ คือปริมาณฟีโรโมนของเส้นทางระหว่างเมือง r และเมือง s ของกลุ่มที่ i

$\eta(r,u)$ คือค่าส่วนกลับของระยะทางระหว่างเมือง r และเมือง u คำนวณได้จาก $1/d_{ru}$

$J_{k,i}(r)$ คือเซตของเมืองที่ถัดจากเมือง r ที่สามารถเลือกได้ของมดตัวที่ k ในกลุ่มที่ i

β คือค่าถ่วงน้ำหนักของค่าระยะทาง

3) ขั้นตอนการปรับค่าฟีโรโมนแบบโลคอล ให้ทำการปรับค่าฟีโรโมนระหว่างสองเมืองใดที่ได้เลือกจากขั้นตอนก่อนหน้า ซึ่งให้ทำการปรับค่าให้ครบทุกกลุ่ม ดังสมการต่อไปนี้

$$\tau_i(r,s) = (1-\rho)\times\tau_i(r,s) + \rho\times\tau_0 \tag{2.33}$$

เมื่อ ρ คือค่าอัตราการระเหย

หลังจากการปรับค่าฟีโรโมนเรียบร้อยแล้ว ถ้าค่า $\tau_i(r,s)$ ที่ได้มีค่าน้อยกว่าค่าฟีโรโมนที่น้อยที่สุดที่ยอมรับได้ (τ_{min}) ให้กำหนดค่าฟีโรโมนใหม่ ดังนี้

$$\tau_i(r,s) = \tau^1_{min} \tag{2.34}$$

4) ขั้นตอนการคำนวณค่าระยะทางรวม หลังจากมดแต่ละตัวได้เลือกเมืองครบทุกเมืองแล้ว ให้ทำการคำนวณค่าระยะทางรวมของมดแต่ละตัวในแต่ละกลุ่ม

5) ขั้นตอนการปรับค่าฟีโรโมนแบบโกลบอล โดยทำการเลือกมดที่มีค่าระยะทางที่สั้นที่สุดของแต่ละกลุ่มเพื่อนำมาใช้ในการปรับค่าฟีโรโมน ดังสมการต่อไปนี้

$$\tau_i(r,s) = (1-\rho)\times\tau_i(r,s) + \rho\times\Delta\tau_i(r,s) \tag{2.35}$$

โดยที่

$$\Delta\tau_i(r,s) = \begin{cases} \dfrac{1}{L_{group\_i\_best}} & ,\text{if } (r.s) \in \text{best route of } i^{th} \text{ group} \\ 0 & ,\text{otherwise} \end{cases} \tag{2.36}$$

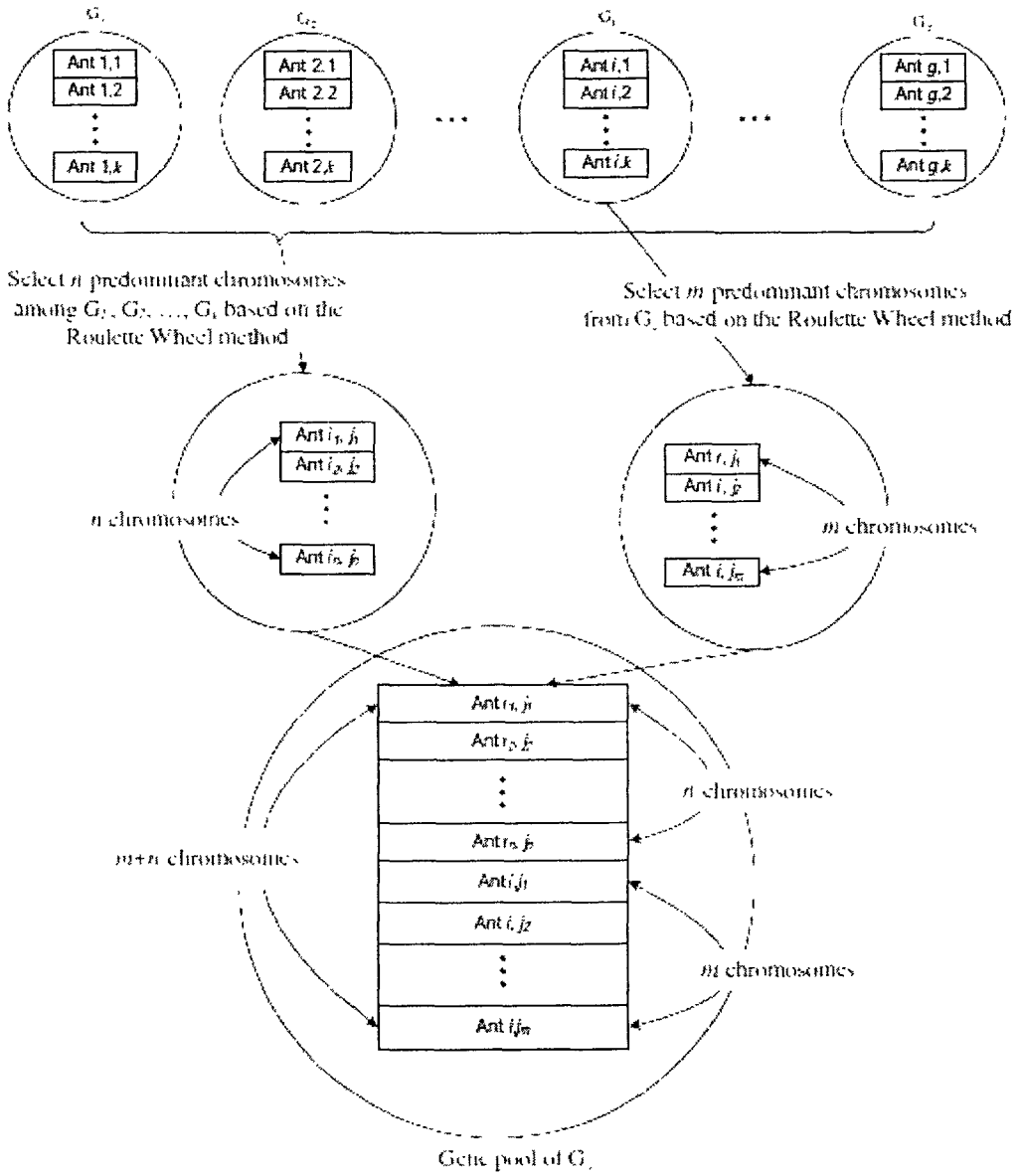
หลังจากการปรับค่าฟีโรโมนเรียบร้อยแล้ว ถ้าค่า $\tau_i(r,s)$ ที่ได้มีค่ามากกว่าค่าฟีโรโมนที่สูงที่สุดที่ยอมรับได้ (τ_{max}) ให้กำหนดค่าฟีโรโมนใหม่ ดังนี้

$$\tau_i(r,s) = \tau^1_{max} \tag{2.37}$$

เมื่อ $L_{group\_i\_best}$ คือค่าระยะทางที่สั้นที่สุดของกลุ่มที่ i

6) ขั้นตอนการคัดเลือก ในขั้นตอนนี้เป็นขั้นตอนการทำงานของจีเนติกอัลกอริทึม มดแต่ละตัวของอัลกอริทึมอาณานิคมจะถูกแทนเป็นโครโมโซมของจีเนติกอัลกอริทึม โดยแต่ละยีนต์แทนด้วยเมือง ขั้นตอนนี้ทำหน้าที่คัดเลือกประชากรที่ดีเพื่อนำไปสร้างประชากรรุ่นลูกต่อไป โดยในแต่

ละกลุ่มที่ i ให้ทำการคัดเลือกมดที่มีค่าระยะทางที่ดีที่สุดจำนวน m ตัวจากกลุ่มของตนเองและทำการคัดเลือกมดที่มีค่าระยะทางที่ดีที่สุดจำนวน n ตัวจากกลุ่มอื่นๆ มารวมกันเพื่อเป็นโครโมโซมมดของแต่ละกลุ่ม ทั้งนี้ให้ทำการคัดเลือกจนครบทุกกลุ่ม ดังรูปที่ 2.10



รูปที่ 2.10 ขั้นตอนการเลือกโครโมโซมมดของอัลกอริทึม PGACS [3]

7) ขั้นตอนการทำครอสโอเวอร์ โดยในแต่ละกลุ่มที่ i ให้ทำการสุ่มเลือกโครโมโซมมดจำนวน 2 ตัว แบ่งเป็นโครโมโซมพ่อและโครโมโซมแม่เพื่อใช้ในการครอสโอเวอร์ โดยต้องมีการสุ่มค่า r เพื่อนำมาเปรียบเทียบกับค่าคงที่ที่กำหนดไว้

<u>กรณีที่ 1</u> ถ้าค่า r ที่สุ่มได้มีค่าน้อยกว่าค่า R_o ซึ่งเป็นค่าคงที่ที่กำหนดไว้ในช่วง [0, 1] ให้เลือกใช้โอเปอร์เรเตอร์ครอสโอเวอร์แบบแกนกระดูก เริ่มต้นโดยการหา “แกนกระดูก” (bone) จาก

เส้นทางที่ซ้ำกันระหว่างโครโมโซมพ่อแม่ จากนั้นให้เลือกแกนกระดูกที่เชื่อมต่อกันยาวที่สุดเป็น "แกนกระดูกหลัก" (base-bone) และเลือกจุดปลายด้านใดด้านหนึ่งให้เป็นเมืองเริ่มต้น หลังจากนั้นให้ทำการเลือกเมืองในลำดับถัดไปจากเส้นทางที่สั้นที่สุดจนกว่าจะครบทุกเมือง ตัวอย่างเช่น โครโมโซมพ่อ (0, 1, 2, 3, 4, 5, 6, 7) และโครโมโซมแม่ (0, 3, 2, 4, 5, 6, 1, 7) ทั้งสองโครโมโซมมีแกนกระดูกคือ (0, 7), (2, 3), (4, 5) และ (5, 6) ดังนั้น แกนกระดูกหลักคือ (4, 5, 6) ให้ทำการเลือกเมืองเริ่มต้น ในที่นี้ สมมติให้เป็นเมือง 6 จากนั้นให้เลือกเมืองในลำดับถัดไปซึ่งจะมี 2 ทางเลือกคือ (6, 7) จากโครโมโซมพ่อและ (6, 1) จากโครโมโซมแม่ ถ้าเส้นทาง (6, 7) มีระยะทางสั้นกว่า (6, 1) เมือง 7 จะถูกเลือกเป็นเมืองถัดไป แต่ถ้าเส้นทาง (6, 1) มีระยะทางสั้นกว่าเมือง 1 จะถูกเลือกเป็นเมืองถัดไป และเมืองที่ถูกเลือกจะนำมารวมเป็นแกนกระดูกหลักต่อไป ให้ทำซ้ำเช่นนี้จนกว่าจะเลือกครบทุกเมือง

<u>กรณีที่ 2</u> ถ้าค่า r ที่สุ่มได้มีค่าน้อยกว่าค่า R_o ให้เลือกใช้โอเปอร์เรเตอร์ครอสโอเวอร์แบบสองจุด โดยการสุ่มเลือกจุดครอสจำนวน 2 จุดเพื่อทำการสลับกลุ่มของยีนต์ระหว่างจุดครอสทั้งสองระหว่างโครโมโซมพ่อและแม่ จากนั้นให้ตรวจสอบการซ้ำกันของยีนต์ของแต่ละโครโมโซม ถ้ามียีนต์ใดที่อยู่นอกขอบเขตของจุดครอสซ้ำกับยีนต์ที่อยู่ในช่วงจุดครอสให้ทำการลบทิ้งและให้เติมเมืองที่ขาดหายไปในยีนต์ดังกล่าวแทน

8) ขั้นตอนการมิวเตชัน โครโมโซมลูกที่ได้จากขั้นตอนก่อนหน้าที่จะถูกนำมามิวเตท ดังนี้

<u>กรณีที่ 1</u> ถ้าค่า i_r ที่สุ่มได้มีค่าน้อยกว่าค่าความน่าจะเป็นในการมิวเตชัน ϕ_r ซึ่งเป็นค่าคงที่ที่กำหนดไว้ในช่วง [0, 1] ให้ทำการมิวเตทโครโมโซมลูกด้วยโอเปอร์เรเตอร์มิวเตชันเส้นทาง โดยการสุ่มเลือกกลุ่มของเมืองจำนวน 2 กลุ่มแล้วทำการสลับตำแหน่งกัน

<u>กรณีที่ 2</u> ถ้าค่า i_p ที่สุ่มได้มีค่าน้อยกว่าค่าความน่าจะเป็นในการมิวเตชัน ϕ_p ซึ่งเป็นค่าคงที่ที่กำหนดไว้ในช่วง [0, 1] ให้ทำการมิวเตทฟีโรโมนด้วยโอเปอร์เรเตอร์มิวเตชันฟีโรโมน เริ่มต้นโดยการสุ่มเลือกเส้นทางระหว่างสองเมืองแล้วทำการสุ่มค่าที่อยู่ในช่วง $[\tau_{min}, \tau_{max}]$ มาแทนที่ค่าฟีโรโมนของเส้นทางที่สุ่มได้

9) ขั้นตอนการคำนวณค่าฟิตเนส ทำหน้าที่คำนวณค่าฟิตเนสของโครโมโซมลูกที่ได้รับการมิวเตทในขั้นตอนก่อนหน้านี้โดยแยกเป็นโครโมโซมลูกของแต่ละกลุ่ม ซึ่งค่าฟิตเนสคำนวณจากค่าระยะทางรวมตามลำดับเมืองในโครโมโซม จากนั้นให้ทำการตรวจสอบการวนซ้ำ ถ้าจำนวนรอบการทำงานของจีเนติกอัลกอริทึมครบตามที่กำหนดไว้ให้ทำขั้นตอนถัดไป แต่ถ้าเงื่อนไขเป็นเท็จให้วนซ้ำขั้นตอนที่ 6 จนกว่าจะครบจำนวนรอบตามที่กำหนดไว้

10) ขั้นตอนการแลกเปลี่ยนข้อมูลระหว่างกลุ่ม ในแต่ละรอบการทำงานจะเลือกวิธีการแลกเปลี่ยนข้อมูลฟีโรโมนระหว่างกลุ่ม โดยมีวิธีในการแลกเปลี่ยนทั้งหมด 7 แบบ ดังนั้น

<u>แบบที่ 1</u> ให้ทำการเลือกกลุ่มที่มีค่าฟิตเนสสูงที่สุด จากนั้นให้นำค่าฟีโรโมนของกลุ่มที่ดีที่สุดไปปรับปรุงค่าฟีโรโมนของกลุ่มอื่นๆ

แบบที่ 2 นำค่าฟีโรโมนของ “กลุ่มเพื่อนบ้าน” (neighbor group) ของกลุ่มที่ i มาปรับปรุงค่าฟีโรโมนของกลุ่มที่ i ซึ่งกลุ่มเพื่อนบ้านของกลุ่มที่ i พิจารณาจากค่าเลขไบนารีในหลักบนสุดที่เหมือนกัน เช่น สมมติว่า กลุ่มที่ i คือ กลุ่มที่ 2 ซึ่งมีเลขไบนารีเป็น 10 จะมีกลุ่มเพื่อนบ้านเป็นกลุ่มที่ 3 ซึ่งมีเลขไบนารีเป็น 11

แบบที่ 3 นำค่าฟีโรโมนของกลุ่มเพื่อนบ้านมาทำการปรับปรุงค่าฟีโรโมนภายในกลุ่มเดียวกันในลักษณะเป็นวงกลม เช่น สมมติว่า กลุ่มที่ 1, 2, 3 และ 4 เป็นกลุ่มเพื่อนบ้านกัน ให้นำค่าฟีโรโมนของกลุ่มที่ 1 ไปปรับปรุงค่าฟีโรโมนของกลุ่มที่ 2 และให้นำค่าฟีโรโมนของกลุ่มที่ 2 ไปปรับปรุงค่าฟีโรโมนของกลุ่มที่ 3 ทำซ้ำวนเป็นวงกลมเช่นนี้จนครบทุกเมือง

แบบที่ 4 นำค่าฟีโรโมนของกลุ่มเพื่อนบ้านของกลุ่มที่ i มาปรับปรุงค่าฟีโรโมนของกลุ่มที่ i ซึ่งกลุ่มเพื่อนบ้านของกลุ่มที่ i พิจารณาจากค่าเลขไบนารีที่เลขต่างกันเพียงหนึ่งบิต เช่น สมมติว่า กลุ่มที่ i คือ กลุ่มที่ 2 ซึ่งมีเลขไบนารีเป็น 10 จะมีกลุ่มเพื่อนบ้าน 2 กลุ่มคือ กลุ่มที่ 1 ซึ่งมีเลขไบนารีเป็น 01 และกลุ่มที่ 3 ซึ่งมีเลขไบนารีเป็น 11

แบบที่ 5 ปรับปรุงฟีโรโมนโดยใช้วิธีการแลกเปลี่ยนค่าฟีโรโมนทั้งแบบที่ 1 และแบบที่ 2

แบบที่ 6 ปรับปรุงฟีโรโมนโดยใช้วิธีการแลกเปลี่ยนค่าฟีโรโมนทั้งแบบที่ 1 และแบบที่ 3

แบบที่ 7 ปรับปรุงฟีโรโมนโดยใช้วิธีการแลกเปลี่ยนค่าฟีโรโมนทั้งแบบที่ 1 และแบบที่ 4

11) ตรวจสอบเงื่อนไขการหยุดการทำงาน ถ้าจำนวนรอบการทำงานครบตามที่กำหนดไว้ให้เลือกโครโมโซมที่มีค่าฟิตเนสดีที่สุด (มีระยะทางสั้นที่สุด) เป็นคำตอบของปัญหา แต่ถ้าเงื่อนไขเป็นเท็จให้วนซ้ำขั้นตอนที่ 2 จนกว่าจะครบจำนวนรอบตามที่กำหนดไว้

งานวิจัยนี้ได้ทำการเปรียบเทียบผลการทดลองกับอัลกอริทึม PACS ที่นำเสนอโดย Chu และคณะ ผลการทดลองพบว่า อัลกอริทึม PGACS ที่นำเสนอในงานวิจัยนี้มีประสิทธิภาพในการค้นหาคำตอบที่ดีกว่าอัลกอริทึมที่เปรียบเทียบด้วย

จากการศึกษางานวิจัยนี้สามารถวิเคราะห์จุดเด่นที่สำคัญได้ดังนี้ อัลกอริทึมที่นำเสนอซึ่งเกิดจากการผสมผสานระหว่างจีเนติกอัลกอริทึมและอัลกอริทึมอาณานิคมมดได้มีการทำงานรวมกันได้อย่างเหมาะสม กล่าวคือ เมื่อมองว่าจีเนติกอัลกอริทึมเป็นโครงสร้างหลัก จะพบว่าอัลกอริทึมอาณานิคมมดจะทำหน้าที่สร้างประชากรเริ่มต้นที่ดีให้กับจีเนติกอัลกอริทึม ซึ่งโดยปกติจีเนติกอัลกอริทึมจะใช้วิธีการสุ่มค่า ดังนั้นจึงส่งผลให้ประสิทธิภาพในค้นหาคำตอบดีขึ้นได้ ในขณะเดียวกัน ถ้ามองว่าอัลกอริทึมอาณานิคมมดเป็นโครงสร้างหลัก จะพบว่าจีเนติกอัลกอริทึมจะทำหน้าที่เป็นโอเปอร์เรเตอร์สำหรับการค้นหาแบบโลคอล ซึ่งส่งผลให้ประสิทธิภาพในค้นหาคำตอบดีขึ้นเช่นกัน นอกจากนี้อัลกอริทึมที่นำเสนอยังมีการแบ่งการทำงานเป็นกลุ่มที่มีการทำงานแบบขนานในการค้นหาคำตอบและมีกลไกในการแลกเปลี่ยนข้อมูลระหว่างกลุ่มอีกด้วย

บทที่ 3
การปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยกลไกการเรียนรู้ด้วยตนเองสำหรับปัญหาออฟติไมเซชัน

การศึกษาในครั้งนี้ได้นำเสนออัลกอริทึมที่เกิดจากการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่นำเสนอโดย Hussein A. Abbass ให้มีกลไกการเรียนรู้ด้วยตนเองเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันให้มีประสิทธิภาพดียิ่งขึ้น ซึ่งจากที่กล่าวไว้ในบทที่ 2 ปัญหาออฟติไมเซชันสามารถแบ่งตามชนิดข้อมูลของพารามิเตอร์ได้เป็น 2 ประเภทคือ ปัญหาออฟติไมเซชันเชิงตัวเลข (numerical optimization problem) และปัญหาออฟติไมเซชันเชิงการจัด (combinatorial optimization problem) ดังนั้น การศึกษาในครั้งนี้จึงได้พัฒนาอัลกอริทึมที่เกิดจากการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมโดยแบ่งเป็น 2 โมเดล คือ

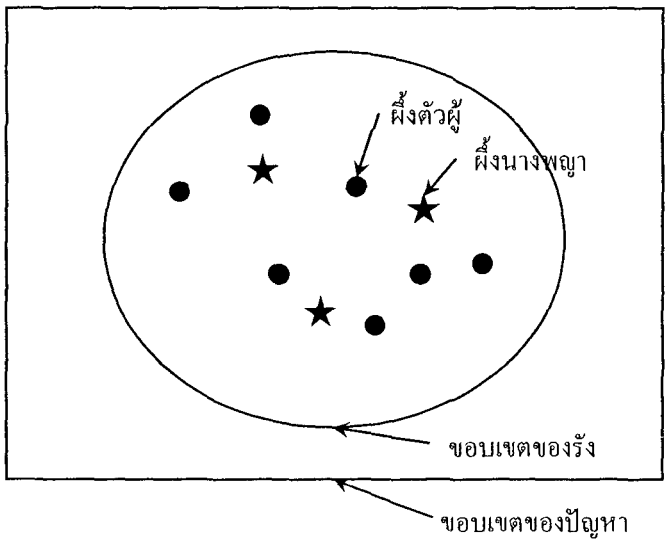
1) อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข โดยได้ทำการทดสอบประสิทธิภาพของอัลกอริทึมที่นำเสนอกับฟังก์ชันมาตรฐานและได้เปรียบเทียบผลการทดลองกับอัลกอริทึมการผสมพันธุ์แบบดั้งเดิม

2) อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด โดยได้ทำการทดสอบประสิทธิภาพของอัลกอริทึมที่นำเสนอกับปัญหาการเดินทางของพนักงานขาย (travelling salesman problem) และได้เปรียบเทียบกับผลการทดลองกับอัลกอริทึม PGACS (parallelized genetic ant colony systems) ที่นำเสนอโดย Chen และ Chien [3]

3.1 การวิเคราะห์ปัญหาของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมถูกพัฒนาขึ้นเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันโดยอัลกอริทึมนี้เกิดจากการเลียนแบบพฤติกรรมการผสมพันธุ์ระหว่างผึ้งนางพญาและผึ้งตัวผู้ ขั้นตอนการทำงานในอัลกอริทึมนี้ได้มีการกำหนดจำนวนของผึ้งนางพญา ผึ้งตัวผู้และตัวอ่อนผึ้งเป็นค่าคงที่ค่าหนึ่งและได้ใช้ค่าฟิตเนสเป็นเครื่องมือในการวัดคุณภาพของคำตอบ นั่นคือ ผึ้งที่มีค่าฟิตเนสสูงจะเป็นคำตอบที่ดีกว่าผึ้งที่มีค่าฟิตเนสที่ต่ำกว่า ดังนั้น การค้นหาผึ้งที่มีค่าฟิตเนสสูงที่สุดจึงเป็นเป้าหมายของการแก้ปัญหาออฟติไมเซชัน อัลกอริทึมนี้แบ่งเป็นขั้นตอนหลัก 3 ขั้นตอน ดังนี้ 1) ขั้นตอนการผสมพันธุ์ 2) ขั้นตอนการสร้างตัวอ่อนผึ้งและการเลี้ยงดู และ 3) ขั้นตอนการเลือกผึ้งนางพญาใหม่ ในขั้นตอนการผสมพันธุ์ ผึ้งนางพญาสามารถเลือกผสมพันธุ์กับผึ้งตัวผู้ได้หลายตัว (multiple mating) แต่ด้วยโครงสร้างของรังที่จำลองขึ้นตามอัลกอริทึมการผสมพันธุ์ของผึ้งแบบ

ดั้งเดิมมีโครงสร้างแบบมีจำนวนผึ้งนางพญาหลายตัวภายในรังเดียวกัน (single-colony multiple-queens model) ดังรูปที่ 3.1 จากโครงสร้างรังลักษณะนี้ทำให้ผึ้งนางพญาทุกตัวเลือกผสมพันธุ์กับผึ้งตัวผู้ภายในรังเดียวกัน หลังจากขั้นตอนการผสมพันธุ์สิ้นสุดลง ผึ้งตัวผู้จะตายทั้งหมด จากนั้นจึงได้เริ่มขั้นตอนการสร้างตัวอ่อนผึ้งโดยการใช้โอเปอร์เรเตอร์ครอสโอเวอร์และโอเปอร์เรเตอร์มิวเตชันเพื่อสร้างตัวอ่อนผึ้งและตัวอ่อนผึ้งที่ได้จะนำมาเข้าสู่ขั้นตอนการเลี้ยงดูโดยการใช้โอเปอร์เรเตอร์ผึ้งงาน โดยขั้นตอนนี้มีการใช้โอเปอร์เรเตอร์ผึ้งงานมากกว่าหนึ่งตัว เพื่อให้มีวิธีที่แตกต่างกันในการปรับปรุงตัวอ่อนผึ้ง เมื่อจบการทำงานในแต่ละรอบการทำงานจะมีการคัดเลือกผึ้งนางพญาใหม่ โดยการเลือกตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงมาแทนที่ผึ้งนางพญาที่มีค่าฟิตเนสที่ต่ำกว่าและทำการสุ่มผึ้งตัวผู้ใหม่ทั้งหมด



รูปที่ 3.1 โครงสร้างรังแบบมีผึ้งนางพญาหลายตัวภายในรังเดียวกันของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

3.1.1 จุดเด่นของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

จากการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมสามารถวิเคราะห์จุดเด่นที่สำคัญได้ ดังนี้

1) ขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมมีความคล้ายคลึงกับจีเนติกอัลกอริทึมซึ่งเป็นอัลกอริทึมที่มีประสิทธิภาพในการแก้ปัญหาออฟติไมเซชัน โดยได้มีการใช้โอเปอร์เรเตอร์การเลือก โอเปอร์เรเตอร์ครอสโอเวอร์และโอเปอร์เรเตอร์มิวเตชันในขั้นตอนการสร้างตัวอ่อนผึ้งเช่นเดียวกับจีเนติกอัลกอริทึม

2) อัลกอริทึมการผสมพันธุ์ของผึ้งมีการใช้โอเปอร์เรเตอร์มากกว่าจีเนติกอัลกอริทึมอยู่หนึ่งโอเปอร์เรเตอร์คือ โอเปอร์เรเตอร์ผึ้งงานซึ่งทำหน้าที่ในการปรับค่าจีโนไทป์ของตัวอ่อนผึ้ง

โดยในอัลกอริทึมนี้มีการใช้โอเปอร์เรเตอร์ผึ้งงานมากกว่าหนึ่งตัว ดังนั้น ตัวอ่อนผึ้งจึงถูกปรับค่าจีโนไทป์ด้วยโอเปอร์เรเตอร์ที่แตกต่างกัน ขั้นตอนการเลี้ยงดูจึงเป็นกลไกหนึ่งที่สามารถช่วยป้องกันการตกในพื้นที่โลคอล (local area) ได้

3) รูปแบบการผสมพันธุ์ระหว่างผึ้งนางพญากับผึ้งตัวผู้เป็นแบบหนึ่งต่อหลาย (one-to-many) กล่าวคือ ผึ้งนางพญาสามารถผสมพันธุ์กับผึ้งตัวผู้ได้มากกว่าหนึ่งตัว ด้วยกลไกนี้ทำให้ตัวอ่อนผึ้งที่ได้จากขั้นตอนการสร้างตัวอ่อนผึ้งมีความหลากหลายเนื่องจากตัวอ่อนผึ้งมีโอกาสสูงที่จะเกิดจากการครอสโอเวอร์ระหว่างผึ้งนางพญาและผึ้งตัวผู้ที่แตกต่างกัน

3.1.2 จุดด้อยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

อย่างไรก็ตาม เมื่อทำการวิเคราะห์หาจุดด้อย พบว่า อัลกอริทึมการผสมพันธุ์ของผึ้งมีจุดที่ควรปรับปรุง ดังนี้

1) โครงสร้างรังของอัลกอริทึมการผสมพันธุ์ของผึ้งเป็นแบบมีจำนวนผึ้งนางพญาหลายตัวภายในรังเดียวกัน ดังรูปที่ 3.1 ถึงแม้ว่าผึ้งนางพญาสามารถเลือกผสมพันธุ์กับผึ้งตัวผู้ได้มากกว่าหนึ่งตัว แต่ผึ้งนางพญาทุกตัวมีการเลือกผสมพันธุ์กับผึ้งตัวผู้ภายในรังตนเองเหมือนกันทั้งหมด จึงเป็นสาเหตุที่ทำให้ผึ้งนางพญาที่มีค่าฟิตเนสใกล้เคียงกันมีโอกาสที่จะเลือกผสมพันธุ์กับเซตของผึ้งตัวผู้ชุดเดียวกันหรือมีความคล้ายคลึงกัน ซึ่งส่งผลให้ตัวอ่อนผึ้งที่สร้างใหม่มีความหลากหลายน้อยลงจึงอาจมีโอกาสตกในพื้นที่โลคอลได้

2) เมื่อจบการทำงานในแต่ละรอบการทำงาน ผึ้งตัวผู้จะถูกฆ่าตายทั้งหมดก่อนการทำงานในรอบถัดไป ดังนั้น จึงต้องมีการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้ใหม่ขึ้นมาแทนที่เสมอ ซึ่งส่งผลทำให้สิ้นเปลืองเวลาในการสร้างจีโนไทป์และการคำนวณค่าฟิตเนสของผึ้งตัวผู้ทุกตัวในทุกรอบการทำงาน

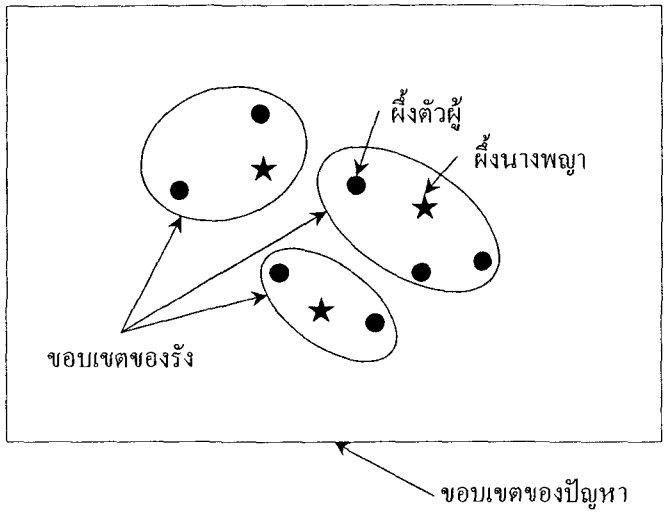
3.2 แนวทางการปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

อัลกอริทึมที่นำเสนอได้ปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมเพื่อแก้ปัญหาจุดด้อยทั้งสองข้อ ดังนี้

1) วิธีการแก้ปัญหาการผสมพันธุ์ระหว่างผึ้งนางพญากับผึ้งตัวผู้ภายในรังเดียวกัน

แนวทางที่ใช้ในการแก้ปัญหานี้ คือ การแบ่งรังเพื่อให้ผึ้งนางพญาได้เลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นๆ ดังนั้นโอกาสที่ผึ้งนางพญาแต่ละตัวจะเลือกผสมพันธุ์กับเซตของผึ้งตัวผู้ชุดเดียวกันหรือมีความคล้ายคลึงจึงมีน้อย จากวิธีการแก้ปัญหานี้อัลกอริทึมที่นำเสนอจึงได้ทำการแบ่งพื้นที่ในการค้นหาคำตอบ (search space) เป็นพื้นที่รังหลายๆ รัง โดยในแต่ละรังประกอบด้วยผึ้งนางพญาหนึ่งตัวเป็นเจ้าของรังและมีผึ้งตัวผู้เป็นสมาชิกของรัง ดังนั้นโครงสร้างรังของอัลกอริทึมที่นำเสนอ

จึงเป็นแบบมีหลายรังและมีผึ้งนางพญาหลายตัว (multiple-colonies multiple-queens model) ดังรูปที่ 3.2



รูปที่ 3.2 โครงสร้างรังแบบมีหลายรังและมีผึ้งนางพญาหลายตัวของอัลกอริทึมที่นำเสนอ

วิธีที่ใช้ในการแบ่งรังของอัลกอริทึมที่นำเสนอนี้ได้ใช้เทคนิคการแบ่งกลุ่มแบบฟัซซีคือ อัลกอริทึมฟัซซีซีมีน (fuzzy c-mean algorithm) โดยกำหนดให้จีโนไทป์ของผึ้งนางพญาเป็นจุดศูนย์กลางของกลุ่ม (centroid) และได้ทำการจัดกลุ่มผึ้งตัวผู้ทั้งหมดให้เป็นสมาชิกในรังใดรังหนึ่งโดยการวัดค่าความเป็นสมาชิก (membership value) จีโนไทป์ของผึ้งตัวผู้ที่มีค่าความเป็นสมาชิกในรังใดมากที่สุดจะถูกจัดให้เป็นสมาชิกของรังนั้นเพียงรังเดียว หลังจากทำการแบ่งรังเรียบร้อยแล้ว ในแต่ละรังจะประกอบด้วยผึ้งนางพญาหนึ่งตัวและผึ้งตัวผู้ ซึ่งมีจำนวนมากหรือน้อยขึ้นอยู่กับความแข็งแรงของผึ้งนางพญาในแต่ละรัง ซึ่งหมายความว่า ถ้าผึ้งนางพญามีค่าฟิตเนสสูงกว่าจะได้รับโอกาสในการสร้างรังได้ใหญ่กว่าผึ้งนางพญาที่มีค่าฟิตเนสที่ต่ำกว่า

อย่างไรก็ตาม ในอัลกอริทึมการผสมพันธุ์แบบดั้งเดิมได้กำหนดจำนวนของผึ้งนางพญาไว้เป็นค่าคงที่ แต่เมื่อนำมาใช้กับอัลกอริทึมที่นำเสนออาจทำให้มีการสร้างรังที่ใกล้กันเกินไป เนื่องจากค่าจีโนไทป์ของผึ้งนางพญาแทนตำแหน่งจุดศูนย์กลางของกลุ่ม ถ้าค่าจีโนไทป์ของผึ้งนางพญามีความเหมือนกันมาก แสดงว่า กลุ่ม (รัง) จะอยู่ใกล้กันไปด้วย ดังนั้น เพื่อป้องกันไม่ให้มีการสร้างรังใกล้กันเกินไป อัลกอริทึมที่นำเสนอจึงได้มีการปรับวิธีการคัดเลือกผึ้งนางพญาใหม่ โดยมีเงื่อนไขในการคัดเลือก ดังนี้ 1) ผึ้งนางพญาที่มีค่าฟิตเนสสูงจะได้รับโอกาสให้สร้างรังก่อนผึ้งนางพญาที่มีค่าฟิตเนสที่ต่ำกว่าเสมอ ดังนั้น ผึ้งนางพญาที่มีค่าฟิตเนสสูงที่สุดจึงได้สร้างรังก่อนเป็นอันดับแรก และ 2) ถ้ารัง (จีโนไทป์) ของผึ้งนางพญาใดมีค่าระยะทางใกล้กับรังที่ถูกสร้างไปก่อนหน้าเกินกว่าที่กำหนดไว้ ผึ้งนางพญาตัวนั้นจะไม่ได้รับเลือกให้เป็นผึ้งนางพญาใหม่ จากหลักการ

การคัดเลือกที่นำเสนอนี้ส่งผลให้ในแต่ละรอบการทำงานมีจำนวนผึ้งนางพญาไม่เท่ากันและไม่มีความจำเป็นต้องกำหนดจำนวนของผึ้งนางพญาไว้ก่อน ดังนั้น สามารถสรุปได้ว่า วิธีการป้องกันไม่ให้มีการสร้างรังใกล้กันจนเกินไปด้วยการปรับวิธีการคัดเลือกผึ้งนางพญาใหม่ ทำให้สร้างจุดเด่นของอัลกอริทึมที่น่าสนใจคือ อัลกอริทึมที่นำเสนอไม่ต้องมีการกำหนดจำนวนของผึ้งนางพญาไว้เป็นค่าคงที่เหมือนกับในอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม นอกจากนี้ ยังหมายความว่าอัลกอริทึมที่นำเสนอมีการแบ่งรังโดยที่ไม่มีการกำหนดจำนวนรังไว้เป็นค่าคงที่ด้วยเช่นกัน

2) วิธีการแก้ปัญหาการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้ใหม่ในทุกรอบการทำงาน

เพื่อลดเวลาในการสร้างจีโนไทป์และการคำนวณค่าฟิตเนสของผึ้งตัวผู้ใหม่ในทุกรอบการทำงาน อัลกอริทึมที่นำเสนอจึงได้มีการกำหนดเงื่อนไขการตายของผึ้งตัวผู้ไว้ 2 กรณีคือ กรณีแรก ผึ้งตัวผู้ตายเมื่อได้รับการผสมพันธุ์กับผึ้งนางพญา และกรณีที่สอง ผึ้งตัวผู้ตายเมื่อมีอายุครบตามที่กำหนดไว้ โดยอัลกอริทึมที่นำเสนอได้มีการกำหนดช่วงชีวิต (lifespan) ของผึ้งตัวผู้ไว้เป็นค่าคงที่ ถ้าผึ้งตัวผู้ไม่ได้รับการผสมพันธุ์กับผึ้งนางพญาจะสามารถดำรงชีวิตต่อไปได้ในรอบการทำงานถัดไปจนกว่าอายุจะครบตามที่กำหนดไว้โดยอายุของผึ้งตัวผู้จะเพิ่มขึ้นครั้งละ 1 ในทุกๆ รอบการทำงาน แต่เมื่อมีการตายของผึ้งตัวผู้เกิดขึ้นจากทั้งสองกรณีข้างต้น อัลกอริทึมที่นำเสนอจะใช้วิธีการสุ่มเลือกผึ้งตัวผู้ใหม่จากตัวอ่อนผึ้งที่ได้ในรอบปัจจุบันมาแทนที่ผึ้งตัวผู้ที่ตายลงเพื่อรักษาจำนวนของผึ้งตัวผู้ให้คงที่ วิธีการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้จากตัวอ่อนผึ้งนี้ช่วยลดเวลาในการประมวลผลเพราะค่าจีโนไทป์และค่าฟิตเนสของตัวอ่อนผึ้งได้ถูกคำนวณไว้ในทุกรอบการทำงานอยู่แล้ว

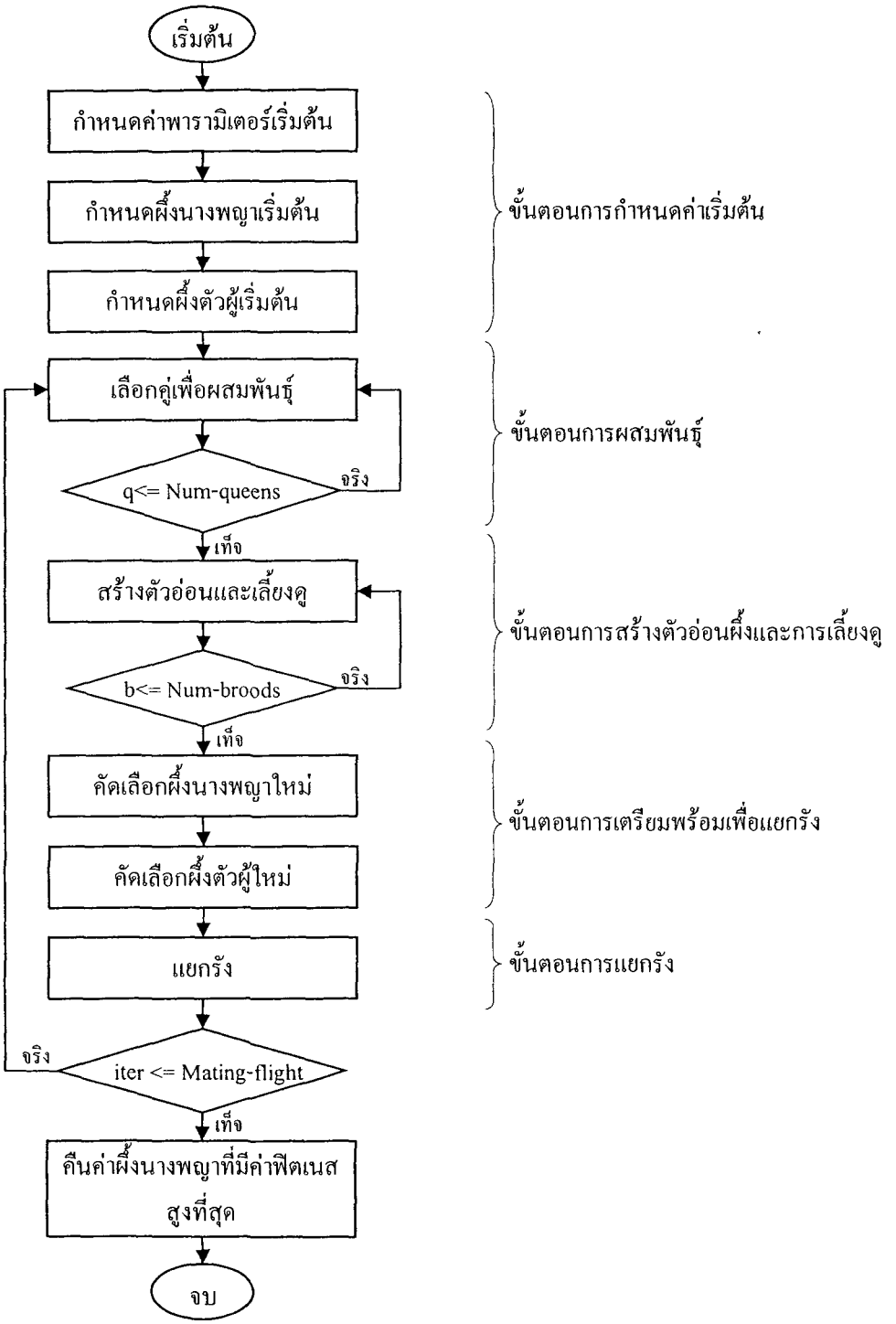
3.3 โครงสร้างการทำงานของอัลกอริทึมที่นำเสนอ

อัลกอริทึมที่นำเสนอแบ่งเป็น 2 โมเดลคือ อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด โดยทั้งสองโมเดลเกิดจากการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมใน 2 แง่มุม คือ 1) ผึ้งนางพญาเลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่น โดยการแบ่งพื้นที่ในการค้นหาเป็นพื้นที่รังหลายๆ รัง และ 2) การคัดเลือกผึ้งตัวผู้ใหม่จากกลุ่มของตัวอ่อนผึ้ง เพื่อแทนที่ผึ้งตัวผู้ที่ตายหลังจากผสมพันธุ์และตายเมื่ออายุครบตามที่กำหนดไว้ ดังนั้น โครงสร้างการทำงานของอัลกอริทึมที่นำเสนอจึงแตกต่างไปจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม ดังรูปที่ 3.3 โดยอัลกอริทึมที่นำเสนอมีขั้นตอนการทำงานเพิ่มขึ้นอีก 2 ขั้นตอนคือ ขั้นตอนการเตรียมพร้อมเพื่อการแยกรังและขั้นตอนการแยกรัง

จากรูปที่ 3.3 โครงสร้างการทำงานของอัลกอริทึมที่นำเสนอประกอบด้วยขั้นตอนหลัก 5 ขั้นตอน คือ

1) ขั้นตอนการกำหนดค่าเริ่มต้น โดยขั้นตอนนี้มีการทำงานหลัก 3 ส่วนคือ การกำหนดค่าพารามิเตอร์เริ่มต้น การกำหนดผึ้งนางพญาเริ่มต้นและการกำหนดผึ้งตัวผู้เริ่มต้น

2) ขั้นตอนการผสมพันธุ์ ขั้นตอนนี้ผึ้งนางพญาแต่ละตัวทำหน้าที่เลือกผสมพันธุ์กับผึ้งตัวผู้ของรังอื่น ยกเว้นกรณีที่มีรังผึ้งเพียงรังเดียว



รูปที่ 3.3 โครงสร้างการทำงานของอัลกอริทึมที่นำเสนอ

3) ขั้นตอนการสร้างตัวอ่อนผึ้งและการเลี้ยงดู โดยแบ่งเป็นขั้นตอนย่อย 2 ขั้นตอน ดังนี้ ขั้นตอนการสร้างตัวอ่อนผึ้ง ซึ่งขั้นตอนนี้มีการใช้โอเปอร์เรเตอร์การเลือก เพื่อสุ่มเลือกผึ้งนางพญาให้ทำหน้าที่วางไข่ ผึ้งนางพญาที่ได้รับเลือกจะใช้โอเปอร์เรเตอร์ครอสโอเวอร์เพื่อทำการครอสโอเวอร์จีโนไทป์ของตนเองกับสเปิร์ม (จีโนไทป์) ของผึ้งตัวผู้ และตัวอ่อนผึ้งที่ได้จะถูกนำไปมิวเตทด้วยโอเปอร์เรเตอร์มิวเตชันต่อไป จากนั้นจะเข้าสู่ขั้นตอนที่สองคือ ขั้นตอนการเลี้ยงดู โดยการใช้โอเปอร์เรเตอร์ผึ้งงานเพื่อทำหน้าที่ปรับค่าจีโนไทป์ของตัวอ่อนผึ้งแต่ละตัว

4) ขั้นตอนการเตรียมพร้อมเพื่อแยกรัง โดยแบ่งเป็นขั้นตอนย่อย 2 ขั้นตอนคือ ขั้นตอนคัดเลือกผึ้งนางพญาใหม่โดยการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังจากผึ้งนางพญาในรอบปัจจุบันและตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าผึ้งนางพญาที่มีค่าฟิตเนสต่ำที่สุด และขั้นตอนการคัดเลือกผึ้งตัวผู้ใหม่โดยการสุ่มเลือกจากตัวอ่อนผึ้งเพื่อแทนที่ผึ้งตัวผู้ที่ตายลง

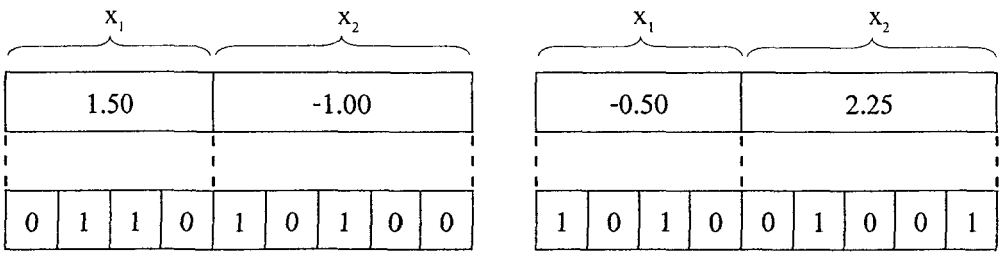
5) ขั้นตอนการแยกรัง ขั้นตอนนี้ทำหน้าที่ในการแบ่งรังผึ้ง ซึ่งแต่ละรังมีผึ้งนางพญาหนึ่งตัวทำหน้าที่เป็นเจ้าของรังและผึ้งตัวผู้ทำหน้าที่เป็นสมาชิกของรัง โดยผึ้งตัวผู้ทั้งหมดจะถูกจัดกลุ่มให้เป็นสมาชิกของรังใดขึ้นอยู่กับค่าความเป็นสมาชิกในแต่ละรัง ผึ้งตัวผู้ที่มีค่าความเป็นสมาชิกในรังใดมากที่สุดจะถูกจัดให้เป็นสมาชิกของรังนั้นๆ

ให้ทำซ้ำขั้นตอนที่ 2-5 จนครบจำนวนรอบที่กำหนดไว้

3.4 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข

3.4.1 การแทนค่าจีโนไทป์ของผึ้ง

ประชากรผึ้งในอัลกอริทึมที่นำเสนอ แบ่งเป็น 3 กลุ่มคือ ผึ้งนางพญา ผึ้งตัวผู้และผึ้งงาน โดยแต่ละกลุ่มมีการแทนค่า (representation) ดังนี้ จีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้แทนค่าด้วยอาร์เรย์ไบนารี แต่ละอิลิเมนต์ของอาร์เรย์แทนค่ายีนต์ซึ่งมีค่าที่เป็นไปได้ 2 ค่าคือ 0 และ 1 ดังรูปที่ 3.4



(ก) จีโนไทป์ของผึ้งนางพญา (ข) จีโนไทป์ของผึ้งตัวผู้

รูปที่ 3.4 การแทนค่าจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้สำหรับปัญหาออฟติไมเซชันเชิงตัวเลข

โดยแต่ละจีโนไทป์ประกอบด้วยเซตของพารามิเตอร์ของปัญหาออฟติไมเซชันที่ใช้ในการทดสอบและพารามิเตอร์แต่ละตัวประกอบด้วยกลุ่มของยีนต์ที่มีจำนวนบิตที่แตกต่างกัน ซึ่งการกำหนดจำนวนบิตที่ต้องใช้ในแต่ละพารามิเตอร์ขึ้นอยู่กับองค์ประกอบ 3 ส่วนคือ บิตที่แทนค่าเครื่องหมาย (sign bit) จำนวน 1 บิต กลุ่มของบิตที่แทนค่าข้อมูลส่วนจำนวนเต็ม (exponent bit) ซึ่งจำนวนบิตที่ใช้ในส่วนนี้ขึ้นอยู่กับขอบเขตของปัญหาและส่วนสุดท้ายคือกลุ่มของบิตที่ใช้แทนค่าข้อมูลส่วนทศนิยม (mantissa bit) ซึ่งจำนวนบิตที่ใช้ในส่วนนี้จะขึ้นอยู่กับความละเอียดของคำตอบที่ต้องการ จากรูปที่ 3.4 จะเห็นว่า จีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้ประกอบด้วยพารามิเตอร์ 2 ตัวคือ (x_1, x_2) โดยสมมติว่า ขอบเขตของพารามิเตอร์ทั้งสองเท่ากับ $x_1 \in [-1, 1]$ และ $x_2 \in [-3, 3]$ (ซึ่งในทางปฏิบัติ ขอบเขตของพารามิเตอร์ปัญหาได้มีการกำหนดไว้ในฟังก์ชันมาตรฐานที่ใช้ในการทดสอบ) และได้มีการกำหนดบิตที่ใช้แทนทศนิยมจำนวน 2 บิต ดังนั้น พารามิเตอร์ x_1 จึงใช้จำนวนบิตทั้งหมด 4 บิต ประกอบด้วย 1) บิตแรกแทนค่าเครื่องหมาย ถ้าบิตเครื่องหมายมีค่าเท่ากับ 0 แสดงว่าแทนเครื่องหมาย + และถ้าบิตเครื่องหมายมีค่าเท่ากับ 1 แสดงว่าแทนเครื่องหมาย –, 2) บิตที่สองแทนค่าจำนวนเต็ม ซึ่งได้ใช้เพียง 1 บิตเนื่องจากขอบเขตของพารามิเตอร์อยู่ในช่วง [-1, 1] และบิตที่เหลือจำนวน 2 บิตใช้แทนจำนวนทศนิยม สำหรับพารามิเตอร์ x_2 ได้แทนค่าด้วยบิตจำนวน 5 บิต ประกอบด้วย 1) บิตแรกแทนค่าเครื่องหมาย, 2) บิตที่สองและสามใช้แทนค่าจำนวนเต็ม ซึ่งได้เลือกใช้จำนวน 2 บิตเพราะขอบเขตของพารามิเตอร์อยู่ในช่วง [-3, 3] และบิตที่เหลือจำนวน 2 บิตใช้แทนจำนวนทศนิยม

สำหรับผึ้งงานได้มีการแทนค่าด้วยฟังก์ชันฮิวริสติก ซึ่งทำหน้าที่ในการค้นหาแบบโลคอล ฟังก์ชันฮิวริสติกที่นำมาใช้ในการศึกษาในครั้งนี้มีจำนวน 5 ฟังก์ชันคือ ฟังก์ชันการกลับบิตแบบสุ่ม (random flip) ฟังก์ชันการการสุ่มเดิน (random walk) ฟังก์ชันการสุ่มค่าใหม่ (random new) ฟังก์ชันการครอสโอเวอร์แบบหนึ่งจุด (one-point crossover) และฟังก์ชันการครอสโอเวอร์แบบสองจุด (two-point crossover) รายละเอียดจะกล่าวในลำดับถัดไป

3.4.2 การคำนวณค่าฟิตเนส

การคำนวณค่าฟิตเนสของผึ้งนางพญา ผึ้งตัวผู้และตัวอ่อนผึ้งแตกต่างกันตามลักษณะของปัญหาที่ใช้ในการทดสอบ อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขได้ทำการทดสอบกับปัญหาออฟติไมเซชันที่อยู่ในลักษณะของฟังก์ชันมาตรฐานทั้งที่เป็นฟังก์ชันค่าต่ำสุด (minimization problem) และฟังก์ชันค่าสูงสุด (maximization problem) ดังนั้น ค่าฟิตเนสของประชากรผึ้งทั้งหมด จึงแบ่งเป็น 2 กรณีคือ การหาค่าฟิตเนส กรณีที่ทดสอบกับฟังก์ชันมาตรฐานที่เป็นฟังก์ชันค่าต่ำสุด สามารถคำนวณได้จากส่วนกลับค่าเอาท์พุทของฟังก์ชัน ตัวอย่างเช่น

$$\text{minimize } f(\bar{x}) = x_1^2 + x_2^2 \qquad (3.1)$$

เมื่อ $\bar{x}$ คือเซตของพารามิเตอร์ที่มีสมาชิกจำนวน 2 ตัวคือ x_1 และ x_2

จากรูปที่ 3.4(ก) ผึ้งนางพญา Q = {1.5, -1} สามารถคำนวณเป็นค่าเอาท์พุทของฟังก์ชันได้จาก $f(\bar{x}) = (1.5)^2 + (-1)^2 = 3.25$ จากนั้นคำนวณค่าฟิตเนสของผึ้งนางพญาได้จาก fitness(Q) = $-f(\bar{x}) = -3.25$ และรูปที่ 3.4(ข) ผึ้งตัวผู้ D = {-0.5, 2.25} คำนวณค่าเอาท์พุทของฟังก์ชันได้จาก $f(\bar{x}) = (-0.5)^2 + (2.25)^2 = 5.3125$ จากนั้นคำนวณค่าฟิตเนสของผึ้งตัวผู้ ได้จาก fitness(D) = $-f(\bar{x})$ = -5.3125 จากวิธีการคำนวณข้างต้น จะเห็นว่า จีโนไทป์ของผึ้งนางพญาเป็นคำตอบที่ดีกว่าจีโนไทป์ของผึ้งตัวผู้ เพราะได้ค่าเอาท์พุทที่มีค่าน้อยกว่า แต่ผึ้งนางพญาจะมีค่าฟิตเนสที่สูงกว่า จึงสรุปได้ว่า ผึ้งที่มีค่าฟิตเนสสูงกว่าคือ ผึ้งที่แข็งแรงดีกว่าและย่อมเป็นคำตอบของปัญหาได้เหมาะสมกว่าด้วย ส่วนกรณีที่สอง การหาค่าฟิตเนส กรณีที่ทดสอบกับฟังก์ชันมาตรฐานที่เป็นฟังก์ชันค่าสูงสุดจะมีลักษณะการคำนวณคล้ายกัน แต่สามารถคำนวณค่าฟิตเนสได้โดยตรงจากค่าเอาท์พุท นั่นคือ fitness(Q) = $f(\bar{x})$

3.4.3 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข

อัลกอริทึมที่นำเสนอ แบ่งเป็นขั้นตอนหลักได้ 5 ขั้นตอน ดังนี้

1) ขั้นตอนการกำหนดค่าเริ่มต้น (initialization process)

การกำหนดค่าเริ่มต้นได้แบ่งเป็น 2 ส่วนคือ การกำหนดค่าพารามิเตอร์เริ่มต้นและการกำหนดรังผึ้งเริ่มต้น ในส่วนของการกำหนดค่าพารามิเตอร์เริ่มต้น มีพารามิเตอร์ที่กำหนดค่า ดังนี้ ขนาดของถุงเก็บสเปิร์มของผึ้งนางพญา จำนวนของผึ้งตัวผู้ จำนวนของตัวอ่อนผึ้ง ค่าความน่าจะเป็นในการมิวเตชันและจำนวนรอบการทำงาน และในส่วนของการกำหนดรังผึ้งเริ่มต้นได้มีการสุ่มสมาชิกภายในรังซึ่งประกอบด้วยผึ้งนางพญาจำนวน 1 ตัวและผึ้งตัวผู้ตามจำนวนที่กำหนดไว้ โดยค่าที่สุ่มกำหนดให้อยู่ในช่วงขอบเขตของปัญหา จากนั้นให้คำนวณค่าฟิตเนสของสมาชิกภายในรังทั้งหมดและทำการจัดเรียงลำดับสมาชิกตามค่าฟิตเนสที่คำนวณได้ สมาชิกที่มีค่าฟิตเนสสูงที่สุดจะถูกเลือกให้เป็นผึ้งนางพญาเริ่มต้นและสมาชิกส่วนที่เหลือกำหนดให้เป็นผึ้งตัวผู้เริ่มต้น

2) ขั้นตอนการผสมพันธุ์ (mating process)

กระบวนการทำงานในขั้นตอนนี้มีการทำงานเหมือนกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม กล่าวคือ ผึ้งนางพญาทุกตัวมีหน้าที่ในการผสมพันธุ์และสามารถเลือกผสมพันธุ์กับผึ้งตัวผู้ได้หลายตัว ผึ้งนางพญาแต่ละตัวจะเลือกผสมพันธุ์กับผึ้งตัวผู้จำนวนมากหรือน้อยขึ้นอยู่กับค่าพลังงานเริ่มต้นและขนาดของถุงเก็บสเปิร์ม เริ่มต้นขั้นตอนการผสมพันธุ์ด้วยการกำหนดค่าเริ่มต้นของค่าพลังงาน (energy) และค่าความเร็วในการบิน (speed) ให้กับผึ้งนางพญาแต่ละตัวโดยการสุ่มค่าอยู่ในช่วง [0.5, 1] จากนั้นผึ้งนางพญาตัดสินใจเลือกผสมพันธุ์กับผึ้งตัวผู้โดยการพิจารณาจากค่าความน่าจะเป็นในการผสมพันธุ์ (marriage probability) ดังสมการที่ 3.2

$$prob(Q,D) = e^{\frac{-difference}{speed(t)}} \quad (3.2)$$

เมื่อ prob(Q,D) คือค่าความน่าจะเป็นในการผสมพันธุ์ระหว่างผึ้งนางพญา Q และผึ้งตัวผู้ D

difference คือค่าผลต่างสมบูรณ์ระหว่างค่าฟิตเนสของผึ้งนางพญา Q และผึ้งตัวผู้ D

speed(t) คือค่าความเร็วในการบินของผึ้งนางพญา Q ณ รอบการทำงานที่ t

ผึ้งนางพญาตัดสินใจเลือกผสมพันธุ์กับผึ้งตัวผู้ที่มีค่าความน่าจะเป็นในการผสมพันธุ์สูงที่สุดและสเปิร์ม (จีโนไทป์) ของผึ้งตัวผู้ที่ได้รับเลือกจะถูกนำมาเก็บไว้ในถุงเก็บสเปิร์มของผึ้งนางพญาเพื่อใช้ในการปฏิสนธิกับไข่ จากนั้นก่อนที่ผึ้งนางพญาจะเริ่มการเลือกคู่ครั้งถัดไป ค่าพลังงานและค่าความเร็วในการบินของผึ้งนางพญาจะถูกปรับลดลง ดังนี้

$$speed(t+1) = \alpha \times speed(t) \quad (3.3)$$

$$energy(t+1) = energy(t) - step \quad (3.4)$$

เมื่อ α คืออัตราการลดลงของค่าความเร็วในการบิน ซึ่งกำหนดให้มีค่าอยู่ในช่วง [0, 1] การศึกษาในครั้งนี้ได้กำหนดค่าให้เท่ากับ 0.95 ตามที่งานวิจัย [1] นำเสนอไว้

step คือปริมาณการลดลงของค่าพลังงาน คำนวณได้จาก

$step = (0.5 \times energy) / spermatheca\text{-}size$

ขั้นตอนการผสมพันธุ์ของผึ้งนางพญาแต่ละตัวสิ้นสุดลงเมื่อถุงเก็บสเปิร์มเต็มหรือค่าพลังงานน้อยกว่าค่าที่กำหนดไว้ซึ่งการศึกษาในครั้งนี้กำหนดค่าให้เท่ากับศูนย์

3) ขั้นตอนการสร้างตัวอ่อนผึ้งและการเลี้ยงดู (breeding and feeding process)

ขั้นตอนนี้ประกอบด้วยขั้นตอนการวางไข่เพื่อสร้างตัวอ่อนผึ้งซึ่งทำหน้าที่โดยผึ้งนางพญาและขั้นตอนการเลี้ยงดูตัวอ่อนผึ้งซึ่งทำหน้าที่โดยผึ้งงาน การทำงานเริ่มต้นด้วยการเลือกผึ้งนางพญาเพื่อทำหน้าที่วางไข่ โดยผึ้งนางพญาถูกสุ่มเลือกด้วยวิธีการเลือกแบบวงล้อหมุน (roullete wheel selection) ผึ้งนางพญาที่มีค่าฟิตเนสสูงจะได้รับโอกาสให้ทำหน้าที่วางไข่สูงกว่าผึ้งนางพญาที่มีค่าฟิตเนสต่ำกว่า จากนั้นผึ้งนางพญาที่ได้รับเลือกจะทำการสุ่มเลือกสเปิร์มจากถุงเก็บสเปิร์มของตนเองเพื่อทำการปฏิสนธิกับไข่ ซึ่งกระบวนการปฏิสนธิคือการครอสโอเวอร์ระหว่างจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้ การศึกษาในครั้งนี้ได้ใช้โอเปอร์เรเตอร์ครอสโอเวอร์แบบสองจุด จากนั้นตัวอ่อนผึ้งที่ได้จะถูกนำไปมิวเตท (mutate) ด้วยโอเปอร์เรเตอร์มิวเตชันโดยการสุ่มบางยีนต์ในจีโน-

ไทป์เพื่อทำการกลับบิตจาก 0 เป็น 1 หรือจาก 1 เป็น 0 กระบวนการสร้างตัวอ่อนผึ้งนี้สิ้นสุดลงเมื่อจำนวนของตัวอ่อนผึ้งครบตามที่กำหนดไว้

หลังจากนั้นจะเข้าสู่ขั้นตอนการเลี้ยงดู โดยมีผึ้งงานซึ่งแทนด้วยฟังก์ชันฮิวริสติกทำหน้าที่ดูแลตัวอ่อนผึ้ง การศึกษาในครั้งนี้ได้ใช้ผึ้งงานจำนวน 5 ตัวซึ่งประกอบด้วยฟังก์ชันการกลับบิตแบบสุ่ม ฟังก์ชันการการสุ่มเดิน ฟังก์ชันการสุ่มค่าใหม่ ฟังก์ชันการครอสโอเวอร์แบบหนึ่งจุดและฟังก์ชันการครอสโอเวอร์แบบสองจุด ตัวอ่อนผึ้งแต่ละตัวจะได้รับการเลี้ยงดูจากผึ้งงานที่แตกต่างกันโดยการสุ่มเลือกด้วยวิธีการเลือกแบบวงล้อหมุนโดยการพิจารณาจากค่าฟิตเนสของผึ้งงาน ซึ่งค่าฟิตเนสของผึ้งงานคำนวณได้จากความสามารถในการปรับค่าฟิตเนสของตัวอ่อนผึ้ง

4) ขั้นตอนการเตรียมพร้อมเพื่อแยกรัง (swarm preparation process)

ขั้นตอนนี้เป็นขั้นตอนที่นำเสนอขึ้นใหม่ เป็นขั้นตอนของการเตรียมความพร้อมก่อนการแบ่งรังซึ่งประกอบด้วย 2 ขั้นตอนย่อยคือ ขั้นตอนการคัดเลือกผึ้งนางพญาใหม่และขั้นตอนการคัดเลือกผึ้งตัวผู้ใหม่เพื่อใช้เป็นประชากรเริ่มต้นของการทำงานในรอบถัดไป กระบวนการของการคัดเลือกผึ้งนางพญาใหม่เป็นการแข่งขันกันระหว่างผึ้งนางพญาและตัวอ่อนผึ้งโดยการพิจารณาจากค่าฟิตเนส ตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าผึ้งนางพญาที่มีค่าฟิตเนสต่ำที่สุดจะถูกเลือกให้เป็นผึ้งนางพญาคู่แข่ง (candidate queen) เพื่อเข้าสู่กระบวนการคัดเลือกต่อไป (ผึ้งนางพญาคู่แข่งหมายรวมถึงผึ้งนางพญาในรอบปัจจุบันด้วย) ทั้งนี้เพื่อป้องกันมิให้ผึ้งนางพญาใหม่มีจำนวนมากเกินไปและป้องกันไม่ให้ผึ้งนางพญาสร้างรังใกล้กันมากจนเกินไป การคัดเลือกผึ้งนางพญาใหม่จึงได้มีการพิจารณาระยะทางระหว่างแต่ละรังด้วย โดยกำหนดให้ผึ้งนางพญาแทนจุดศูนย์กลางของตำแหน่งรังและมีกฎเกณฑ์ว่า ผึ้งนางพญาคู่แข่งที่มีความแข็งแรงมากกว่าจะได้รับโอกาสให้สร้างรังได้ก่อนผึ้งนางพญาคู่แข่งที่มีความแข็งแรงน้อยกว่า ดังนั้น ก่อนการเริ่มต้นคัดเลือกผึ้งนางพญาใหม่จึงต้องทำการจัดเรียงผึ้งนางพญาคู่แข่งตามค่าฟิตเนส ผึ้งนางพญาคู่แข่งที่มีค่าฟิตเนสสูงที่สุด แสดงว่ามีความแข็งแรงมากที่สุดจึงได้รับเลือกให้เป็นผึ้งนางพญาใหม่ตัวแรกเสมอ จากนั้นจึงเริ่มการพิจารณาผึ้งนางพญาคู่แข่งลำดับที่สองโดยการคำนวณระยะทางระหว่างผึ้งนางพญาคู่แข่งลำดับที่สองกับผึ้งนางพญาใหม่ด้วยวิธีการวัดระยะทางแบบยูคลิเดียน (Euclidean distance) ระยะทางที่น้อยที่สุดระหว่างผึ้งนางพญาคู่แข่งลำดับที่สองกับผึ้งนางพญาใหม่ที่ถูกเลือกก่อนหน้า ซึ่งคำนวณได้จากสมการที่ 3.5 จะถูกนำมาพิจารณาเป็นเงื่อนไขการเลือก ดังนี้ ถ้าผึ้งนางพญาคู่แข่งในลำดับที่สองมีค่าระยะทางที่น้อยที่สุดที่ห่างจากผึ้งนางพญาใหม่ที่ถูกเลือกก่อนหน้ามากกว่าค่าที่กำหนดไว้ ผึ้งนางพญาลำดับที่สองจะถูกเลือกให้เป็นผึ้งนางพญาใหม่อีกตัว แต่ถ้ามีค่าระยะทางที่น้อยที่สุดน้อยกว่าค่าที่กำหนดไว้ ตามสมการที่ 3.6 แสดงว่าผึ้งนางพญาคู่แข่งลำดับที่สองควรเป็นสมาชิกภายในรังของผึ้งนางพญาใหม่ ดังนั้น จึงให้ทำการลบผึ้งนางพญาคู่แข่งลำดับที่สองออกจากลิสต์ อย่างไรก็ตาม ถ้าผึ้งนางพญาคู่แข่งลำดับที่สองเป็นตัวอ่อนผึ้งให้นำรวมอยู่ในกลุ่มของตัวอ่อนผึ้งคงเดิม แต่ถ้า

ผึ้งนางพญาคู่แข่งลำดับที่สองเคยเป็นผึ้งนางพญามาก่อนแล้วให้ทำการคัดทิ้งได้ หลังจากนั้นกระบวนการคัดเลือกจะดำเนินการพิจารณาผึ้งนางพญาคู่แข่งในลำดับถัดไปจนกว่าจะครบทุกตัว

$$N = \arg\min_{1 \le n \le S} d(C, q_n) \tag{3.5}$$

$$d(C, q_N) \le \xi \tag{3.6}$$

โดยที่

$$\xi = (5.82574 \times 10^4)\,\text{Area} + 0.58878 \tag{3.7}$$

และ

$$\text{Area} = \prod_{i=1}^{m} (x_i^{max} - x_i^{min}) \tag{3.8}$$

เมื่อ $d(C, q_n)$ คือระยะทางแบบยูคลิเดียนระหว่างผึ้งนางพญาคู่แข่ง C กับผึ้งนางพญาใหม่ตัวที่ n

N คือลำดับของผึ้งนางพญาใหม่ที่อยู่ใกล้ผึ้งนางพญาคู่แข่งมากที่สุด

S คือจำนวนของผึ้งนางพญาใหม่ที่ถูกเลือกไว้ก่อนหน้าแล้ว

ξ คือระยะทางระหว่างรังที่น้อยที่สุดที่ยอมรับได้ ซึ่งคำนวณได้จากสมการที่ 3.7 ซึ่งสมการนี้ได้มาจากการรันทดสอบเพื่อกำหนดค่าระยะทางที่น้อยที่สุดที่เหมาะสมสำหรับแต่ละปัญหา จากนั้นจึงนำค่าระยะทางที่ได้ทั้งหมดมาทำการวิเคราะห์การถดถอย (regression analysis) เพื่อคำนวณหาสัมประสิทธิ์การถดถอย (regression coeffcient)

Area คือขนาดพื้นที่ที่ใช้ในการค้นหาคำตอบ ซึ่งคำนวณหาได้จากผลคูณระหว่างขอบเขตของพารามิเตอร์ทั้งหมดของปัญหาที่ทดสอบ ทั้งนี้จำนวนพารามิเตอร์และขอบเขตของปัญหาขึ้นอยู่กับนิยามของปัญหาที่ใช้ทดสอบ

x_i^{max} คือค่าขอบเขตบนของพารามิเตอร์ตัวที่ i ซึ่งได้มีการกำหนดไว้ในนิยามของปัญหาที่ทดสอบ

x_i^{min} คือค่าขอบเขตล่างของพารามิเตอร์ตัวที่ i ซึ่งได้มีการกำหนดไว้ในนิยามของปัญหาที่ทดสอบ

m คือจำนวนพารามิเตอร์ทั้งหมด

จากกระบวนการของการคัดเลือกผึ้งนางพญาใหม่ที่กล่าวมาทั้งหมด สรุปได้ว่า

- ผึ้งนางพญาในรอบปัจจุบันอาจถูกเลือกให้เป็นผึ้งนางพญาใหม่อีกครั้ง แสดงว่า ผึ้งนางพญายังคงรักษารังของตนเองไว้ได้

- ผึ้งนางพญาในรอบปัจจุบันที่ไม่ได้ถูกเลือกให้เป็นผึ้งนางพญาในรอบถัดไปจะถูกคัดทิ้ง
- ตัวอ่อนผึ้งที่ได้รับเลือกให้เป็นผึ้งนางพญาคู่แข่งแต่ไม่ได้รับเลือกให้เป็นผึ้งนางพญาใหม่จะถูกนำไปรวมกับกลุ่มของตัวอ่อนผึ้งเช่นเดิม

สำหรับกระบวนการของการคัดเลือกผึ้งตัวผู้ใหม่ ได้ทำการเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่เหลือจากการคัดเลือกผึ้งนางพญาใหม่เพื่อรักษาจำนวนของผึ้งตัวผู้ในระบบให้คงที่ตามที่ได้กำหนดไว้ โดยผึ้งตัวผู้มีเงื่อนไขการตาย 2 กรณีคือ กรณีแรก ผึ้งตัวผู้ตายเมื่อมีการผสมพันธุ์กับผึ้งนางพญา และกรณีที่สอง ผึ้งตัวผู้ตายเมื่อมีอายุครบตามที่กำหนดไว้ ซึ่งการศึกษาในครั้งนี้ได้กำหนดช่วงชีวิตของผึ้งตัวผู้เท่ากับ 3 และกำหนดอายุเริ่มต้นเท่ากับ 1 เมื่อมีการทำงานในรอบถัดไป อายุของผึ้งตัวผู้จะเพิ่มขึ้นครั้งละ 1 จนกว่าอายุจะครบตามที่กำหนดไว้ ดังนั้น ผึ้งตัวผู้ที่ไม่ได้รับการผสมพันธุ์กับผึ้งนางพญาจะสามารถดำรงชีวิตต่อไปได้ในรอบการทำงานถัดไปจนกว่าอายุจะครบตามที่กำหนดไว้ แต่เมื่อผึ้งตัวผู้ตายลงจะมีการสุ่มเลือกตัวอ่อนผึ้งเข้ามาแทนที่ ดังนั้น กล่าวโดยสรุปได้ว่า การคัดเลือกผึ้งตัวผู้ใหม่คือ การสุ่มเลือกตัวอ่อนผึ้งมาแทนที่ผึ้งตัวผู้ที่ตายลงเพื่อนำมารวมกับผึ้งตัวผู้ที่ยังมีชีวิต

5) ขั้นตอนการแยกรัง (swarm process)

ขั้นตอนนี้เป็นขั้นตอนที่นำเสนอขึ้นใหม่เช่นเดียวกับขั้นตอนที่ผ่านมา ขั้นตอนนี้เป็นขั้นตอนของการแบ่งรังเพื่อทำการจัดกลุ่มผึ้งตัวผู้ให้เข้าเป็นสมาชิกในรังของผึ้งนางพญา โดยผึ้งตัวผู้จะเป็นสมาชิกในรังของผึ้งนางพญาได้เพียงรังเดียวและผึ้งตัวผู้เก่าที่เคยเป็นสมาชิกของรังอื่นมาก่อนจะถูกจัดกลุ่มใหม่เช่นกัน หลักการที่ใช้ในการจัดกลุ่มนี้พิจารณาจากค่าความเป็นสมาชิกของผึ้งตัวผู้ในรังของผึ้งนางพญาทุกตัว ถ้าค่าความเป็นสมาชิกในรังใดมีค่าสูงที่สุดจะถูกเลือกเป็นรังของผึ้งตัวผู้ ดังสมการที่ 3.9

$$\mu(d_p, q_n) = \exp\left(-\frac{\left\| d_p - q_n \right\|^2}{\sigma_{q_n}^2} \right) \tag{3.9}$$

โดยที่

$$\sigma_{q_n} = \max_k \left[d(q_n, q_k) \times \frac{f(q_n)}{f(q_n) + f(q_k)} \right] \tag{3.10}$$

เมื่อ $\mu(d_p, q_n)$ คือค่าความเป็นสมาชิกของผึ้งตัวผู้ลำดับที่ p ในรังของผึ้งนางพญาลำดับที่ n

σ_{q_n} คือขนาดรังของผึ้งนางพญาลำดับที่ n ซึ่งตัวแปรนี้ใช้ในการกำหนดความกว้างของระฆัง (bell curve) ในสมการที่ 3.9 ซึ่งเป็นสมการเกาส์เซียน (gaussian function)

จากสมการที่ 3.10 จะเห็นว่า ถ้า q_n มีค่าฟิตเนสสูงจะได้รับโอกาสให้สร้างรังได้ใหญ่กว่าผึ้งนางพญาที่มีค่าฟิตเนสที่ต่ำกว่า

$f(\cdot)$ คือฟังก์ชันฟิตเนส

$d(q_n, q_k)$ คือระยะทางแบบยูคลิเดียนระหว่างผึ้งตัวผู้ลำดับที่ p และผึ้งนางพญาลำดับที่ n

อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข

กำหนดค่าพารามิเตอร์เริ่มต้น
สร้างประชากรเริ่มต้นโดยการสุ่ม
เลือกประชากรที่มีค่าฟิตเนสสูงที่สุดเป็นผึ้งนางพญาเริ่มต้นและกำหนดให้ประชากรที่เหลือเป็นผึ้งตัวผู้
For ทำซ้ำจนครบตามจำนวนรอบที่กำหนดไว้
  For ทำซ้ำจนครบตามจำนวนผึ้งนางพญา
    กำหนดค่าพลังงานและค่าความเร็วในการบินของผึ้งนางพญาโดยการสุ่ม
    While ทำซ้ำเมื่อค่าพลังงานของผึ้งนางพญาน้อยกว่า E_{min} และถุงเก็บสเปิร์มยังไม่เต็ม
      คัดเลือกผึ้งตัวผู้ที่มีค่าความน่าจะเป็นในการผสมพันธุ์สูงที่สุดเพื่อทำการผสมพันธุ์กับผึ้งนางพญา
      คัดลอกสเปิร์ม (จีโนไทป์) ของผึ้งตัวผู้ที่ได้รับเลือกเก็บไว้ในถุงเก็บเสปิร์มของผึ้งนางพญา
      ปรับลดค่าพลังงานและค่าความเร็วในการบินของผึ้งนางพญา
    End while
  End for
  While ทำซ้ำจนครบตามจำนวนตัวอ่อนผึ้งที่กำหนดไว้
    เลือกผึ้งนางพญาเพื่อทำหน้าที่ครอสโอเวอร์ด้วยวิธีวงล้อหมุน
    สุ่มเลือกสเปิร์มจากถุงเก็บสเปิร์มของผึ้งนางพญาที่ได้รับเลือก
    สร้างโครโมโซมลูกด้วยการครอสโอเวอร์ระหว่างจีโนไทป์ของผึ้งนางพญากับสเปิร์มที่ได้รับเลือก
    มิวเตทโครโมโซมลูกด้วยโอเปอร์เรเตอร์มิวเตชัน
    ปรับค่าโครโมโซมลูกด้วยโอเปอร์เรเตอร์ผึ้งงาน
    คำนวณค่าฟิตเนสของโอเปอร์เรเตอร์ผึ้งงาน
  End while
  คำนวณค่าช่วงชีวิตของผึ้งตัวผู้ที่ไม่ได้รับการผสมพันธุ์
  คัดเลือกผึ้งนางพญาคู่แข่งจากตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าผึ้งนางพญาที่มีค่าฟิตเนสน้อยที่สุด
  เรียงลำดับผึ้งนางพญาคู่แข่งตามค่าฟิตเนสจากค่ามากไปหาค่าน้อย
  กำหนดให้ผึ้งนางพญาคู่แข่งที่มีค่าฟิตเนสสูงที่สุดเป็นผึ้งนางพญาใหม่ลำดับที่หนึ่ง
  For ทำซ้ำจนครบตามจำนวนผึ้งนางพญาคู่แข่ง (โดยเริ่มจากลำดับที่สอง)
    If ระยะทางระหว่างผึ้งนางพญาคู่แข่งและผึ้งนางพญาใหม่ที่ได้รับเลือกก่อนหน้า > ค่าที่กำหนดไว้
      กำหนดให้ผึ้งนางพญาคู่แข่งเป็นผึ้งนางพญาใหม่
    Else
      If ผึ้งนางพญาคู่แข่งเคยได้รับเลือกให้เป็นผึ้งนางพญามาก่อนแล้ว
        ลบผึ้งนางพญาคู่แข่งทิ้ง
      Else
        ย้ายผึ้งนางพญาคู่แข่งเข้าไปรวมกับตัวอ่อนผึ้งที่ไม่ได้รับเลือกให้เป็นผึ้งนางพญาคู่แข่ง
      End if
    End if
  End for
  คัดเลือกผึ้งตัวผู้ที่มีอายุน้อยกว่าที่กำหนดไว้ให้เป็นผึ้งตัวผู้ที่ยังมีชีวิตอยู่ในรอบถัดไป
  สุ่มเลือกผึ้งตัวผู้ใหม่จากตัวอ่อนผึ้งเพื่อแทนที่ผึ้งตัวผู้ที่ตายลง
  แยกรังโดยการจัดกลุ่มผึ้งตัวผู้ให้เป็นสมาชิกในรังของผึ้งนางพญา
End for
Return ผึ้งนางพญาที่มีค่าฟิตเนสสูงที่สุด

รูปที่ 3.5 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข

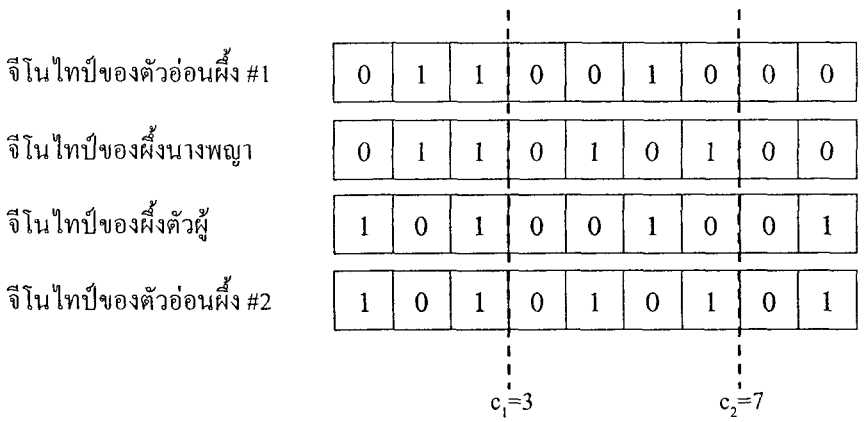
การทำงานในขั้นตอนนี้จะสิ้นสุดลงเมื่อมีการแบ่งรังได้เรียบร้อยแล้ว สมาชิกในรังที่สร้างขึ้นทั้งหมดจะเป็นประชากรเริ่มต้นของรอบถัดไป

6) ทำซ้ำขั้นตอนที่ 2-5 จนกระทั่งจำนวนรอบการทำงานครบตามที่กำหนดไว้ จึงหยุดการทำงานและผึ้งนางพญาที่มีค่าฟิตเนสสูงที่สุดถือเป็นคำตอบที่ดีที่สุดที่อัลกอริทึมสามารถค้นหาได้

3.4.4 ขั้นตอนการทำงานของโอเปอร์เรเตอร์

1) โอเปอร์เรเตอร์ครอสโอเวอร์

การศึกษาในครั้งนี้ได้เลือกใช้โอเปอร์เรเตอร์ครอสโอเวอร์แบบสองจุด โอเปอร์เรเตอร์นี้ทำหน้าที่สร้างตัวอ่อนผึ้งได้ครั้งละ 2 ตัว โดยเริ่มต้นการทำงานด้วยการสุ่มเลือกจุดครอสจำนวน 2 จุด กำหนดให้จุด c_1 และ c_2 แทนจุดครอส จากนั้นให้คัดลอกยีนต์ของผึ้งนางพญาจากตำแหน่งแรกจนถึงตำแหน่งที่ c_1 และตำแหน่งที่ c_2+1 จนถึงตำแหน่งสุดท้ายไปยังจีโนไทป์ของตัวอ่อนผึ้งตัวที่ 1 ในตำแหน่งเดียวกันและทำการคัดลอกยีนต์ของผึ้งตัวผู้ตำแหน่งที่ c_1+1 จนถึงตำแหน่ง c_2 ไปยังจีโนไทป์ของตัวอ่อนผึ้งตัวเดิมในตำแหน่งที่ว่างอยู่ สำหรับการสร้างตัวอ่อนผึ้งตัวที่ 2 จะทำการคัดลอกยีนต์ของผึ้งตัวผู้จากตำแหน่งแรกจนถึงตำแหน่งที่ c_1 และตำแหน่งที่ c_2+1 จนถึงตำแหน่งสุดท้ายไปยังจีโนไทป์ของตัวอ่อนผึ้งตัวที่ 2 ในตำแหน่งเดียวกันและทำการคัดลอกยีนต์ของผึ้งนางพญาตำแหน่งที่ c_1+1 จนถึงตำแหน่ง c_2 ไปยังจีโนไทป์ของตัวอ่อนผึ้งตัวเดิมในตำแหน่งที่ว่างอยู่ ดังรูปที่ 3.6



รูปที่ 3.6 โอเปอร์เรเตอร์ครอสโอเวอร์แบบสองจุด

2) โอเปอร์เรเตอร์มิวเตชัน

โอเปอร์เรเตอร์มิวเตชันที่เลือกใช้ในอัลกอริทึมนี้เป็นโอเปอร์เรเตอร์มิวเตชันมาตรฐานที่ใช้สำหรับการแทนค่าแบบไบนารี การทำงานของโอเปอร์เรเตอร์นี้เริ่มต้นที่บิตแรกโดยการสุ่มค่าตัวแปรซึ่งอยู่ในช่วง [0, 1] กำหนดให้เป็นตัวแปร r_1 ถ้าค่า r_1 ที่สุ่มได้น้อยกว่าหรือเท่ากับค่าความน่าจะเป็นในการมิวเตชันให้ทำการกลับบิตจาก 0 เป็น 1 หรือจาก 1 เป็น 0 และถ้าค่า r_1 มากกว่าค่าความ

น่าจะเป็นในการมิวเตชันจะไม่มีการเปลี่ยนค่าบิต จากนั้นให้ทำซ้ำเช่นเดียวกันจนกระทั่งดำเนินการครบทุกบิต

3) โอเปอร์เรเตอร์ผึ้งงาน

จากที่กล่าวไว้ข้างต้น การศึกษาในครั้งนี้ใช้ผึ้งงานจำนวน 5 ตัวซึ่งประกอบด้วยฟังก์ชันการกลับบิตแบบสุ่ม ฟังก์ชันการการสุ่มเดิน ฟังก์ชันการสุ่มค่าใหม่ ฟังก์ชันการครอสโอเวอร์แบบหนึ่งจุดและฟังก์ชันการครอสโอเวอร์แบบสองจุด โดยแต่ละโอเปอร์เรเตอร์มีขั้นตอนการทำงาน ดังนี้

- ฟังก์ชันการกลับบิตแบบสุ่ม

ฟังก์ชันนี้เริ่มต้นการทำงานโดยการสุ่มเลือกพารามิเตอร์จากจีโนไทป์ของตัวอ่อนผึ้งจำนวน 1 ตัว จากนั้นทำการกลับบิตจาก 0 เป็น 1 หรือจาก 1 เป็น 0 ในทุกตำแหน่งบิตของพารามิเตอร์ที่ถูกเลือกเพื่อแทนที่ค่าพารามิเตอร์เดิมในจีโนไทป์ของตัวอ่อนผึ้ง

- ฟังก์ชันการสุ่มเดิน

ฟังก์ชันนี้ทำการสุ่มบิตจากจีโนไทป์ของตัวอ่อนผึ้งจำนวน 1 บิตเพื่อทำการกลับบิตจาก 0 เป็น 1 หรือจาก 1 เป็น 0

- ฟังก์ชันการสุ่มค่าใหม่

ฟังก์ชันนี้ทำการสุ่มจีโนไทป์ใหม่จำนวน 1 จีโนไทป์ เพื่อนำมาแทนที่จีโนไทป์เดิมของตัวอ่อนผึ้ง

- ฟังก์ชันการครอสโอเวอร์แบบหนึ่งจุด

ฟังก์ชันนี้มีการทำงานเหมือนกับการครอสโอเวอร์แบบหนึ่งจุด ก่อนการเริ่มต้นการครอส-โอเวอร์ต้องทำการสุ่มค่าจีโนไทป์ใหม่จำนวน 1 จีโนไทป์เพื่อใช้ในการครอสโอเวอร์กับจีโนไทป์ของตัวอ่อนผึ้ง จากนั้นจึงทำการสุ่มเลือกจุดครอสจำนวน 1 จุด ซึ่งกำหนดให้เป็นจุด c แล้วให้ทำการคัดลอกยีนต์ของจีโนไทป์ใหม่จากตำแหน่งแรกจนถึงตำแหน่งที่ c ไปแทนที่ยีนต์ในตำแหน่งเดียวกันของจีโนไทป์ของตัวอ่อนผึ้ง

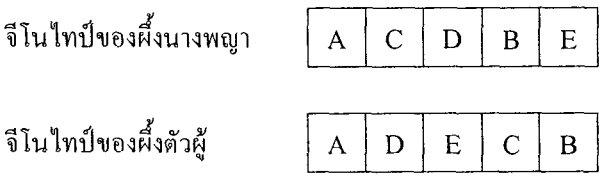
- ฟังก์ชันการครอสโอเวอร์แบบสองจุด

ฟังก์ชันนี้มีลักษณะการทำงานเหมือนกับฟังก์ชันการครอสโอเวอร์แบบหนึ่งจุด ยกเว้นจำนวนจุดที่ใช้ในการครอสโอเวอร์ ฟังก์ชันการครอสโอเวอร์แบบสองจุดมีการสุ่มจุดครอสจำนวน 2 จุด กำหนดให้จุด c_1 และ c_2 แทนจุดครอส ก่อนการเริ่มต้นการครอสโอเวอร์ต้องทำการสุ่มค่าจีโน-ไทป์ใหม่จำนวน 1 จีโนไทป์เพื่อใช้ในการครอสโอเวอร์กับจีโนไทป์ของตัวอ่อนผึ้ง จากนั้นทำการคัดลอกยีนต์ของจีโนไทป์ใหม่ตำแหน่งที่ c_1 จนถึงตำแหน่ง c_2 ไปแทนที่ยีนต์ในตำแหน่งเดียวกันของจีโนไทป์ของตัวอ่อนผึ้ง

3.5 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด

3.5.1 การแทนค่าจีโนไทป์ของผึ้ง

ประชากรผึ้งในอัลกอริทึมที่นำเสนอ แบ่งเป็น 3 กลุ่มคือ ผึ้งนางพญา ผึ้งตัวผู้และผึ้งงาน เช่นเดียวกับการแทนค่าจีโนไทป์ในปัญหาออฟติไมเซชันเชิงตัวเลข แต่จีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้ถูกแทนค่าด้วยอาร์เรย์ของพารามิเตอร์ที่เป็นออบเจ็กต์ของปัญหา ซึ่งแต่ละอิลิเมนต์ของอาร์เรย์แทนด้วยหนึ่งออบเจ็กต์ ตัวอย่างเช่น ปัญหาการเดินทางของพนักงานขายที่มีจำนวน 5 เมือง ประกอบด้วยเมือง A, B, C, D และ E จากตัวอย่างนี้ พารามิเตอร์คือ เมือง ดังนั้น ค่าจีโนไทป์ของผึ้งนางพญา Q = {A, C, D, B, E} และผึ้งตัวผู้ D = {A, D, E, C, B} สามารถแทนค่าจีโนไทป์ได้ ดังรูปที่ 3.7



รูปที่ 3.7 การแทนค่าจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้สำหรับปัญหาออฟติไมเซชันเชิงการจัด

สำหรับผึ้งงานได้มีการแทนค่าด้วยฟังก์ชันฮิวริสติก ซึ่งทำหน้าที่ในการค้นหาแบบโลคอล ฟังก์ชันฮิวริสติกที่นำมาใช้ในการศึกษาในครั้งนี้เป็นโอเปอร์เรเตอร์มิวเตชันที่นิยมกับปัญหาออฟติไมเซชันเชิงการจัด มีจำนวน 6 ฟังก์ชันคือ ฟังก์ชันการเคลื่อนตำแหน่ง (displacement mutation) ฟังก์ชันการแลกเปลี่ยนค่า (exchange mutation) ฟังก์ชันการแทรกตำแหน่ง (insertion mutation) ฟังก์ชันการกลับค่าอย่างง่าย (simple inversion mutation) ฟังก์ชันการกลับค่า (inversion mutation) และฟังก์ชันการเรียงสลับ (scramble mutation) ซึ่งรายละเอียดของแต่ละฟังก์ชันจะกล่าวในลำดับถัดไป

3.5.2 การคำนวณค่าฟิตเนส

ค่าฟิตเนสของผึ้งนางพญาและผึ้งตัวผู้สามารถคำนวณได้จากค่าฟังก์ชันเป้าหมาย กรณีที่เป็นปัญหาออฟติไมเซชันเชิงการจัดที่ต้องการหาค่าฟังก์ชันเป้าหมายที่มีค่าสูงสุด ค่าฟิตเนสจะแปรผันตามค่าฟังก์ชันเป้าหมาย แต่ถ้ากรณีที่เป็นปัญหาของการหาค่าฟังก์ชันเป้าหมายที่มีค่าต่ำสุด ค่าฟิตเนสจะแปรผกผันกับค่าฟังก์ชันเป้าหมาย การศึกษาในครั้งนี้ได้ทดสอบกับปัญหาการเดินทางของพนักงานขาย ซึ่งมีเป้าหมายเพื่อค้นหาเส้นทางที่มีระยะทางสั้นที่สุดที่ผ่านเมืองทุกเมือง ดังนั้นปัญหาการเดินทางของพนักงานขายจึงจัดให้เป็นปัญหาที่มีฟังก์ชันเป้าหมายเป็นค่าต่ำสุด โดยฟังก์ชันเป้าหมายของปัญหานี้ ดังสมการที่ 3.11

$$\text{minimize } f(t) = \sum_{i=1}^{n} d_{i\pi(i)} \tag{3.11}$$

เมื่อ t คือเซตของเมืองที่เกิดจากการเรียงสับเปลี่ยนจำนวน n เมือง ซึ่งเรียกว่า ทัวร์ (tour)

$f(\cdot)$ คือฟังก์ชันเป้าหมาย

$\pi(i)$ คือเมืองลำดับถัดไปที่เลือกเดินทางต่อจากเมือง i

$d_{i\pi(i)}$ คือระยะทางระหว่างเมือง i ไปยังเมืองลำดับถัดไป

จากรูปที่ 3.7 จีโนไทป์ของผึ้งนางพญาแทนทัวร์ t ในสมการที่ 3.11 ซึ่งสามารถคำนวณค่าฟังก์ชันเป้าหมายได้ เท่ากับ $f(t) = d_{AC} + d_{CD} + d_{DB} + d_{BE} + d_{EA}$ และจีโนไทป์ของผึ้งตัวผู้ สามารถคำนวณค่าฟังก์ชันเป้าหมายได้ เท่ากับ $f(t) = d_{AD} + d_{DE} + d_{EC} + d_{CB} + d_{BA}$ จากนั้นให้คำนวณค่าฟิตเนสของผึ้งนางพญาและผึ้งตัวผู้ได้จาก fitness(Q) = -f(t) หรือ fitness(Q) = 1/f(t) ก็ได้ จะเห็นว่า จากวิธีการคำนวณค่าฟิตเนสนี้ ผึ้งที่เป็นคำตอบที่เหมาะสมที่สุดคือ ผึ้งที่ให้ค่าฟังก์ชันเป้าหมายต่ำที่สุด แต่มีค่าฟิตเนสสูงที่สุด

3.5.3 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด

อัลกอริทึมที่นำเสนอ แบ่งเป็นขั้นตอนหลักได้ 5 ขั้นตอน ดังนี้

1) ขั้นตอนการกำหนดค่าเริ่มต้น

ในขั้นตอนนี้มีการทำงานเหมือนกับอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข โดยการกำหนดค่าเริ่มต้นได้แบ่งเป็น 2 ส่วนคือ การกำหนดค่าพารามิเตอร์เริ่มต้นและการกำหนดรังผึ้งเริ่มต้น ในส่วนของการกำหนดค่าพารามิเตอร์เริ่มต้น มีพารามิเตอร์ที่กำหนดค่า ดังนี้ ขนาดของถุงเก็บสเปิร์มของผึ้งนางพญา จำนวนของผึ้งตัวผู้ จำนวนของตัวอ่อนผึ้ง ค่าความน่าจะเป็นในการมิวเตชัน อัตราค่าความเหมือน (similarity rate) และจำนวนรอบการทำงาน และในส่วนของการกำหนดรังผึ้งเริ่มต้นได้มีการสุ่มค่าสมาชิกภายในรังซึ่งประกอบด้วยผึ้งนางพญาจำนวน 1 ตัวและผึ้งตัวผู้ตามจำนวนที่กำหนดไว้ โดยค่าที่สุ่มกำหนดให้อยู่ในช่วงขอบเขตของปัญหา จากนั้นให้คำนวณค่าฟิตเนสของสมาชิกภายในรังทั้งหมดและทำการจัดเรียงลำดับสมาชิกตามค่าฟิตเนสที่คำนวณได้ สมาชิกที่มีค่าฟิตเนสสูงที่สุดจะถูกเลือกให้เป็นผึ้งนางพญาเริ่มต้นและสมาชิกส่วนที่เหลือกำหนดให้เป็นผึ้งตัวผู้เริ่มต้น

2) ขั้นตอนการผสมพันธุ์

กระบวนการทำงานในขั้นตอนนี้มีการทำงานเหมือนกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม กล่าวคือ ผึ้งนางพญาทุกตัวมีหน้าที่ในการผสมพันธุ์และสามารถเลือกผสมพันธุ์กับผึ้งตัวผู้ได้หลายตัว ผึ้งนางพญาแต่ละตัวจะผสมพันธุ์กับผึ้งตัวผู้จำนวนมากหรือน้อยขึ้นอยู่กับค่า

พลังงานเริ่มต้นและขนาดของถุงเก็บสเปิร์ม เริ่มต้นขั้นตอนการผสมพันธุ์ด้วยการกำหนดค่าเริ่มต้นของค่าพลังงานและค่าความเร็วในการบินให้กับผึ้งนางพญาแต่ละตัวโดยการสุ่มค่าที่อยู่ในช่วง [0.5, 1] จากนั้นผึ้งนางพญาตัดสินใจเลือกผสมพันธุ์กับผึ้งตัวผู้ โดยการพิจารณาจากค่าความน่าจะเป็นในการผสมพันธุ์ ดังสมการที่ 3.12

$$prob(Q,D) = e^{\frac{-difference}{speed(t)}} \tag{3.12}$$

เมื่อ prob(Q,D) คือค่าความน่าจะเป็นในการผสมพันธุ์ระหว่างผึ้งนางพญา Q และผึ้งตัวผู้ D

difference คือค่าผลต่างสมบูรณ์ระหว่างค่าฟิตเนสของผึ้งนางพญา Q และผึ้งตัวผู้ D

speed(t) คือค่าความเร็วในการบินของผึ้งนางพญา Q ณ รอบการทำงานที่ t

ผึ้งนางพญาตัดสินใจเลือกผสมพันธุ์กับผึ้งตัวผู้ที่มีค่าความน่าจะเป็นในการผสมพันธุ์สูงที่สุดและสเปิร์ม (จีโนไทป์) ของผึ้งตัวผู้ที่ดีได้รับเลือกจะถูกนำมาเก็บไว้ในถุงเก็บสเปิร์มของผึ้งนางพญาเพื่อใช้ในการปฏิสนธิกับไข่ จากนั้นก่อนที่ผึ้งนางพญาจะเริ่มการเลือกคู่ครั้งถัดไป ค่าพลังงานและค่าความเร็วในการบินของผึ้งนางพญาจะถูกปรับลดลง ดังนี้

$$speed(t+1) = \alpha \times speed(t) \tag{3.13}$$

$$energy(t+1) = energy(t) - step \tag{3.14}$$

เมื่อ α คืออัตราการลดลงของค่าความเร็วในการบิน ซึ่งกำหนดให้มีค่าอยู่ในช่วง [0, 1] การศึกษาในครั้งนี้ได้กำหนดค่าให้เท่ากับ 0.95 ตามที่งานวิจัย [1] นำเสนอไว้

step คือปริมาณการลดลงของค่าพลังงาน คำนวณได้จาก

$step = (0.5 \times energy) / spermatheca\text{-}size$

ขั้นตอนการผสมพันธุ์ของผึ้งนางพญาแต่ละตัวสิ้นสุดลงเมื่อถุงเก็บสเปิร์มเต็มหรือค่าพลังงานน้อยกว่าค่าที่กำหนดไว้ซึ่งการศึกษาในครั้งนี้กำหนดค่าให้เท่ากับศูนย์

3) ขั้นตอนการสร้างตัวอ่อนผึ้งและการเลี้ยงดู

ขั้นตอนนี้เริ่มต้นด้วยการเลือกผึ้งนางพญาเพื่อทำหน้าที่วางไข่ โดยผึ้งนางพญาถูกสุ่มเลือกด้วยวิธีการเลือกแบบวงล้อหมุน ผึ้งนางพญาที่มีค่าฟิตเนสสูงจะได้รับโอกาสให้ทำหน้าที่วางไข่สูงกว่าผึ้งนางพญาที่มีค่าฟิตเนสต่ำกว่า จากนั้นผึ้งนางพญาที่ได้รับเลือกจะทำการสุ่มเลือกสเปิร์มจากถุงเก็บสเปิร์มของตนเองเพื่อทำการปฏิสนธิกับไข่ ซึ่งกระบวนการปฏิสนธิคือการครอสโอเวอร์

ระหว่างจีโนไทป์ของผึ้งนางพญาและผึ้งตัวผู้ การศึกษาในครั้งนี้ได้ใช้โอเปอร์เรเตอร์ครอสโอเวอร์แบบวงกลม (cycle crossover) จากนั้นตัวอ่อนผึ้งที่ได้จะถูกนำไปมิวเตทด้วยโอเปอร์เรเตอร์มิวเตชันแบบแลกเปลี่ยนค่าในตำแหน่งติดกัน (adjacent exchange mutation) หลังจากนั้นจะเข้าสู่ขั้นตอนการเลี้ยงดู โดยมีผึ้งงานซึ่งแทนด้วยฟังก์ชันฮิวริสติกทำหน้าที่ดูแลตัวอ่อนผึ้ง การศึกษาในครั้งนี้ได้ใช้ผึ้งงานจำนวน 6 ตัวซึ่งประกอบด้วยฟังก์ชันการเคลื่อนตำแหน่ง ฟังก์ชันการแลกเปลี่ยนค่า ฟังก์ชันการแทรกตำแหน่ง ฟังก์ชันการกลับค่าอย่างง่าย ฟังก์ชันการกลับค่า และฟังก์ชันการเรียงสลับ ซึ่งรายละเอียดของแต่ละฟังก์ชันจะกล่าวในหัวข้อถัดไป ตัวอ่อนผึ้งแต่ละตัวจะได้รับการเลี้ยงดูจากผึ้งงานที่แตกต่างกันโดยการสุ่มเลือกด้วยวิธีการเลือกแบบวงล้อหมุนโดยการพิจารณาจากค่าฟิตเนสของผึ้งงาน ซึ่งค่าฟิตเนสของผึ้งงานคำนวณได้จากความสามารถในการปรับค่าฟิตเนสของตัวอ่อนผึ้ง

4) ขั้นตอนการเตรียมพร้อมเพื่อแยกรัง

ขั้นตอนการเตรียมพร้อมเพื่อแยกรังประกอบด้วย 2 ขั้นตอนย่อยคือ ขั้นตอนการคัดเลือกผึ้งนางพญาใหม่และขั้นตอนการคัดเลือกผึ้งตัวผู้ใหม่เพื่อใช้เป็นประชากรเริ่มต้นของการทำงานในรอบถัดไป กระบวนการของการคัดเลือกผึ้งนางพญาใหม่เป็นการแข่งขันกันระหว่างผึ้งนางพญาและตัวอ่อนผึ้งโดยการพิจารณาจากค่าฟิตเนส ตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าผึ้งนางพญาที่มีค่าฟิต-เนสต่ำที่สุดจะถูกเลือกให้เป็นผึ้งนางพญาคู่แข่งเพื่อเข้าสู่กระบวนการคัดเลือกต่อไป (ผึ้งนางพญาคู่แข่งหมายรวมถึงผึ้งนางพญาในรอบปัจจุบันด้วย) ทั้งนี้เพื่อป้องกันมิให้ผึ้งนางพญาใหม่มีจำนวนมากเกินไปและป้องกันไม่ให้ผึ้งนางพญาสร้างรังใกล้กันมากจนเกินไป การคัดเลือกผึ้งนางพญาใหม่จึงได้มีการพิจารณาค่าความเหมือน (similarity value) ของผึ้งนางพญาคู่แข่งด้วย โดยวิธีการวัดความเหมือนได้นำเสนอขึ้นใหม่ในอัลกอริทึมนี้เพื่อใช้ในการวัดความเหมือนระหว่างจีโนไทป์ที่มีการแทนค่าแบบข้อความ ซึ่งแตกต่างจากการแทนค่าด้วยเลขจำนวนจริงที่สามารถใช้การวัดระยะทางระหว่างรังเป็นเกณฑ์การพิจารณาได้ การวัดค่าความเหมือนวิธีนี้ ถ้าผึ้งนางพญามีค่าความเหมือนใกล้เคียงกัน แสดงว่า จีโนไทป์มีความคล้ายคลึงกันมาก (เลือกใช้เส้นทางคล้ายกัน) และด้วยสาเหตุที่จีโนไทป์มีความคล้ายคลึงกันมาก จึงทำให้ผึ้งนางพญาซึ่งแทนจุดศูนย์กลางของรังมีการสร้างรังใกล้กันไปด้วย ดังนั้น การคัดเลือกผึ้งนางพญาใหม่จึงมีกฎเกณฑ์ ดังนี้ 1) ผึ้งนางพญาคู่แข่งที่มีความแข็งแรงมากกว่าจะได้รับโอกาสให้สร้างรังได้ก่อนผึ้งนางพญาคู่แข่งที่มีความแข็งแรงน้อยกว่าและ 2) ผึ้งนางพญาใหม่แต่ละตัวต้องมีค่าความเหมือนมากกว่าค่าที่กำหนดไว้ ดังนั้น ก่อนการคัดเลือกผึ้งนางพญาใหม่จึงต้องทำการจัดเรียงผึ้งนางพญาคู่แข่งตามค่าฟิตเนส ผึ้งนางพญาคู่แข่งที่มีค่าฟิตเนสสูงที่สุด แสดงว่า มีความแข็งแรงมากที่สุดจึงได้รับเลือกให้เป็นผึ้งนางพญาใหม่ตัวแรกเสมอ จากนั้นจึงเริ่มการพิจารณาผึ้งนางพญาคู่แข่งลำดับที่สองโดยการวัดค่าความเหมือน ค่าความเหมือนระหว่างผึ้งนางพญาคู่แข่งลำดับที่สองกับผึ้งนางพญาใหม่ที่ได้เลือกก่อนหน้า ซึ่งคำนวณได้จากสมการที่ 3.16 จะถูกนำมาพิจารณาเป็นเงื่อนไขการเลือก ดังนี้ ถ้าผึ้งนางพญาคู่แข่งในลำดับที่

สองมีค่าความเหมือนมากกว่าค่าความเหมือนที่กำหนดไว้ ผึ้งนางพญาลำดับที่สองจะถูกเลือกให้เป็นผึ้งนางพญาใหม่อีกตัว ดังสมการที่ 3.15 แต่ถ้ามีค่าความเหมือนน้อยกว่าค่าที่กำหนดไว้ แสดงว่าผึ้งนางพญาคู่แข่งลำดับที่สองควรเป็นสมาชิกภายในรังของผึ้งนางพญาใหม่ ดังนั้น จึงให้ทำการลบผึ้งนางพญาคู่แข่งลำดับที่สองออกจากลิสต์ อย่างไรก็ตาม ถ้าผึ้งนางพญาคู่แข่งลำดับที่สองเป็นตัวอ่อนผึ้งให้นำรวมอยู่ในกลุ่มของตัวอ่อนผึ้งคงเดิม แต่ถ้าผึ้งนางพญาคู่แข่งลำดับที่สองเคยเป็นผึ้งนางพญามาก่อนแล้วให้ทำการคัดทิ้งได้ หลังจากนั้นกระบวนการคัดเลือกจะดำเนินการพิจารณาผึ้งนางพญาคู่แข่งในลำดับถัดไปจนกว่าจะครบทุกตัว

$$\text{similarity}(q_N, C_k) \geq \xi \tag{3.15}$$

โดยที่

$$\text{similarity}(q_N, C_k) = \left| v(q_N) - v(C_k) \right| \tag{3.16}$$

$$v(C_k) = \frac{\sum_{e_{ij} \in T_k} w(e_{ij})}{|T_k|} \tag{3.17}$$

$$w(e_{ij}) = \sum_{C_k \in S_{e_{ij}}} \left| \bar{f}(C_k) \times r(C_k) \right| \tag{3.18}$$

เมื่อ similarity(q_N, C_k) คือค่าความเหมือนระหว่างผึ้งนางพญาคู่แข่งที่ k กับผึ้งนางพญาใหม่ที่ N

N คือลำดับของผึ้งนางพญาใหม่ที่มีค่าความเหมือนใกล้เคียงกับผึ้งนางพญาคู่แข่งมากที่สุด

ξ คือค่าความเหมือนที่น้อยที่สุดที่ยอมรับได้ แนะนำให้กำหนดค่าเท่ากับ 0.05 เนื่องจากค่าความเหมือนของผึ้งนางพญามีค่าสูงที่สุดเท่ากับ 1 ดังนั้นจำนวนของผึ้งนางพญาที่สามารถสร้างได้จึงไม่เกิน 1/0.05 = 20 ตัว ถ้ามีกำหนดค่าความเหมือนที่น้อยที่สุดที่ยอมรับได้ไว้มากกว่า 0.05 จะส่งผลให้จำนวนของผึ้งนางพญาที่สามารถสร้างได้ในแต่ละรอบการทำงานน้อยลง แต่ถ้ามีกำหนดค่าความเหมือนที่น้อยที่สุดที่ยอมรับได้ไว้น้อยกว่า 0.05 จะทำให้จำนวนของผึ้งนางพญาที่สามารถสร้างได้ในแต่ละรอบการทำงานมากยิ่งขึ้น ถ้ายิ่งมีจำนวนผึ้งนางพญามากขึ้นย่อมส่งผลทำให้เวลาที่ใช้ในการประมวลผลยิ่งมากขึ้นด้วยเช่นกัน

$v(\cdot)$ คือค่าถ่วงน้ำหนักเฉลี่ย

T_k คือเซตของเส้นเชื่อมที่ผึ้งนางพญาคู่แข่งเลือกใช้

$w(e_{ij})$ คือค่าถ่วงน้ำหนักของเส้นเชื่อมระหว่างเมือง i ไปยังเมือง j โดยที่ค่าถ่วงน้ำหนักนี้คำนวณได้จากความถี่ในการใช้เส้นทางของผึ้งนางพญาคู่แข่งทั้งหมด ซึ่งมีค่าอยู่ในช่วง [0,1] ถ้ามีการเลือกใช้เส้นทางใดสูงค่าถ่วงน้ำหนักจะมีค่าเข้าใกล้หนึ่ง

S คือจำนวนของผึ้งนางพญาใหม่ที่ถูกเลือกไว้ก่อนหน้าแล้ว

$\bar{f}(\cdot)$ คือค่าฟิตเนสที่ผ่านการนอร์มอลไลเซชัน (normalization) มีค่าอยู่ในช่วง [0,1] คำนวณได้ ดังสมการที่ 3.19

$$\bar{f}(C_k) = \frac{f(C_k)}{\sum_{u=1}^{K} f(C_u)} \tag{3.19}$$

K คือจำนวนของผึ้งนางพญาคู่แข่งทั้งหมด

$r(\cdot)$ คือค่าคะแนนการจัดลำดับ (ranking score) ซึ่งมีค่าอยู่ในช่วง [0,1] ค่านี้แปรผันตามค่าฟิตเนส ถ้าผึ้งนางพญาคู่แข่งมีค่าฟิตเนสสูงที่สุดจะมีค่าคะแนนเท่ากับหนึ่งเสมอ ส่วนผึ้งนางพญาตัวอื่นจะมีค่าคะแนนการจัดลำดับลดหลั่นตามค่าฟิตเนส ดังสมการที่ 3.20

$$r(C_k) = \frac{f(C_k)}{\max_{u \in K}[f(C_u)]} \tag{3.20}$$

จากกระบวนการของการคัดเลือกผึ้งนางพญาใหม่ที่กล่าวมาทั้งหมด สรุปได้ว่า

- ผึ้งนางพญาในรอบปัจจุบันอาจถูกเลือกให้เป็นผึ้งนางพญาใหม่อีกครั้ง แสดงว่า ผึ้งนางพญายังคงรักษารังของตนเองไว้ได้
- ผึ้งนางพญาในรอบปัจจุบันที่ไม่ได้ถูกเลือกให้เป็นผึ้งนางพญาในรอบถัดไปจะถูกคัดทิ้ง
- ตัวอ่อนผึ้งที่ได้รับเลือกให้เป็นผึ้งนางพญาคู่แข่งแต่ไม่ได้รับเลือกให้เป็นผึ้งนางพญาใหม่จะถูกนำไปรวมกับกลุ่มของตัวอ่อนผึ้งเช่นเดิม

สำหรับกระบวนการของการคัดเลือกผึ้งตัวผู้ใหม่ ได้ทำการเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่เหลือจากการคัดเลือกผึ้งนางพญาใหม่เพื่อรักษาจำนวนของผึ้งตัวผู้ในระบบให้คงที่ตามที่ได้กำหนดไว้ โดยผึ้งตัวผู้มีเงื่อนไขการตาย 2 กรณี คือ กรณีแรก ผึ้งตัวผู้ตายเมื่อมีการผสมพันธุ์กับผึ้งนางพญา และกรณีที่สอง ผึ้งตัวผู้ตายเมื่อมีอายุครบตามที่กำหนดไว้ ซึ่งการศึกษาในครั้งนี้ได้กำหนดช่วงชีวิตของผึ้งตัวผู้เท่ากับ 3 และกำหนดอายุเริ่มต้นเท่ากับ 1 เมื่อมีการทำงานในรอบถัดไป อายุของผึ้งตัวผู้

จะเพิ่มขึ้นครั้งละ 1 จนกว่าอายุจะครบตามที่กำหนดไว้ ดังนั้น ผึ้งตัวผู้ที่ไม่ได้รับการผสมพันธุ์กับผึ้งนางพญาจะสามารถดำรงชีวิตต่อไปได้ในรอบการทำงานถัดไปจนกว่าอายุจะครบตามที่กำหนดไว้ แต่เมื่อผึ้งตัวผู้ตายลงจะมีการสุ่มเลือกตัวอ่อนผึ้งเข้ามาแทนที่เพื่อนำมารวมกับผึ้งตัวผู้ที่ยังมีชีวิต

อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด

กำหนดค่าพารามิเตอร์เริ่มต้น
สร้างประชากรเริ่มต้นโดยการสุ่ม
เลือกประชากรที่มีค่าฟิตเนสสูงที่สุดเป็นผึ้งนางพญาเริ่มต้นและกำหนดให้ประชากรที่เหลือเป็นผึ้งตัวผู้
For ทำซ้ำจนครบตามจำนวนรอบที่กำหนดไว้
For ทำซ้ำจนครบตามจำนวนผึ้งนางพญา
กำหนดค่าพลังงานและค่าความเร็วในการบินของผึ้งนางพญาโดยการสุ่ม
While ทำซ้ำเมื่อค่าพลังงานของผึ้งนางพญาน้อยกว่า E_{min} และถุงเก็บสเปิร์มยังไม่เต็ม
คัดเลือกผึ้งตัวผู้ที่มีค่าความน่าจะเป็นในการผสมพันธุ์สูงที่สุดเพื่อทำการผสมพันธุ์กับผึ้งนางพญา
คัดลอกสเปิร์ม (จีโนไทป์) ของผึ้งตัวผู้ที่ได้รับเลือกเก็บไว้ในถุงเก็บเสปิร์มของผึ้งนางพญา
ปรับลดค่าพลังงานและค่าความเร็วในการบินของผึ้งนางพญา
End while
End for
While ทำซ้ำจนครบตามจำนวนตัวอ่อนผึ้งที่กำหนดไว้
เลือกผึ้งนางพญาเพื่อทำหน้าที่ครอสโอเวอร์ด้วยวิธีวงล้อหมุน
สุ่มเลือกสเปิร์มจากถุงเก็บสเปิร์มของผึ้งนางพญาที่ได้รับเลือก
สร้างโครโมโซมลูกด้วยการครอสโอเวอร์ระหว่างจีโนไทป์ของผึ้งนางพญากับสเปิร์มที่ได้รับเลือก
มิวเตทโครโมโซมลูกด้วยโอเปอร์เรเตอร์มิวเตชัน
ปรับค่าโครโมโซมลูกด้วยโอเปอร์เรเตอร์ผึ้งงาน
คำนวณค่าฟิตเนสของโอเปอร์เรเตอร์ผึ้งงาน
End while
คำนวณค่าช่วงชีวิตของผึ้งตัวผู้ที่ไม่ได้รับการผสมพันธุ์
คัดเลือกผึ้งนางพญาคู่แข่งจากตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าผึ้งนางพญาที่มีค่าฟิตเนสน้อยที่สุด
เรียงลำดับผึ้งนางพญาคู่แข่งตามค่าฟิตเนสจากค่ามากไปหาค่าน้อย
กำหนดให้ผึ้งนางพญาคู่แข่งที่มีค่าฟิตเนสสูงที่สุดเป็นผึ้งนางพญาใหม่ลำดับที่หนึ่ง
For ทำซ้ำจนครบตามจำนวนผึ้งนางพญาคู่แข่ง (โดยเริ่มจากลำดับที่สอง)
If ค่าความเหมือนระหว่างผึ้งนางพญาคู่แข่งและผึ้งนางพญาใหม่ที่ได้รับเลือกก่อนหน้า > ค่าที่กำหนดไว้
กำหนดให้ผึ้งนางพญาคู่แข่งเป็นผึ้งนางพญาใหม่
Else
If ผึ้งนางพญาคู่แข่งเคยได้รับเลือกให้เป็นผึ้งนางพญามาก่อนแล้ว
ลบผึ้งนางพญาคู่แข่งทิ้ง
Else
ย้ายผึ้งนางพญาคู่แข่งเข้าไปรวมกับตัวอ่อนผึ้งที่ไม่ได้รับเลือกให้เป็นผึ้งนางพญาคู่แข่ง
End if
End if
End for
คัดเลือกผึ้งตัวผู้ที่มีอายุน้อยกว่าที่กำหนดไว้ให้เป็นผึ้งตัวผู้ที่ยังมีชีวิตอยู่ในรอบถัดไป
สุ่มเลือกผึ้งตัวผู้ใหม่จากตัวอ่อนผึ้งเพื่อแทนที่ผึ้งตัวผู้ที่ตายลง
แยกรังโดยการจัดกลุ่มผึ้งตัวผู้ให้เป็นสมาชิกในรังของผึ้งนางพญา
End for
Return ผึ้งนางพญาที่มีค่าฟิตเนสสูงที่สุด

รูปที่ 3.8 อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด

5) ขั้นตอนการแยกรัง

ขั้นตอนการแยกรัง เป็นการจัดกลุ่มผึ้งตัวผู้ให้เข้าเป็นสมาชิกในรังของผึ้งนางพญา โดยผึ้งตัวผู้จะเป็นสมาชิกในรังของผึ้งนางพญาได้เพียงรังเดียวและผึ้งตัวผู้เก่าที่เคยเป็นสมาชิกของรังอื่นมาก่อนจะถูกจัดกลุ่มใหม่เช่นกัน หลักการที่ใช้ในการจัดกลุ่มนี้พิจารณาจากค่าความเหมือน ถ้าผึ้งตัวผู้มีค่าความเหมือนใกล้เคียงกับค่าความเหมือนของผึ้งนางพญาใดมากที่สุด ผึ้งตัวผู้จะถูกจัดให้เป็นสมาชิกในรังของผึ้งนางพญาตัวนั้น ดังสมการที่ 3.21

$$N = \arg\min_{1 \le n \le S} \left(\text{similarity}(d_p, q_n)\right) \tag{3.21}$$

เมื่อ $\text{similarity}(d_p, q_n)$ คือค่าความเหมือนระหว่างผึ้งตัวผู้ที่ p กับผึ้งนางพญาที่ n

N คือลำดับของผึ้งนางพญาที่มีค่าความเหมือนใกล้เคียงกับผึ้งตัวผู้มากที่สุด

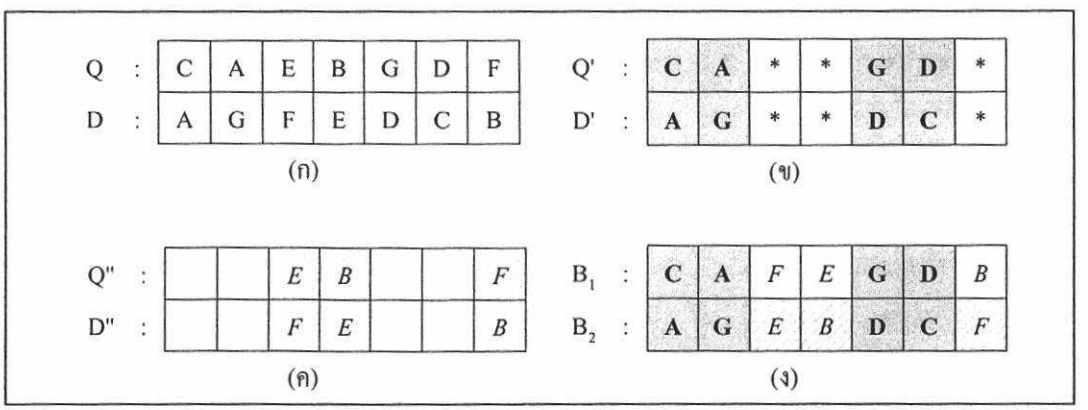
การทำงานในขั้นตอนนี้จะสิ้นสุดลงเมื่อมีการแบ่งแยกรังได้เรียบร้อยแล้ว สมาชิกในรังที่สร้างขึ้นทั้งหมดจะเป็นประชากรเริ่มต้นของรอบถัดไป

6) ทำซ้ำขั้นตอนที่ 2-5 จนกระทั่งจำนวนรอบการทำงานครบตามที่กำหนดไว้ จึงหยุดการทำงานและผึ้งนางพญาที่มีค่าฟิตเนสสูงที่สุดถือเป็นคำตอบที่ดีที่สุดที่อัลกอริทึมสามารถค้นหาได้

3.5.4 ขั้นตอนการทำงานของโอเปอร์เรเตอร์

1) โอเปอร์เรเตอร์ครอสโอเวอร์

การศึกษาในครั้งนี้ได้เลือกใช้โอเปอร์เรเตอร์ครอสโอเวอร์แบบวงกลมที่นำเสนอโดย Oliver, Smith และ Holland [21] โอเปอร์เรเตอร์นี้ทำการสร้างจีโนไทป์ของตัวอ่อนผึ้งรุ่นลูกด้วยการเลือกเมืองมาจากจีโนไทป์ของรุ่นพ่อแม่โดยยังคงรักษาลำดับตำแหน่งของเมืองและมีวิธีการเลือกเมืองลักษณะเป็นวงกลม ดังรูปที่ 3.9



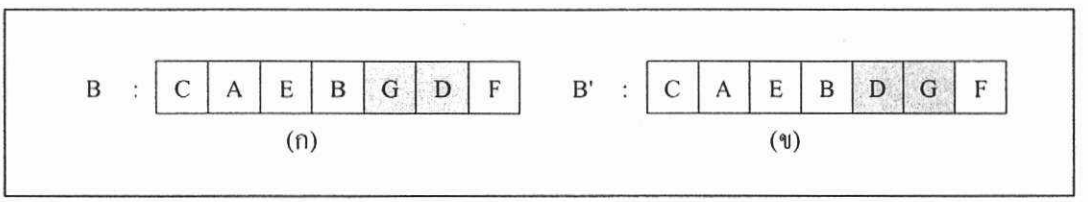
รูปที่ 3.9 การทำงานของโอเปอร์เรเตอร์ครอสโอเวอร์แบบวงกลม

ขั้นตอนการทำงานโอเปอร์เรเตอร์ครอสโอเวอร์แบบวงกลม สรุปได้ดังนี้

- สุ่มเลือกตำแหน่งเมืองเริ่มต้นของวงกลมจากจีโนไทป์ของผึ้งนางพญา Q กำหนดให้เป็นตำแหน่งแรกคือ เมือง C
- เลือกเมืองลำดับถัดไปให้มีลักษณะเป็นวงกลมด้วยมีวิธีการดังนี้ หาตำแหน่งของเมือง C ในจีโนไทป์ของผึ้งตัวผู้ D จะได้ตำแหน่งที่ 6 จากนั้นให้เลือกเมืองจากจีโนไทป์ของผึ้งนางพญาในตำแหน่งที่ 6 คือ เมือง D จากนั้นให้ทำซ้ำเช่นเดิมคือ หาตำแหน่งของเมือง D ในจีโนไทป์ของผึ้งตัวผู้ D จะได้ ตำแหน่งที่ 5 จากนั้นให้เลือกเมืองจากจีโนไทป์ของผึ้งนางพญาในตำแหน่งที่ 5 คือ เมือง G ให้ทำซ้ำเช่นนี้จนกระทั่งวนกลับมาอยู่ที่ตำแหน่งเริ่มต้น ในที่นี้คือ ตำแหน่งแรก เมื่อเลือกเป็นวงกลมแล้ว จะได้ C-D-G-A ตามลำดับ ดังรูปที่ 3.9(ข)
- เมืองที่ไม่ได้รับเลือกจากขั้นตอนก่อนหน้านี้ ดังรูปที่ 3.9(ค) จะถูกคัดลอกไปยังจีโนไทป์ของตัวอ่อนผึ้งแต่ละตัวแบบสลับกัน จากรูปที่ 3.9(ง) จะเห็นว่า จีโนไทป์ของตัวอ่อนผึ้งตัวแรกมีตำแหน่งเมืองที่เลือกเป็นวงกลมเหมือนจีโนไทป์ของผึ้งนางพญาและมีตำแหน่งเมืองที่ไม่ได้รับเลือกเหมือนจีโนไทป์ของผึ้งตัวผู้ ส่วนจีโนไทป์ของตัวอ่อนผึ้งอีกตัวมีตำแหน่งเมืองที่เลือกเป็นวงกลมเหมือนจีโนไทป์ของผึ้งตัวผู้และมีตำแหน่งเมืองที่ไม่ได้รับเลือกเหมือนจีโนไทป์ของผึ้งนางพญา

2) โอเปอร์เรเตอร์มิวเตชัน

การศึกษาในครั้งนี้ได้เลือกใช้โอเปอร์เรเตอร์มิวเตชันแบบแลกเปลี่ยนค่าในตำแหน่งติดกัน ซึ่งนำเสนอโดย Murata และ Ishibuchi [22] โอเปอร์เรเตอร์นี้ทำการสุ่มตำแหน่งที่ต้องการมิวเตทตั้งแต่ตำแหน่งที่ 1-7 เพื่อทำการสลับเมืองในตำแหน่งติดกัน จากรูปที่ 3.10 ตำแหน่งที่สุ่มได้คือตำแหน่งที่ 5 จากนั้นให้ทำการสลับเมืองในตำแหน่งที่ 5 และ 6



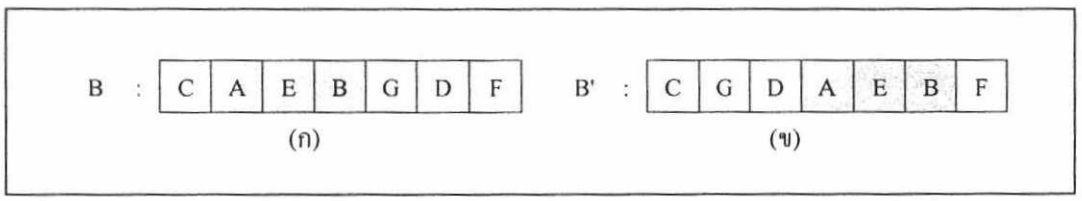
รูปที่ 3.10 การทำงานของโอเปอร์เรเตอร์มิวเตชันแบบแลกเปลี่ยนค่าในตำแหน่งติดกัน

3) โอเปอร์เรเตอร์ผึ้งงาน

จากที่กล่าวไว้ข้างต้น การศึกษาในครั้งนี้ใช้ผึ้งงานจำนวน 6 ตัวซึ่งประกอบด้วยฟังก์ชันการเคลื่อนตำแหน่ง ฟังก์ชันการแลกเปลี่ยนค่า ฟังก์ชันการแทรกตำแหน่ง ฟังก์ชันการกลับค่าอย่างง่าย ฟังก์ชันการกลับค่า และฟังก์ชันการเรียงสลับ โดยแต่ละโอเปอร์เรเตอร์มีขั้นตอนการทำงาน ดังนี้

- ฟังก์ชันการเคลื่อนตำแหน่ง

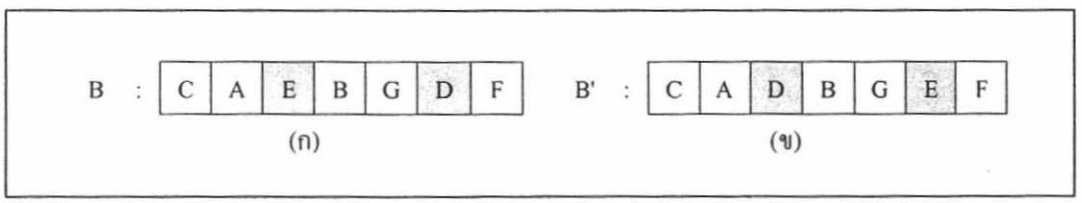
ฟังก์ชันนี้นำเสนอโดย Michalewicz [23] เริ่มต้นการทำงานโดยการสุ่มเลือกช่วงของเมืองเพื่อทำการย้ายไปยังตำแหน่งอื่น จากรูปที่ 3.11 ช่วงตำแหน่งที่สุ่มได้คือ ตำแหน่งที่ 2-4 คือ เมือง A-E-B จากนั้นให้ทำการย้ายเมืองดังกล่าวไปยังตำแหน่งสุ่มใหม่ กำหนดให้เป็นตำแหน่งที่ 4 ดังนั้น เมือง A, E และ B จะถูกย้ายไปยังตำแหน่ง 4, 5 และ 6 ตามลำดับ โดยที่ทำการเลื่อนตำแหน่งของเมือง G และ D มาแทนในตำแหน่งที่ 2 และ 3 ตามลำดับ



รูปที่ 3.11 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบเคลื่อนตำแหน่ง

- ฟังก์ชันการแลกเปลี่ยนตำแหน่ง

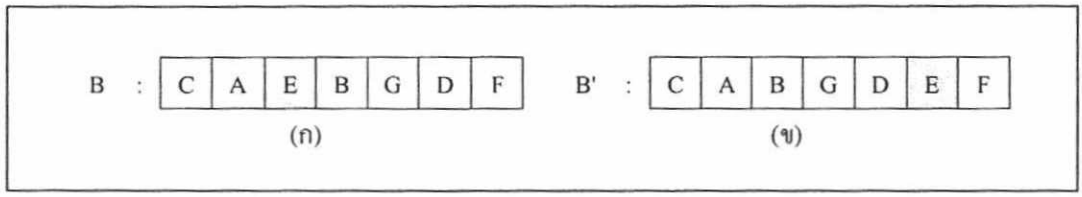
ฟังก์ชันนี้นำเสนอโดย Banzhaf [24] เริ่มต้นการทำงานโดยการสุ่มเลือกเมืองเพื่อทำการสลับกับเมืองในตำแหน่งอื่น จากรูปที่ 3.12 กำหนดตำแหน่งที่สุ่มได้คือ ตำแหน่งที่ 3 และ 6 ดังนั้นให้ทำการสลับเมือง E ในตำแหน่งที่ 3 กับเมือง D ในตำแหน่งที่ 6 โดยที่เมืองอื่นๆ ไม่มีการเปลี่ยนตำแหน่งใดๆ



รูปที่ 3.12 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบแลกเปลี่ยนตำแหน่ง

- ฟังก์ชันการแทรกตำแหน่ง

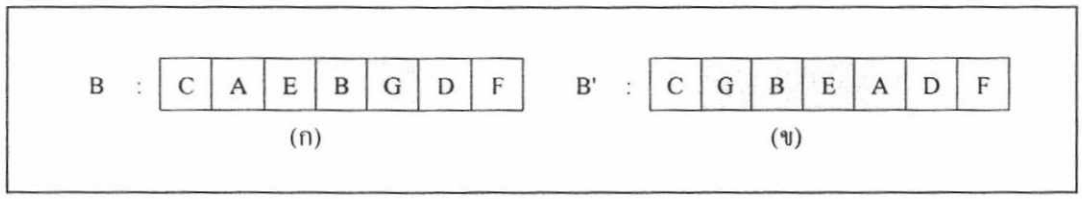
ฟังก์ชันนี้นำเสนอโดย Fogel [25] เริ่มต้นการทำงานโดยการสุ่มเลือกเมืองเพื่อย้ายไปแทรกในตำแหน่งอื่น จากรูปที่ 3.13 กำหนดตำแหน่งที่สุ่มได้คือ ตำแหน่งที่ 3 และ 6 จากนั้นให้ทำการย้ายเมือง E ในตำแหน่งที่ 3 ไปอยู่ในตำแหน่งที่ 6 โดยการแทรกไว้ก่อนหน้าเมือง F ในตำแหน่งที่ 7 ดังนั้น เมืองในตำแหน่งที่ 4-6 เดิม (B-G-D) จะถูกเลื่อนตำแหน่งมายังตำแหน่งที่ 2-5 ตามลำดับด้วย



รูปที่ 3.13 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบแทรกตำแหน่ง

- ฟังก์ชันการกลับค่าอย่างง่าย

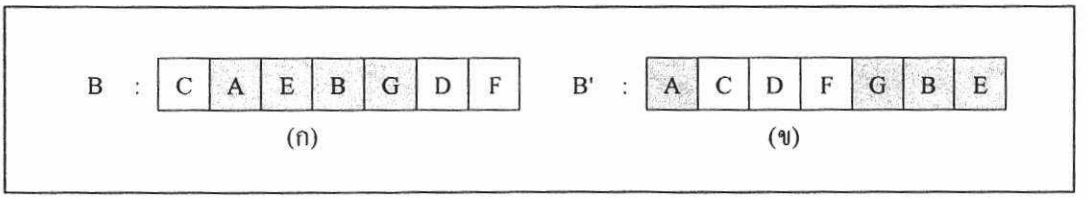
ฟังก์ชันนี้นำเสนอโดย Holland [26] เริ่มต้นการทำงานโดยการสุ่มเลือกช่วงเมืองเพื่อทำการสลับตำแหน่งจากซ้ายไปขวาและให้แทนที่ลงในตำแหน่งเดิม จากรูปที่ 3.14 กำหนดช่วงตำแหน่งที่สุ่มได้คือ ตำแหน่งที่ 2-5 ซึ่งคือ เมือง A-E-B-G จากนั้นให้ทำการสลับตำแหน่งจากซ้ายไปขวา จึงได้ลำดับใหม่คือ G-B-E-A จากนั้นให้ทำการแทนที่ในตำแหน่งเดิม



รูปที่ 3.14 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบกลับค่าอย่างง่าย

- ฟังก์ชันการกลับค่า

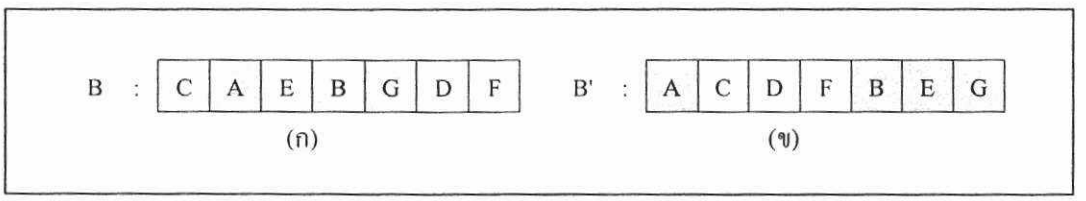
ฟังก์ชันนี้นำเสนอโดย Fogel [27] ซึ่งมีการทำงานเหมือนกับฟังก์ชันการกลับค่าอย่างง่าย เริ่มต้นการทำงานโดยการสุ่มเลือกช่วงเมืองเพื่อทำการสลับตำแหน่งจากซ้ายไปขวา แต่แตกต่างกันตรงที่ว่า ให้ทำการแทนที่ลงในตำแหน่งสุ่มอื่นๆ จากรูปที่ 3.15 กำหนดช่วงตำแหน่งที่สุ่มได้คือ ตำแหน่งที่ 2-5 ซึ่งคือเมือง A-E-B-G จากนั้นให้ทำการสลับตำแหน่งจากซ้ายไปขวา จึงได้ลำดับใหม่คือ G-B-E-A เพื่อทำการแทนที่ในตำแหน่งสุ่มใหม่ กำหนดให้เป็น ตำแหน่ง 5 ดังนั้น เมือง G, B, E และ A จะย้ายไปยังตำแหน่งที่ 5, 6, 7 และ 1 ตามลำดับ



รูปที่ 3.15 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบกลับค่า

- ฟังก์ชันการเรียงสลับ

ฟังก์ชันนี้นำเสนอโดย Ulder, Aarts, Bandelt, van Laarhoven และ Pesch [28] ซึ่งมีการทำงานเหมือนกับฟังก์ชันการกลับค่า เริ่มต้นการทำงานโดยการสุ่มเลือกช่วงเมืองเพื่อทำการสลับตำแหน่ง แต่แตกต่างกันตรงที่ว่า ไม่ได้ทำการสลับตำแหน่งจากซ้ายไปขวา แต่ฟังก์ชันนี้ทำการเรียงสลับ (permute) เมืองเพื่อแทนที่ลงในตำแหน่งสุ่มอื่นๆ จากรูปที่ 3.16 กำหนดช่วงตำแหน่งที่สุ่มได้คือ ตำแหน่งที่ 2-5 ซึ่งคือเมือง A-E-B-G จากนั้นให้ทำการเรียงสลับลำดับเมืองใหม่ ตัวอย่างเช่น B-E-G-A เพื่อทำการแทนที่ในตำแหน่งสุ่มใหม่ กำหนดให้เป็น ตำแหน่ง 5 ดังนั้น เมือง B, E, G และ A จะย้ายไปยังตำแหน่งที่ 5, 6, 7 และ 1 ตามลำดับ



รูปที่ 3.16 การทำงานของโอเปอร์เรเตอร์ผึ้งงานแบบเรียงสลับ

บทที่ 4
การทดสอบการปรับปรุงอัลกอริทึมการผสมพันธุ์ของผึ้งด้วยกลไกการเรียนรู้ด้วยตนเองสำหรับปัญหาออฟติไมเซชัน

การศึกษาในครั้งนี้ได้พัฒนาอัลกอริทึมที่เกิดจากการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมใน 2 แง่มุมคือ วิธีการผสมพันธุ์ระหว่างรัง เพื่อการแก้ปัญหาการผสมพันธุ์ระหว่างผึ้งนางพญากับผึ้งตัวผู้ภายในรังเดียวกัน โดยการแบ่งรังเพื่อให้ผึ้งนางพญาได้มีโอกาสเลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นๆ และวิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งเพื่อการแก้ปัญหาการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้ใหม่ในทุกรอบการทำงาน ด้วยวิธีการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้จากตัวอ่อนผึ้งเพื่อลดเวลาในการประมวลผลเพราะค่าจีโนไทป์และค่าฟิตเนสของตัวอ่อนผึ้งได้ถูกคำนวณไว้ในทุกรอบการทำงานอยู่แล้ว งานวิจัยนี้ได้นำเสนออัลกอริทึมที่เกิดจากการปรับปรุงขั้นตอนการทำงานตามที่กล่าวมา โดยแบ่งเป็น 2 โมเดลเพื่อให้ครอบคลุมกับรูปแบบปัญหาคือ อัลกอริทึมที่นำเสนอเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันเชิงตัวเลขและอัลกอริทึมที่นำเสนอเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันเชิงการจัด ซึ่งได้กล่าวรายละเอียดในบทที่ผ่านมาแล้ว

การทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอ ได้แบ่งเป็น 5 ส่วน ดังนี้

1) การทดสอบเพื่อศึกษาปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

2) การทดสอบเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาและผึ้งตัวผู้ที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่

3) การทดสอบเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาและผึ้งตัวผู้ที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยไม่มีการกำหนดจำนวนของผึ้งนางพญา

4) การทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข โดยได้ทำการทดสอบกับฟังก์ชันมาตรฐานและได้เปรียบเทียบผลการทดลองกับอัลกอริทึมการผสมพันธุ์แบบดั้งเดิม

5) การทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด โดยได้ทำการทดสอบกับปัญหาการเดินทางของพนักงานขาย (travelling salesman problem) และได้เปรียบเทียบกับผลการทดลองในงานวิจัยของ Chen และ Chien [3]

4.1 ข้อมูลที่ใช้ในการทดสอบ

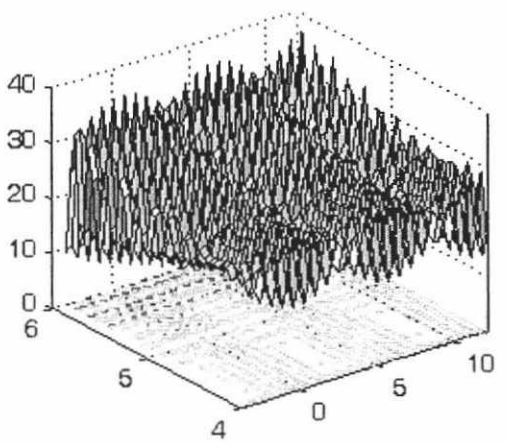
4.1.1 ฟังก์ชันมาตรฐานสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข

1) ฟังก์ชัน Michalewicz

ฟังก์ชันนี้เป็นฟังก์ชันสำหรับการหาค่าสูงสุด (maximization problem) ที่นำเสนอโดย Michalewicz [23] ดังสมการที่ 4.1

$$f(\vec{x}) = 21.5 + x_1 \times \sin(4\pi x_1) + x_2 \times \sin(20\pi x_2) \tag{4.1}$$

เมื่อ x_1 มีค่าอยู่ในช่วง [-3.0, 12.1] และ x_2 มีค่าอยู่ในช่วง [4.1, 5.8] โดยที่มีค่าฟังก์ชันที่สูงที่สุดเท่ากับ 38.850294479 และมีคำตอบที่เหมาะสมที่สุดคือ x_1 มีค่าอยู่ระหว่าง [11.625543700, 11.625545700] และ x_2 มีค่าเท่ากับ 5.725044250 กราฟของฟังก์ชันนี้แสดง ดังรูปที่ 4.1



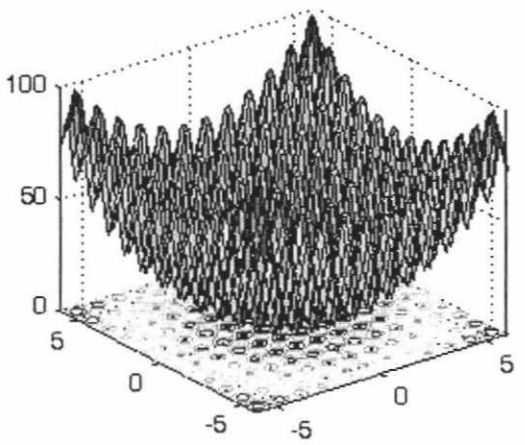
รูปที่ 4.1 กราฟของฟังก์ชัน Michalewicz

2) ฟังก์ชัน Rastrigin

ฟังก์ชันนี้เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด (minimization problem) ที่นำเสนอโดย Törn และ Zilinskas [29] ดังสมการที่ 4.2

$$f(\vec{x}) = 10m + \sum_{i=1}^{m} \left(x_i^2 - 10\cos(2\pi x_i) \right) \tag{4.2}$$

เมื่อ x_i มีค่าอยู่ในช่วง [-5.12, 5.12] และ m คือจำนวนมิติของปัญหา ฟังก์ชันนี้มีค่าฟังก์ชันที่ต่ำที่สุดเท่ากับ 0 และมีคำตอบที่เหมาะสมที่สุดคือ $\vec{x}^*$ เท่ากับ (0, 0, ..., 0) กราฟของฟังก์ชันนี้แสดง ดังรูปที่ 4.2



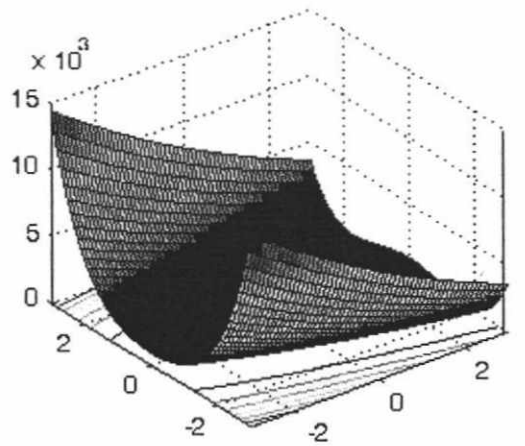
รูปที่ 4.2 กราฟของฟังก์ชัน Rastrigin

3) ฟังก์ชัน Rosenbrocks' valley

ฟังก์ชันนี้เป็นฟังก์ชันการหาค่าต่ำสุดที่นำเสนอโดย De Jong [30] ดังสมการที่ 4.3

$$f(\vec{x}) = \sum_{i=1}^{m-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2\right] \tag{4.3}$$

เมื่อ x_i มีค่าอยู่ในช่วง [-2.048, 2.048] และ m คือจำนวนมิติของปัญหา ฟังก์ชันนี้มีค่าฟังก์ชันที่ต่ำที่สุดเท่ากับ 0 และมีคำตอบที่เหมาะสมที่สุดคือ $\vec{x}^*$ เท่ากับ (1, 1, ..., 1) กราฟของฟังก์ชันนี้แสดงดังรูปที่ 4.3



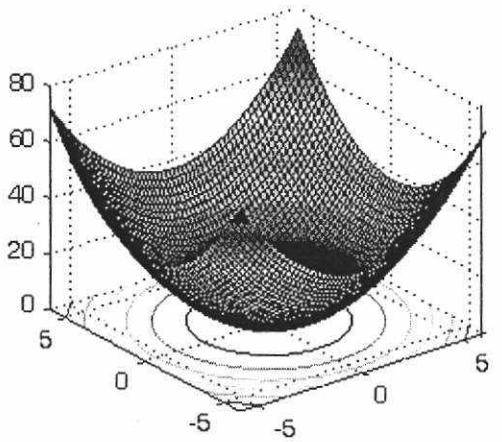
รูปที่ 4.3 กราฟของฟังก์ชัน Rosenbrocks' valley

4) ฟังก์ชัน Sphere

ฟังก์ชันนี้เป็นฟังก์ชันการหาค่าต่ำสุดที่นำเสนอโดย De Jong [30] ดังสมการที่ 4.4

$$f(\vec{x}) = \sum_{i=1}^{m} x_i^2 \tag{4.4}$$

เมื่อ x_i มีค่าอยู่ในช่วง [-5.12, 5.12] และ m คือจำนวนมิติของปัญหา ฟังก์ชันนี้มีค่าฟังก์ชันที่ต่ำที่สุดเท่ากับ 0 และมีคำตอบที่เหมาะสมที่สุดคือ $\vec{x}^*$ เท่ากับ (0, 0, ..., 0) กราฟของฟังก์ชันนี้แสดง ดังรูปที่ 4.4



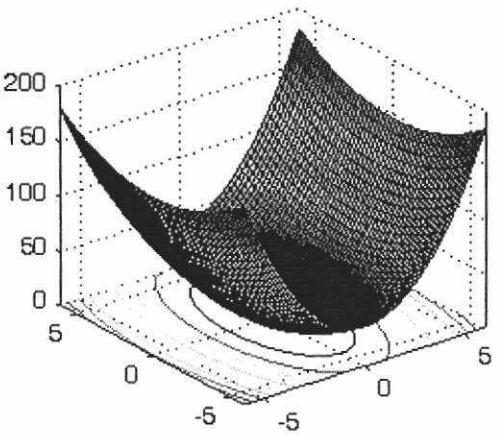
รูปที่ 4.4 กราฟของฟังก์ชัน Sphere

5) ฟังก์ชัน Weighted sphere

ฟังก์ชันนี้เป็นฟังก์ชันสำหรับการหาค่าต่ำสุดที่นำเสนอโดย Pohlheim [31] ดังสมการที่ 4.5

$$f(\vec{x}) = \sum_{i=1}^{m} i^2 x_i^2 \tag{4.5}$$

เมื่อ x_i มีค่าอยู่ในช่วง [-5.12, 5.12] และ m คือจำนวนมิติของปัญหา ฟังก์ชันนี้มีค่าฟังก์ชันที่ต่ำที่สุดเท่ากับ 0 และมีคำตอบที่เหมาะสมที่สุดคือ $\vec{x}^*$ เท่ากับ (0, 0, ..., 0) กราฟของฟังก์ชันนี้แสดง ดังรูปที่ 4.5



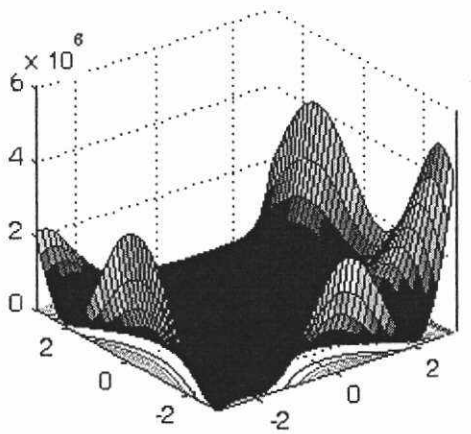
รูปที่ 4.5 กราฟของฟังก์ชัน Weighted sphere

6) ฟังก์ชัน Goldstein-Price

ฟังก์ชันนี้เป็นฟังก์ชันสำหรับการหาค่าต่ำสุดที่นำเสนอโดย Goldstein และ Price [32] ดังสมการที่ 4.6

$$f(\vec{x}) = \left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+16x_1x_2+3x_2^2\right)\right]\times \left[30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right] \quad (4.6)$$

เมื่อ x_1 และ x_2 มีค่าอยู่ในช่วง [-2.0, 2.0] ฟังก์ชันนี้มีค่าฟังก์ชันที่ต่ำที่สุดเท่ากับ 3 และมีคำตอบที่เหมาะสมที่สุดคือ $\vec{x}^*$ เท่ากับ (0, -1) กราฟของฟังก์ชันนี้แสดง ดังรูปที่ 4.6

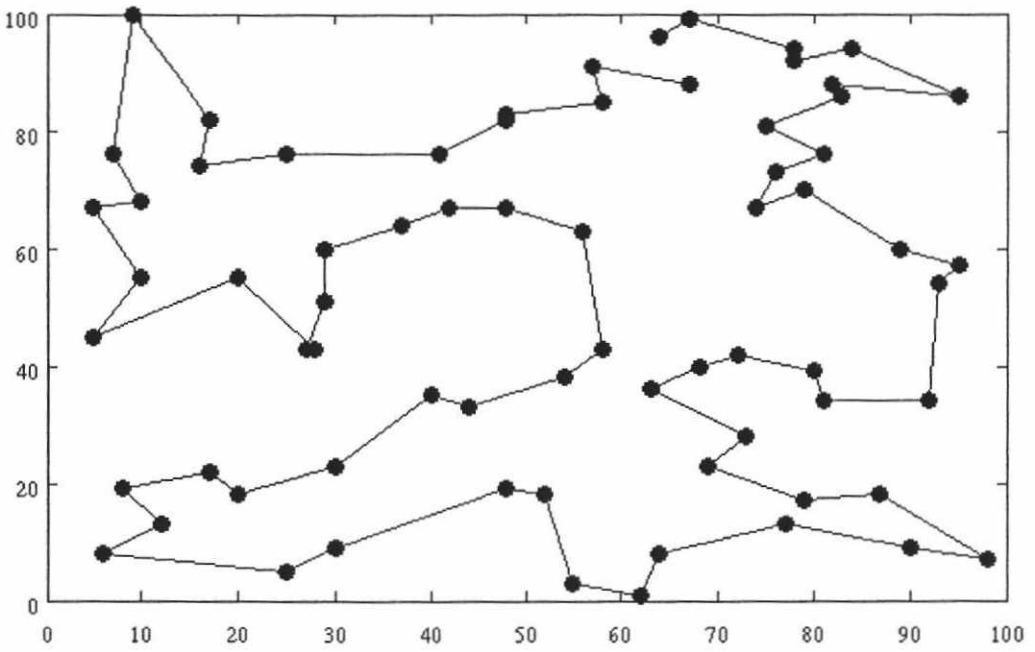


รูปที่ 4.6 กราฟของฟังก์ชัน Goldstein-Price

4.1.2 ชุดข้อมูลมาตรฐานสำหรับปัญหาออฟติไมเซชันเชิงการจัด

1) ข้อมูล ST70

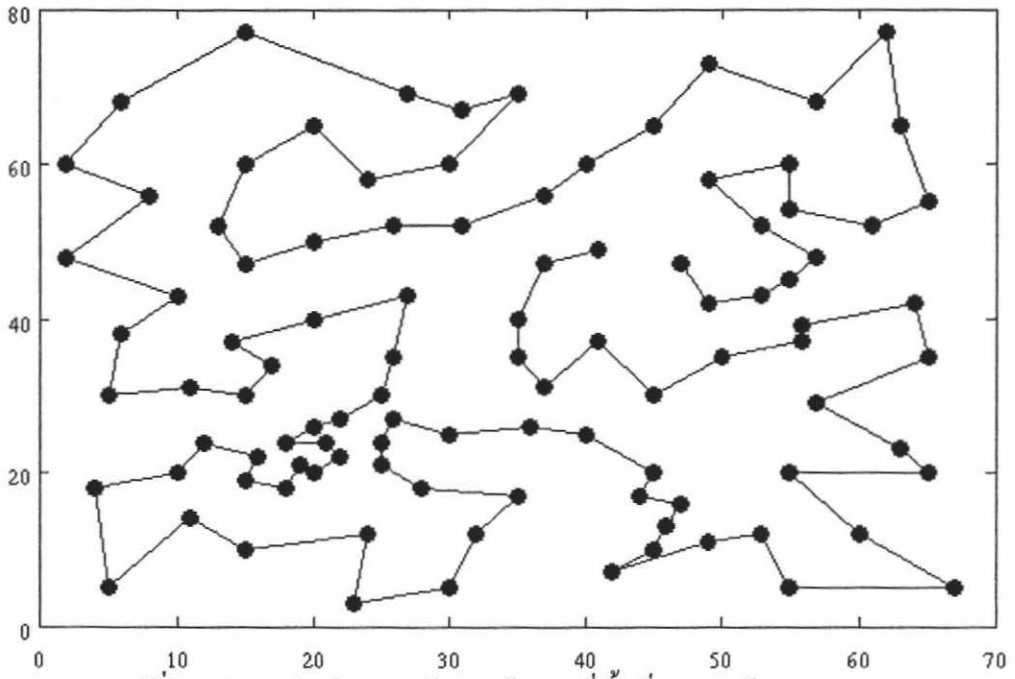
ข้อมูล ST70 [33] เป็นข้อมูลทดสอบมาตรฐานของปัญหาการเดินทางของพนักงานขายแบบสมมาตรที่นำเสนอโดย Smith และ Thompson ข้อมูลนี้มีเมืองทดสอบจำนวน 70 เมือง โดยแต่ละเมืองคำนวณระยะทางระหว่างกันด้วยวิธีการหาระยะทางแบบยูคลิเดียน ระยะทางที่สั้นที่สุดมีค่าเท่ากับ 675 ดังรูปที่ 4.7



รูปที่ 4.7 ตำแหน่งเมืองและข้อมูลเส้นทางที่สั้นที่สุดของข้อมูล ST70

2) ข้อมูล EIL101

ข้อมูล EIL101 [33] เป็นข้อมูลทดสอบมาตรฐานของปัญหาการเดินทางของพนักงานขายแบบสมมาตรที่นำเสนอโดย Christofides และ Eilon ข้อมูลนี้มีเมืองทดสอบจำนวน 101 เมือง โดยแต่ละเมืองคำนวณระยะทางระหว่างกันด้วยวิธีการหาระยะทางแบบยูคลิเดียน ระยะทางที่สั้นที่สุดมีค่าเท่ากับ 629 ดังรูปที่ 4.8



รูปที่ 4.8 ตำแหน่งเมืองและข้อมูลเส้นทางที่สั้นที่สุดของข้อมูล EIL101

4.2 เครื่องมือที่ใช้ในการวัดประสิทธิภาพ

1) ค่าผลลัพธ์ (output) คือ ค่าข้อมูลที่ได้จากการทดลอง โดยได้ทำการเปรียบเทียบทั้งค่าคำตอบที่ดีที่สุดและค่าคำตอบที่แย่ที่สุด แทนด้วย y_{best} และ y_{worst} ตามลำดับ

2) ค่าเฉลี่ย (mean) คือ ผลรวมของค่าผลลัพธ์ที่ได้จากการทดลองแล้วหารด้วยจำนวนครั้งในการทดลอง เพื่อหาตัวแทนของค่าผลลัพธ์ที่ได้จากการทำการทดลองซ้ำหลายๆ ครั้ง

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{4.7}$$

เมื่อ $\bar{y}$ คือค่าเฉลี่ยของค่าผลลัพธ์ที่ได้จากการทดลอง

y_i คือค่าผลลัพธ์ที่ได้จากการทดลองที่ i

n คือจำนวนข้อมูลทั้งหมด

3) ส่วนเบี่ยงเบนมาตรฐาน (standard deviation: S.D.) เป็นค่าสถิติที่ใช้วัดการกระจายของข้อมูล

$$S.D. = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}} \tag{4.8}$$

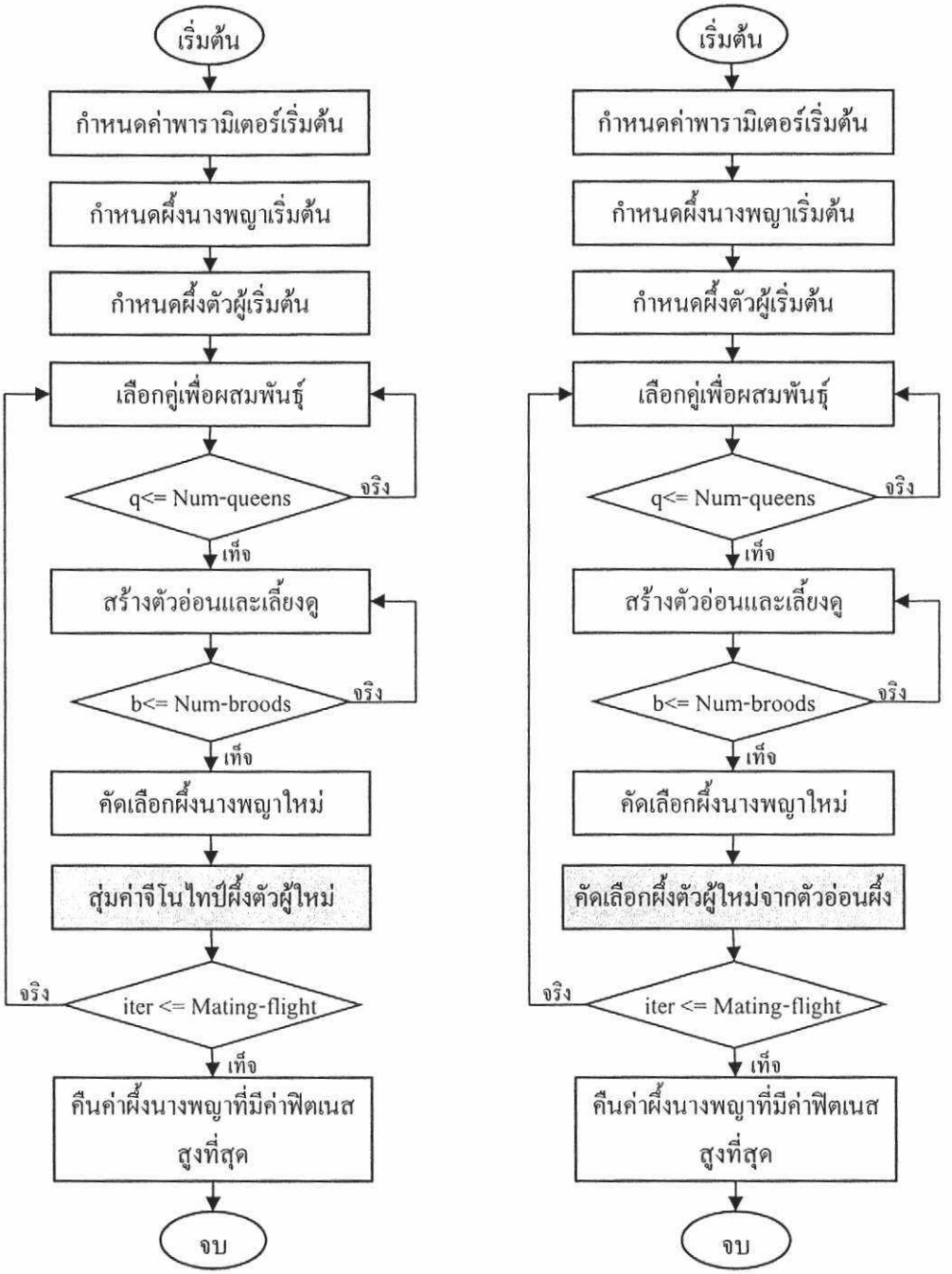
เมื่อ $\bar{y}$ คือค่าเฉลี่ยของค่าผลลัพธ์ที่ได้จากการทดลอง

y_i คือค่าผลลัพธ์ที่ได้จากการทดลองที่ i

n คือจำนวนข้อมูลทั้งหมด

4.3 การทดสอบเพื่อศึกษาปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง

การทดลองที่ 1 เป็นการทดลองเพื่อศึกษาปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยการทดลองนี้ได้ทำการเปรียบเทียบการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมโดยที่มีการปรับเปลี่ยนกลไกการคัดเลือกผึ้งตัวผู้แบบดั้งเดิมให้เป็นการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งแบบที่นำเสนอ จากรูปที่ 4.9 แสดงการเปรียบเทียบโครงสร้างการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (MBO) และอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง (MBO-newDrone) ขั้นตอนที่มีการแรเงาไว้คือขั้นตอนที่มีความแตกต่างกันระหว่างทั้งสองโมเดล ซึ่งเป็นปัจจัยที่ต้องการศึกษา



(ก) โครงสร้างการทำงานของอัลกอริทึม MBO (ข) โครงสร้างการทำงานของอัลกอริทึม MBO-newDrone

รูปที่ 4.9 เปรียบเทียบโครงสร้างการทำงานระหว่างอัลกอริทึม MBO และ
อัลกอริทึม MBO-newDrone

4.3.1 การกำหนดค่าพารามิเตอร์

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง มีการกำหนดค่าพารามิเตอร์ทั้งหมดเหมือนกัน ดังตารางที่ 4.1 จากตารางจะเห็นว่า มีการกำหนดค่าจำนวนของผึ้งนางพญาไว้เป็นค่าคงที่จำนวน 5 ค่า เนื่องจากจำนวนของผึ้งนางพญามีความสัมพันธ์กับปัญหาที่ทดสอบ การกำหนดจำนวนของผึ้งนางพญามากหรือน้อยขึ้นอยู่กับความยากง่ายของปัญหา อย่างไรก็ตาม ถ้ากำหนดมากเกินไปจะทำให้มีการคำนวณเพิ่มมากขึ้นด้วย ในการทดลองนี้จึงได้มีการกำหนดจำนวนผึ้งนางพญาไว้หลายค่าเพื่อให้ครอบคลุมกับปัญหาที่ใช้ในการทดสอบ

ตารางที่ 4.1 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม MBO และอัลกอริทึม MBO-newDrone

พารามิเตอร์	MBO	MBO-newDrone
จำนวนของผึ้งนางพญา	1, 10, 20, 30, 40	1, 10, 20, 30, 40
จำนวนของผึ้งตัวผู้	200	200
จำนวนของตัวอ่อนผึ้ง	500	500
ขนาดของถุงเก็บสเปิร์ม	100	100
ค่าความน่าจะเป็นในการมิวเตชัน	0.2	0.2
จำนวนรอบการทำงาน	400	400

4.3.2 ผลการทดลอง

การทดลองที่ 1 ได้ทำการทดสอบอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งกับฟังก์ชันออฟติไมเซชันจำนวน 6 ฟังก์ชัน คือ

1) ฟังก์ชัน Michalewicz

2) ฟังก์ชัน Rastrigin

3) ฟังก์ชัน Rosenbrocks' valley

4) ฟังก์ชัน Sphere

5) ฟังก์ชัน Weighted sphere

6) ฟังก์ชัน Goldstein-Price

โดยในแต่ละฟังก์ชันได้มีการสุ่มข้อมูลด้วยฟังก์ชัน rand() ของโปรแกรม MATLAB เพื่อใช้เป็นประชากรเริ่มต้นของอัลกอริทึมที่ทดสอบจำนวน 10 ชุด ดังนั้น จากการกำหนดค่าพารามิเตอร์ ดังตารางที่ 4.1 อัลกอริทึมแต่ละตัวมีการรันจำนวน 50 ครั้งต่อหนึ่งฟังก์ชัน โดยแบ่งเป็น การกำหนดผึ้งนางพญา 1 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา 10 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา

20 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา 30 ตัว มีการรันจำนวน 10 ครั้งและผึ้งนางพญา 40 ตัว มีการรันจำนวน 10 ครั้ง ดังนั้น แต่ละอัลกอริทึมจึงมีการรันทั้งหมด 300 ครั้ง (50 ครั้ง × 6 ฟังก์ชัน)

เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบคือ Intel ® core ™ i7 2.8 GHz (3 GB of RAM) และใช้โปรแกรม MATLAB เวอร์ชัน 7.6.0.324 ผลการทดลองที่ได้มีการเปรียบเทียบประสิทธิภาพของทั้งสองอัลกอริทึมด้วย 5 ตัวชี้วัด คือ 1) ค่าคำตอบที่ดีที่สุด (y_{best}) 2) ค่าคำตอบที่แย่ที่สุด (y_{worst}) 3) ค่าเฉลี่ยของคำตอบทั้งหมด ($\bar{y}$) 4) ส่วนเบี่ยงเบนมาตรฐาน (S.D.) และ 5) เวลา (time) ดังตารางที่ 4.2 และตารางที่ 4.3 โดยแต่ละฟังก์ชันมีการแสดงผลการทดลองแบ่งเป็น 6 ส่วนคือ

1) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 1 ตัว

2) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 10 ตัว

3) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 20 ตัว

4) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 30 ตัว

5) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 40 ตัว

6) ผลการทดลองจากการรันจำนวน 50 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 1, 10, 20, 30 และ 40 ตัว

ตารางที่ 4.2 ผลการทดลองของอัลกอริทึม MBO

ชื่อฟังก์ชัน	y_{best}	y_{worst}	$\bar{y}$	S.D.	time
1) Michalewicz					
1	38.8467061922	38.8221357062	38.8435623860	0.0075639503	0 : 02 : 15
10	38.8479863775	38.8451574963	38.8462690132	0.0009611662	0 : 03 : 59
20	38.8479863775	38.8429356528	38.8467167687	0.0015351384	0 : 06 : 06
30	38.8479863775	38.8221357062	38.8416788098	0.0095241986	0 : 08 : 29
40	38.8478852297	38.8308819157	38.8426697789	0.0057158746	0 : 11 : 06
ค่าเฉลี่ย	**38.8479863775**	**38.8221357062**	**38.8441793513**	**0.0061479250**	**0 : 06 : 23**
2) Rastrigin					
1	0.0001892008	0.0009459972	0.0003405608	0.0002325774	0 : 02 : 06
10	0.0000000000	0.0009459972	0.0003594795	0.0003830993	0 : 03 : 44
20	0.0000000000	0.0015135926	0.0004919187	0.0004901353	0 : 05 : 44
30	0.0000000000	0.0037838681	0.0007000261	0.0011766232	0 : 08 : 02
40	0.0000000000	0.0018919660	0.0009838309	0.0005185177	0 : 10 : 38
ค่าเฉลี่ย	**0.0000000000**	**0.0037838681**	**0.0005751632**	**0.0006664008**	**0 : 06 : 03**

ตารางที่ 4.2 ผลการทดลองของอัลกอริทึม MBO (ต่อ)

ชื่อฟังก์ชัน	y_{best}	y_{worst}	$\bar{y}$	S.D.	time
3) Rosenbrocks' valley					
1	0.0000009538	0.0000238987	0.0000051569	0.0000069992	0 : 01 : 56
10	0.0000000000	0.0000953674	0.0000133576	0.0000296803	0 : 03 : 35
20	0.0000000000	0.0000238987	0.0000054436	0.0000072105	0 : 05 : 36
30	0.0000009538	0.0000344501	0.0000082203	0.0000115631	0 : 07 : 54
40	0.0000000000	0.0000778443	0.0000172498	0.0000247243	0 : 10 : 28
ค่าเฉลี่ย	**0.0000000000**	**0.0000953674**	**0.0000098856**	**0.0000184355**	**0 : 05 : 54**
4) Sphere					
1	0.0000000000	0.0000038147	0.0000011444	0.0000011723	0 : 02 : 06
10	0.0000000000	0.0000152588	0.0000054359	0.0000053957	0 : 03 : 45
20	0.0000000000	0.0000238419	0.0000048637	0.0000072274	0 : 05 : 45
30	0.0000000000	0.0000047684	0.0000020027	0.0000013820	0 : 08 : 02
40	0.0000009537	0.0000123978	0.0000057220	0.0000036523	0 : 10 : 35
ค่าเฉลี่ย	**0.0000000000**	**0.0000238419**	**0.0000038338**	**0.0000046501**	**0 : 06 : 03**
5) Weighted sphere					
1	0.0000009537	0.0000276566	0.0000069618	0.0000085068	0 : 02 : 10
10	0.0000009537	0.0000152588	0.0000062943	0.0000048670	0 : 03 : 50
20	0.0000009537	0.0000276566	0.0000091553	0.0000096337	0 : 05 : 52
30	0.0000009537	0.0000152588	0.0000055313	0.0000049001	0 : 08 : 12
40	0.0000009537	0.0000238419	0.0000075340	0.0000070719	0 : 10 : 48
ค่าเฉลี่ย	**0.0000009537**	**0.0000276566**	**0.0000070953**	**0.0000070586**	**0 : 06 : 10**
6) Goldstein-Price					
1	3.0000000000	3.0004467901	3.0002267636	0.0002107386	0 : 02 : 04
10	3.0000000000	3.0014791604	3.0004263183	0.0004728057	0 : 03 : 49
20	3.0000000000	3.0009606348	3.0003158873	0.0002942444	0 : 05 : 55
30	3.0000000000	3.0004467901	3.0000859347	0.0001813459	0 : 08 : 20
40	3.0000000000	3.0016525937	3.0007899476	0.0005845918	0 : 11 :03
ค่าเฉลี่ย	**3.0000000000**	**3.0016525937**	**3.0003689703**	**0.0004379715**	**0 : 06 : 14**

ตารางที่ 4.3 ผลการทดลองของอัลกอริทึม MBO-newDrone

ชื่อฟังก์ชัน	y_{best}	y_{worst}	$\bar{y}$	S.D.	time
1) Michalewicz					
1	38.8479863775	38.8460333996	38.8474004842	0.0009433780	0 : 02 : 01
10	38.8479863775	38.8448543621	38.8470164286	0.0011095155	0 : 03 : 20
20	38.8479863775	38.8448543621	38.8465214754	0.0009087426	0 : 04 : 45
30	38.8479863775	38.8451574963	38.8471342199	0.0009522720	0 : 06 : 11
40	38.8479863775	38.8288018259	38.8425397566	0.0075584838	0 : 07 : 36
ค่าเฉลี่ย	**38.8479863775**	**38.8288018259**	**38.8461224729**	**0.0038155704**	**0 : 04 : 47**
2) Rastrigin					
1	0.0000000000	0.0003784017	0.0001702808	0.0001656633	0 : 01 : 48
10	0.0000000000	0.0007567963	0.0002459604	0.0002004279	0 : 02 : 57
20	0.0001892008	0.0009459972	0.0005297595	0.0003655672	0 : 04 : 12
30	0.0001892008	0.0009459972	0.0004162397	0.0002931054	0 : 05 : 26
40	0.0001892008	0.0015135926	0.0005865191	0.0004672867	0 : 06 : 41
ค่าเฉลี่ย	**0.0000000000**	**0.0015135926**	**0.0003897519**	**0.0003448854**	**0 : 04 : 13**
3) Rosenbrocks' valley					
1	0.0000000000	0.0000009538	0.0000005723	0.0000004925	0 : 01 : 41
10	0.0000000000	0.0000038162	0.0000011447	0.0000014780	0 : 02 : 51
20	0.0000000000	0.0000152821	0.0000029595	0.0000050023	0 : 04 : 04
30	0.0000000000	0.0000238987	0.0000037255	0.0000072867	0 : 05 : 18
40	0.0000000000	0.0000085904	0.0000020996	0.0000034504	0 : 06 : 32
ค่าเฉลี่ย	**0.0000000000**	**0.0000238987**	**0.0000021003**	**0.0000042818**	**0 : 04 : 05**
4) Sphere					
1	0.0000000000	0.0000019073	0.0000007629	0.0000006032	0 : 01 : 49
10	0.0000000000	0.0000047684	0.0000020981	0.0000015443	0 : 02 : 59
20	0.0000000000	0.0000047684	0.0000025749	0.0000019622	0 : 04 : 14
30	0.0000000000	0.0000038147	0.0000007629	0.0000012556	0 : 05 : 27
40	0.0000000000	0.0000095367	0.0000024796	0.0000030557	0 : 06 : 41
ค่าเฉลี่ย	**0.0000000000**	**0.0000095367**	**0.0000017357**	**0.0000019714**	**0 : 04 : 14**

ตารางที่ 4.3 ผลการทดลองของอัลกอริทึม MBO-newDrone (ต่อ)

ชื่อฟังก์ชัน	y_{best}	y_{worst}	$\bar{y}$	S.D.	time
5) Weighted sphere					
1	0.0000000000	0.0000076294	0.0000026703	0.0000026521	0 : 01 : 50
10	0.0000000000	0.0000085831	0.0000031471	0.0000034838	0 : 03 : 01
20	0.0000000000	0.0000162125	0.0000076294	0.0000058616	0 : 04 : 10
30	0.0000000000	0.0000076294	0.0000021935	0.0000025051	0 : 05 : 26
40	0.0000000000	0.0000152588	0.0000035286	0.0000045858	0 : 06 : 50
ค่าเฉลี่ย	**0.0000000000**	**0.0000162125**	**0.0000038338**	**0.0000043276**	**0 : 04 : 15**
6) Goldstein-Price					
1	3.0000000000	3.0004467901	3.0001546375	0.0002076409	0 : 01 : 45
10	3.0000000000	3.0008559803	3.0002642965	0.0002826912	0 : 02 : 57
20	3.0000000000	3.0014791604	3.0003470101	0.0004340190	0 : 04 : 16
30	3.0000000000	3.0009628051	3.0004122369	0.0003152421	0 : 05 : 34
40	3.0000000000	3.0016525937	3.0004844779	0.0004805035	0 : 06 :53
ค่าเฉลี่ย	**3.0000000000**	**3.0016525937**	**3.0003325318**	**0.0003623757**	**0 : 04 : 17**

4.3.3 สรุปผลการทดลอง

การทดลองที่ 1 ได้ทำการทดสอบอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งกับฟังก์ชันออฟติไมเซชันจำนวน 6 ฟังก์ชัน จากผลการทดลอง สามารถสรุปได้ดังนี้

1) ฟังก์ชัน Michalewicz

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งสามารถค้นหาคำตอบที่ดีที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่ดีที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20 และ 30 ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งค้นหาคำตอบที่ดีที่สุดได้ครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลอง โดยที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยสูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 0.02% และ 0.01% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.10(ก) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งสูงกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Michalewicz เป็นฟังก์ชันสำหรับ

การหาค่าสูงสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีดั้งเดิม นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 25.07% อีกด้วย

2) ฟังก์ชัน Rastrigin

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20, 30 และ 40 ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 1 และ 10 โดยที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 60.00% และ 32.24% ตามลำดับ โดยมีค่าส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.10(ข) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Rastrigin เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีดั้งเดิม นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 30.30% อีกด้วย

3) ฟังก์ชัน Rosenbrocks' valley

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20 และ 40 ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลอง โดยที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 74.94% และ 78.75% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.10(ค) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Rosenbrocks' valley เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีดั้งเดิม นอกจากนี้

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 30.79% อีกด้วย

4) ฟังก์ชัน Sphere

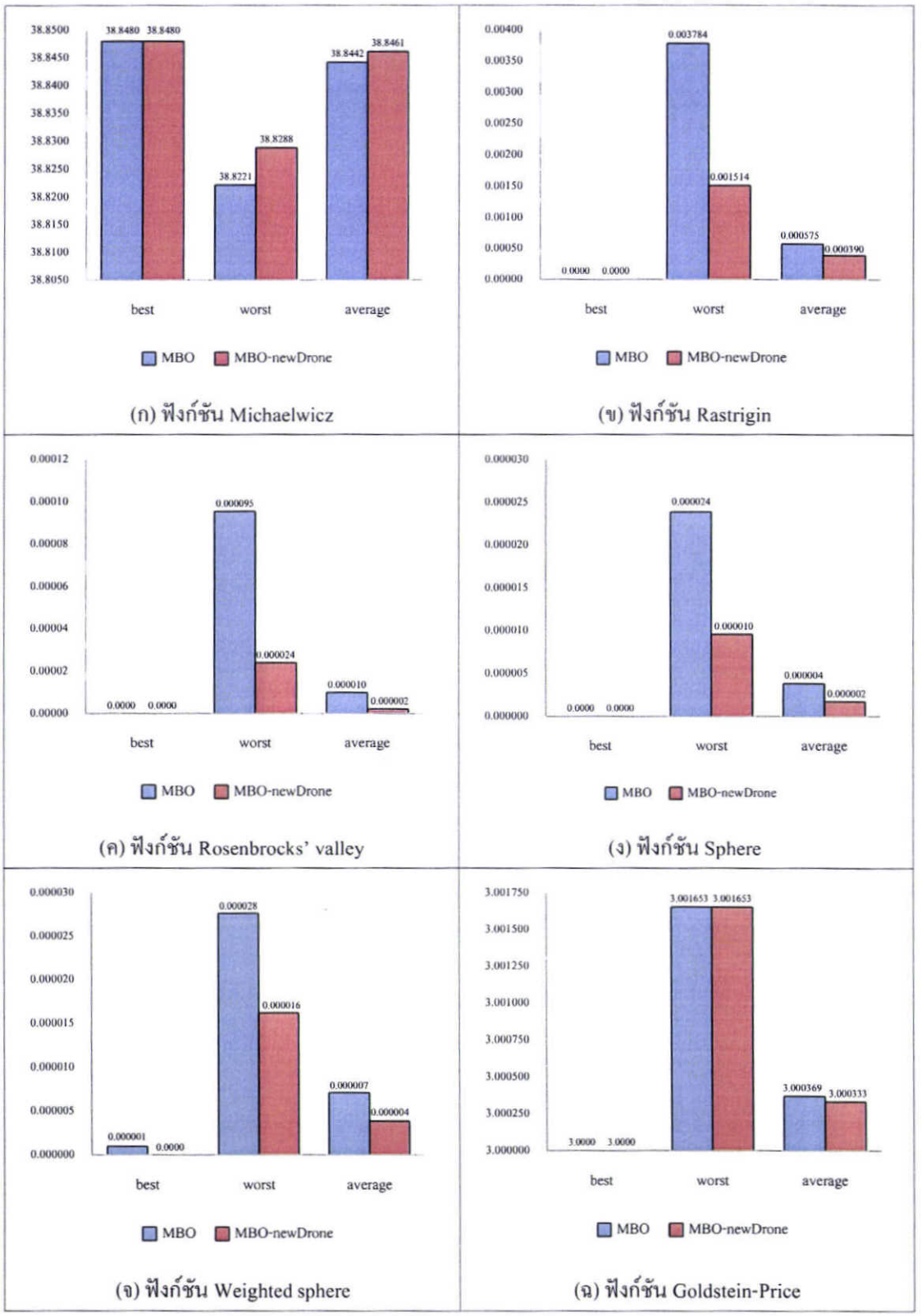
อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 1, 10, 20 และ 30 ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลอง โดยที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 60.00% และ 54.73% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.10(ง) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Sphere เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีดั้งเดิม นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 30.03% อีกด้วย

5) ฟังก์ชัน Weighted sphere

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุดในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลองและยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 41.38% และ 45.97% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย อย่างไรก็ตามอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมมีความผิดพลาดในการค้นหาคำตอบที่เหมาะสมที่สุดเพียงเล็กน้อยเท่านั้น จากรูปที่ 4.10(จ) แท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งต่ำกว่าเส้นกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Weighted sphere เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดีขึ้นกว่าวิธีดั้งเดิม นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 31.08% อีกด้วย

6) ฟังก์ชัน Goldstein-Price

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 5 ค่าจำนวนผึ้งนางพญาที่ทดลองและได้ค่าคำตอบที่แย่ที่สุดเท่ากันด้วย โดยที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งได้ค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง 0.001% และได้ค่าส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย ซึ่งถือว่าค่าคำตอบเฉลี่ยมีความแตกต่างกันน้อยมาก จากรูปที่ 4.10(ฉ) แท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งสูงใกล้เคียงกันกับแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่นำเสนอนี้มีผลต่อการค้นหาคำตอบที่เหมาะสมที่สุดของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมเล็กน้อยสำหรับฟังก์ชันทดสอบนี้ อย่างไรก็ตาม แม้ว่าคำตอบของทั้งสองอัลกอริทึมมีความแตกต่างกันเล็กน้อย แต่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งได้ช่วยลดเวลาในการประมวลผลลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมถึง 31.28%



รูปที่ 4.10 กราฟแสดงการเปรียบเทียบผลการทดลองที่ 1 ระหว่างอัลกอริทึม MBO และอัลกอริทึม MBO-newDrone

4.4 การทดสอบเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรัง (แบบกำหนดจำนวนผึ้งนางพญา)

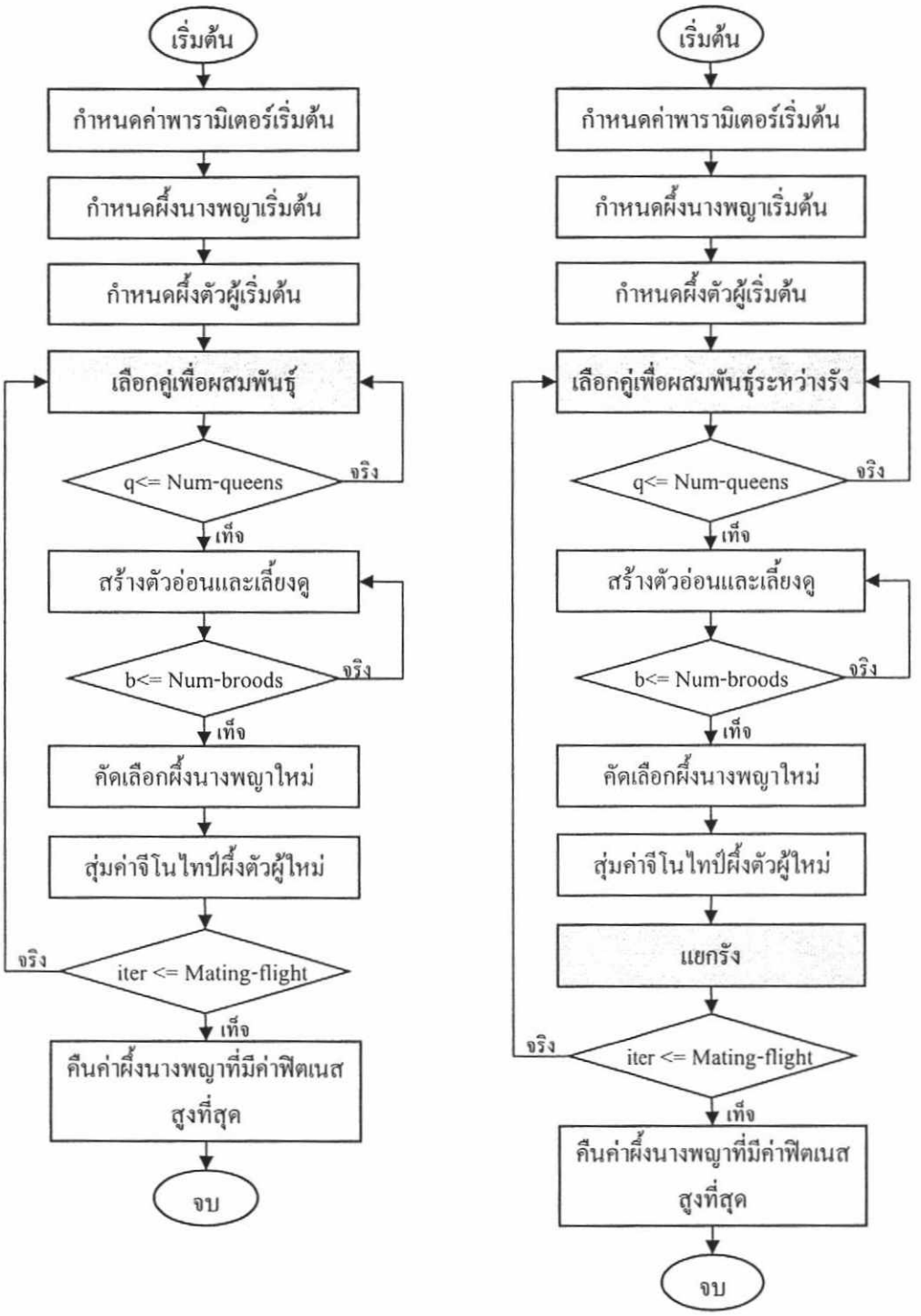
การทดลองที่ 2 เป็นการทดลองเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยการทดลองนี้ได้เปรียบเทียบการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรัง โดยได้มีการปรับปรุงให้มีการแบ่งรัง แต่ละรังมีผึ้งนางพญาหนึ่งตัว ผึ้งตัวผู้ทุกตัวจะถูกจัดให้เป็นสมาชิกของรังใดรังหนึ่ง จากนั้นผึ้งนางพญาแต่ละตัวจะเลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นๆ ทั้งนี้ได้มีการกำหนดจำนวนของผึ้งนางพญาไว้เป็นค่าคงที่ จากรูปที่ 4.11 แสดงการเปรียบเทียบโครงสร้างการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (MBO) และอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนของผึ้งนางพญาเป็นค่าคงที่ (MBO-fixSwarming) ขั้นตอนที่มีการแรเงาไว้คือ ขั้นตอนที่มีความแตกต่างกันระหว่างทั้งสองโมเดล ซึ่งเป็นปัจจัยที่ต้องการศึกษา

4.4.1 การกำหนดค่าพารามิเตอร์

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนของผึ้งนางพญาเป็นค่าคงที่ ได้มีการกำหนดค่าพารามิเตอร์ทั้งหมดเหมือนกัน ดังตารางที่ 4.4 จากตารางจะเห็นว่า อัลกอริทึมทั้งสองมีการกำหนดค่าจำนวนของผึ้งนางพญาไว้เป็นค่าคงที่จำนวน 5 ค่า เหมือนกับในการทดลองที่ผ่านมา เนื่องจากจำนวนของผึ้งนางพญามีความสัมพันธ์กับปัญหาที่ทดสอบ การกำหนดจำนวนของผึ้งนางพญามากน้อยขึ้นอยู่กับความยากง่ายของปัญหา อย่างไรก็ตาม ถ้ากำหนดมากเกินไปอาจทำให้มีการคำนวณมากขึ้นด้วย ในการทดลองนี้จึงได้มีการกำหนดจำนวนผึ้งนางพญาไว้หลายค่าเพื่อให้ครอบคลุมกับปัญหาที่ทดสอบ

ตารางที่ 4.4 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม MBO และอัลกอริทึม MBO-fixSwarming

พารามิเตอร์	MBO	MBO-fixSwarming
จำนวนของผึ้งนางพญา	1, 10, 20, 30, 40	1, 10, 20, 30, 40
จำนวนของผึ้งตัวผู้	200	200
จำนวนของตัวอ่อนผึ้ง	500	500
ขนาดของถุงเก็บสเปิร์ม	100	100
ค่าความน่าจะเป็นในการมิวเตชัน	0.2	0.2
จำนวนรอบการทำงาน	400	400



(ก) โครงสร้างการทำงานของอัลกอริทึม MBO (ข) โครงสร้างการทำงานของอัลกอริทึม MBO-fixSwarming

รูปที่ 4.11 เปรียบเทียบโครงสร้างการทำงานระหว่างอัลกอริทึม MBO และอัลกอริทึม MBO-fixSwarming

4.4.2 ผลการทดลอง

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ ได้ทำการทดสอบกับฟังก์ชันออฟติไมเซชันจำนวน 6 ฟังก์ชัน คือ

1) ฟังก์ชัน Michalewicz

2) ฟังก์ชัน Rastrigin

3) ฟังก์ชัน Rosenbrocks' valley

4) ฟังก์ชัน Sphere

5) ฟังก์ชัน Weighted sphere

6) ฟังก์ชัน Goldstein-Price

โดยได้ใช้ชุดข้อมูลเริ่มต้นชุดเดียวกันกับการทดลองที่ 1 เพื่อให้สามารถเปรียบเทียบผลการทดลองร่วมกันได้ โดยชุดข้อมูลดังกล่าวเกิดจากการสุ่มข้อมูลด้วยฟังก์ชัน rand() ของโปรแกรม MATLAB เพื่อใช้เป็นประชากรเริ่มต้นของอัลกอริทึมที่ทดสอบจำนวน 10 ชุด จากการกำหนดค่าพารามิเตอร์ดังตารางที่ 4.4 อัลกอริทึมแต่ละตัวมีการรันจำนวน 50 ครั้งต่อหนึ่งฟังก์ชัน โดยแบ่งเป็นการกำหนดผึ้งนางพญา 1 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา 10 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา 20 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา 30 ตัว มีการรันจำนวน 10 ครั้งและผึ้งนางพญา 40 ตัว มีการรันจำนวน 10 ครั้ง ดังนั้น แต่ละอัลกอริทึมจึงมีการรันทั้งหมด 300 ครั้ง (50 ครั้ง × 6 ฟังก์ชัน) ทั้งนี้ ผลการทดลองของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม ได้นำมาจากการทดลองที่ 1 (ตารางที่ 4.2) ในการทดลองนี้จึงได้ทำการรันเฉพาะอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่เท่านั้น

เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบ คือ Intel ® core ™ i7 2.8 GHz (3 GB of RAM) และใช้โปรแกรม MATLAB เวอร์ชัน 7.6.0.324 ผลการทดลองที่ได้มีการเปรียบเทียบประสิทธิภาพของทั้งสองอัลกอริทึมด้วย 5 ตัวชี้วัด คือ 1) ค่าคำตอบที่ดีที่สุด (y_{best}) 2) ค่าคำตอบที่แย่ที่สุด (y_{worst}) 3) ค่าเฉลี่ยของคำตอบทั้งหมด ($\bar{y}$) 4) ส่วนเบี่ยงเบนมาตรฐาน (S.D.) และ 5) เวลา (time) ดังตารางที่ 4.5 โดยมีการแสดงผลการทดลองแบ่งเป็น 6 ส่วนคือ

1) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 1 ตัว

2) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 10 ตัว

3) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 20 ตัว

4) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 30 ตัว

5) ผลการทดลองจากการรันจำนวน 10 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 40 ตัว

6) ผลการทดลองจากการรันจำนวน 50 ครั้ง เมื่อกำหนดจำนวนผึ้งนางพญา 1, 10, 20, 30 และ 40 ตัว

ตารางที่ 4.5 ผลการทดลองของอัลกอริทึม MBO-fixSwarming

ชื่อฟังก์ชัน	y_{best}	y_{worst}	$\bar{y}$	S.D.	time
1) Michalewicz					
1	38.8463365338	38.8448543621	38.8459188458	0.0005789448	0 : 02 : 15
10	38.8479863775	38.8448543621	38.8471848753	0.0011216025	0 : 06 : 10
20	38.8479863775	38.8412524228	38.8465989577	0.0020984733	0 : 10 : 30
30	38.8479863775	38.8460333996	38.8474373881	0.0007617577	0 : 14 : 49
40	38.8479863775	38.8417566154	38.8464439644	0.0018013729	0 : 19 : 09
ค่าเฉลี่ย	**38.8479863775**	**38.8412524228**	**38.8467168063**	**0.0014500654**	**0 : 10 : 35**
2) Rastrigin					
1	0.0001892008	0.0024595614	0.0010784257	0.0009227827	0 : 02 : 04
10	0.0000000000	0.0009459972	0.0003784003	0.0002820413	0 : 05 : 43
20	0.0000000000	0.0018919660	0.0003405573	0.0005696880	0 : 09 : 43
30	0.0000000000	0.0017027651	0.0006432758	0.0005584068	0 : 13 : 44
40	0.0000000000	0.0032162727	0.0007000268	0.0009524568	0 : 17 : 45
ค่าเฉลี่ย	**0.0000000000**	**0.0032162727**	**0.0006281372**	**0.0007256775**	**0 : 09 : 48**
3) Rosenbrocks' valley					
1	0.0000009538	0.0000085904	0.0000020037	0.0000024830	0 : 01 : 55
10	0.0000009538	0.0000085904	0.0000022899	0.0000025134	0 : 05 : 04
20	0.0000009538	0.0000778443	0.0000102652	0.0000238701	0 : 08 : 28
30	0.0000000000	0.0000344501	0.0000050673	0.0000107132	0 : 11 : 53
40	0.0000000000	0.0000344501	0.0000073617	0.0000120513	0 : 15 : 18
ค่าเฉลี่ย	**0.0000000000**	**0.0000778443**	**0.0000053975**	**0.0000128324**	**0 : 08 : 32**
4) Sphere					
1	0.0000000000	0.0000019073	0.0000006676	0.0000006437	0 : 02 : 06
10	0.0000000000	0.0000085831	0.0000033379	0.0000025132	0 : 05 : 47
20	0.0000000000	0.0000095367	0.0000041962	0.0000032791	0 : 09 : 50
30	0.0000000000	0.0000123978	0.0000031471	0.0000038424	0 : 13 : 51
40	0.0000000000	0.0000171661	0.0000034332	0.0000050504	0 : 17 : 58
ค่าเฉลี่ย	**0.0000000000**	**0.0000171661**	**0.0000029564**	**0.0000034748**	**0 : 09 : 54**

ตารางที่ 4.5 ผลการทดลองของอัลกอริทึม MBO-fixSwarming (ต่อ)

ชื่อฟังก์ชัน	y_{best}	y_{worst}	$\bar{y}$	S.D.	time
5) Weighted sphere					
1	0.0000000000	0.0000076294	0.0000024796	0.0000026673	0 : 02 : 10
10	0.0000000000	0.0000190735	0.0000061035	0.0000057079	0 : 06 : 02
20	0.0000000000	0.0000123978	0.0000049591	0.0000045582	0 : 10 : 14
30	0.0000000000	0.0000190735	0.0000069618	0.0000064219	0 : 14 : 17
40	0.0000000000	0.0000276566	0.0000062943	0.0000088234	0 : 18 : 27
ค่าเฉลี่ย	**0.0000000000**	**0.0000276566**	**0.0000053596**	**0.0000059600**	**0 : 10 : 14**
6) Goldstein-Price					
1	3.0000000000	3.0004467901	3.0001614293	0.0001861275	0 : 02 : 01
10	3.0000000000	3.0009606348	3.0003876953	0.0003203661	0 : 05 : 16
20	3.0000000000	3.0014791604	3.0002817865	0.0004568261	0 : 08 : 50
30	3.0000000000	3.0014737575	3.0002194656	0.0004550386	0 : 12 : 24
40	3.0000000000	3.0004467901	3.0002230992	0.0001736821	0 : 15 : 57
ค่าเฉลี่ย	**3.0000000000**	**3.0014791604**	**3.0002546952**	**0.0003363171**	**0 : 08 : 54**

4.4.3 สรุปผลการทดลอง

การทดลองที่ 2 ได้ทำการทดสอบอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่กับฟังก์ชันออฟติไมเซชันจำนวน 6 ฟังก์ชัน จากผลการทดลอง สามารถสรุปได้ดังนี้

1) ฟังก์ชัน Michalewicz

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่ดีที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่ดีที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20 และ 30 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ค้นหาคำตอบที่ดีที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20, 30 และ 40 นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยสูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 0.05% และ 0.01% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.12(ก) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้ง

แบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สูงกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Michalewicz เป็นฟังก์ชันสำหรับการหาค่าสูงสุด) แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีการผสมพันธุ์ภายในรังเดียวกัน แต่อย่างไรก็ตาม การทำงานในขั้นตอนการแยกรังของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ทำให้เวลาในการประมวลผลเพิ่มขึ้นจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 65.80%

2) ฟังก์ชัน Rastrigin

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน ซึ่งทั้งสองอัลกอริทึมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20, 30 และ 40 เหมือนกัน แต่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ได้ค่าคำตอบที่แย่ที่สุดดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 15.00% ในทางกลับกันอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมได้ค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ 9.21% และมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่า ดังรูปที่ 4.12(ข) จากผลการทดลองนี้แสดงให้เห็นว่า อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่มีช่วงคำตอบระหว่างค่าคำตอบที่ดีที่สุดและค่าคำตอบที่แย่ที่สุดแคบลง แต่ค่าคำตอบทั้งหมดมีการกระจายตัวสูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม นอกจากนี้การทำงานในขั้นตอนการแยกรังของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ได้ใช้เวลาในการประมวลผลเพิ่มขึ้นจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 61.98%

3) ฟังก์ชัน Rosenbrocks' valley

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20 และ 40 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 30 และ 40 นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึม

การผสมพันธุ์ของผึ้งแบบดั้งเดิม 18.37% และ 45.40% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.12(ค) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Rosenbrocks' valley เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีการผสมพันธุ์ภายในรังเดียวกัน แต่อย่างไรก็ตาม การทำงานในขั้นตอนการแยกรังของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ทำให้เวลาในการประมวลผลเพิ่มขึ้นจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 44.63%

4) ฟังก์ชัน Sphere

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 1, 10, 20 และ 30 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ค้นหาคำตอบที่เหมาะที่สุดได้ครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลอง นอกจากนี้ยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 28.00% และ 22.89% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.12(ง) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Sphere เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีการผสมพันธุ์ภายในรังเดียวกัน แต่อย่างไรก็ตามการทำงานในขั้นตอนการแยกรังของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ทำให้เวลาในการประมวลผลเพิ่มขึ้นจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 63.64%

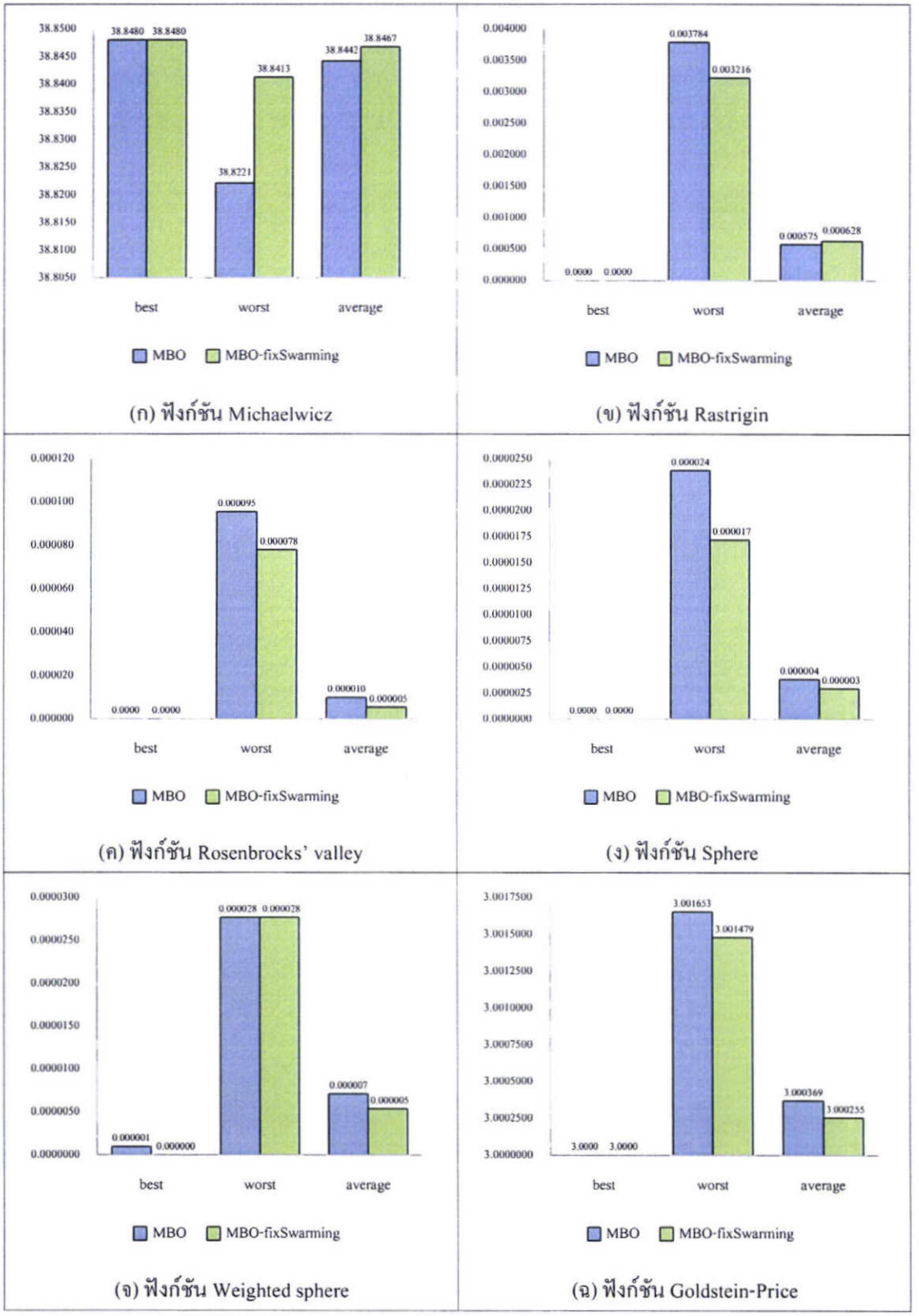
5) ฟังก์ชัน Weighted sphere

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุด (จากตารางที่ 2) ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่เหมาะสมที่สุดครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลองและได้ค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 24.46% และมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย อย่างไรก็ตามทั้งสองอัลกอริทึมสามารถค้นหา

คำตอบที่แย่ที่สุดได้เท่ากัน จากรูปที่ 4.12(จ) แท่งกราฟค่าคำตอบที่เหมาะสมที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยที่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีการผสมพันธุ์ภายในรังเดียวกัน นอกจากนี้กลไกการทำงานในขั้นตอนการแยกรังของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ทำให้เวลาในการประมวลผลเพิ่มขึ้นจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 65.95%

6) ฟังก์ชัน Goldstein-Price

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ได้ครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลอง นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยที่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 0.01% และ 0.003% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.12(ฉ) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยที่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Goldstein-Price เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีการผสมพันธุ์ภายในรังเดียวกัน อย่างไรก็ตามการทำงานในขั้นตอนการแยกรังของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ทำให้เวลาในการประมวลผลเพิ่มขึ้นจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 42.78%



รูปที่ 4.12 กราฟแสดงการเปรียบเทียบผลการทดลองระหว่างอัลกอริทึม MBO และอัลกอริทึม MBO-fixSwarming

4.5 การทดสอบเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรัง (แบบไม่มีการกำหนดจำนวนผึ้งนางพญา)

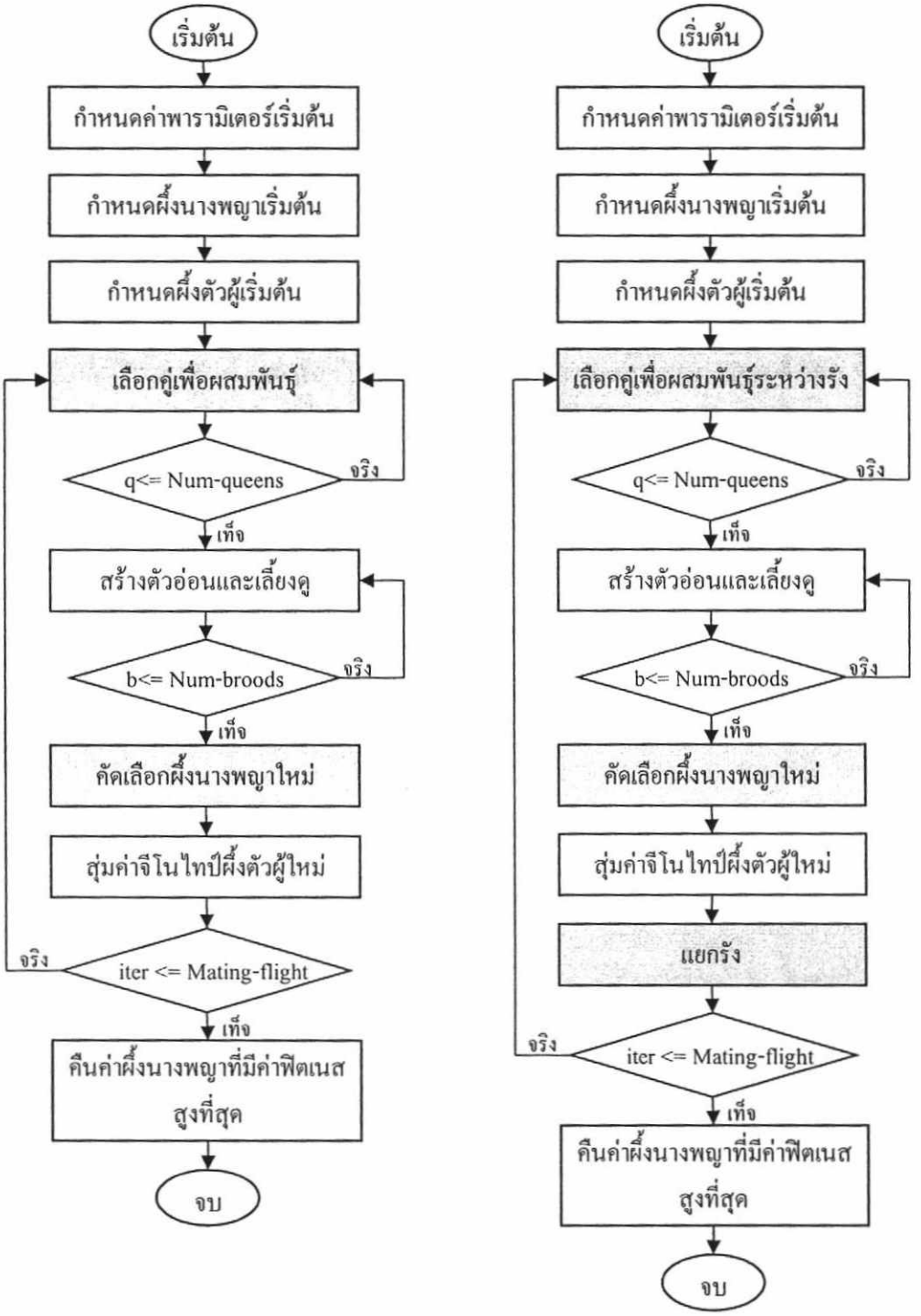
การทดลองที่ 3 เป็นการทดลองเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม แต่มีข้อแตกต่างจากการทดลองที่ 2 คือ การทดลองที่ 2 ได้กำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ แต่ในการทดลองนี้ไม่มีการกำหนดจำนวนผึ้งนางพญาในแต่ละรอบการทำงาน การทดสอบได้ทำการเปรียบเทียบการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยได้มีการปรับปรุงให้มีการแบ่งรัง แต่ละรังมีผึ้งนางพญาหนึ่งตัว ผึ้งตัวผู้ทุกตัวจะถูกจัดให้เป็นสมาชิกของรังใดรังหนึ่ง จากนั้นผึ้งนางพญาแต่ละตัวจะเลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นๆ ทั้งนี้ในแต่ละรอบการทำงานได้มีกลไกการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง ดังนั้นจึงไม่มีการกำหนดจำนวนของผึ้งนางพญาเป็นค่าคงที่ จากรูปที่ 4.13 แสดงการเปรียบเทียบโครงสร้างการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (MBO) และอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนของผึ้งนางพญา (SMBO-oldDrone) ขั้นตอนที่มีการแรเงาไว้คือ ขั้นตอนที่มีความแตกต่างกันระหว่างทั้งสองโมเดล ซึ่งเป็นปัจจัยที่ต้องการศึกษา

4.5.1 การกำหนดค่าพารามิเตอร์

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนของผึ้งนางพญา ได้มีการกำหนดค่าพารามิเตอร์ต่างๆ ดังตารางที่ 4.6 อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมมีการกำหนดค่าเหมือนในการทดลองที่ผ่านมา แต่ในอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนของผึ้งนางพญาไม่จำเป็นต้องมีการระบุจำนวนของผึ้งนางพญา

ตารางที่ 4.6 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม MBO และอัลกอริทึม SMBO-oldDrone

พารามิเตอร์	MBO	SMBO-oldDrone
จำนวนของผึ้งนางพญา	1, 10, 20, 30, 40	-
จำนวนของผึ้งตัวผู้	200	200
จำนวนของตัวอ่อนผึ้ง	500	500
ขนาดของถุงเก็บสเปิร์ม	100	100
ค่าความน่าจะเป็นในการมิวเตชัน	0.2	0.2
จำนวนรอบการทำงาน	400	400



(ก) โครงสร้างการทำงานของอัลกอริทึม MBO (ข) โครงสร้างการทำงานของอัลกอริทึม SMBO-oldDrone

รูปที่ 4.13 เปรียบเทียบโครงสร้างการทำงานของอัลกอริทึม MBO และอัลกอริทึม SMBO-oldDrone

4.5.2 ผลการทดลอง

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญา ได้ทำการทดสอบกับฟังก์ชันออฟติไมเซชันจำนวน 6 ฟังก์ชัน คือ

1) ฟังก์ชัน Michalewicz

2) ฟังก์ชัน Rastrigin

3) ฟังก์ชัน Rosenbrocks' valley

4) ฟังก์ชัน Sphere

5) ฟังก์ชัน Weighted sphere

6) ฟังก์ชัน Goldstein-Price

โดยได้ใช้ชุดข้อมูลเริ่มต้นชุดเดียวกันกับการทดลองที่ 1 เพื่อให้สามารถเปรียบเทียบผลการทดลองร่วมกันได้ โดยชุดข้อมูลดังกล่าวเกิดจากการสุ่มข้อมูลด้วยฟังก์ชัน rand() ของโปรแกรม MATLAB เพื่อใช้เป็นประชากรเริ่มต้นของอัลกอริทึมที่ทดสอบจำนวน 10 ชุด จากการกำหนดค่าพารามิเตอร์ดังตารางที่ 4.6 อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมมีการรันจำนวน 50 ครั้งต่อหนึ่งฟังก์ชัน โดยแบ่งเป็นการกำหนด ผึ้งนางพญา 1 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา 10 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา 20 ตัว มีการรันจำนวน 10 ครั้ง ผึ้งนางพญา 30 ตัว มีการรันจำนวน 10 ครั้งและผึ้งนางพญา 40 ตัว มีการรันจำนวน 10 ครั้ง ดังนั้น อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมมีการรันทั้งหมด 300 ครั้ง (50 ครั้ง × 6 ฟังก์ชัน) ส่วนอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนของผึ้งนางพญามีการรันจำนวน 10 ครั้งต่อหนึ่งฟังก์ชัน เนื่องจากอัลกอริทึมที่นำเสนอมีกลไกการคัดเลือกผึ้งนางพญาที่เหมาะสมในแต่ละรอบการทำงานจึงไม่จำเป็นต้องมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ ดังนั้น อัลกอริทึมที่นำเสนอจึงมีการรันจำนวน 60 ครั้ง (10 ครั้ง × 6 ฟังก์ชัน) ทั้งนี้ ผลการทดลองของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมได้นำมาจากการทดลองที่ 1 (ตารางที่ 4.2) ในการทดลองนี้จึงได้ทำการรันเฉพาะอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาเท่านั้น

เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบคือ Intel ® core ™ i7 2.8 GHz (3 GB of RAM) และใช้โปรแกรม MATLAB เวอร์ชัน 7.6.0.324 ผลการทดลองที่ได้มีการเปรียบเทียบประสิทธิภาพของทั้งสองอัลกอริทึมด้วย 5 ตัวชี้วัด คือ 1) ค่าคำตอบที่ดีที่สุด (y_{best}) 2) ค่าคำตอบที่แย่ที่สุด (y_{worst}) 3) ค่าเฉลี่ยของคำตอบทั้งหมด ($\bar{y}$) 4) ส่วนเบี่ยงเบนมาตรฐาน (S.D.) และ 5) เวลา (time) ดังตารางที่ 4.7

ตารางที่ 4.7 ผลการทดลองของอัลกอริทึม SMBO-oldDrone

ชื่อฟังก์ชัน	y_{best}	y_{worst}	$\bar{y}$	S.D.	time
1) Michalewicz	38.8479863775	38.8424314603	38.8469055838	0.0022287574	0 : 02 : 50
2) Rastrigin	0.0000000000	0.0024595614	0.0006432751	0.0007256666	0 : 04 : 07
3) Rosenbrocks' valley	0.0000000000	0.0000344501	0.0000045898	0.0000105878	0 : 01 : 57
4) Sphere	0.0000000000	0.0000095367	0.0000027657	0.0000028944	0 : 02 : 06
5) Weighted sphere	0.0000000000	0 0000190735	0.0000057220	0.0000050464	0 : 02 : 06
6) Goldstein-Price	3.0000000000	3.0004467901	3.0002474188	0.0002220261	0 : 02 : 00

4.5.3 สรุปผลการทดลอง

การทดลองที่ 3 ได้ทำการทดสอบอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญากับฟังก์ชันออฟติไมเซชันจำนวน 6 ฟังก์ชัน จากผลการทดลอง สามารถสรุปได้ดังนี้

1) ฟังก์ชัน Michalewicz

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่ดีที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่ดีที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20 และ 30 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญามีกลไกในการหาจำนวนผึ้งนางพญาในแต่ละรอบการทำงานด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.14(ก) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยสูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 0.05% และ 0.01% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.15(ก) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาสูงกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Michalewicz เป็นฟังก์ชันการหาค่าสูงสุด) แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังและการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีดั้งเดิม นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการ

กำหนดจำนวนผึ้งนางพญายังทำให้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 55.61% อีกด้วย

2) ฟังก์ชัน Rastrigin

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20, 30 และ 40 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญามีกลไกในการหาจำนวนผึ้งนางพญาในแต่ละรอบการทำงานด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.14(ข) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังได้ค่าคำตอบที่แย่ที่สุดดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 35.00% แต่ในทางกลับกันอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมได้ค่าคำตอบเฉลี่ยที่ดีกว่า 11.84% โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย แต่อย่างไรก็ตามอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาช่วยทำให้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 31.96%

3) ฟังก์ชัน Rosenbrocks' valley

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20 และ 40 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญามีกลไกในการหาจำนวนผึ้งนางพญาในแต่ละรอบการทำงานด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.14(ค) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 63.88% และ 53.57% ตามลำดับ จากรูปที่ 4.15(ค) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Rosenbrocks' valley เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่าวิธีการเลือกผสมพันธุ์ระหว่างรังและการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่

นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าวิธีดั้งเดิม นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังทำให้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 66.95% อีกด้วย

4) ฟังก์ชัน Sphere

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 1, 10, 20 และ 30 ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญามีกลไกในการหาจำนวนผึ้งนางพญาในแต่ละรอบการทำงานด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.14(ง) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 60.00% และ 27.86% ตามลำดับ จากรูปที่ 4.15(ง) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Sphere เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังและการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้น นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังทำให้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 65.29% อีกด้วย

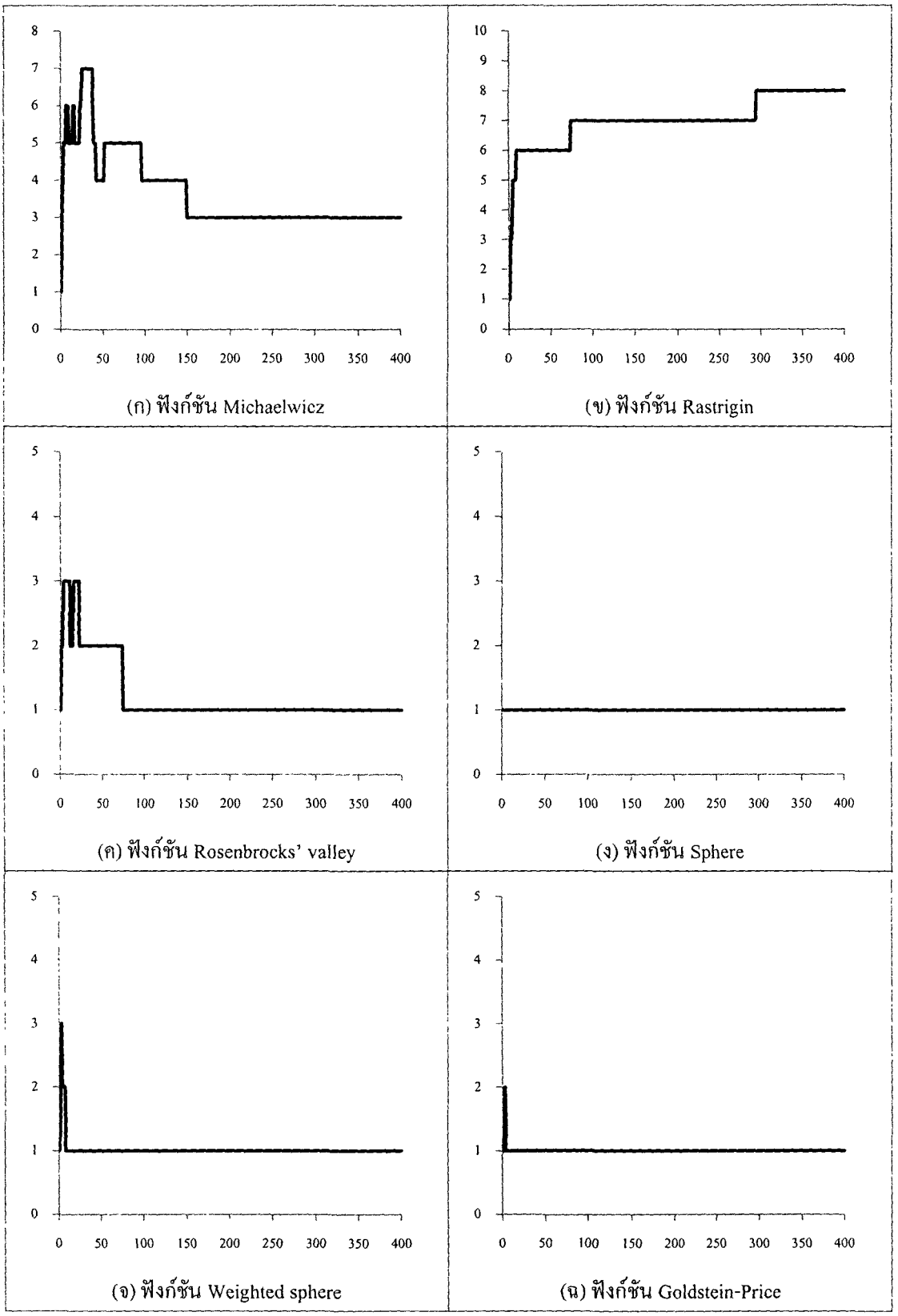
5) ฟังก์ชัน Weighted sphere

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุดในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังที่นำเสนอโดยไม่มีการกำหนดจำนวนผึ้งนางพญาสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ โดยในแต่ละรอบการทำงานมีการหาจำนวนผึ้งนางพญาในแต่ละรอบการทำงานด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.14(จ) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 31.03% และ 19.36% ตามลำดับ จากรูปที่ 4.15(จ) แท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญา

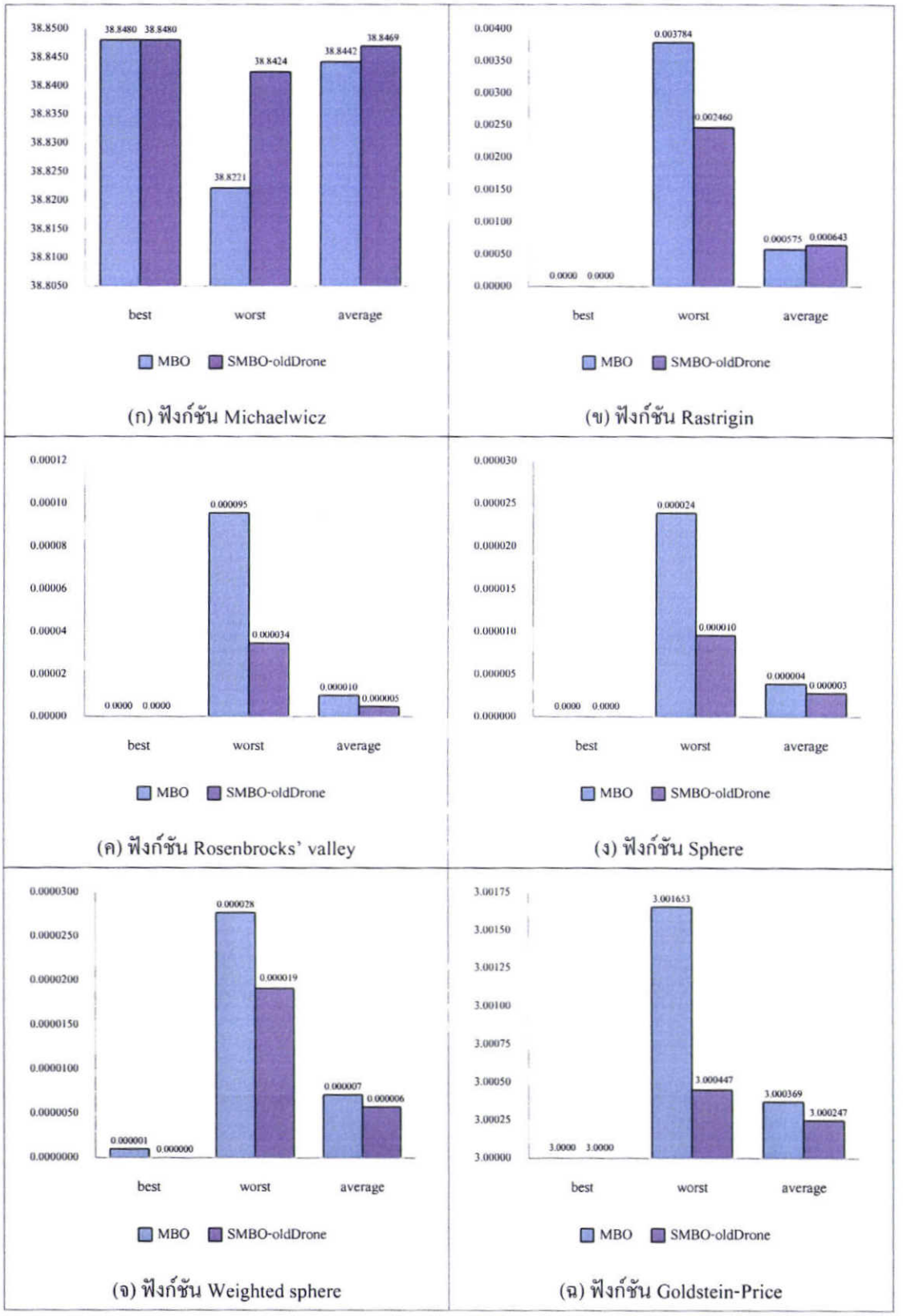
ต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Weighted sphere เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังและการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้น นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังทำให้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 65.95% อีกด้วย

6) ฟังก์ชัน Goldstein-Price

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังที่นำเสนอโดยไม่มีการกำหนดจำนวนผึ้งนางพญาสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลอง ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญามีกลไกในการหาจำนวนผึ้งนางพญาในแต่ละรอบการทำงานด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.14(ฉ) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 0.04% และ 0.003% ตามลำดับ จากรูปที่ 4.15(ฉ) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Goldstein-Price เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการเลือกผสมพันธุ์ระหว่างรังและการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้น นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังทำให้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 67.91% อีกด้วย



รูปที่ 4.14 จำนวนของผึ้งนางพญาจากการรันอัลกอริทึม SMBO-oldDrone



รูปที่ 4.15 กราฟแสดงการเปรียบเทียบผลการทดลองระหว่างอัลกอริทึม MBO และอัลกอริทึม SMBO-oldDrone

4.6 การทดสอบอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข

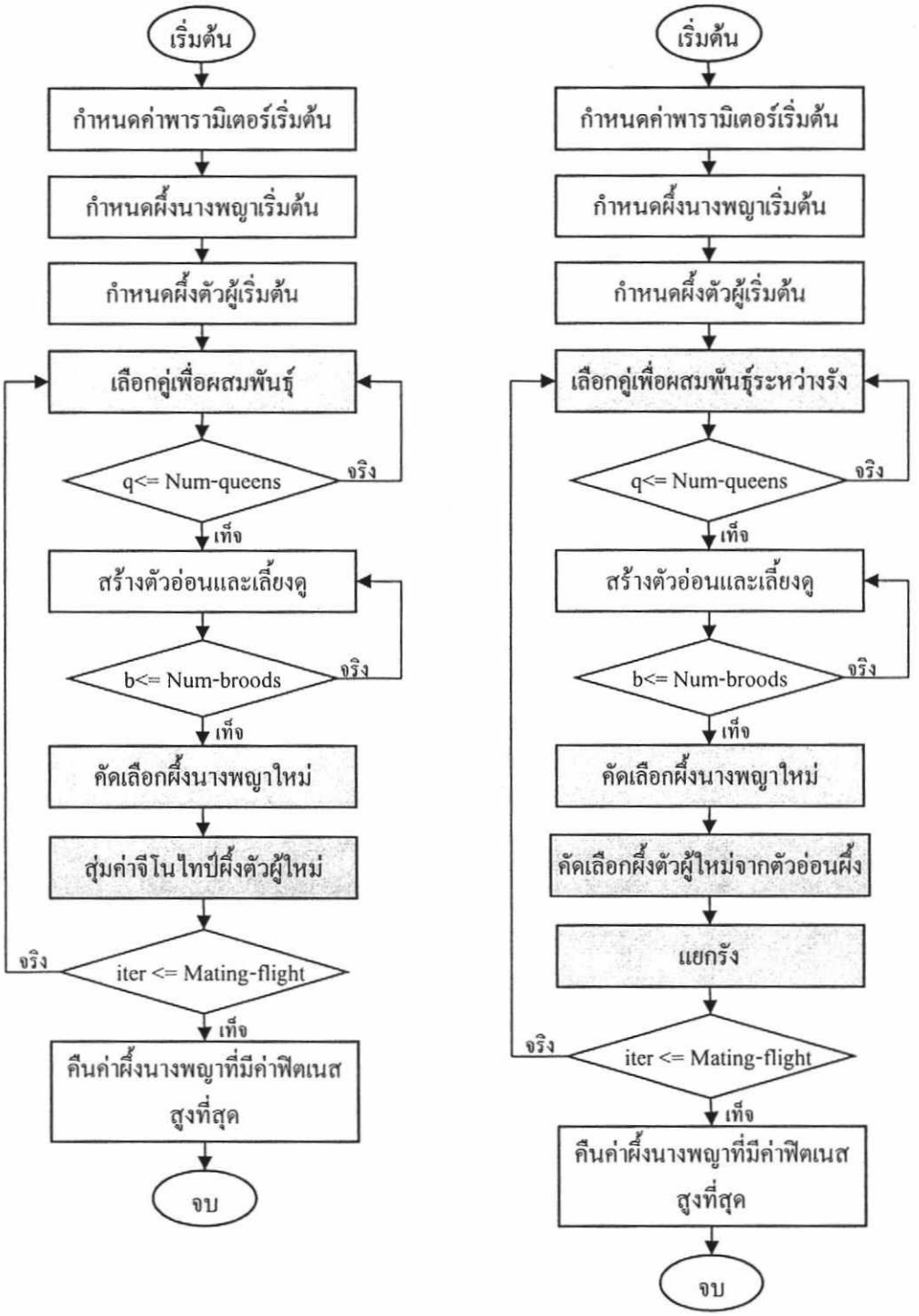
การทดลองที่ 4 เป็นการทดลองเพื่อเปรียบเทียบประสิทธิภาพการทำงานระหว่างอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขซึ่งเกิดจากการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 2 ส่วน คือ 1) วิธีการผสมพันธุ์ระหว่างรัง เพื่อการแก้ปัญหาการผสมพันธุ์ระหว่างผึ้งนางพญากับผึ้งตัวผู้ภายในรังเดียวกันด้วยการแบ่งรังเพื่อให้ผึ้งนางพญาได้มีโอกาสเลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นๆ นอกจากนี้ในแต่ละรอบการทำงานมีการคัดเลือกผึ้งนางพญาที่เหมาะสมในการสร้างรังอีกด้วย และ 2) วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งเพื่อการแก้ปัญหาการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้ใหม่ในทุกรอบการทำงาน ด้วยวิธีการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้จากตัวอ่อนผึ้งเพื่อลดเวลาในการประมวลผล เพราะค่าจีโนไทป์และค่าฟิตเนสของตัวอ่อนผึ้งได้ถูกคำนวณไว้ในทุกรอบการทำงานอยู่ก่อนแล้ว จากรูปที่ 4.16 แสดงการเปรียบเทียบโครงสร้างการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (MBO) และอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข (SMBO-numerical) ขั้นตอนที่มีการแรเงาไว้คือ ขั้นตอนที่มีความแตกต่างกันระหว่างทั้งสองโมเดล ซึ่งเป็นปัจจัยที่ต้องการศึกษา

4.6.1 การกำหนดค่าพารามิเตอร์

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข ได้มีการกำหนดค่าพารามิเตอร์ต่างๆ ดังตารางที่ 4.8 จากตารางจะเห็นว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมมีการกำหนดค่าพารามิเตอร์เหมือนในการทดลองที่ผ่านมา ในขณะที่อัลกอริทึมที่นำเสนอสำหรับใช้ในการแก้ปัญหาออฟติไมเซชันเชิงตัวเลขไม่จำเป็นต้องมีการกำหนดจำนวนผึ้งนางพญาไว้เป็นค่าคงที่ เพราะในแต่ละรอบการทำงานมีกลไกการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังโดยอัตโนมัติ

ตารางที่ 4.8 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม MBO และอัลกอริทึม SMBO-numerical

พารามิเตอร์	MBO	SMBO-numerical
จำนวนของผึ้งนางพญา	1, 10, 20, 30, 40	-
จำนวนของผึ้งตัวผู้	200	200
จำนวนของตัวอ่อนผึ้ง	500	500
ขนาดของถุงเก็บสเปิร์ม	100	100
ค่าความน่าจะเป็นในการมิวเตชัน	0.2	0.2
จำนวนรอบการทำงาน	400	400



(ก) โครงสร้างการทำงานของอัลกอริทึม MBO (ข) โครงสร้างการทำงานของอัลกอริทึม SMBO-numerical

รูปที่ 4.16 เปรียบเทียบโครงสร้างการทำงานของอัลกอริทึม MBO และ
อัลกอริทึม SMBO-numerical

4.6.2 ผลการทดลอง

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติ-ไมเซชันเชิงตัวเลขได้ทำการทดสอบกับฟังก์ชันออฟติไมเซชันจำนวน 6 ฟังก์ชัน คือ

1) ฟังก์ชัน Michalewicz

2) ฟังก์ชัน Rastigin

3) ฟังก์ชัน Rosenbrocks' valley

4) ฟังก์ชัน Sphere

5) ฟังก์ชัน Weighted sphere

6) ฟังก์ชัน Goldstein-Price

โดยได้ใช้ชุดข้อมูลเริ่มต้นชุดเดียวกันกับการทดลองที่ 1 เพื่อให้สามารถเปรียบเทียบผลการทดลองร่วมกันได้ โดยชุดข้อมูลดังกล่าวเกิดจากการสุ่มข้อมูลด้วยฟังก์ชัน rand() ของโปรแกรม MATLAB เพื่อใช้เป็นประชากรเริ่มต้นของอัลกอริทึมที่ทดสอบจำนวน 10 ชุด จากการกำหนดค่าพารามิเตอร์ดังตารางที่ 4.8 อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม มีการรันจำนวน 50 ครั้งต่อหนึ่งฟังก์ชัน โดยแบ่งเป็นการกำหนด ผึ้งนางพญา 1 ตัว มีการรันจำนวน 10 ครั้ง กำหนดผึ้งนางพญา 10 ตัว มีการรันจำนวน 10 ครั้ง กำหนดผึ้งนางพญา 20 ตัว มีการรันจำนวน 10 ครั้ง กำหนดผึ้งนางพญา 30 ตัว มีการรันจำนวน 10 ครั้งและกำหนดผึ้งนางพญา 40 ตัว มีการรันจำนวน 10 ครั้ง ดังนั้น การทดลองนี้อัลกอริทึมการผสมพันธุ์ของผึ้งมีการรันทั้งหมด 300 ครั้ง (50 ครั้ง × 6 ฟังก์ชัน) ส่วนอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข มีการรันจำนวน 10 ครั้งต่อหนึ่งฟังก์ชัน เนื่องจากอัลกอริทึมที่นำเสนอมีกลไกการคัดเลือกพญาผึ้งที่เหมาะสมในแต่ละรอบการทำงานจึงไม่จำเป็นต้องมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ ดังนั้น อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขจึงมีการรันจำนวน 60 ครั้ง (10 ครั้ง × 6 ฟังก์ชัน) ทั้งนี้ ผลการทดลองของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมได้นำมาจากการทดลองที่ 1 (ตารางที่ 4.2) ในการทดลองนี้จึงได้ทำการรันเฉพาะอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขเท่านั้น

เครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบคือ Intel ® core ™ i7 2.8 GHz (3 GB of RAM) และใช้โปรแกรม MATLAB เวอร์ชัน 7.6.0.324 ผลการทดลองที่ได้มีการเปรียบเทียบประสิทธิภาพของทั้งสองอัลกอริทึมด้วย 5 ตัวชี้วัด คือ 1) ค่าคำตอบที่ดีที่สุด (y_{best}) 2) ค่าคำตอบที่แย่ที่สุด (y_{worst}) 3) ค่าเฉลี่ยของคำตอบทั้งหมด ($\bar{y}$) 4) ส่วนเบี่ยงเบนมาตรฐาน (S.D.) และ 5) เวลา (time) ดังตารางที่ 4.9

ตารางที่ 4.9 ผลการทดลองของอัลกอริทึม SMBO-numerical

ชื่อฟังก์ชัน	y_{best}	y_{worst}	$\bar{y}$	S.D.	time
1) Michalewicz	38.8479863775	38.8460333996	38.8476226021	0.0006609825	0 : 02 : 36
2) Rastrigin	0.0000000000	0.0009459972	0.0002270403	0.0002792073	0 : 03 : 30
3) Rosenbrocks' valley	0.0000000000	0.0000152821	0 0000023867	0.0000046630	0 : 01 : 49
4) Sphere	0.0000000000	0.0000019073	0.0000007629	0.0000007523	0 : 01 : 51
5) Weighted sphere	0.0000000000	0.0000047684	0.0000020027	0.0000018234	0 : 01 : 45
6) Goldstein-Price	3.0000000000	3.0004467901	3.0002232854	0.0001739478	0 : 01 : 45

4.6.3 สรุปผลการทดลอง

การทดลองที่ 4 ได้ทำการทดสอบอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมกับอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขกับฟังก์ชันออฟติไมเซชันจำนวน 6 ฟังก์ชัน จากผลการทดลอง สามารถสรุปได้ดังนี้

1) ฟังก์ชัน Michalewicz

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติ-ไมเซชันเชิงตัวเลขสามารถค้นหาคำตอบที่ดีที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่ดีที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20 และ 30 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขไม่ต้องมีการกำหนดจำนวนผึ้งนางพญา โดยในแต่ละรอบการทำงานได้มีกลไกในการหาจำนวนผึ้งนางพญาด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.17(ก) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยสูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 0.06% และ 0.01% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.18(ก) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไม-เซชันเชิงตัวเลขสูงกว่าเส้นกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Michalewicz เป็นฟังก์ชันสำหรับการหาค่าสูงสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง วิธีการเลือกผสมพันธุ์ระหว่างรังและวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 59.27%

2) ฟังก์ชัน Rastrigin

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20, 30 และ 40 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขไม่ต้องมีการกำหนดจำนวนผึ้งนางพญา โดยในแต่ละรอบการทำงานได้มีกลไกในการหาจำนวนผึ้งนางพญาด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.17(ข) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 75.00% และ 60.53% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.18(ข) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Rastrigin เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง วิธีการเลือกผสมพันธุ์ระหว่างรังและวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 42.15%

3) ฟังก์ชัน Rosenbrocks' valley

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 10, 20 และ 40 (จากตารางที่ 4.2) ในขณะที่อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขไม่ต้องมีการกำหนดจำนวนผึ้งนางพญา โดยในแต่ละรอบการทำงานได้มีกลไกในการหาจำนวนผึ้งนางพญาด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.17(ค) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 83.98% และ 75.86% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.18(ค) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Rosenbrocks' valley เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง วิธีการเลือกผสมพันธุ์ระหว่างรังและวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าอัลกอริทึม

การผสมพันธุ์ของผึ้งแบบดั้งเดิม นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 69.21%

4) ฟังก์ชัน Sphere

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติ-ไมเซชันเชิงตัวเลขสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ เมื่อกำหนดจำนวนผึ้งนางพญาเท่ากับ 1, 10, 20 และ 30 ในขณะที่อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขไม่ต้องมีการกำหนดจำนวนผึ้งนางพญา โดยในแต่ละรอบการทำงานได้มีกลไกในการหาจำนวนผึ้งนางพญาด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.17(ง) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 92.00% และ 80.10% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.18(ง) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Sphere เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง วิธีการเลือกผสมพันธุ์ระหว่างรังและวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 69.42%

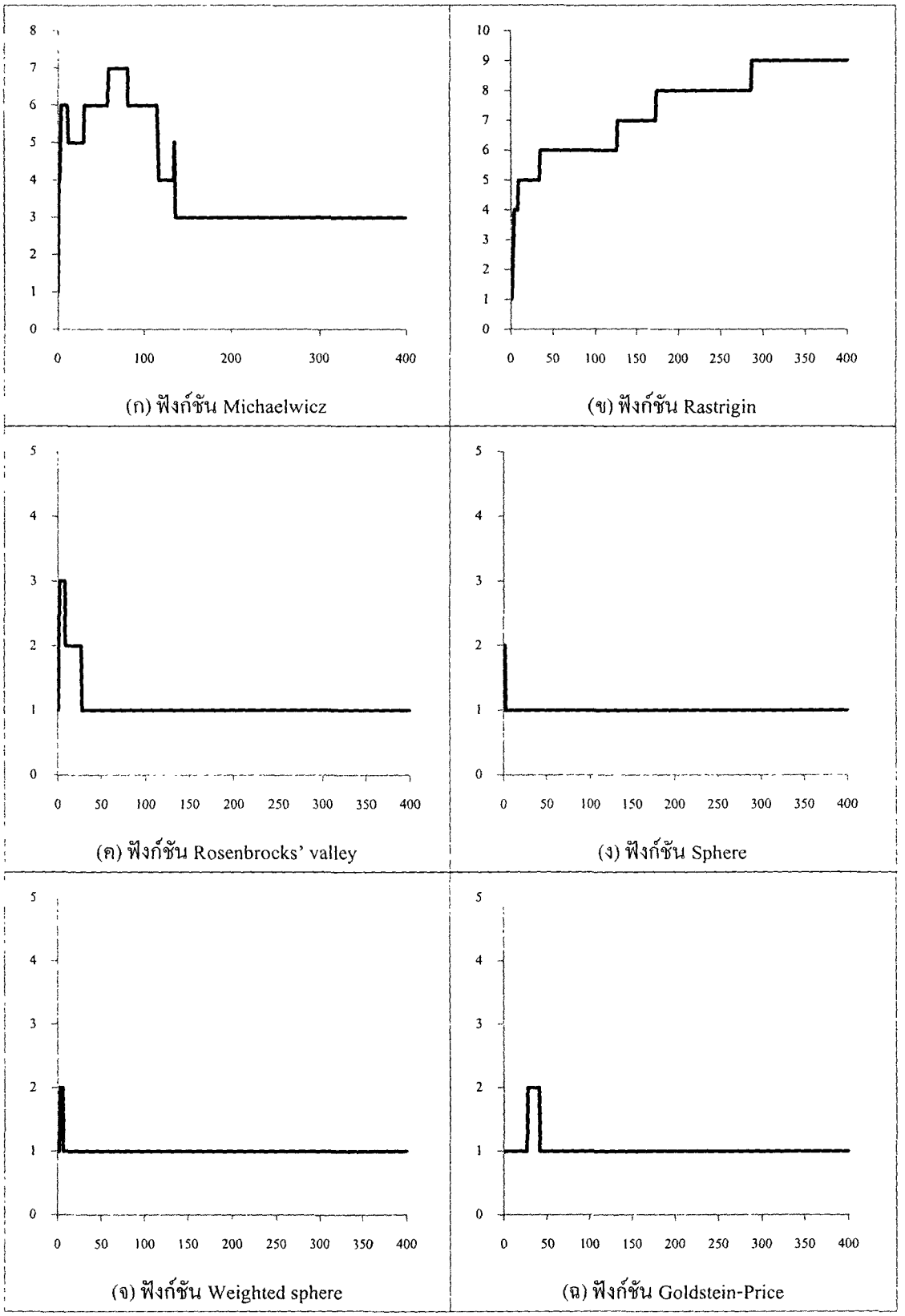
5) ฟังก์ชัน Weighted sphere

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุดในขณะที่อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ โดยในแต่ละรอบการทำงานมีการหาจำนวนผึ้งนางพญาในแต่ละรอบการทำงานด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.17(จ) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 82.76% และ 71.77% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.18(จ) แท่งกราฟของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Weighted phere เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง วิธีการเลือกผสมพันธุ์ระหว่างรังและวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม นอกจากนี้อัลกอริทึมที่

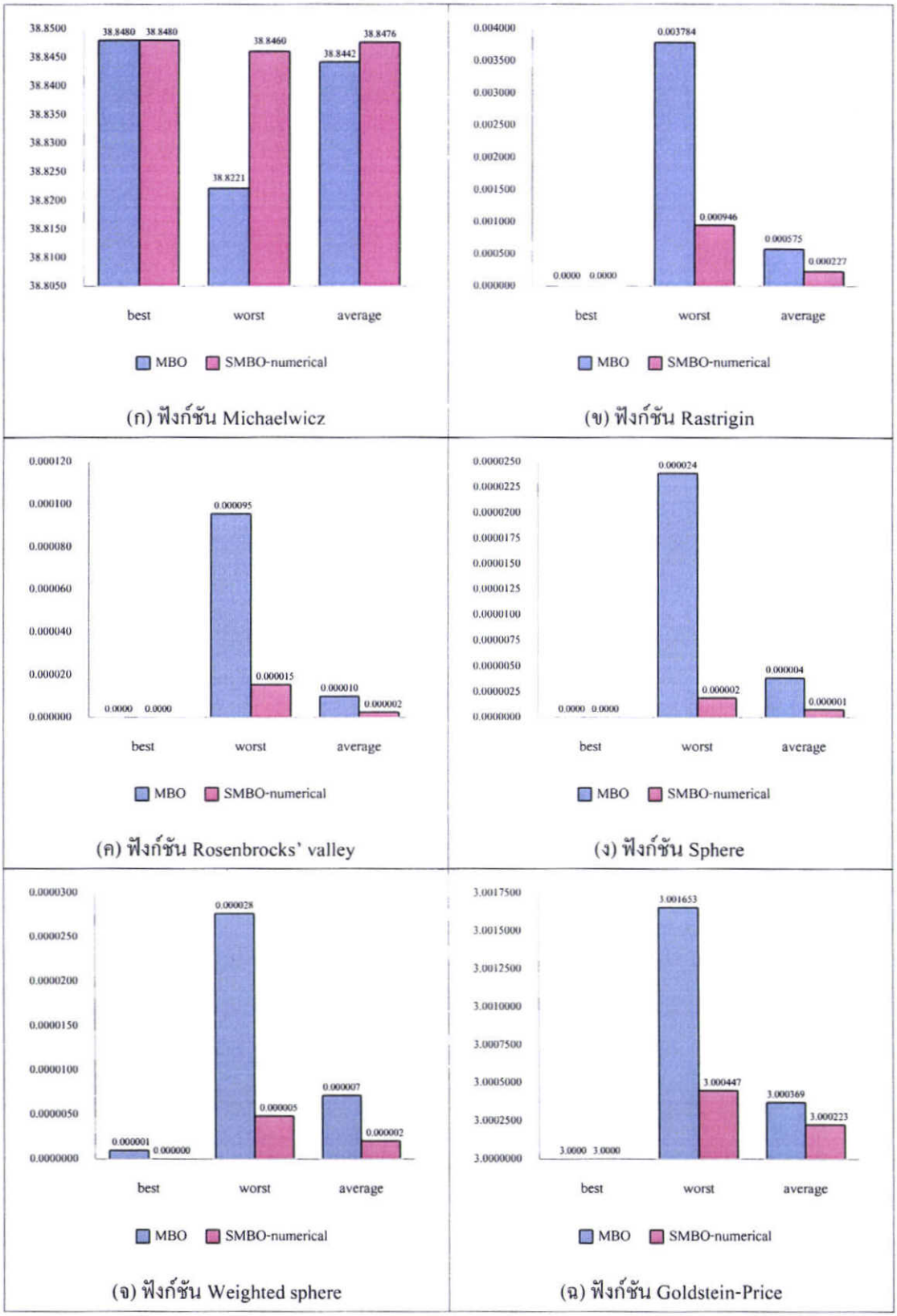
นำเสนอสำหรับออฟติไมเซชันเชิงตัวเลขยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 71.62%

6) ฟังก์ชัน Goldstein-Price

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้เหมือนกัน โดยอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 5 ค่าตามจำนวนผึ้งนางพญาที่ทดลอง ในขณะที่อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขไม่ต้องมีการกำหนดจำนวนผึ้งนางพญา โดยในแต่ละรอบการทำงานได้มีกลไกในการหาจำนวนผึ้งนางพญาด้วยวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรัง รูปที่ 4.17(ฉ) แสดงจำนวนของผึ้งนางพญาในแต่ละรอบการทำงาน นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังได้ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 0.04% และ 0.004% ตามลำดับ โดยมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากรูปที่ 4.18(ฉ) แท่งกราฟค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขต่ำกว่าแท่งกราฟของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม (ฟังก์ชัน Goldstein-Price เป็นฟังก์ชันสำหรับการหาค่าต่ำสุด) แสดงว่า วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง วิธีการเลือกผสมพันธุ์ระหว่างรังและวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังที่นำเสนอนี้ทำให้การค้นหาคำตอบที่เหมาะสมที่สุดมีประสิทธิภาพดียิ่งขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังใช้เวลาในการประมวลผลลดลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 71.93%



รูปที่ 4.17 จำนวนของผึ้งนางพญาจากการรันอัลกอริทึม SMBO-numerical



รูปที่ 4.18 กราฟแสดงการเปรียบเทียบผลการทดลองระหว่างอัลกอริทึม MBO และอัลกอริทึม SMBO-numerical

4.7 การทดสอบอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด

การทดลองที่ 5 เป็นการทดลองเพื่อเปรียบเทียบประสิทธิภาพการทำงานระหว่างอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด (SMBO-combinatorial) เทียบกับอัลกอริทึม PGACS (Parallelized genetic ant colony systems) ที่นำเสนอโดย Chen และ Chien [3] ซึ่งเป็นอัลกอริทึมที่เกิดจากการผสมผสานระหว่างอัลกอริทึมอาณานิคมมดกับจีเนติกอัลกอริทึม การทดลองนี้ได้มีการทดสอบกับปัญหาการเดินทางของพนักงานขายซึ่งประกอบด้วยข้อมูล ST70 และข้อมูล EIL101 และได้ทำการเปรียบเทียบผลการทดลองของอัลกอริทึมที่นำเสนอกับผลการทดลองที่นำมาจากงานวิจัยของ Chen และ Chien [3] ด้วย

4.7.1 การกำหนดค่าพารามิเตอร์

อัลกอริทึมที่นำเสนอสำหรับใช้ในการแก้ปัญหาออฟติไมเซชันเชิงการจัด มีการกำหนดค่าพารามิเตอร์ต่างๆ ดังตารางที่ 4.10 โดยมีการกำหนดจำนวนรอบในการทำงานของข้อมูล ST70 และข้อมูล EIL101 เท่ากับ 60,000 และ 80,000 ตามลำดับ

ตารางที่ 4.10 การกำหนดค่าพารามิเตอร์ของอัลกอริทึม SMBO-combinatorial

พารามิเตอร์	SMBO-combinatorial
จำนวนของผึ้งตัวผู้	200
จำนวนของตัวอ่อนผึ้ง	500
ขนาดของถุงเก็บสเปิร์ม	100
ค่าความน่าจะเป็นในการมิวเตชัน	0.2
จำนวนรอบในการทำงาน	60000, 80000
ค่าความเหมือนที่ยอมรับได้	0.05

4.7.2 ผลการทดลอง

การทดลองนี้ได้ทำการเปรียบเทียบประสิทธิภาพกับอัลกอริทึม PGACS ที่นำเสนอโดย Chen และ Chien [3] โดยได้ทำการทดสอบกับปัญหาการเดินทางของพนักงานขายจำนวน 2 ข้อมูล คือ ข้อมูล ST70 และข้อมูล EIL101 การทดลองเพื่อวัดประสิทธิภาพของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดได้ทำการทดสอบโดยการรัน 10 ครั้งต่อหนึ่งข้อมูลด้วยเครื่องคอมพิวเตอร์ Intel ® core ™ i7 2.8 GHz (3 GB of RAM) และใช้โปรแกรม MATLAB เวอร์ชัน 7.6.0.324 ผลการทดลองที่ได้ได้ทำการเปรียบเทียบประสิทธิภาพของทั้งสองอัลกอริทึม

ด้วย 4 ตัวชี้วัด คือ 1) ค่าคำตอบที่ดีที่สุด (y_{best}) 2) ค่าคำตอบที่แย่ที่สุด (y_{worst}) 3) ค่าเฉลี่ยของคำตอบทั้งหมด ($\bar{y}$) และ 4) ส่วนเบี่ยงเบนมาตรฐาน (S.D.)

ผลการทดลองของอัลกอริทึม PGACS และอัลกอริทึม SMBO-combinatorial ของข้อมูล ST70 และข้อมูล EIL101 ดังตารางที่ 4.11 และตารางที่ 4.12 ตามลำดับ ทั้งนี้ผลการทดลองของอัลกอริทึม PGACS ได้นำมาจากงานวิจัย [3]

ตารางที่ 4.11 การเปรียบเทียบผลการทดลองของอัลกอริทึม PGACS และอัลกอริทึมที่นำเสนอ SMBO-combinatorial สำหรับข้อมูล ST70

การทดลอง	**PGACS [3]**	**SMBO-combinatorial**
1	677	675
2	677	675
3	677	675
4	677	675
5	677	675
6	677	675
7	677	675
8	677	679
9	677	675
10	684	679
y_{best}	677	**675**
y_{worst}	684	**679**
$\bar{y}$	678	**676**
S D.	2.2135943621	**1.686548085**
time	-	8 : 11 : 36

ตารางที่ 4.12 การเปรียบเทียบผลการทดลองของอัลกอริทึม PGACS และอัลกอริทึมที่นำเสนอ SMBO-combinatorial สำหรับข้อมูล EIL101

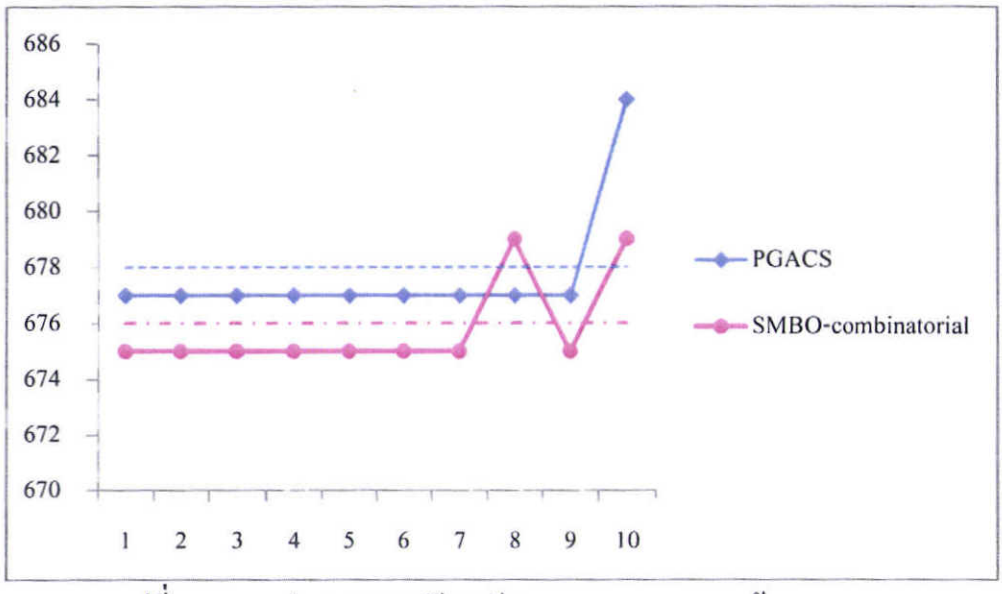
การทดลอง	PGACS [3]	SMBO-combinatorial
1	645	637
2	645	638
3	645	638
4	642	636
5	649	638
6	640	635
7	642	631
8	642	634
9	646	637
10	640	633
y_{best}	640	**631**
y_{worst}	649	**638**
$\bar{y}$	644	**636**
S.D.	2.8751811537	**2.4060109910**
time	-	10 : 38 : 29

4.7.3 สรุปผลการทดลอง

การทดลองที่ 5 ได้ทำการเปรียบเทียบประสิทธิภาพของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด (SMBO-combinatorial) กับอัลกอริทึม PGACS ที่นำเสนอโดย Chen และ Chien [3] ผลการทดลอง สามารถสรุปได้ ดังนี้

1) ข้อมูล ST70

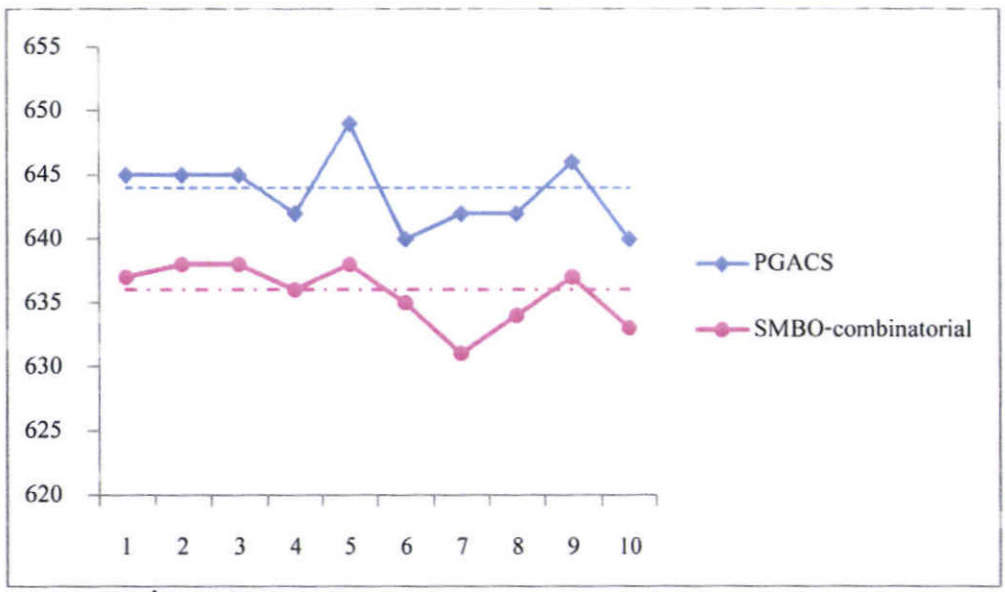
อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดสามารถค้นหาค่าคำตอบที่เหมาะสมที่สุดได้ ในขณะที่อัลกอริทึม PGACS ไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้โดย อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดสามารถค้นหาค่าคำตอบที่ดีที่สุด ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยได้ดีกว่าอัลกอริทึม PGACS 0.30%, 0.73% และ 0.30% ตามลำดับ รวมทั้งมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าอีกด้วย จากผลการทดลองนี้ สรุปได้ว่า อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดมีประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดสูงกว่าอัลกอริทึม PGACS สำหรับข้อมูล ST70



รูปที่ 4.19 กราฟแสดงการเปรียบเทียบผลการทดลองของข้อมูล ST70

2) ข้อมูล EIL101

อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดสามารถค้นหาค่าคำตอบได้ต่ำกว่าอัลกอริทึม PGACS ดังรูปที่ 4.20 โดยสามารถค้นหาค่าคำตอบที่ดีที่สุด ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยได้ดีกว่าอัลกอริทึม PGACS 1.41%, 1.69% และ 1.24% ตามลำดับ รวมทั้งมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย จากผลการทดลองนี้ สรุปได้ว่า อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดมีประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดสูงกว่าอัลกอริทึม PGACS สำหรับข้อมูล EIL101



รูปที่ 4.20 กราฟแสดงการเปรียบเทียบผลการทดลองของข้อมูล EIL101

4.8 สรุปผลการทดสอบอัลกอริทึมที่นำเสนอ

1) สรุปผลการทดสอบการศึกษาปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง (MBO-newDrone) มีประสิทธิภาพในการค้นหาคำตอบได้สูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 6 ฟังก์ชัน ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ 5 ฟังก์ชัน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งยังได้ค่าคำตอบที่แย่ที่สุด ค่าคำตอบเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมในทุกฟังก์ชัน จากผลการทดลองที่ 1 สรุปได้ว่า ปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งทำให้ประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดดีขึ้นโดยเฉลี่ย 35.28% และกลไกการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งช่วยลดเวลาในการประมวลลดลงโดยเฉลี่ย 29.76%

2) สรุปผลการทดสอบการศึกษาปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาและผึ้งตัวผู้ที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

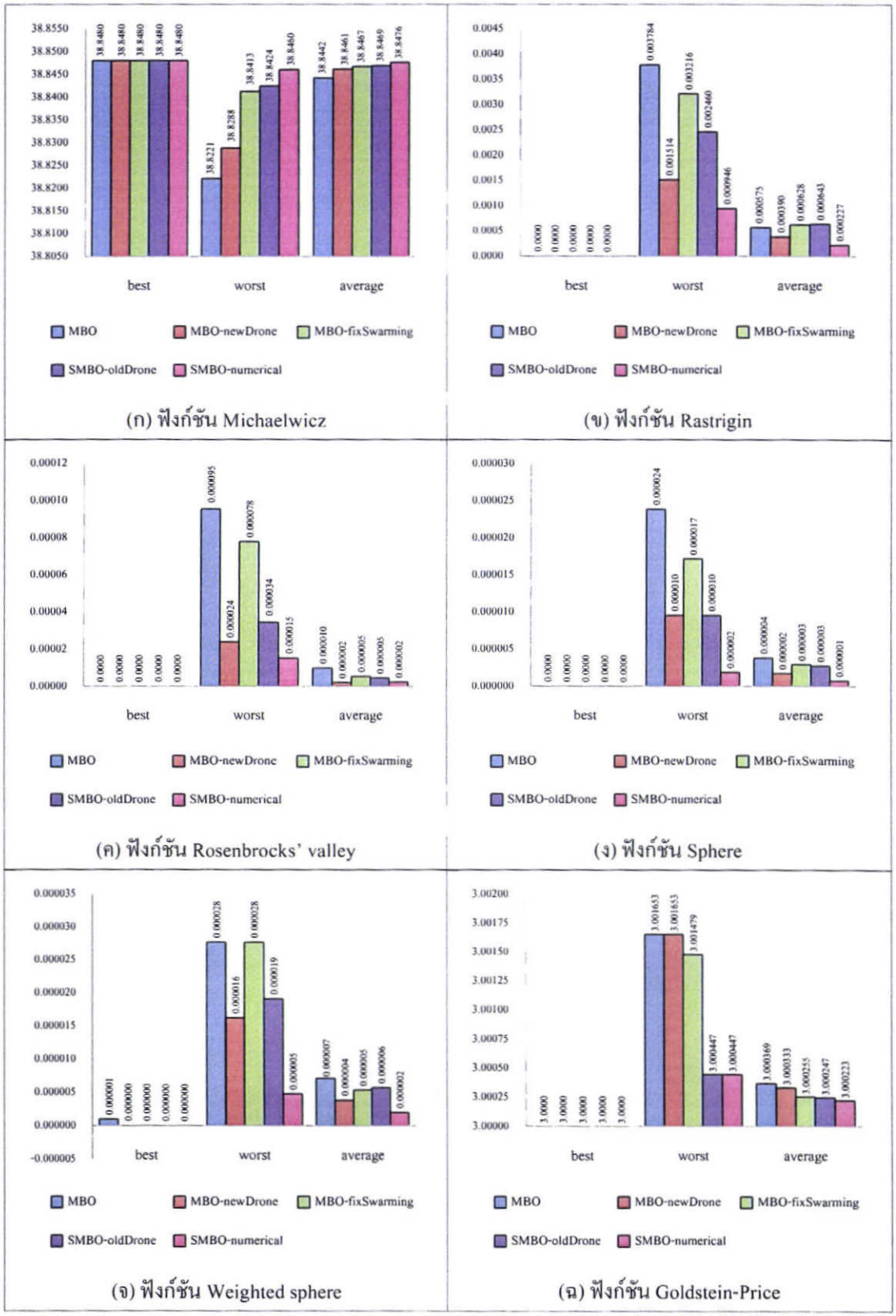
อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ (MBO-fixSwarming) มีประสิทธิภาพในการค้นหาคำตอบได้สูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 6 ฟังก์ชัน ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ 5 ฟังก์ชัน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ยังได้ค่าคำตอบที่แย่ที่สุด ค่าคำตอบเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมในทุกฟังก์ชัน ยกเว้นฟังก์ชัน Rastrigin ที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ได้ค่าคำตอบเฉลี่ยน้อยกว่า 9.21% และมีส่วนเบี่ยงเบนมาตรฐานที่น้อยกว่า จากผลการทดลองที่ 2 สรุปได้ว่า ปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาโดยที่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ทำให้ประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดดีขึ้นโดยเฉลี่ย 18.55% แต่ใช้เวลาในการประมวลสูงขึ้นโดยเฉลี่ย 57.46%

3) สรุปผลการทดสอบการศึกษาปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาและผึ้งตัวผู้ที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมโดยไม่มีการกำหนดจำนวนของผึ้งนางพญา

อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญา (SMBO-oldDrone) มีประสิทธิภาพในการค้นหาคำตอบได้สูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 6 ฟังก์ชัน ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ 5 ฟังก์ชัน นอกจากนี้อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญายังได้ค่าคำตอบที่แย่ที่สุด ค่าคำตอบเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมในทุกฟังก์ชัน ยกเว้นฟังก์ชัน Rastrigin ที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมที่มีการผสมพันธุ์ระหว่างรังโดยไม่มีการกำหนดจำนวนผึ้งนางพญาได้ค่าคำตอบเฉลี่ยน้อยกว่า 11.84% และมีส่วนเบี่ยงเบนมาตรฐานที่น้อยกว่า จากผลการทดลองที่ 3 สรุปได้ว่า ปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาโดยมีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังทำให้ประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดดีขึ้นโดยเฉลี่ย 20.16% และวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมช่วยลดเวลาในการประมวลลดลงโดยเฉลี่ย 58.95%

4) สรุปผลการทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข โดยได้ทำการทดสอบกับปัญหาออฟติไมเซชันแบบฟังก์ชันและได้เปรียบเทียบผลการทดลองกับอัลกอริทึมการผสมพันธุ์แบบดั้งเดิม

อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข (SMBO-numerical) เป็นการรวมการทำงานของ 2 ปัจจัยคือ การคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งและการผสมพันธุ์ระหว่างรังของผึ้งนางพญาโดยที่มีการคัดเลือกจำนวนผึ้งนางพญาที่เหมาะสม จากผลการทดลอง พบว่า อัลกอริทึมที่นำเสนอมีประสิทธิภาพในการค้นหาคำตอบได้สูงกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมโดยสามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ครบทั้ง 6 ฟังก์ชัน ในขณะที่อัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมค้นหาคำตอบที่เหมาะสมที่สุดได้ 5 ฟังก์ชัน นอกจากนี้อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขยังได้ค่าคำตอบที่แย่ที่สุด ค่าคำตอบเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานดีกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมในทุกฟังก์ชัน จากผลการทดลองที่ 4 สามารถสรุปได้ว่า อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขมีประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดดีขึ้นโดยเฉลี่ย 48.05% และช่วยลดเวลาในการประมวลลดลงโดยเฉลี่ย 63.93%



รูปที่ 4.21 กราฟแสดงการเปรียบเทียบประสิทธิภาพของแต่ละอัลกอริทึมโดยแยกตามฟังก์ชัน

5) สรุปผลการทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด โดยได้ทำการทดสอบกับปัญหาการเดินทางของพนักงานขายและได้เปรียบเทียบกับผลการทดลองในงานวิจัยของ Chen และ Chien [3]

การทดลองที่ 5 ได้ทำการทดสอบอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดและอัลกอริทึม PGACS กับข้อมูล ST70 และข้อมูล EIL101 ผลการทดสอบ พบว่า อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดสามารถค้นหาค่าคำตอบที่ดีที่สุด ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยได้ดีกว่าอัลกอริทึม PGACS ในทั้งสองชุดข้อมูล โดยที่อัลกอริทึมที่นำเสนอนี้สามารถค้นหาคำตอบที่เหมาะสมที่สุดของข้อมูล ST70 ได้ ในขณะที่อัลกอริทึม PGACS ไม่สามารถหาคำตอบที่เหมาะสมที่สุดได้ จากผลการทดลองนี้ สรุปได้ว่า อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดมีประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดได้ดีกว่าอัลกอริทึม PGACS

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปงานวิจัย

การศึกษาในครั้งนี้ได้พัฒนาอัลกอริทึมที่เกิดจากการปรับปรุงขั้นตอนการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยแบ่งเป็น 2 โมเดล คือ อัลกอริทึมที่นำเสนอเพื่อใช้ในการแก้ปัญหาออฟติไมเซชันเชิงตัวเลขและอัลกอริทึมที่นำเสนอเพื่อใช้ในการแก้ปัญหาออฟติ-ไมเซชันเชิงการจัด ซึ่งแต่ละโมเดลมีความแตกต่างจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม 2 แง่มุม คือ 1) วิธีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งเพื่อการแก้ปัญหาการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้ใหม่ในทุกรอบการทำงาน ด้วยวิธีการสุ่มค่าจีโนไทป์ของผึ้งตัวผู้จากตัวอ่อนผึ้งทำให้ลดเวลาในการประมวลผลเพราะค่าจีโนไทป์และค่าฟิตเนสของตัวอ่อนผึ้งได้ถูกคำนวณไว้ในทุกรอบการทำงานอยู่แล้วและ 2) วิธีการผสมพันธุ์ระหว่างรัง โดยการแบ่งพื้นที่ในการค้นหาเป็นพื้นที่รังหลายๆ รัง เพื่อให้ผึ้งนางพญาได้มีโอกาสเลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นๆ

หลักการทำงานโดยย่อของอัลกอริทึมที่นำเสนอเริ่มต้นจากการกำหนดค่าพารามิเตอร์และประชากรเริ่มต้น ประชากรที่มีค่าฟิตเนสสูงที่สุดจะได้รับเลือกให้เป็นผึ้งนางพญาจำนวนหนึ่งตัว ประชากรส่วนที่เหลือจะได้รับเลือกให้เป็นผึ้งตัวผู้ ผึ้งนางพญาทุกตัวมีหน้าที่ในการผสมพันธุ์และสามารถผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นๆ ได้หลายตัว ยกเว้นกรณีที่มีรังเพียงรังเดียว ผึ้งนางพญาแต่ละตัวจะผสมพันธุ์กับผึ้งตัวผู้จำนวนมากหรือน้อยขึ้นอยู่กับค่าพลังงานเริ่มต้นและขนาดของถุงเก็บสเปิร์ม โดยผึ้งนางพญาตัดสินใจเลือกผสมพันธุ์จากค่าความน่าจะเป็นในการผสมพันธุ์ ผึ้งตัวผู้ที่มีค่าความน่าจะเป็นในการผสมพันธุ์สูงที่สุดจะถูกเลือกเพื่อผสมพันธุ์และสเปิร์มของผึ้งตัวผู้จะถูกเก็บไว้ในถุงเก็บสเปิร์มของผึ้งนางพญาเพื่อใช้ในการปฏิสนธิกับไข่ในช่วงการวางไข่ การผสมพันธุ์ของผึ้งนางพญาแต่ละตัวสิ้นสุดลง เมื่อถุงเก็บสเปิร์มเต็มหรือค่าพลังงานน้อยกว่าค่าที่กำหนดไว้ ในช่วงฤดูวางไข่ ผึ้งนางพญาจะถูกสุ่มเลือกด้วยวิธีการเลือกแบบวงล้อหมุนเพื่อทำหน้าที่สร้างตัวอ่อนผึ้ง ผึ้งนางพญาที่ได้รับเลือกเริ่มต้นสร้างตัวอ่อนผึ้งโดยการสุ่มเลือกสเปิร์มจากถุงเก็บสเปิร์มเพื่อทำการครอสโอเวอร์กับจีโนไทป์ของตนเอง จากนั้นตัวอ่อนผึ้งที่ได้จะถูกนำไปมิวเตท กระบวนการสร้างตัวอ่อนผึ้งสิ้นสุดลงเมื่อจำนวนของตัวอ่อนผึ้งครบตามที่กำหนดไว้ หลังจากนั้นผึ้งงานซึ่งแทนด้วยฟังก์ชันฮิวริสติกจะทำหน้าที่ดูแลตัวอ่อนผึ้งโดยการปรับค่าจีโนไทป์ของตัวอ่อนผึ้ง ซึ่งประสิทธิภาพในการปรับค่าจีโนไทป์ตัวอ่อนผึ้งของผึ้งงานแต่ละตัวจะถูกนำไปคำนวณเป็นค่าฟิต-เนสของผึ้งงานนั้นด้วย เมื่อมีทั้งประชากรรุ่นพ่อแม่และประชารุ่นลูกเรียบร้อยแล้ว ขั้นตอนถัดไปคือ การคัดเลือกผึ้งนางพญาใหม่ที่เหมาะสมสำหรับการสร้างรังโดยการพิจารณาเปรียบเทียบจากค่าฟิตเนสระหว่างผึ้งนางพญาในรอบปัจจุบันกับตัวอ่อนผึ้ง ตัวอ่อนผึ้งที่มีค่าฟิตเนสสูงกว่าผึ้งนางพญา

ที่มีค่าฟิตเนสที่ต่ำที่สุดจะถูกเลือกให้เป็นผึ้งนางพญาคู่แข่ง หลังจากนั้นจึงทำการเลือกผึ้งนางพญาใหม่ที่เหมาะสมสำหรับการสร้างรังโดยที่ผึ้งนางพญาใหม่ต้องไม่ได้สร้างรังใกล้กันเกินกว่าค่าที่กำหนดไว้ จากนั้นจึงเริ่มการคัดเลือกผึ้งตัวผู้ใหม่ ผึ้งตัวผู้ที่ตายลงหลังจากการผสมพันธุ์หรือตายเมื่อครบอายุตามที่กำหนดไว้จะถูกแทนที่ด้วยตัวอ่อนผึ้งด้วยวิธีการสุ่ม และก่อนการทำงานรอบถัดไป ผึ้งตัวผู้ทั้งหมดจะถูกพิจารณาให้เป็นสมาชิกในรังของผึ้งนางพญาเพียงรังใดรังหนึ่ง การทำงานจะทำซ้ำจนกระทั่งครบจำนวนที่กำหนดไว้ ผึ้งนางพญาที่มีค่าฟิตเนสสูงที่สุดถือเป็นคำตอบของปัญหา

การทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอ ได้แบ่งเป็น 5 ส่วน ดังนี้

1) การทดสอบเพื่อศึกษาปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม

2) การทดสอบเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาและผึ้งตัวผู้ที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยมีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่

3) การทดสอบเพื่อศึกษาปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาและผึ้งตัวผู้ที่ส่งผลต่อประสิทธิภาพการทำงานของอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิม โดยไม่มีการกำหนดจำนวนของผึ้งนางพญา

4) การทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข โดยได้ทำการทดสอบกับฟังก์ชันมาตรฐาน จำนวน 6 ฟังก์ชันคือ ฟังก์ชัน Michalewicz ฟังก์ชัน Rastrigin ฟังก์ชัน Rosenbrocks' valley ฟังก์ชัน Sphere ฟังก์ชัน Weighted Sphere และฟังก์ชัน Goldstein-Price และได้เปรียบเทียบผลการทดลองกับอัลกอริทึมการผสมพันธุ์แบบดั้งเดิม

5) การทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด โดยได้ทำการทดสอบกับปัญหาการเดินทางของพนักงานขายที่นำมาจากฐานข้อมูล TSPLIB คือ ข้อมูล ST70 และข้อมูล EIL101 และได้เปรียบเทียบกับผลการทดลองในงานวิจัยของ Chen และ Chien [3]

5.2 สรุปผลการทดสอบ

1) ปัจจัยการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งทำให้ประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดดีขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมโดยเฉลี่ย 35.28% และกลไกการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งช่วยลดเวลาในการประมวลลดลงโดยเฉลี่ย 29.76%

2) ปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาโดยที่มีการกำหนดจำนวนผึ้งนางพญาเป็นค่าคงที่ทำให้ประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดดีขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมโดยเฉลี่ย 18.55% แต่ใช้เวลาในการประมวลสูงขึ้นโดยเฉลี่ย 57.46%

3) ปัจจัยการผสมพันธุ์ระหว่างรังของผึ้งนางพญาโดยที่มีการคัดเลือกผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังทำให้ประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดดีขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมโดยเฉลี่ย 20.16% และวิธีการคัดเลือกผึ้งนางพญาที่เหมาะสมช่วยลดเวลาในการประมวลลดลงโดยเฉลี่ย 58.95%

4) อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลข ซึ่งเกิดจากการรวมการทำงานของ 2 ปัจจัยคือ การคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้งและการผสมพันธุ์ระหว่างรังของผึ้งนางพญาโดยที่มีการคัดเลือกจำนวนผึ้งนางพญาที่เหมาะสมมีประสิทธิภาพในการค้นหาคำตอบที่เหมาะสมที่สุดดีขึ้นกว่าอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมโดยเฉลี่ย 48.05% และช่วยลดเวลาในการประมวลลดลงโดยเฉลี่ย 63.93%

5) อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดสามารถค้นหาค่าคำตอบที่ดีที่สุด ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยได้ดีกว่าอัลกอริทึม PGACS ทั้งข้อมูล ST70 และข้อมูล EIL101 สำหรับข้อมูล ST70 อัลกอริทึมที่นำเสนอสามารถค้นหาค่าคำตอบที่ดีที่สุด ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยได้ดีกว่าอัลกอริทึม PGACS 0.30%, 0.73% และ 0.30% ตามลำดับ นอกจากนี้อัลกอริทึมที่นำเสนอยังสามารถค้นหาค่าคำตอบที่เหมาะสมที่สุดได้ ในขณะที่อัลกอริทึม PGACS ไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ สำหรับข้อมูล EIL101 อัลกอริทึมที่นำเสนอสามารถค้นหาค่าคำตอบที่ดีที่สุด ค่าคำตอบที่แย่ที่สุดและค่าคำตอบเฉลี่ยได้ดีกว่าอัลกอริทึม PGACS 1.41%, 1.69% และ 1.24% ตามลำดับ รวมทั้งมีส่วนเบี่ยงเบนมาตรฐานที่ต่ำกว่าด้วย

5.3 จุดเด่นของงานวิจัย

1. ผึ้งนางพญาเลือกผสมพันธุ์กับผึ้งตัวผู้ได้มากกว่าหนึ่งตัวและสามารถเลือกผสมพันธุ์กับผึ้งตัวผู้จากรังอื่นๆ ได้จึงทำให้ตัวอ่อนผึ้งที่เกิดจากการครอสโอเวอร์ระหว่างผึ้งนางพญาและผึ้งตัวผู้มีโอกาสสูงที่จะเกิดจากจีโนไทป์พ่อและแม่ที่แตกต่างกันมากกว่าการเลือกผสมพันธุ์ภายในรังเดียวกัน ดังนั้น วิธีการเลือกผสมพันธุ์ลักษณะนี้จึงเป็นกลไกหนึ่งที่ช่วยให้สามารถสร้างตัวอ่อนผึ้งที่มีความหลากหลายได้มากยิ่งขึ้น

2. อัลกอริทึมที่นำเสนอมีการใช้โอเปอร์เรเตอร์การเลือก โอเปอร์เรเตอร์ครอสโอเวอร์ โอเปอร์เรเตอร์มิวเตชันและโอเปอร์เรเตอร์ผึ้งงานสำหรับการสร้างตัวอ่อนผึ้ง โดยที่มีการใช้โอเปอร์เรเตอร์ผึ้งงานมากกว่าหนึ่งตัว ทำให้ตัวอ่อนผึ้งที่สร้างขึ้นใหม่ถึงแม้ว่าจะเกิดจากจีโนไทป์พ่อและแม่เดียวกัน แต่มีโอกาสที่จะถูกเลี้ยงดูโดยโอเปอร์เรเตอร์ผึ้งงานที่แตกต่างกันได้ ดังนั้นการเลี้ยงดูจึงเป็นกลไกหนึ่งที่ช่วยป้องกันการตกในพื้นที่โลคอลได้

3. อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขและอัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัด มีกลไกการคัดเลือกจำนวนผึ้งนางพญาที่เหมาะสมในแต่ละ

รอบการทำงานทำให้ผู้ใช้ไม่จำเป็นต้องมีการกำหนดจำนวนของผึ้งนางพญาไว้เป็นค่าคงที่ เพราะการกำหนดจำนวนของผึ้งนางพญาให้เหมาะสมกับปัญหาที่ทดสอบเป็นอุปสรรคอย่างหนึ่งของผู้ใช้

4. ผึ้งนางพญาที่มีค่าฟิตเนสสูงที่สุดจะได้รับเลือกให้เป็นผึ้งนางพญาใหม่ในรอบการทำงานถัดไปโดยอัตโนมัติจึงทำให้ยังคงรักษาคำตอบที่ดีที่สุดไว้เสมอ และผึ้งนางพญาที่มีค่าฟิตเนสสูงจะได้รับโอกาสที่จะถูกเลือกให้เป็นผึ้งนางพญาใหม่สูงกว่าผึ้งนางพญาที่มีค่าฟิตเนสที่ต่ำกว่า นอกจากนี้ ผึ้งนางพญาที่มีค่าฟิตเนสสูงยังมีโอกาสที่จะสร้างรังได้ใหญ่กว่าผึ้งนางพญาที่มีค่าฟิตเนสที่ต่ำกว่าด้วย

5. อัลกอริทึมที่นำเสนอช่วยลดเวลาในการประมวลลงจากอัลกอริทึมการผสมพันธุ์ของผึ้งแบบดั้งเดิมได้ เพราะมีกลไกการคัดเลือกจำนวนผึ้งนางพญาที่เหมาะสมสำหรับการสร้างรังและมีการคัดเลือกผึ้งตัวผู้จากตัวอ่อนผึ้ง

5.4 ปัญหาที่พบในการงานวิจัย

อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงตัวเลขที่ใช้ในการทดสอบกับฟังก์ชันมาตรฐานใช้เวลาประมวลผลน้อย แต่อัลกอริทึมที่นำเสนอสำหรับปัญหาออฟติไมเซชันเชิงการจัดที่ใช้ในการทดสอบกับปัญหาการเดินทางของพนักงานขายได้ใช้เวลาในการประมวลผลนาน ดังนั้นจึงควรปรับปรุงการทำงานของโอเปอร์เรเตอร์ให้สามารถทำงานได้รวดเร็วขึ้น

5.5 แนวทางในการพัฒนาต่อในอนาคต

1. พัฒนาอัลกอริทึมที่นำเสนอร่วมกับอัลกอริทึมอื่นๆ เพื่อให้มีประสิทธิภาพในการแก้ปัญหาออฟติไมเซชันมากยิ่งขึ้น เช่น การนำอัลกอริทึมที่นำเสนอมาผสมผสานกับอัลกอริทึมการค้นหาแบบตาบู (tabu search) โดยการปรับใช้อัลกอริทึมการค้นหาแบบตาบูให้ทำหน้าที่เป็นหน่วยจัดเก็บข้อมูลประสบการณ์ในการค้นหาให้กับอัลกอริทึมที่นำเสนอ เป็นต้น

2. พัฒนาอัลกอริทึมที่นำเสนอเพื่อให้สามารถใช้ในการแก้ปัญหาอื่นๆ ได้นอกเหนือจากปัญหาออฟติไมเซชัน เช่น การพัฒนาอัลกอริทึมที่นำเสนอให้สามารถทำงานร่วมกับอัลกอริทึมโครงข่ายประสามเทียมแบบย้อนกลับ (backpropagation neural network) เพื่อประยุกต์ใช้ในการแก้ปัญหาการจัดกลุ่มข้อมูล (classification) โดยการปรับใช้อัลกอริทึมที่นำเสนอในขั้นตอนการปรับค่าถ่วงน้ำหนักของอัลกอริทึมโครงข่ายประสาทเทียมแบบย้อนกลับ เป็นต้น

เอกสารอ้างอิง

[1] Abbass H. A. "MBO : Marriage in honey-bee optimization (A haplometrosis polygynous swarming approach)" **Proceeding of the Congress on Evolutionary Computation (CEC2001)**, 2001. pp. 207–214.

[2] Haddad O. B., Afshar A. and Marino M. A. "Honey-bee mating optimization (HBMO) algorithm: A new heuristic approach for water resources optimization" **Water Resources Management**, vol. 20, 2006. pp. 661–680.

[3] Chen S.-M, Chien C.-Y. "Parallelized genetic ant colony systems for solving the traveling salesman problem" **Expert Systems with Application**, vol. 38, 2011. pp. 3873-3883.

[4] Nocedal J., Wright S. J. **Numerical Optimization (Springer series in Operation Research and Financial Engineering.** New York : Springer. 2009.

[5] Belegundu A. D., Chandrupatla T. R. **Optimization Concept and Applications in Engineering**. New York : Cambridge University Press. 2011.

[6] Fonseca C. M., Flemming P. J. "An overview of evolutionary algorithms in multiobjective optimization," **Evolutionary Computation**, vol. 3(1). 1995. pp. 1–16.

[7] Papadimitriou C. H., Steiglitz K. **Combinatorial optimization: Algorithm and Complexity.** New Jersey : Prentice-Hall. 1982.

[8] Gutin G., Yeo A. and Zverovitch A. "Exponential neighborhoods and Domination analysis for the TSP" in Gutin G., Punnen A. **The traveling salesman problem and its variations. Boston** : Kluwer Academic Publishers. 2002.

[9] Whitehead S. B., Shaw F. R. **Honeybees and their management.** New York : Van Nostrand. 1951.

[10] Gould J. L., Gould C. G. **The honey bee.** New York : Scientific American Library. 1995.

[11] Gen M., Cheng R. **Genetic Algorithm and Engineering Design**. New York : John Wiley and Sons. 1997.

[12] Mohan S., Jinesh Babu K. S. "Optimal Water Distribution Network Design with Honey-Bee Mating Optimization" **Journal of computing in civil engineering**, 2010. pp. 117-126.

[13] Alperovits E., Shamir U. "Design of optimal water distribution systems" **Water Resource Res.**, vol. 13(6). 1977. pp. 885–900.

[14] Fujiwara O., Khang D. B. "A two-phase decomposition method for optimal design of looped water distribution networks" **Water Resource Res.**, vol. 26(4), 1990. pp. 539–549.

[15] Li-ning X., Ying-wu C. and Huai-ping C. "An intelligent genetic algorithm designed for global optimization of multi-minima functions" **Applied Mathematics and Computation**, vol. 178, 2006. pp. 355–371.

[16] Wang L., Tang F. and Wu H. "Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation" **Applied Mathematics and Computation**, vol. 171, 2005. pp. 1141–1156.

[17] Kao Y., Zahara E. "A hybrid genetic algorithm and particle swarm optimization for multimodal functions" **Applied Soft Computing**, 2008. pp. 849–857.

[18] Abd-El-Waheda W. F., Mousa A. A. and El-Shorbagy M. A. "Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems" **Journal of Computational and Applied Mathematics**, vol. 235, 2011. pp. 1446–1453.

[19] Duran Toksarı M. "Minimizing the multimodal functions with Ant Colony Optimization approach" **Expert Systems with Applications**, vol. 36, 2009. pp. 6030–6035.

[20] Dixon L. C. W., Szegö G. P. "The global optimization problem an introduction" in Dixon L. C. W., Szegr G. P. (Eds.) **Towards global optimization**. Amsterdam: North-Holland, 1978. pp. 1–15.

[21] Oliver I. M., Smith D. J. and Holland J. R. C. "A study of permutation crossover operators on the TSP" **Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications,** 1987. pp. 224-230.

[22] Murata T., Ishibuchi H. "Performance evaluation of genetic algorithms for flowshop scheduling problems" **Proceedings of the First IEEE Conference on Evolutionary computation**, 1994. pp. 812–817.

[23] Michalewicz Z. **Genetic Algorithms + Data Structures = Evolution Programs**. Berlin : Springer-Verlag. 1992.

[24] Banzhaf W. "The Molecular travelling salesman" **Biological Cybernetics**, vol. 64, 1990. pp. 7-14.

[25] Fogel D. B. "An evolutionary approach to the travelling salesman problem" **Biological Cybernetics**, vol. 60, 1988. pp. 139-144.

[26] Holland J. **Adaptation in natural and artificial systems.** Ann Arbor : University of Michigan Press. 1975.

[27] Fogel D. B. "Applying evolutionary programming to selected travelling salesman problems" **International Journal of Cybernetics and Systems,** vol. 24, 1993. pp. 27-36.

[28] Ulder N. L. J., Aarts E. H. L., Bandelt H. -J., van Laarhoven P. J. M. and Pesch E. "Genetic Local Search Algorithms for the Traveling Salesman Problem" **Proceedings of the 1st Workshop on Parallel Problem Solving from Nature**, 1990. pp. 106-116.

[29] Törn A., Zilinskas A. **Global optimization (Lecture Notes in Compute Science).** Berlin : Springer-Verlag. 1989.

[30] De Jong K. A. "**An analysis of the behavior of a class of genetic adaptive systems.**" Doctoral dissertion of University of Michigan. 1975.

[31] Pohlheim H. "The genetic and evolutionary algorithm toolbox for Matlab." [Online]. Available : http://www.geatbx.com/docu/fcnindex-01.html. 2006.

[32] Goldstein A. A., Price J. F. "On descent from local minima" **Mathematics of Computation**, vol. 25(115), 1971. pp. 569-574.

[33] Reinelt G. "TSPLIB - A traveling salesman library" **ORSA Journal on Computing**, vol. 3(4), 1991. pp. 376-384.

ภาคผนวก

ภาคผนวก ก
ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

Thammano A., Poolsamran P. "SMBO: A self-organizing model of marriage in honey-bee optimization" **Expert Systems with Applications**, vol. 39, 2012. pp. 5576-5583.

Poolsamran P., Thammano A. "The modified marriage in honey-bee optimization for function optimization problems" **Procedia Computer Science**, vol. 6, 2011. pp. 335--342.

Poolsamran P., Thammano A. "The modified marriage in honey-bee optimization for multiobjective optimization problems," **Proceeding of 6th International IEEE Conference on Intelligent Systems (IS'12)**, 2012.

Expert Systems with Applications 39 (2012) 5576–5583

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

SMBO: A self-organizing model of marriage in honey-bee optimization

Arit Thammano *, Patcharawadee Poolsamran

Computational Intelligence Laboratory, Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand

ARTICLE INFO

Keywords:
Marriage in honey-bee optimization
Artificial bee
Function optimization
Swarm intelligence

ABSTRACT

This paper proposes a novel swarm intelligence technique, which is an adaptation of Abbass's marriage in honey-bee optimization (MBO), with the aim to achieve better overall performance than the original version of the MBO while also lowering the computation time for finding the optimal solution. The original MBO has been proven to be one of the best swarm intelligence algorithms for solving optimization problems. However, many parameters need to be properly set in order for the MBO to perform at its best. Therefore, long computation time caused by a large number of trial and error iterations involved in trying to find the right combination of parameters is unavoidable. The framework of the proposed algorithm is similar to the original MBO, which is based on the marriage behavior of honey-bees. In order to improve the efficiency of the MBO algorithm, several aspects of the original MBO have been adapted, such as (1) the proposed algorithm is adapted to obtain the ability to automatically search for the proper number of queens, (2) the proposed algorithm divides the problem space into several colonies, each of which has its own queen. In order to keep the number of colonies to a minimum, the proposed algorithm, therefore, encourages the queens to compete with each other for a larger colony and also urges the newly-born brood which is fitter than the queen of the colony to overthrow the queen. (3) the fuzzy c-means algorithm is employed to assign the drones to the proper colonies. The proposed algorithm has been evaluated and compared to the original MBO algorithm. The experimental results on six benchmark problems demonstrate the potential of the proposed algorithm in offering an efficient and effective solution to the problem.

1. Introduction

Finding the globally optimal solution for the complex multi-variable optimization problems may require huge computation time, especially when the number of variables is large. Therefore, the use of metaheuristics, which sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time, has received more and more attention in the last 30 years (Blum, Aguilera, Roli, & Sampels, 2008). Recently, many researchers have focused their attention on a new class of biologically inspired metaheuristic algorithms, called swarm intelligence. Swarm intelligence techniques are inspired by the social behavior of flocking animals such as colonies of ants, flocks of birds, schools of fish, and swarms of bees. The main difference between the swarm intelligence algorithms and the local search algorithms, such as hill climbing, simulated annealing, and tabu search, is that the swarm intelligence algorithms are population-based metaheuristics, while the local search algorithms use a single solution at each step (Pham et al., 2006).

Two of the most successfully swarm intelligence algorithms are Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). They have been applied successfully to a variety of problems such as the traveling salesman problem (Dorigo & Di Caro, 1999), scheduling problems (Blum & Sampels, 2004), vehicle routing problems (Rizzoli, Montemanni, Lucibello, & Gambardella, 2007), and classification problems (Zahiria & Seyedin, 2007). Besides the ongoing popularity of ACO and PSO, studies of honey bee behavior are in an increasing trend during the last few years. There are several algorithms which imitate the behavior of bees in nature. Those algorithms can be categorized into two groups based on the behavioral characteristics of honey bees. The first group is based on the food foraging behavior while the second group is inspired by the marriage behavior. Examples of the first group include the Artificial Bee Colony (ABC) (Karaboga, 2005), the Bee System (BS) (Lučič & Teodorović, 2003), the Bee Colony Optimization (BCO) (Teodorović & Dell'Orco, 2005), and the Bees Algorithm (BA) (Pham et al., 2006). At the beginning of the foraging process, the scout bees are sent out to search for promising flower patches. When the scout bees return to the hive, they perform a waggle dance to share their discoveries with others. This dance is actually the bee language which contains three pieces of information: the direction to the flower patch, the distance from the hive to the flower patch, and the amount of nectar

* Corresponding author. Tel.: +66 2 723 4964; fax: +66 2 723 4910.
E-mail addresses: arit@it.kmitl.ac.th (A. Thammano), patcharp@buu.ac.th (P. Poolsamran).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.11.060

available in the flower patch. The colony then evaluates the relative merit of different flower patches presented by the scout bees. After the evaluation, the forager bees follow the scout bees back to the flower patches to collect nectar. The more nectar a flower patch has, the more forager bees are sent there (Baykasoğlu, Ozbakır, & Tapkan, 2007; Pham et al., 2006). The foraging based algorithms have been applied to solve many problems such as optimization problems (Karaboga & Basturk, 2007, 2008), vehicle routing problems (Teodorović & Dell'Orco, 2005), and pattern recognition problems (Pham, Otri, Ghanbarzadeh, & Koç, 2006).

The other approach is inspired by the marriage behavior of honey bees. In nature, a honey-bee colony consists of the queen(s), drones, workers, and broods. The queen is the only bee in the colony with fully developed ovaries. She mates only once in her lifetime, but with seven to twenty drones (Abbass & Teo, 2003). After mating, she begins laying hundreds of eggs a day (National Research Council, 1991). Those that are fertilized become female bees, either queens or workers, while those that remain unfertilized develop into drones. The Marriage in Honey-bee Optimization (MBO) is the best representation of this group. Afshar, Bozorg Haddad, Marino, and Adams (2007) successfully applied the honey bee mating optimization (HBMO) algorithm to solve the single reservoir operation optimization problems. Later, Fathian and Amiri (2008) proposed a two-stage hybrid system which is a combination of the self-organizing feature map (SOM) and the MBO algorithm to solve the clustering problems. They compared the proposed hybrid algorithm with other metaheuristic algorithms, such as ant colony optimization, genetic algorithm, simulated annealing, and tabu search. Results show that the proposed algorithm is the best performer both in terms of finding the optimal solution and the processing time. Curkovic and Jerbic (2007) tested the performance of the MBO in solving the nonlinear Diophantine equations and the path planning problem. Results show that the MBO outperforms GA on both problems.

Even though many studies suggest that the MBO is a very powerful optimization technique, one main drawback of the MBO is the time it takes for the resolution when the number of queens increases. Therefore, the aim of this paper is to propose an improved version of the MBO algorithm. The proposed algorithm can significantly reduce the computation time required for finding the optimal solution, and at the same time achieve a better performance for the nonlinear, single objective, multi-variable optimization problems.

The rest of this paper is organized as follows: Section 2 briefly describes the original MBO algorithm. Section 3 presents the proposed algorithm and also points out the modifications made to the original MBO algorithm in order to achieve a better performance. The description of the benchmark functions, their global optimum, and the experimental results are given in Section 4. Finally, Section 5 is the conclusion.

2. Marriage in honey-bee optimization

The MBO belongs to a class of swarm intelligence. It attempts to model the marriage behavior of real honey-bees in a bee colony. A honey-bee colony typically consists of the queen(s), drones, workers, and broods. Each member has a specific task to do in the colony. The queen is the mother of all other bees in the colony. The most important function of the queen is to lay eggs to produce offspring. Drones are the fathers of the colony. The only task of drones is to locate and mate with the queen during the mating flight. In the marriage process, the queen starts waggling dance before she flies far from the hive in order to mate with the drones, usually from other colonies (Modern Agro, 2006). The queen generally mates with the drones that fly the highest and are able to catch her (Horn, 2005). After the mating, the drones immediately die. The queen continues to mate until her spermatheca, where sperm of the participated drones are stored, is full. Once her spermatheca is full of sperm, the queen will return to the hive and begin laying both fertilized and unfertilized eggs. The queen lays fertilized eggs by randomly retrieving sperm from her spermatheca to fertilize the eggs. In fact, only queens and workers arise from the fertilized eggs, while drones arise from the unfertilized eggs. Workers are female bees which do all the work in the colony, for instance, nest construction, food foraging, feeding, and brood care (Abbass, 2001; Abbass & Teo, 2003; Fathian & Amiri, 2008; Pai, Yang, & Chang, 2009).

The MBO algorithm can be divided into four main stages as presented in Fig. 1. The first stage starts with initializing the parameters of the MBO algorithms. The parameters needed to be defined are the number of queens, the number of drones, the number of workers, the size of the queen's spermatheca, the number of potential broods, the mutation rate, and the number of mating flights.

The second stage is the mating process. Before the mating flights begin, both energy and speed of each queen are randomly initialized with values in the range [0.5, 1] (Abbass & Teo, 2003).

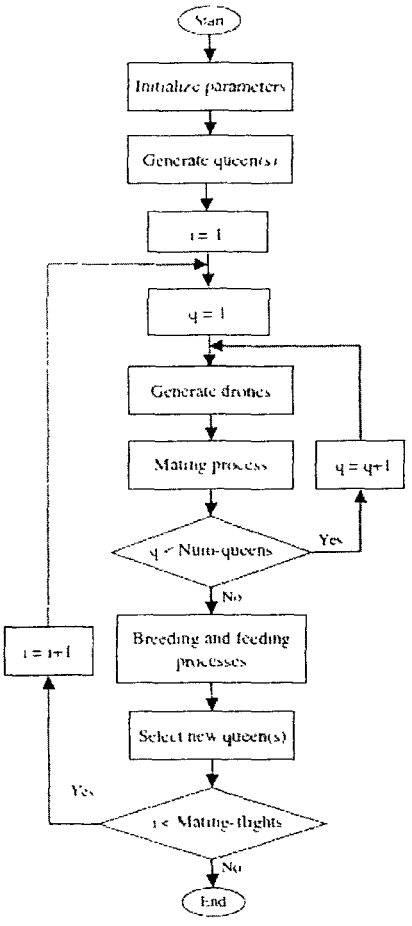


Fig. 1. The flowchart of MBO algorithm.

During the mating flight, a queen mates with the drone that has the highest marriage probability.

$$prob(q_n, d_p) = e^{\frac{-\Delta f}{S(t)}} \quad (1)$$

where prob (q_n, d_p) is the probability of a successful mating between the queen q_n and the drone d_p; $\Delta(f)$ is the absolute difference between the fitness value of the queen q_n and the fitness value of the drone d_p, $S(t)$ is the speed of the queen at time t. The probability of mating is high when either the queen's speed is high or the fitness of the drone is as good as the queen's. If the mating is successful, the selected drone's genotype will be stored in the queen's spermatheca. After each transition, the queen's speed and energy decline in accordance with the following equations (Abbass, 2001):

$$S(t+1) = \alpha \times S(t) \quad (2)$$

$$E(t+1) = E(t) - \gamma \quad (3)$$

where $E(t)$ is the energy of the queen at time t; α is a decay factor, which has a value between 0 and 1; γ, the amount of energy reduction after each transition, is computed by using the following equation (Abbass & Teo, 2003):

$$\gamma = \frac{0.5 \times E(0)}{M} \quad (4)$$

where M is the queen's spermatheca size. The queen continues to mate with other drones until her energy is lower than the predetermined value of E_{min} or her spermatheca is full. Then the queen returns to her hive to begin breeding. This second stage is repeated until all queens complete their mating flights.

The third stage is the breeding and feeding processes. When all queens complete their mating flight, they return to their hive and the breeding process begins. Similar to the genetic algorithm, the breeding process is based on the three operations of selection, crossover, and mutation. In the selection operation, a queen is chosen with a probability proportional to her fitness value, and then a sperm is randomly selected from the chosen queen's spermatheca. A new brood is generated by crossover the selected sperm's genotype with the queen's genotype. Mutation is then applied to the new brood in order to provide greater variety to the solutions.

In the feeding process, the genotype of the newly-born brood is further improved by a worker, which is chosen in proportion to its fitness. There are several workers in a hive. Each worker represents different heuristic function such as GSAT, random walk, WalkSAT, and one-point crossover (Abbass & Teo, 2003). Their fitness values are updated according to the rate of improvement achieved on the broods. When the number of broods is equal to the number of potential broods, the breeding and feeding processes are complete.

Finally, the fourth stage starts by sorting the new improved broods according to their fitness values. In a colony with a single queen, the queen is replaced with the fittest brood if the latter is fitter than the former. Similarly, in a colony with multiple queens, the weaker queens are replaced with the younger and fitter broods until none of the remaining broods is fitter than any of the queens. All remaining broods are then killed and the new mating flight begins. This loop is repeated until a predetermined number of iteration is reached or the convergence criteria are met. A pseudo code of the MBO algorithm is illustrated in Fig. 2.

3. Methodology

The self-organizing model of marriage in honey-bee optimization (SMBO) presented in this section is an improvement over the original MBO in several respects. Firstly, the SMBO divides the problem space into several colonies, each of which has its own queen. The size of each colony depends on the fitness of the corresponding queen. The fitter queen rules the larger colony than

```
Algorithm: Marriage in Honey-bee Optimization
    Generate the initial population and define the initial parameters.
    Evaluate the fitness of all individuals in the population
    Select the N best individuals from the population and assign them to be the queens
    For a predefined number of mating flights
        For each queen in the queen list
            Initialize the queen's energy and speed
            Generate a pool of drones and evaluate their fitness values
            While energy > E_min and Spermatheca is not full
                Select a drone with the highest marriage probability
                Add sperm of the selected drone in the queen's spermatheca
                Update the queen's energy and speed.
            End while
        End for
        While a predefined number of broods is not reached
            Select a queen using the roulette wheel selection method
            Randomly select a sperm from the spermatheca of the selected queen.
            Generate the broods by crossover the queen's genotype with the selected sperm's genotype
            Mutate the newly created broods
            Use the workers to improve the mutated broods
            Update the fitness of the selected workers
        End while
        Evaluate the fitness of all the broods
        While the best brood is better than the worst queen
            Replace the least fit queen with the best brood
            Remove the best brood from the brood list
        End while
        Kill all remaining broods
    End for
Return the best queen
```

Fig. 2. Pseudo code for MBO algorithm

the weaker one. Secondly, the parameter which controls the territorial boundary of the colony is introduced in the proposed algorithm to prevent the queens from staying too close to each other. The newly-born brood of one colony which is fitter than the queen of the colony must overthrow the queen, instead of founding a new colony. As a consequence, the proposed algorithm can limit the number of colonies to a very small number. Thirdly, all drones have their own life span. As opposed to the original MBO, the unmated drones survive until their life span is reached. Fourthly, the queen is prevented from mating with the drones from her own colony in order to reduce the probability of trapping in the local optimum area. However, if there is no drone in the other colonies, the queen will mate with her own drones. Lastly, the clustering technique based on the fuzzy logic concept is employed to assign all drones to the proper colonies.

The proposed algorithm is divided into five stages: (1) the population initialization, (2) the mating process, (3) the breeding and feeding processes, (4) the selection of the new queens and drones, and (5) the founding of the new colonies. In the first stage, the initial population is created by randomly generating N individuals. Each individual, represented by the binary string, is a candidate solution to the problem. Its length depends on the number of variables of the problem considered, the range of each variable, and the decimal point precision. For example, in Fig. 3, let us consider a problem of finding the values of x_1 and x_2 which minimize the function $f(x_1,x_2)$, where x_1 and x_2 are real numbers with a level of precision of 0.25, $x_1 \in [-7,7]$ and $x_2 \in [-2,2]$. In this example, the length of each individual is 11 bits. Out of 11 bits, the first six bits represent the value of x_1, which consists of a sign bit, three exponent bits, and two mantissa bits. The remaining bits represent the value of x_2, which consists of a sign bit, two exponent bits, and two mantissa bits. Then the fitness value of each individual is evaluated. The best individual of the initial population is selected to be the queen of the initial hive, while the rest is assigned to be the drones.

In the second stage, the mating process starts by randomly initializing the queen's energy and speed. Then the probability that the queen q_n will mate with the drone d_n is evaluated by using the Eq. (1). The probability of mating is high when either the fitness of the drone is as good as that of the queen, or the queen's speed is high. The drone with the highest probability is selected to mate with the queen. The selected drone's genotype is stored in the queen's spermatheca. Then the selected drone dies and is removed from the drone population. After each mating, the queen's speed and energy decline in accordance with the Eqs. (2) and (3). However, the queen continues to mate with other drones until her spermatheca is full or her energy is lower than the predetermined value of E_{min}.

When all queens complete their mating flight, they return to their hives and the third stage begins. In the third stage, a queen is chosen from the pool of queens using the roulette wheel selection scheme. Then the chosen queen reproduces the broods by crossover her genotype with the genotype of the randomly selected sperm from her own spermatheca. In this paper, the two-point crossover is used, as shown in Fig. 4. Next, the mutation operation is carried out on every new brood created. Since the brood's genotype is a binary string, the standard binary mutation operation, flipping

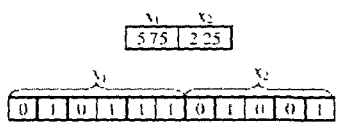


Fig. 3. An example of the bee's genotype.

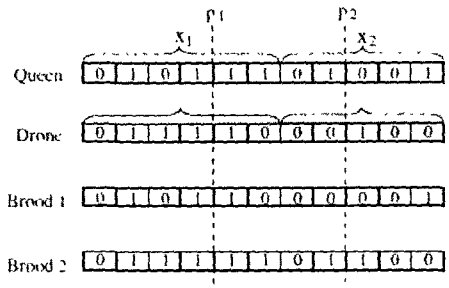


Fig. 4. An example of the crossover operation.

the value of the chosen bit from 1 to 0 or vice versa, is used on each brood. The last process of the third stage is the feeding process. The workers, which represent the local search heuristic functions, are used to improve the genotypes of the newly-born broods. For each brood, a worker is selected from the worker pool by using the roulette wheel selection scheme. The heuristic function, which the selected worker represents, is then applied to the brood. This research utilizes five local search heuristic functions, namely random flip, random new, random walk, two point crossover, and one-point crossover. Therefore, there are total of 5 workers in this research.

In the fourth stage, the fitness values of all the broods created in the third stage are determined. Then the current queens and the broods which are fitter than the weakest queen are nominated as the candidates for the queens of the next generation. The concept of selecting the queens of the next generation is that the candidate with better fitness value is given priority over the weaker one. Therefore, before the selection process begins, the candidates are ranked according to their fitness values. The top-ranked candidate is promoted to be the first queen. The second-ranked candidate, who is next in turn, will be chosen as another queen if it is not a member of the colony which is ruled by the previously selected queen. The candidate C is said to be a member of the colony N if the criterion in (5) is met.

$$d(C,q_N) < \xi \tag{5}$$

$$N = \underset{1 \le n \le S}{\arg\min}\, d(C,q_n) \tag{6}$$

$$\xi = 0.582574 \times 10^{-5} A + 0.058878 \tag{7}$$

$$A = \prod_{i=1}^{m} \left(x_i^{max} - x_i^{min}\right) \tag{8}$$

where $d(C,q_n)$ is the Euclidean distance between the candidate C and the nth queen.

S is the number of previously selected queens.

N is the index of a colony whose queen is the closest match to the candidate C.

ξ is the territorial boundary of the colony. If the candidate stays closer to the queen of the N colony than ξ, the candidate will be considered as a member of the N colony. However, if the distances between the candidate and all previously selected queens are greater than the predefined boundary ξ, the candidate will be assigned as a new queen.

x_i^{max} is the upper bound of the input x_i.

x_i^{min} is the lower bound of the input x_i.

m is the dimensionality of the problem space.

The selection process continues with the lower-ranked candidates until all the candidates have been considered. It is worth to note that in the above selection process the current queen might

not be re-selected to be a new queen. After the selection process ends, the current queens which have not been re-selected will be killed and removed from the population. However, the unselected candidates which are nominated from the group of broods will not be killed; instead, they will be grouped with the weaker broods who have not been nominated as the queen candidate. The members of this new group will have a chance to be a drone by filling the vacancies left by the death of the older drones. In this research, there are two causes of death of a drone: (1) death after mating with the queen, and (2) the end of its life span. Life span of a drone is predetermined by the researcher in the first stage of the algorithm. The drones that mate with the queen immediately die whereas the unmated ones stay alive to the next generation. However, their life spans are reduced by one. When the life span reaches zero, the unmated drone dies and is removed from the drone population. In each generation, many drones die from either of the above two causes. In order to maintain the number of drones at the predetermined value, replacements for the dead drones are randomly selected one by one from the newly formed group, mentioned earlier, until the predetermined value is reached.

The fifth stage is the colony founding stage. The main task of the fifth stage is to assign all drones, both old and new, to the proper colonies. It should be noted that even though the old drone currently belongs to one of the colonies, it must be reassigned again at this stage. Some might stay with the same queen, while others might not. In this stage, the concepts of the clustering algorithm and the fuzzy membership function are employed in clustering drones with similar contents together. Given $D = \{d_1, d_2, \ldots, d_p, \ldots, d_P\}$ is a set of drones, and P is the total number of drones. The fuzzy membership value of the drone d_p in the colony owned by the queen q_n is calculated according to the following equation:

$$\mu(d_p, q_n) = e^{\left(\frac{-\|y_p - c_n\|^2}{\sigma_{q_n}^2}\right)} \tag{9}$$

$$\sigma_{q_n} = \max_k \left[d(q_n, q_k) \times \frac{f(q_n)}{f(q_n) + f(q_k)} \right] \tag{10}$$

where $\mu(d_p, q_n)$ is a membership value of the drone d_p in the colony of the queen q_n.

σ_{q_n} denotes the size of the colony of the queen q_n. The value of σ_{q_n} is proportional to the fitness of the corresponding queen q_n.

$k = 1, 2, \ldots$, the total number of queens.

$f(q_n)$ and $f(q_k)$ are the fitness values of the queen q_n and q_k respectively.

Algorithm: A Self-organizing Model of Marriage in Honey-bee Optimization (SMBO)

```
Define the initial parameters
Create the initial population
Select the best individual of the initial population to be the queen and assign the rest to be the drones
For a predefined number of mating flights
    For each queen in the queen list
        Initialize the queen's energy and speed
        While energy > E_min and spermatheca is not full
            Select a drone with the highest marriage probability to mate with the queen
            Add sperm of the selected drone in the queen's spermatheca
            Update the queen's energy and speed
        End while
    End for
    While a predefined number of broods is not reached
        Select a queen using the roulette wheel selection method
        Randomly select a sperm from the spermatheca of the selected queen.
        Reproduce the broods by crossover the queen's genotype with the genotype of the selected sperm
        Mutate the newly created broods
        Use the workers to improve the mutated broods
        Update the fitness of the selected workers
    End while
    Update a life span of every unmated drone
    Create a list of candidates for the queens of the next generation.
    Rank the candidates according to their fitness values from the highest to the lowest
    Assign the top-ranked candidate as the first new queen.
    For each candidate in the ranked list, starting from the second-ranked to the last-ranked candidates
        If the distances between the candidate and all the new queens are greater than
            Assign the candidate as a new queen.
        Else
            If the candidate is one of the current queens
                Remove the candidate from the population
            Else
                Add the candidate to a new list to be used as potential candidates for the new drones
            End if
        End if
    End for
    Create a list of potential candidates for the new drones
    Randomly select the new drones to fill the vacancies left by the death of the old drones.
    Assign each drone to a proper colony
End
Return the best queen
```

Fig. 5. Pseudo code for SMBO algorithm.

$d(q_n, q_k)$ is the Euclidean distance between the queen q_n and its neighboring queen q_k.

Subsequently, each drone determines its new hive by considering its membership value in each colony. The colony with the maximum membership value is selected as the new home for the drone.

$$K = \arg\max_i [\mu(d_p, q_k)]. \tag{11}$$

Once all drones are assigned to their new colony, the termination criterion will be checked. If it is not met, the whole algorithm will repeat again. However, a new group of colonies which contains both queens and drones will be used as the population for the next generation. A pseudo code of the proposed SMBO algorithm is presented in Fig. 5.

4. Experimental results

4.1. Test functions

The performance of the proposed algorithm is evaluated and compared to the original MBO algorithm. To test the performance, both algorithms were used to find the optimal solutions of the following six widely used benchmark functions: the Michalewicz's function, the Rastrigin's function, the Rosenbrock's valley function, the sphere function, the weighted sphere function, and the Goldstein–Price's function. In an optimization problem, the goal is to find the values of the decision variables that maximize or minimize the objective function without violating any of the constraints. The above six functions used in this paper represent both maximization and minimization problems. For clearer understanding, the equations of the six functions used in this paper are briefly described as follows:

a. The Michalewicz's function (Michalewicz, 1996)

$$f(\vec{x}) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2) \tag{12}$$

where $-3.0 \leqslant x_1 \leqslant 12.1$ and $4.1 \leqslant x_2 \leqslant 5.8$. The global maximum of this Michalewicz's function is equal to 38.850294479, attained at $11.625543700 \leqslant x_1 \leqslant 11.625545700$ and $x_2 = 5.725044250$. The graph of this function is shown in Fig. 6a.

b. The Rastrigin's function (Törn & Zilinskas, 1989)

$$f(\vec{x}) = 10m + \sum_{i=1}^{m} (x_i^2 - 10\cos(2\pi x_i)) \tag{13}$$

where m is the dimensionality of the problem space. The search space is restricted to $-5.12 \leqslant x_i \leqslant 5.12$, $i = 1,2,\ldots,m$. Its global minimum of zero is attained at the point $\vec{x} = (0,0,\ldots,0)$. The two-dimensional graph of this function is shown in Fig. 6b.

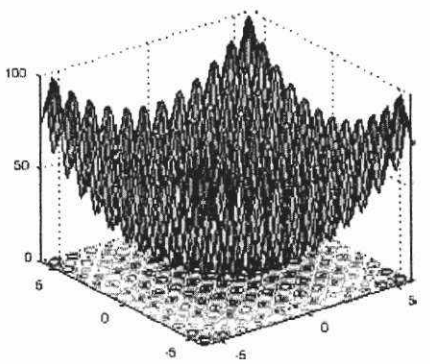


Fig. 6b. The Rastrigin's function.

c. The Rosenbrock's valley function (De Jong, 1975)

$$f(\vec{x}) = \sum_{i=1}^{m-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right] \tag{14}$$

The search domain is $-2.048 \leqslant x_i \leqslant 2.048$, $i = 1,2,\ldots,m$. The two-dimensional Rosenbrock's valley function has a global minimum at the point $\vec{x} = (1,1)$ where $f(\vec{x}) = 0$. The graph of this function is shown in Fig. 6c.

d. The sphere function (De Jong, 1975)

$$f(\vec{x}) = \sum_{i=1}^{m} x_i^2 \tag{15}$$

where m is the dimensionality of the problem space. The search space is restricted to $-5.12 \leqslant x_i \leqslant 5.12$, $i = 1,2,\ldots,m$. The global minimum of the sphere function is equal to zero, attained at $\vec{x} = (0,0,\ldots,0)$. The two-dimensional plot of this function is shown in Fig. 6d.

e. The weighted sphere function (Pohlheim, 2006)

$$f(\vec{x}) = \sum_{i=1}^{m} i^2 x_i^2 \tag{16}$$

where m is the dimensionality of the problem space. The search space is restricted to $-5.12 \leqslant x_i \leqslant 5.12$, $i = 1,2,\ldots,m$. Its global

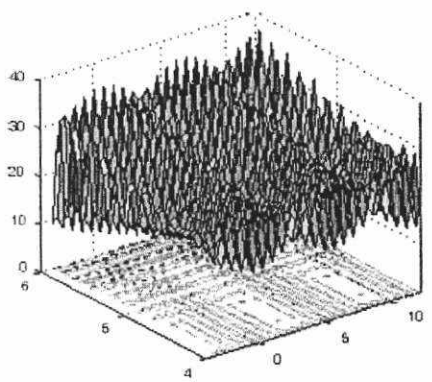


Fig. 6a. The Michalewicz's function.

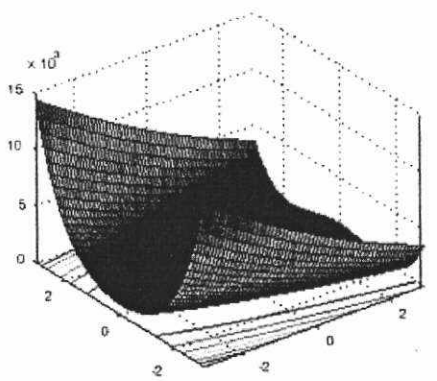


Fig. 6c. The Rosenbrock's valley function.

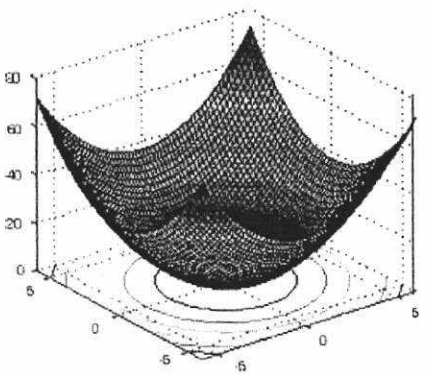


Fig. 6d. The sphere function.

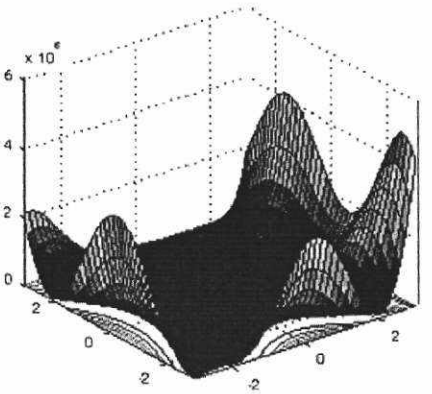


Fig. 6f. The Goldstein–Price's function.

minimum of zero is attained at the point $\vec{x} = (0, 0, \ldots, 0)$. The two-dimensional plot of this function is shown in Fig. 6e.

f. The Goldstein–Price's function (Goldstein & Price, 1971)

$$f(\vec{x}) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right] \quad (17)$$

where $-2.0 \leq x_1 \leq 2.0$ and $-2.0 \leq x_2 \leq 2.0$. This function has a global minimum at the point $\vec{x} = (0, -1)$ where it attains a value of 3. The graph of this function is shown in Fig. 6f.

4.2. Results and discussions

One of the most difficult steps in conducting an experiment is defining the values of the parameters of both the original and the proposed MBO algorithms. For the original MBO algorithm, the parameters needed to be defined are the number of queens, the number of drones in each generation, the size of the queen's spermatheca, the number of broods in each generation, the number of mating flights, and the mutation rate. For the proposed algorithm, however, only five parameters are needed to be defined which are the number of drones in each generation, the size of the queen's spermatheca, the number of broods in each generation, the number of mating flights, and the mutation rate. Since the performance of the MBO-based intelligent systems strongly depends on proper selection of system parameters, as most of other intelligent systems, the parameters of both models are varied to various values, as shown in Tables 1 and 2, in order to get the best out of both models. For each parameter setting, moreover, ten experimental repetitions are performed with different initial populations each time. For each test function, therefore, fifty experiments need to be conducted in case of the original MBO algorithm while only ten experiments are performed in case of the proposed algorithm. This is the first advantage of the proposed algorithm. In contrast to the original MBO algorithm, where the experiments with various numbers of queens have to be conducted to find out the optimal number, the proposed algorithm determines the proper number of queens automatically. From this ability, the computation time required for finding the optimal solution is several orders of magnitude less than the original MBO algorithm.

The performance of both the original MBO and the proposed SMBO algorithms in finding the optimal solutions of the six benchmark functions is shown in Table 3. Table 3 illustrates the best solution, the worst solution, and the average and the standard deviation over the number of experiments carried out for each problem. The following are summaries of what we observed from the experimental results:

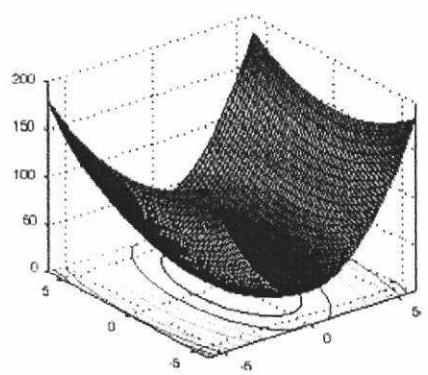


Fig. 6e. The weighted sphere function.

Table 1
The parameter used in the original MBO.

Parameters	Values
Number of queens	1, 10, 20, 30, 40
Spermatheca size	100
Number of drones	200
Number of broods	500
Number of mating flights	400
Mutation rate	0.2

Table 2
The parameter used in the proposed model.

Parameters	Values
Spermatheca size	100
Number of drones	200
Number of broods	500
Number of mating flights	400
Mutation rate	0.2

Table 3
The result of the standard MBO and the proposed model.

	The original MBO				The proposed model			
	Best	Worst	Average	S.D.	Best	Worst	Average	S.D.
Michalewicz	38.847986378	38.813182555	38.841151503	0.008631782	38.847986378	38.846033400	38.847231956	0.000824442
Rastrigin	0.000000000	0.010026249	0.002073527	0.002428718	0.000000000	0.030642884	0.007301366	0.011495355
Rosenbrock	0.000000000	0.000158124	0.000035956	0.000044132	0.000000000	0.000061408	0.000016757	0.000021074
Sphere	0.000000000	0.000070572	0.000019855	0.000019248	0.000000000	0.000001907	0.000000763	0.000000752
Weighted sphere	0.000000954	0.000282288	0.000038662	0.000048092	0.000000000	0.000007629	0.000001907	0.000002771
GoldStein–Price	3.000000000	3.012672906	3.000899331	0.001878220	3.000000000	3.000446790	3.000151214	0.000202508

- Both algorithms can achieve optimal solutions for Rastrigen's function, Rosenbrock's valley function, the sphere function, and Goldstein–Price's function. The proposed algorithm achieved the optimal solution 20% of the time, 30% of the time, 40% of the time, and 60% of the time for the Rastrigen's function, the Rosenbrock's valley function, the sphere function, and the Goldstein–Price's function respectively. On the other hand, the original MBO algorithm only found the optimal solution 6% of the time, 4% of the time, 2% of the time, and 24% of the time for the Rastrigen's function, the Rosenbrock's valley function, the sphere function, and the Goldstein–Price's function respectively.
- For the weighted sphere function, only the proposed algorithm attains the global minimum of zero. The proposed algorithm achieves the optimal solution 30% of the time while the best solution that the original MBO gets is 0.000000954.
- For the Michalewicz function, however, both algorithms cannot find the optimal solution. While the global maximum of the function is equal to 38.850294479, 38.847986378 is the best that both algorithms can achieve. The proposed algorithm gets the best solution 40% of the time, more than double the success rate of 16% obtained by the original MBO algorithm.
- When considering the worst solution obtained by both algorithms, the solutions obtained by the proposed algorithm are closer to the optimal solution than those obtained by the original MBO algorithm for all benchmark functions except the Rastrigen's function.
- When considering the average over the number of experiments carried out for each problem, the proposed algorithm outperforms the original MBO algorithm for all benchmark functions except the Rastrigen's function.

In summary, the performance comparisons with the original MBO show that the proposed algorithm is superior to the original MBO both in terms of the computation time and the ability to find the optimal solution. The proposed algorithm performs very well in solving all 6 benchmark functions.

5. Conclusion

This paper presents a new swarm intelligence algorithm, which is an improved version of Abbass's MBO algorithm. The main drawback of the original MBO algorithm is its very long computation time. Even though the proposed algorithm is still mainly based on the marriage behavior of honey-bees, various assumptions and processes have been modified to improve the algorithm efficiency. The performance of the proposed algorithm is evaluated against the original MBO algorithm. The experimental results show that the proposed algorithm outperforms the original MBO algorithm both in terms of the computation time required for finding the optimal solution and the ability to achieve the optimal solution.

References

Abbass, H.A. (2001). MBO: Marriage in honey bees optimization: A haplometrosis polygynous swarming approach. In *Proceedings of the congress on evolutionary computation* (pp. 207–214).

Abbass, H. A., & Teo, J. (2003). A true annealing approach to the marriage in honey-bees optimization algorithm. *International Journal of Computational Intelligence and Applications, 3*(2), 199–211.

Afshar, A., Bozorg Haddad, O., Marino, M. A., & Adams, B. J. (2007). Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. *Journal of the Franklin Institute, 344*(5), 452–462.

Baykasoğlu, A., Ozbakir, L., & Tapkan, P. (2007). Artificial bee colony algorithm and its application to generalized assignment problem. *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*. I-Tech Education and Publishing.

Blum, C., & Sampels, M. (2004). An ant colony optimization algorithm for shop scheduling problems. *Journal of Mathematical Modelling and Algorithms, 3*(3), 285–308.

Blum, C., Aguilera, M. J. B., Roli, A., & Sampels, M. (2008). *Hybrid metaheuristics: An introduction. Hybrid Metaheuristics: An emerging approach to optimization*. Springer-Verlag.

Curkovic, P., & Jerbic, B. (2007). Honey-bees optimization algorithm applied to path planning problem. *The International Journal of Simulation Modeling, 6*(3), 154–164.

De Jong, K.A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. Doctoral dissertion, University of Michigan.

Dorigo, M., & Di Caro, G. (1999). Ant Colony Optimization: A new meta-heuristic. In *Proceedings of the congress on evolutionary computation* (pp. 1470–1477).

Fathian, M., & Amiri, B. (2008). A honeybee-mating approach for cluster analysis. *The International Journal of Advanced Manufacturing Technology, 38*(7–8), 809–821.

Goldstein, A. A., & Price, J. F. (1971). On descent from local minima. *Mathematics of Computation, 25*(115), 569–574.

Horn, T. (2005). *Bees in America: How the honey bee shaped a nation*. The University Press of Kentucky.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report – TR06, Erciyes University.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*(3), 459–471.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing, 8*(1), 687–697.

Lucic, P., & Teodorovic, D. (2003). Computing with bees: Attacking complex transportation engineering problems. *International Journal on Artificial Intelligence Tools, 12*(3), 375–394.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures= Evolution Programs*. Springer-Verlag.

Modern Agro (2006). About the honey bee. http://www.modernagro.com/about_the_honey_bee.html.

National Research Council (1991). Microlivestock: Little-known Small Animals with a Promising Economic Future. National Academy Press.

Pai, P., Yang, S., & Chang, P. (2009). Forecasting output of integrated circuit industry by support vector regression models with marriage honey-bees optimization algorithms. *Expert Systems with Applications, 36*(7), 10746–10751.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The bee algorithm – A novel tool for complex optimisation problem. *Intelligent Production Machines and Systems*, 454–459.

Pham, D.T., Otri, S., Ghanbarzadeh, A., & Koç, E. (2006). Application of the bees algorithm to the training of learning vector quantisation networks for control chart pattern recognition. In *Proceedings of the 2nd IEEE international conference on information and communication technologies: From theory to applications* (pp. 1624–1629).

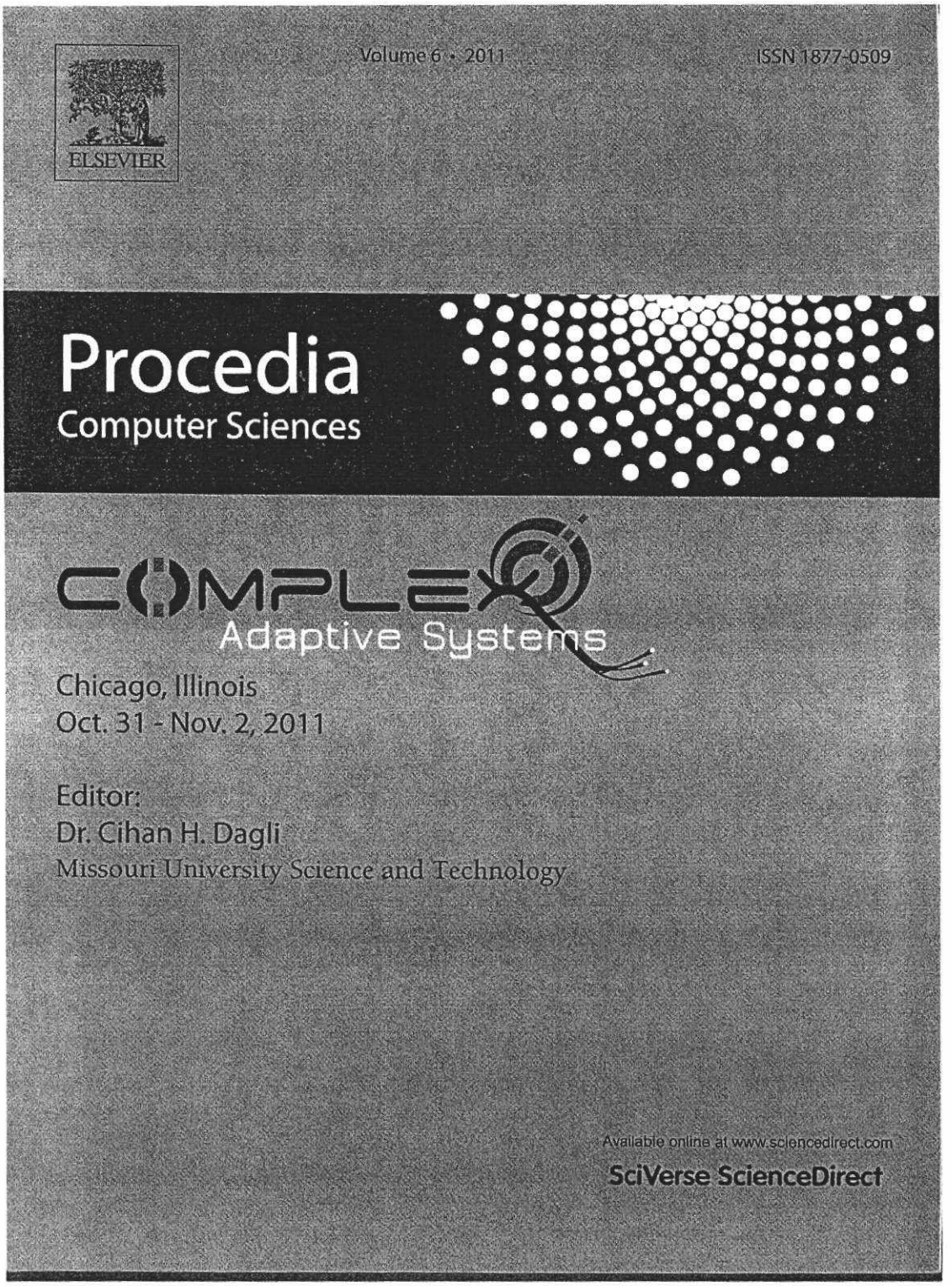
Pohlheim, H. (2006). The genetic and evolutionary algorithm toolbox for Matlab. http://www.geatbx.com/docu/fcnindex-01.html.

Rizzoli, A. E., Montemanni, R., Lucibello, E., & Gambardella, L. M. (2007). Ant colony optimization for real-world vehicle routing problems: from theory to applications. *Swarm Intelligence, 1*(2), 135–151.

Teodorovic, D., & Dell'Orco, M. (2005). Bee Colony Optimization – A cooperative learning approach to complex transportation problems. In *Proceedings of the 10th Meeting of the EURO Working Group on Transportation* (pp. 51–60).

Torn, A. & Zilinskas, A. (1989). *Global optimization (Lecture Notes in Computer Science)*. Springer-Verlag.

Zahiria, S. H., & Seyedin, S. A. (2007). Swarm intelligence based classifiers. *Journal of the Franklin Institute, 344*(5), 362–376.


ELSEVIER
Volume 6 • 2011
ISSN 1877-0509
Procedia
Computer Sciences
COMPLEX
Adaptive Systems
Chicago, Illinois
Oct. 31 - Nov. 2, 2011
Editor:
Dr. Cihan H. Dagli
Missouri University Science and Technology
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 6 (2011) 335–342

Procedia Computer Science

Complex Adaptive Systems, Volume 1
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2011- Chicago, IL

A Modified Marriage in Honey-Bee Optimization for Function Optimization Problems

Patcharawadee Poolsamran*, Arit Thammano

Computational Intelligence Laboratory, Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand

Abstract

Many researches have implemented a genetic algorithm with real-coded chromosomes to solve a wide variety of problems. Their experimental results suggested that the real-coded genetic algorithms gave superior results to binary-coded genetic algorithm on most of the test problems. Inspired by the above founding, this study aims to (1) propose a modified Marriage in Honey-bee Optimization (MBO) technique (2) compare the performance of the proposed technique to that of the real-coded GA technique. In this study, two main ideas are proposed. Firstly, to handle the real encoding of genotypes, we present a new crossover operator and a new heuristic worker, named the scroll-based worker, for manipulating the real value of genes. Secondly, to reduce the number of user-defined parameters, we provide the original MBO with a self-organizing capability. With a self-organizing capability, the proposed model can automatically determine the proper number of queens itself. The experimental results on five benchmark test functions show that the proposed model is very effective in solving the function optimization problems.

Keywords: Real encoding; Marriage in Honey-bee Optimization; Function Optimization Problems

1. Introduction

The Evolutionary Algorithms (EAs) is a population-based metaheuristic optimization method. In general, the EAs consist of three mechanisms inspired by theories of natural evolutions [1, 2, 3]. Firstly, the selection operation is used to select two parent chromosomes for reproduction. This selection process is typically probabilistic, that is, the highly fit individual is allowed to create more offsprings than the lesser one. Secondly, the recombination

* Corresponding author. Tel.: +66-2-723-4964; fax: +66-2-723-4910.
E-mail address: patcharp@buu.ac.th.

1877-0509 © 2011 Published by Elsevier Ltd.
doi:10.1016/j.procs.2011.08.061

operation, commonly called crossover, is a method used to produce the offsprings that inherits some characteristics of their parents. Finally, the mutation operation slightly changes some genes of the chromosome with the hope that the fitness of the chromosome will increase. Chromosome representation is the first important step for a successful operation of the EAs. For the function optimization problems of continuous variables, there are two ways to encode the chromosome, binary representation and real representation. In the former, the chromosome is represented by a sequence of binary bits; therefore, the precision of the solution is restricted by the length of the chromosome. The most common binary representations are the binary, gray, and unary encodings [3]. The latter is a real encoding. In the real encoding, the chromosome is represented by a sequence of floating point numbers. Therefore, the precision of the solution depends on the precision of the machine used for computations. Using the above two encoding methods, a number evolutionary algorithms have been implemented such as in the reference [4], [1], and [5]. The previous studies suggested that the real encoding is a very efficient method. In comparison to the binary representation, it provides a higher precision and has faster convergence speed.

The MBO was proposed by Hussein A. Abbass [6] for solving 3-SAT problems. The MBO is a recent evolutionary metaheuristic inspired by the marriage behavior in real honey-bee. There are several applications of this approach that have been presented; for instance, Bozorg Haddad, Afshar and Mariño [7] applied the conventional MBO to solve a real-world reservoir operation problem. The performance of their model is well comparable with the performance of the genetic algorithm. In this paper, we proposed the real-coded MBO algorithm together with a new crossover and worker. For performance evaluation, the results of the proposed algorithm on five benchmark functions are presented and compared with the results of the real-coded GA.

The remaining paper is organized as follow. The next section introduces a description of conventional MBO. Section 3 details the proposed methodology. Then, a detail of parameter identification, five benchmark test functions, and an analysis of experimental results are given in section 4. Finally, the conclusions will be discussed in section 5.

2. The Marriage in Honey-bee Optimization

In nature, honey-bees can be classified, based on their roles, into three groups, the queen, the drones, and the workers. The role of the queen is to mate with several drones and to lay the eggs that will later develop into numerous broods. The role of the drone is mainly to mate with the queen. The task of the worker bees is to (1) feed the queen and the broods (2) do all the maintenance work of the hive. The MBO algorithm, which attempts to model the marriage behavior of real honey-bees in nature, is also classified its artificial bees into three groups: the queen, the drones, and the workers. The queen and the drones represent the candidate solutions to the problem whereas the workers represent the heuristics used to perform local search on the solutions. In short, the mating process between the queen and the drones, and the heuristic local search operation are the ways in manipulating the candidate solutions to find the optimal solution.

The algorithm of the original MBO is listed as follows:

- Step 1: Initialization

Six parameters, the number of queens, the queen's spermatheca size, the number of drones, the number of broods, the mutation probability, and the convergence criteria, are to be assigned the values by the user. Subsequently, the initial population is created by randomly generating N individuals. The fitter individuals are selected to be the initial queens of the hive, while the rest is assigned to be the drones.

- Step 2: Mating process

To begin the process, the queen's energy and speed are randomly initialized. A decision of the queen whether to mate with a particular drone is based on the probability of marriage as follows:

$$Prob(Q,D) = \exp\left(\frac{-\Delta(f)}{S(t)}\right) \qquad (1)$$

where $\Delta(f)$ denotes an absolute difference of the fitness values between the queen Q and the drone D; S(t) is the speed of the queen at time t. If the mating is successful, the sperm of the drone is added to the queen's spermatheca.

Then, the drone dies immediately after mating. After each successful mating, the queen's speed and energy decrease according to the following equations:

$$S(t+1) = \alpha * S(t) \tag{2}$$

$$E(t+1) = E(t) - \gamma \tag{3}$$

where α is a factor in the range [0, 1] and γ is the amount of energy reduction after each transition

$$\gamma = \frac{0.5 * initial\_\gamma}{M} \tag{4}$$

where M is the queen's spermatheca size. This stage is repeated until the energy of the queen is lower than the predefined threshold or her spermatheca is full.

- Step 3: Breeding and feeding processes

When all queens complete their mating, they start breeding to generate new broods. The breeding process starts with randomly selecting a sperm from the queen's spermatheca, and then followed by crossover the selected sperm's genotype with the queen's genotype to form a brood as illustrated in Fig 1. Next, the mutation operator is applied to the new brood in order to provide greater variety to the solutions. Finally, the genotype of the newly-born brood is further improved by a worker bee, which is chosen in proportion to its fitness. Each worker represents one of many local search heuristics, such as GSAT, WalkSAT, random walk, random new, random flip, and one-point crossover [6, 8]. When the number of broods is equal to the required quantity, this process is complete.

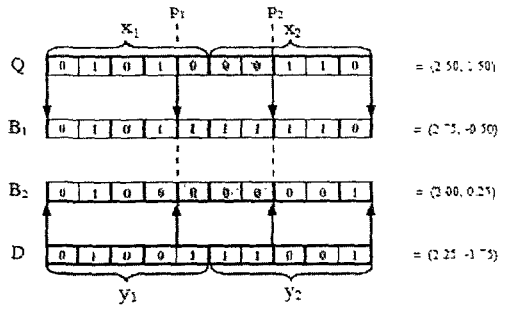


Fig. 1 Binary crossover operation

- Step 4: Updating process

In this process, some queens are replaced with fitter broods until no brood that is fitter than any of the queens exists. Thus, the population of the next generation certainly has equally or better quality. Before new mating flight begins, the remaining broods have been killed.

All above processes are repeated until the convergence criteria are met.

3. The proposed methodology

This paper presents two main issues: (1) applying the real encoding to the MBO approach and (2) reducing the number of user-defined parameters of the MBO algorithm. In the original MBO, the binary string is typically used to encode the genotype. Since the genotypes represent the candidate solutions, the encoding scheme should be

designed so that it can cover the possible solution When a high precision of solution is required such as in the function optimization problems the binary encoding scheme might not be appropriate. Therefore, in this paper, the real encoding is used instead of binary numbers; the crossover operator, the mutation operator, and the local search heuristics, which are used to manipulate the genotype, are also redesigned to match the change in encoding method. The followings are the details of proposed operators:

3.1. The properties of operators

3.1.1 The crossover operator

The concept of this new crossover is to (1) clone the strong individual in order to keep it alive as long as possible and (2) improve the weak individual by combining the genotypes of the queen and the drone. In this paper, the retaining probability is used to indicate the degree of strength of each individual. The retaining probability of the i^{th} individual can calculated as follow:

$$P_{retain}(i) = \exp(-\frac{|f_i - f_{best}|}{\sigma^2}) \tag{5}$$

where f_i and f_{best} are the fitness values of the i^{th} individual and the fittest individual respectively. σ is the width of the Gaussian function. In this new crossover operator, the operation begins by generating a random real number between 0 and 1; then comparing it to the retaining probability. If a random number is smaller than the retaining probability, the parent will be cloned to form a pair of broods. Otherwise, the two broods will be created by using equations (6) and (7).

$$g_i^{b_1} = \begin{cases} g_i^q & ;r_i \le P_{retain}(Q) \\ g_i^q + c_i \cdot \Delta P_{retain}(Q,D) \cdot g_i^d & ;otherwise \end{cases} \tag{6}$$

$$g_i^{b_2} = \begin{cases} g_i^d & ;r_i \le P_{retain}(D) \\ g_i^d + c_i \cdot \Delta P_{retain}(Q,D) \cdot g_i^q & ;otherwise \end{cases} \tag{7}$$

where g_i^* is the i^{th} gene; c_i is a random number in the range [-0.5, 0.5], b_1 and b_2 are the first and second broods respectively. $P_{retain}(\cdot)$ is the retaining probability in the range [0, 1]. $\Delta P_{retain}(Q, D)$ is the difference of the retaining probability between the queen Q and the drone D

It is clearly that the individuals with greater retaining probability are given higher chance of survival than the inferior individuals Therefore, the weakest individual has the smallest chance of reserving its characteristic

3.1.2. The mutation operator

The main idea of the mutation operator is to slightly change the value of the real-coded gene with the hope that the fitness of the genotype will increase. For each gene, the operation begins by generating a random real number between 0 and 1, and then compares it to the predefined mutation probability, which is usually set to a small number If a random number is smaller than the mutation probability, the gene will be slightly adjusted by adding a small number in the range [-0.5, 0.5]

3.1.3. The heuristic worker operator

This proposed model employs two kinds of workers The first worker, called the random one worker, is presented in the reference [6]. The other, called the scroll-based worker, is proposed in this paper. The scroll-based worker shown in Fig 2 is designed specifically for real encoding. Let g_i denotes the i^{th} gene in the genotype; g_i' is the 1^{th}

digit to the left of the decimal point in the gene g_i; g_i' is the r^{th} digit to the right of the decimal point in the gene g_i. The value of g_i is adjusted based on the following equations.

$$g_i^* = g_i^* + \Delta \tag{8}$$

$$\Delta = random\{-1, 0, 1\} \tag{9}$$

where Δ is used to scroll up or down (to increase or decrease) the value of a digit. If $\Delta = 1$, the value of a digit will be increased by 1. If $\Delta = -1$, the value of a digit will be decreased by 1. If $\Delta = 0$, the value of a digit will not be altered

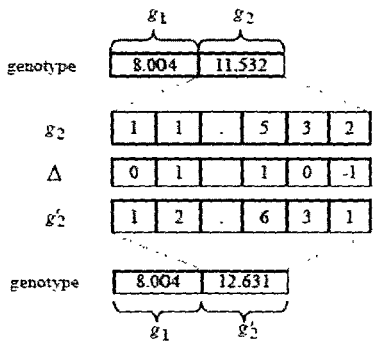


Fig. 2. The scroll-based worker

3.2 The proposed algorithm

The procedure starts with one randomly generated queen's genotype. The very first task that has to be done is to mate with several drones in order to reproduce broods. Next is the breeding process in which the queen's genotype is crossover with the genotype of a randomly selected sperm from the queen's spermatheca. If the number of broods is equal to the required quantity, the breeding process completes. The newly-born broods are then improved by a mutation operator and a randomly selected worker. The broods which are fitter than the queen will be promoted to be in a list of candidates for the queen. Hence, the number of queens will be increasing. However, if none of the broods is fitter than any of the queens, the old queens will continue ruling their own colonies. Note that if a list of the queens is changed, all the colonies will be merged together into a single entity; then the new colonies are rebuilt by a group of new queens. In order to keep the number of queens to a minimum, the queens are prohibited from being too close to each other. Since some drones die after mating with the queens, in order to maintain the number of drones, the vacancies left by the death will be replaced by the broods with high fitness values.

Before the new mating flight begins, the colony reconstruction process has to be done. In this process, the drones are clustered into several colonies, the centroids of which are the queens by implication. The clustering technique used in this study is based on the fuzzy c-mean concept. Given $D = \{d_1, d_2, \ldots, d_p, \ldots, d_P\}$ and $Q = \{q_1, q_2, \ldots, q_s, \ldots, q_S\}$ are sets of drones and queens, respectively. The membership value of the drone d_p in the colony of the queen q_s is calculated according to the following Gaussian membership function:

$$\mu(d_p, q_s) = \exp\left(\frac{-\sum (d_p - q_s)^2}{\sigma_{q_s}^2}\right) \tag{10}$$

where σ_{q_s} indicates the width of the Gaussian function, which is the ratio of the fitness value of the queen q_s to the fitness value of its neighbor.

$$\sigma_{q_s} = \max\{\sigma_{q_s}^k\} \tag{11}$$

$$\sigma_{q_s}^k = d(q_s, q_k) \times \frac{f(q_s)}{f(q_s) + f(q_k)} \quad ; s \neq k \tag{12}$$

Each drone chooses its colony by considering its membership value in each colony; the colony with the maximum membership value is selected as a new home for the drone. This process finishes after all drones have already been assigned to suitable colonies

4. Results and Discussions

To test the performance, the proposed model is used to find the optimal solutions of five benchmark functions: F1 (the Ackley's function), F2 (the Griewank's function), F3 (the sphere function), F4 (the weighted-sphere function), and F5 (the Schwefel's double sum function) Then its performance is compared with that of the real-coded GA The equations of the above benchmark functions are shown in Table 1.

Table 1 The five benchmark functions

Functions	Formula	Domain range
F1· Ackley	$f(\vec{x}) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}} - e^{\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)}$	$x_i \in [-30, 30]$
F2 Griewank	$f(\vec{x}) = \sum_{i=1}^{2} \frac{x_i^2}{4000} - \prod_{i=1}^{2} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$x_i \in [-600, 600]$
F3 sphere	$f(\vec{x}) = \sum_{i=1}^{2} x_i^2$	$x_i \in [-5.12, 5.12]$
F4 weighted- sphere	$f(\vec{x}) = \sum_{i=1}^{2} i^2 x_i^2$	$x_i \in [-5.12, 5.12]$
F5· Schwefel's double sum	$f(\vec{x}) = \sum_{i=1}^{2} (\sum_{j=1}^{i} x_j)^2$	$x_i \in [-65.536, 65.536]$

Since the performance of most evolutionary algorithms strongly depends on proper selection of system parameters, the parameters of both models are varied to various values in order to get the best out of both models. The best performances of both models are obtained when their system parameters are defined as follows:

For the proposed model, the number of drones is set to 200, the number of broods is set to 500, the queen's spermatheca size is set to 100, the mutation probability is set to 0.2, and the number of mating flights is set at 2000.

For the real-coded GA, the population size is set to 240, the crossover probability is set to 0.7, the mutation probability is set to 0.2, and the number of generations is set at 2000.

For each benchmark functions, both algorithms are run 10 times. Each run starts with a different initial population and stops when the number of generations is equal to the predefined number The performance of the two algorithms is measured in terms of the effectiveness and the success rate. The former is measured the closeness of the resulting solution to the global optimum while the latter is the number of times the algorithm successfully

located the global optimum out of 100 trials. The experimental results of both the proposed model and the real-coded GA in finding the optimal solutions of five benchmark functions are shown in Table 2.

Table 2 The effectiveness of the proposed model and the real-coded GA

		Experimental results			
		Best	Worst	Average	S.D
F1	Real-coded GA	7.276253E-06	2.192872E-05	1.296436E-05	5.448939E-06
	Proposed Model	0	0	0	0
F2	Real-coded GA	3.000000E-12	1.257130E-01	5.127757E-02	4.718468E-02
	Proposed Model	0	7.396041E-03	2.218812E-03	3.572627E-03
F3	Real-coded GA	1.996650E-13	2.850000E-10	6.750000E-11	9.240881E-11
	Proposed Model	0	0	0	0
F4	Real-coded GA	4.000000E-12	5.140000E-10	9.210000E-11	1.646751E-10
	Proposed Model	0	0	0	0
F5	Real-coded GA	5.000000E-12	2.021000E-09	5.098000E-10	6.821282E-10
	Proposed Model	0	0	0	0

In Table 2, the effectiveness of the two algorithms is showed using four statistical data: the best, the worst, the average, and the standard deviation obtained from 10 runs. The experimental results show that the best, the worst, the average, and the standard derivation of the results offered by the proposed model are better than the real-coded GA in all benchmark functions. However, the differences between the two algorithms are small.

The results in terms of the success rate (Table 3) show that the proposed model is able to obtain the global optimum of all test functions, however, the real-coded GA is unable to find the global optimum. For F1, F3, F4, and F5 functions, the proposed model achieved the global optimum of the functions in all the runs. For F2 function, the proposed model attained the global optimum with a success rate of 70%. This clearly indicates that the proposed model has better capability in finding the global optimum than the real-coded GA.

Table 3 The success rate of the proposed model and the real-coded GA

Function	Success rate	
	Real-coded GA	Proposed Model
F1 Ackley	0%	100%
F2 Griewank	0%	70%
F3 sphere	0%	100%
F4 weighted- sphere	0%	100%
F5 Schwefel's double sum	0%	100%

5. Conclusions

To reduce the number of user-defined parameters, we provide the original MBO with a self-organizing capability. The proposed model can automatically determine the proper number of queens itself. Results on five benchmark test functions show that the proposed model is very effective in solving the function optimization problems.

References

1 Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin Heidelberg, 1996

2 M. Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems, Addison Wesley, Harlow, 2002

3 F. Rothlauf, Representations for genetic and evolutionary algorithms, Springer-Verlag, Berlin Heidelberg, 2006.
4 C. Z Janikow and Z Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms, In Proc. of the 4th Int. Conf. on Genetic Algorithms (1991) 31-36.
5 N. Tutkun, Parameter estimation in mathematical models using the real coded genetic algorithms, Int J. of Expert Syst. Appl. 36(2) (2009) 3342-3345.
6 H A. Abbass, MBO. Marriage in honey bees optimization: A haplometrosis polygynous swarming approach. In Proc of the Congress on Evolutionary Computation (2001) 207-214
7. O Bozorg Haddad. A. Afshar, and M. A. Mariño. Honey-bee mating optimization (HBMO) algorithm A new heuristic approach for water resources optimization. Water Resources Management 20 (2006) 661-680.
8 H A Abbass and J. Teo, A true annealing approach to the marriage in honey-bees optimization algorithm, IJCIA 3 (2003) 199-211

SECOND ANNOUNCEMENT AND CALL FOR PAPERS

The Institute of Electrical and Electronics Engineers, Inc.

IEEE

Intelligent Systems IS'2012: Methodology, Models, Applications in Emerging Technologies

Organized by

IEEE IM/CS/SMC Joint Chapter of Bulgaria
IEEE Computational Intelligence Chapter of Bulgaria
Federation of Scientific-Technical Unions of Bulgaria
John Atanassov Union on Automatics and Informatics, Bulgaria
Institute of ICT, Bulgarian Academy of Sciences
SULSIT State University, Sofia, Bulgaria

September 6-8, 2012
Sofia, Bulgaria
http://www.ieee-is.org

MAIN TOPICS

- Advanced Intelligent Systems
- Artificial Intelligence
- Bioinformatics and Computational Biology
- Business and Finance
- Case-based and Temporal Reasoning
- Cognitive Aspects of Intelligent Systems
- Computational Intelligence
- Data Fusion
- Data Mining and Knowledge Discovery
- Data Processing
- Decision Support Systems
- Evolutionary Computation
- Education, e-Learning
- Enviromental Engineering
- Fuzzy Sets and Logics, and Their Extensions
- Genetic Algorithms
- Health, Medicine,Bioengineering
- Human-Machine Interaction
- Intelligent Behavior
- Intelligent Control
- Intelligent Information Security Systems
- Intelligent Measurement
- Intelligent Network Technologies
- Intelligent Signal Processing
- Knowledge Management
- Machine Learning
- Multi-Agent Systems
- Neural Networks and Neuro-Fuzzy Systems
- Ontology-Based Intelligent Systems
- Petri Nets and Their Extensions
- Power Industry
- Process Control
- Robotics and Manufacturing
- Swarm Intelligence and Ant Colony Optimization
- Telecommunication
- Transportation
- WEB-Mining

PROGRAM COMMITTEE

Honorary Chair: L. A. Zadeh (US)
General Co-Chairs: R. Yager (US), V. Sgurev (BG)
International Program Committee Co-Chairs:
J. Kacprzyk (PL), M. Hadjiski (BG)

P. Albertos (ES)
P. Angelov (GB)
M. Angelova (GB)
K. Atanassov (BG)
R. Babuska (NL)
I. Bratko (SL)
F. Čapkovic (SK)
P. Chountas (GB)
I. Dumitrache (RO)
M. Dumitrescu (RO)
E. El-Darzi (GB)
S. Fidanova (BG)
D. Filev (US)
G. Fogel (US)
A. Fradkov (RU)
T. Fukuda (JP)
A. Gegov (GB)
R. Gonçalves (PT)
V. Gorodetsky (RU)
F. Herrera (ES)
K. Hirota (JP)
G. Horvath (HU)
L. Jain (AU)
P. Jorrand (FR)
V. Jotsov (BG)
I. Kalaykov (SE)
N. Kasabov (NZ)
O. Kaynak (TR)
V. Kecman (NZ)
J. Keller (US)
G. Klir (US)
T. Kolemisevska-Gugulovska (MK)
A. Kordon (US)
M Krawczak (PL)
V. Kreinovich (US)
K. Leiviska (FI)
S. McClean (GB)
P. Melo-Pinto (PT)
J. Mendel (US)
D. Nauck (GB)
E. Oja (FI)
G. Osipov (RU)
N. Pal (IN)
G. Palm (DE)
R. Palm (DE)
H. Papadopoulos (CY)
T. Parisini (IT)
W. Pedrycz (CA)
K. Peeva (BG)
V. Piuri (IT)
M. Polycarpou (CY)
I. Popchev (BG)
Y. Popkov (RU)
L. Rutkowski (PL)
A. Sala (ES)
T. Samad (US)
A. Shannon (AU)
S. Strmcnik (SL)
V. Stefanuk (RU)
E. Szmidt (PL)
H. Tamura (JP)
M. Thuillard (CH)
A. Topalov (BG)
G. Vatchkov (JP)
K. Warwick (GB)
T. Yildirim (TR)
G. Yen (US)
J. Zurada (US)

Technical Program Chair: V. Jotsov (BG)

CONFERENCE COMMITTEE

Chair of NOC: K. Atanassov
Co-Chair of NOC: L. Doukovska

CORRESPONDENCE & SUBMISSION

All correspondence must be directed to Prof. K. Atanassov, DSc, DSc, PhD or Assoc. Prof. L. Doukovska, PhD, via e-mail:

ieee-is2012@bas.bg

All paper submissions must be directed to the EDAS system:

http://edas.info

DEADLINES

- Expression of interest: Now open
- Submission of draft papers: March 27, 2012
- Notification of acceptance: April 15, 2012 (via e-mail)
- Submission of final papers: May 5, 2012
- Registration of authors: With submission of camera-ready papers

LOCATION

The capital of Bulgaria is located in the western part of the country at the foot of Mountain Vitosha. It is the 12th largest city by population in the EU. Many of the major universities, institutions, and businesses of Bulgaria are concentrated in Sofia. It is also a center of public and political life, cultural events and media, as well as research institutes.
Sofia offers numerous sightseeing places and tourist attractions: gorgeous rich museums, beautiful medieval churches and art galleries, as well as many night clubs and pubs. A notable site is the 10th century Boyana Church in the skirts of Vitosha, which is one of the UNESCO World Heritage site.

St. Alexander Nevsky Cathedral in the centre of Sofia

Sixth International IEEE Conference
Intelligent Systems, Sofia, Bulgaria
6-8 September 2012
www.ieee-is.org

IEEE IM/CS/SMC Joint Chapter of Bulgaria
IEEE CIS Chapter of Bulgaria

TO Miss Patcharawadee Poolsamran, Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Thailand, the first author of accepted paper "A Modified Marriage in Honey-Bee Optimization for Multiobjective Optimization Problems" to be published in IEEE sources.

Dear Miss Poolsamran,

I would like to inform you that judging upon the reviews received, your paper *have been accepted* for publications in the Proceedings of 6th International IEEE conference on Intelligent Systems IEEE IS'12, to be held in Sofia, Bulgaria, 6-8 September 2012. Consider this letter also as a confirmation that you are invited to present your papers at oral presentation during the conference.

EDAS.info reviewing system shows up the following IEEE IS'12 conference acceptance ratio: 39 7%

Please, keep in mind that the organizing committee assures that papers comply with the IEEE style for papers, and you are solely responsible for any other errors you made in your manuscript. The organizing committee keeps its right not to publish a manuscript if the authors fail to meet the format requirements of their manuscripts, if the copyright form has not been submitted, etc.

I take the chance to congratulate you with your paper's successful completion of the review process. and I am looking forward to meeting you or your representatives at the conference!

Sofia, Bulgaria
May 15, 2012

Technical Program Chair:

/Full Prof. Vladimir Jotsov, D. Sc., Ph.D./
Deputy Dean, Faculty 'Information Sciences'
State University for Library Studies and IT
Email: **jotsov@ieee.org**
Phone: 00359 878 220 438

A Modified Marriage in Honey-Bee Optimization for Multiobjective Optimization Problems

Patcharawadee Poolsamran[1] and Arit Thammano[2]
Computational Intelligence Laboratory
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
[1]patcharp@buu.ac.th and [2]arit@it.kmitl.ac.th

Abstract—This paper proposes a modified marriage in honey-bee optimization for solving multiobjective optimization problems. Unlike the original marriage in honey-bee optimization, the proposed algorithm divides the objective space into several colonies, each of which has its own queen. The fitness of each solution is based on 3 parameters: the size of the colony, the number of dominating solutions, and the number of dominated solutions. The nondominated solutions with highest fitness values are preferentially assigned to be the queens while the rest are assigned to be the drones. Next, all drones are assigned to the colony according to their distances from the queens of the colonies. In order to maximize a genetic variance in the population, the multiple mating is used. The multiple mating requires the queen to mate with drones from the other colonies. The proposed algorithm has been evaluated and compared to two state-of-the-art metaheuristic algorithms: the Pareto archived evolution strategy and the nondominated sorting genetic algorithm. The experimental results on 5 different ZDT benchmark functions illustrate that the proposed algorithm is able to converge to the true Pareto fronts and has better spread of solutions, as compared with the published results of the two state-of-the-art algorithms.

Keywords- swarm intelligence; marriage in honey-bee optimization; multiobjective optimization problem

I. INTRODUCTION

In the past decade, the increasing demand in developing the effective and efficient techniques to solve complex real-world problems with multiple objectives has encouraged the development of many bio-inspired algorithms. In contrast to the single-objective optimization problem, there is no unique optimal solution when considering multiple conflicting objectives. Therefore, the solving of multiobjective optimization problems is to determine a set of optimal solutions representing the best tradeoff between multiple conflicting objectives, called the Pareto-optimal set. In fact, finding the Pareto-optimal solutions is NP-complete [1], which is impractical to solve in reasonable computing time. The bio-inspired algorithms seem to be the most attractive approaches for this kind of problems because 1) they are population-based metaheuristics that can find multiple solutions in a single run, and 2) they typically require small computational time. A number of different bio-inspired algorithms have been applied to multiobjective optimization problems such as the genetic algorithm (GA), the artificial immune systems (AIS), and the particle swarm optimization (PSO).

In the mid-eighties, the vector evaluated genetic algorithm (VEGA) [2] is the first effort to improve the genetic algorithm to search for a set of Pareto-optimal solutions in a single run. After the development of VEGA, numerous variations of GA approach have been proposed. To date, there are several well-known GAs for the MOP such as the nondominated sorting genetic algorithm (NSGA-II) [3], the multi-objective genetic algorithm (MOGA) [4], the strength pareto evolutionary algorithm (SPEA-II) [5], and the pareto archived evolution strategy (PAES) [6]. In recent years, Wong, Yeung, and Lau [7] has proposed a novel hybrid method based on the genetic algorithm and the artificial immune system, called the hybrid artificial immune system, for solving both unconstrained and constrained multiobjective problems of global container repositioning.

In addition, the marriage in honey-bee optimization (MBO), one of recently proposed bio-inspired models is inspired by the process of real honey-bee mating. The MBO algorithm was first proposed by Hussein A. Abbass [8] for solving 3-SAT problems. Since then, there are several applications of this algorithm that have been presented; for instance, Bozorg Haddad, Afshar and Mariño [9] applied the original MBO to solve a reservoir operation problem. The performance of their model is well comparable with the performance of the genetic algorithm.

In the recent study, Thammano and Poolsamran [10] proposed a modified version of the MBO

algorithm, called a self-organizing model of marriage in honey-bee optimization (SMBO). Their proposed model has been tested on six benchmark functions, including both maximization and minimization problems. The performance comparisons with the original MBO show that their proposed algorithm is superior to the original MBO both in terms of the computation time and the ability to achieve the optimal solution. Later, Poolsamran and Thammano [11] proposed a real-coded marriage in honey-bee optimization together with the new crossover and worker operators. The experimental results, compared with the results of the real-coded GA, on five benchmark test functions show that the proposed model is very effective in solving the single objective optimization problems. In this paper, we propose a modified marriage in honey-bees optimization for the multiobjective optimization problem. For performance evaluation, the results of the proposed algorithm on five ZDT functions are presented and compared with the published results of two state-of-the-art algorithms: the PAES algorithm and the NSGA-II (real-coded) algorithm.

The remaining paper is organized as follow. The next section introduces a definition of the multiobjective optimization problem and a description of the original marriage in honey-bees optimization. Section 3 describes the proposed algorithm. Then, a detail of parameter identification, a performance metric, and an analysis of experimental results are given in section 4. Finally, the conclusions are discussed in section 5.

II Background

A. Multiobjective optimization problem

A multiobjective optimization involves the process of simultaneously optimizing m conflicting objectives subject to certain constraints. In a multiobjective optimization problem, rather than finding a single solution as in a single objective optimization problem, it aims to find a set of solutions as close to the pareto-optimal set as possible, and at the same time maintain diversity among the individual solutions.

$$\begin{aligned} \text{Min} \quad & F(x^*) = (f_1(x^*), f_2(x^*), \ldots, f_m(x^*)) \\ \text{subject to} \quad & q_i(x) \le 0 \quad (i = 1, 2, \ldots, k), \\ & r_j(x) = 0 \quad (j = 1, 2, \ldots, l) \end{aligned} \tag{1}$$

where q_i is the i^{th} inequality constraint, k is the number of inequality constraints, r_j is the j^{th} equality constraint, and l is the number of equality constraints. According to the concept of Pareto dominance, a solution x_1 is said to dominate the solution x_2 if the following conditions are true: 1) no component of $F(x_1)$ is worse than the corresponding component of $F(x_2)$, and 2) at least one component of $F(x_1)$ is better than the corresponding component of $F(x_2)$. Any solution that is not dominated by others is regarded as a nondominated solution. A solution x^* is Pareto-optimal solution if there is no other solution x that can improve at least one objective function without detriment to another objective function.

B. Marriage in honey bee optimization (MBO)

In nature, honey-bees can be classified, based on their roles, into three groups: the queen, the drone, and the worker. The role of the queen is to mate with several drones and then lay eggs that later develop into numerous broods. These broods are fed and cared for by the workers until they mature. The MBO algorithm, which attempts to model the marriage behavior of real honey-bees in nature, is also classified its artificial bees into the queen, the drone and the worker. The queens and the drones represent the candidate solutions to the problem; the workers represent the local search heuristics for improving the genotypes of the broods. The mating process between the queens and the drones, followed by the feeding process by the workers will enable the algorithm to find the optimal solution. The algorithm of the original MBO is listed as follows:

1) The initialization process

Six parameters, the number of queens, the size of queen's spermatheca, the number of drones, the number of broods, the mutation probability, and the number of mating flights, are defined by the user. Subsequently, the population which comprises the queens and the drones is randomly generated. The fitter individuals are selected as initial queens while the rest are assigned as drones.

2) The mating process

Before the mating flights begin, both energy and speed of each queen are randomly initialized. Then the queen mates with the drone that has the highest probability of marriage as shown in the following equation.

$$\text{Prob}(q_i, d_p) = \exp\left\{\frac{-\Delta(f)}{S(t)}\right\} \tag{2}$$

where $\Delta(f)$ denotes an absolute difference of fitness value between the queen q_i and the drone d_p, $S(t)$ is the speed of the queen at time t. If the mating is successful, a sperm of the participated drone is added to the queen's spermatheca. Then, the drone dies immediately after mating. This stage is repeated until the energy of the queen is lower than the predefined threshold or the queen's spermatheca is full. After each mating transition, the queen's speed and energy are decreased in accordance with the following equations.

$$S(t+1) = \alpha * S(t) \tag{3}$$

$$E(t+1) = E(t) - \gamma \tag{4}$$

where α is a decay factor in the range [0, 1].γ is the amount of energy reduction after each transition: it is calculated according to the following equation:

$$\gamma = \frac{0.5 * initial\_\gamma}{M} \tag{5}$$

where M is the size of the queen's spermatheca.

3) The breeding and the feeding process

The breeding process begins after all of the queens complete their mating. During the breeding process, the queen chooses a sperm from her spermatheca by using the roulette wheel selection scheme. Then new broods are created by crossover the genotype of the queen with the genotype of the selected sperm. This breeding process is repeated until the number of broods reaches the required quantity. Afterward, the genotypes of all newly-born broods are improved by using both binary mutation and worker operators. The way of performing binary mutation is to flip a bit from 1 to 0 or vice versa. In the feeding process, the workers, which represent the local search heuristic functions such as GSAT, WalkSAT, the random walk, the random new, the random flip, and the one-point crossover [8], are chosen in proportion to their fitness and are then applied to the genotypes of the broods.

4) The competition process

Finally, the weaker queens are replaced with the fitter broods until there is no brood remaining that is fitter than any of the queens. In doing this, the population of the next generation certainly has equally or better quality. Before the new mating flight begins, all remaining broods are killed.

All above processes are repeated until the convergence criteria are met.

III Proposed Methodology

A. Fitness evaluation

In this proposed model, the fitness value of the i^{th} individual consists of two terms: 1) the first term is the winning probability (P_{win}) which measures the degree to which the i^{th} individual dominates other members in the colony, and 2) the second term is the losing probability (P_{loss}). The losing probability measures the degree to which the i^{th} individual is dominated by other members in the colony. That is, the fitness value of the i^{th} individual is high when the i^{th} individual is a nondominated solution which dominates many members in a colony.

$$\text{fitness}(i) = \omega_1 P_{win}^2 + \omega_2 (1 - P_{loss}^2) \tag{6}$$

$$P_{win} = \frac{N_c}{N_{pop}} \cdot \frac{N_{win}}{N_c} = \frac{N_{win}}{N_{pop}} \tag{7}$$

$$P_{loss} = \frac{N_c}{N_{pop}} \cdot \frac{N_{loss}}{N_c} = \frac{N_{loss}}{N_{pop}} \tag{8}$$

where ω_1 and ω_2 are the weight factors; N_c is the number of members in the colony c; N_{pop} is the population size; N_{win} is the number of members in the colony c which is dominated by the i^{th} individual; N_{loss} is the number of members in the colony c that dominate the i^{th} individual.

B. Proposed algorithm

The proposed algorithm is divided into five stages. 1) the initialization process, 2) the mating process, 3) the breeding and the feeding processes, 4) the swarm preparation process, and 5) the swarm process. In the first stage, six parameters, the maximum number of queens, the size of the queen's spermatheca, the number of drones, the number of broods, the mutation probability, and the number of mating flights, are defined by the user. Next, the initial population is created by randomly generating N_{pop} individuals. Each individual is represented by a vector of real numbers. Its length depends on the number of decision variables. Then the fitness value of each individual is evaluated based on the individual's dominated and dominating numbers. The individual having the largest dominating number and, at the same time, having the smallest dominated number is selected to be the queen of the initial colony, while the rest is assigned to be the drones.

In the second stage, the mating process starts by initializing the queen's energy and speed with random values in the range [0.5, 1]. On the mating flight, each queen will mate with several drones. The probability that the queen q_i will mate the drone d_p is evaluated by using (2). The drone with the highest probability is selected to mate with the queen. Then the selected drone's genotype (sperm) is directly transferred to the queen's spermatheca. After each mating, the queen's energy and speed decline in accordance with (3) – (5). However, the queen continues to mate until her spermatheca is full or her energy is lower than the predefined threshold. In this study, the threshold is set to zero.

When all queens complete their mating flight, they return to their hives for the starting of the third stage. In the breeding process, a queen is chosen by using the roulette wheel selection scheme. Then the chosen queen reproduces the broods by crossover her genotype with the genotype of the randomly selected sperm from her own spermatheca. In this paper, the retaining crossover operator [11] is used. The retaining probability of each individual is calculated by using the following equation:

$$P_{retain}(i) = \frac{fitness(i)}{fitness_{max}} \tag{9}$$

Next, the mutation operator is applied to the newly-born broods. Unlike the original MBO, the mutated broods are classified into their mother's colony. The other process of the third stage is the feeding process. The worker operators, which represent the local search heuristic functions, are used to improve the genotypes of the mutated broods. For each brood, by using the roulette wheel selection scheme, a worker is selected in proportion to its fitness value. This paper utilizes two local search heuristic functions: the random new and the scroll-based operators [11]. The fitness value of each worker is updated according to the cumulative rate of improvement achieved on the broods. This third stage is complete when the number of broods is equal to the required number of broods.

The fourth stage, the swarm preparation process, consists of the new population selection and the clustering processes. In the process of new population selection, the new population for next generation is created by combining the N_{pop} best individuals from a group of old population and the newly-born broods together. To enhance the convergence speed, the new population selection process uses an elitist-preserving approach. All nondominated bees (queens, drones, and broods) of every colony are selected by default to be added to the candidate pool. If the number of the nondominated bees is less than the population size, a group of dominated bees in each colony whose fitness values are higher than the average fitness of the colony will then be selected to be added to the candidate pool. This selection process continues with the remaining dominated bees in each colony until the number of members in the candidate pool is greater than or equal to the population size. Therefore, in the case that the number of members in the candidate pool is greater than the population size, some candidates will be discarded to maintain the population size. Before the deletion begins, the minimum distance in the objective space between each candidate and its nearest neighbors, and the average of these minimum distances are calculated. Then the weakest candidate among all candidates whose distance is less than the average distance is deleted from the candidate pool. The deletion continues with the remaining candidates until the number of members in the candidate pool is equal to the population size. After the deletion is complete, all remaining candidates are appointed to be the new population. Next, the new population is clustered into two groups based on the dominance concept: the new queens and the new drones. The nondominated individuals are assigned as the new queens while the rest are assigned as the new drones. However, if the number of the new queens is greater than the maximum number of queens, some nondominated individuals will be removed from being the queen and added to a group of new drones by using the similar method as in the deletion. That is, the weakest nondominated individual whose distance is less than the average distance is removed from being the queen until the number of the new queens is equal to the specified number. The swarm preparation process is complete when the new queens and the new drones have already been prepared in the required quantities.

The fifth stage is the swarm process. In this last stage, the drones are clustered into several colonies, the centroids of which are the queens by implication. In choosing the colony, the drone determines its new colony by considering the distance between itself and the queen of each colony. The colony holding the minimum distance is selected as a new home for the drone. Given $D = \{d_1, d_2, \ldots, d_p, \ldots d_P\}$ and $Q = \{q_1, q_2, \ldots, q_s, \ldots q_S\}$ are the sets of drones and queens respectively. The distance in the objective space between the drone d_p and the queen q_s is calculated according to the following equation:

$$d(d_p, q_s) = \left\| F(q_s) - F(d_p) \right| \tag{10}$$

$$K = \arg\min_{p} [d(d_p, q_s)] \tag{11}$$

where $F(q_s)$ is the fitness value of the queen q_s, $F(d_p)$ is the fitness value of the drone d_p, K is the index of the colony selected as a new home for the drone. The swarm process finishes after all drones have already been assigned to suitable colonies.

Finally, the termination criterion is checked. If it is not met, the whole algorithm will repeat again. In this study, the termination criterion is the number of mating flights defined by the user.

IV. Results and Discussions

In this paper, the ZDT test suite proposed by Zitzler, Deb, and Thiele [12] is used to test the performance of the proposed algorithm. The ZDT test suite comprises six test functions. This study uses five of those six test functions, which are ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. Each test function represents different characteristics including convexity, nonconvexity, discreteness, multimodality, and non-uniformity. All test functions have two objective functions. The results on five benchmark test functions of the proposed algorithm are compared with the published results of the PAES algorithm and the NSGA-II (real-coded) algorithm. The performance of all compared algorithms is measured in terms of the convergence metric ψ and the diversity metric Δ [3] as follows.

- The convergence metric ψ determines the average distance between each solution obtained from the algorithm and the nearest solution on the true Pareto front. The smaller the value of this matric, the better the convergence toward the pareto-optimal front.

$$\psi = \frac{\sum_{i=1}^{n} d_i}{n} \tag{12}$$

where d_i is the Euclidean distance between the i^{th} solution obtained from the algorithm and its nearest solution on the true Pareto front. n is the number of obtained solutions.

- The diversity metric Δ is used to measure the spread of the obtained solutions.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n} (d_i - \bar{d})}{d_f + d_l + (n-1)\bar{d}} \qquad (13)$$

where d_f is the Euclidean distance between the extreme solutions of the true Pareto-front. d_l is the Euclidean distance between the boundary solutions of the known Pareto front. d_i is the Euclidean distance between consecutive solutions in the known Pareto front. $\bar{d}$ is the average of all distances d_i.

Table I shows the mean and the variance (the figures in brackets) of the convergence metrics obtained by using three algorithms PAES, NSGA-II (real-coded), and the proposed algorithm. Results of the PAES algorithm and the NSGA-II algorithm shown in Table I and II are obtained from [3]. For the proposed algorithm, the results are the mean and the variance of 10 different runs; each run starts with a different initial population. The parameters of the proposed algorithm are set as follow: the maximum number of queens is set to 100, the number of drones is set to 100, the number of broods is set to 100, the queen's spermatheca size is set to 20, the mutation probability is set to 0.1, and the number of mating flights is set at 10000.

From Table I, the proposed algorithm achieves better convergence than the NSGA-II algorithm for all test functions, and produces better convergence than the PAES algorithm for ZDT1, ZDT2, ZDT3, and ZDT4.

Table II shows the mean and the variance (the figures in brackets) of the diversity metrics obtained by using all three algorithms. The proposed algorithm performs better than the PAES algorithm for all test functions, and outperforms the NSGA-II algorithm for the following three test functions, ZDT3, ZDT4, and ZDT6. In addition, the variances of the convergence metric and the diversity metric obtained with the proposed algorithm are also very small.

V. Conclusions

This paper proposes a modified marriage in honey-bee optimization for solving optimization problems with multiple conflicting objectives. Unlike the original marriage in honey-bee optimization, the proposed algorithm divides the objective space into several colonies, each of which has its own queen. Therefore, the multiple mating behavior of the queen, in which the queen performs with several drones from other colonies, helps maximizing a genetic variance in the population, which in turn improves the search ability of the algorithm. The experimental results on different ZDT test functions show that the proposed algorithm is well capable of converging to the true Pareto front with better spread of solutions than the other two state-of-the-art models.

TABLE I MEAN AND VARIANCE OF THE CONVERGENCE METRIC

Test Functions	Convergence Metric		
	PAES	*NSGA-II (real-coded)*	*The proposed model*
ZDT1	0.082085 (0.008679)	0.033482 (0.004750)	**0.030974** (0.001666)
ZDT2	0.126276 (0.036877)	0.072391 (0.031689)	**0.067908** (0.000717)
ZDT3	0.023872 (0.000010)	0.114500 (0.007940)	**0.022747** (0.001014)
ZDT4	0.854816 (0.527238)	0.513053 (0.118460)	**0.203611** (0.188460)
ZDT6	**0.085469** (0.006664)	0.296564 (0.013135)	0.439687 (0.005091)

TABLE II MEAN AND VARIANCE OF THE DIVERSITY METRIC

Test Functions	Diversity Metric		
	PAES	*NSGA-II (real-coded)*	*The proposed model*
ZDT1	1.229794 (0.004839)	**0.390307** (0.001876)	0.551515 (0.000827)
ZDT2	1.165942 (0.007682)	**0.430776** (0.004721)	0.563713 (0.000525)
ZDT3	0.789920 (0.001653)	0.738540 (0.019706)	**0.538479** (0.000263)
ZDT4	0.870458 (0.101399)	0.702612 (0.064648)	**0.504928** (0.000033)
ZDT6	1.153052 (0.003916)	0.668025 (0.009923)	**0.538407** (0.000290)

REFERENCES

[1] T. Bäck, Evolutionary Algorithms in Theory and Practice New York, NY: Oxford University Press, 1996

[2] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in Proc. of an Int. Conf. on Genetics and Their Applications, pp. 93–100, 1985.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transaction on Evolutionary Computation, vol. 6, pp. 182–197, 2002.

[4] C. M. Fonseca and P. J. Flemming, "An overview of evolutionary algorithms in multiobjective optimization," Evolutionary Computation, vol. 3(1), pp. 1–16, 1995.

[5] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in Proc. of Evolutionary Methods for Design Optimization and Control, pp. 19–26, 2001

[6] J. Knowles and D. Corne, "The Pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization," in Proc. of the 1999 Congress on Evolutionary Computation, pp. 98–105, 1999

[7] E. Y. C. Wong, H. S. C. Yeung, and H. Y. K. Lau, "Immunity-based hybrid evolutionary algorithm for multi-objective optimization in global container repositioning," Engineering Applications of Artificial Intelligence, vol. 22, pp. 842–854, 2009

[8] H. A. Abbass, "MBO: Marriage in honey bees optimization: A haplometrosis polygynous swarming approach," in Congress on Evolutionary Computation, pp. 207–214, 2001

[9] O. Bozorg-Haddad, A. Afshar, and M. A. Mariño, "Honey-bee mating optimization (HBMO) algorithm: A new heuristic approach for water resources optimization," Water Resources Management, vol. 20, pp. 661–680, 2006

[10] A. Thammano and P. Pookzamran, "SMBO: A self-organizing model of marriage in honey-bee optimization," Int. J. of Expert Syst. Appl., (impress).

[11] P. Pookzamran and A. Thammano, "The modified marriage in honey-bee optimization for function optimization problems," Procedia Computer Science, vol. 6, pp. 335–342, 2011.

[12] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," Evolutionary Computation, vol. 8(2), pp. 173–195, 2000

ประวัติผู้เขียน

ชื่อ – นามสกุล	นางสาวพัชรวดี พูลสำราญ
วันเกิด	28 สิงหาคม 2522
ที่อยู่	132 ม. 6 ต. บ้านซ่อง อ. พนมสารคาม จ. ฉะเชิงเทรา 24120
ประวัติการศึกษา	วท.บ. (วิทยาการคอมพิวเตอร์) เกียรตินิยมอันดับ 2 มหาวิทยาลัยบูรพา วท.ม. (เทคโนโลยีสารสนเทศ) สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ประสบการณ์ในการทำงาน	2544 - ปัจจุบัน อาจารย์ประจำสาขาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์และสังคมศาสตร์ มหาวิทยาลัยบูรพา วิทยาเขตสระแก้ว 2551 รักษาการคณบดีคณะวิทยาศาสตร์และสังคมศาสตร์ มหาวิทยาลัยบูรพา วิทยาเขตสระแก้ว